ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

**Кафедра комплексной информационной безопасности
электронно-вычислительных систем (КИБЭВС)**

А.А. Шелупанов, В.Н. Кирнос

ИНФОРМАТИКА. БАЗОВЫЙ КУРС

Учебник
(*в четырех частях*)

Часть 3
Основы алгоритмизации и программирования в среде Visual C++ 2005

*Допущено Сибирским региональным отделением
Учебно-методического объединения Вузов по образованию
в области информационной безопасности для межвузовского
использования в качестве учебного пособия по специальности* 090105
«Комплексное обеспечение информационной безопасности автоматизированных систем»

В-Спектр
Томск – 2008

УДК 681.3.06(075)
ББК 32.973-018я7
Ш-49

Ш 44 **Шелупанов А.А., Кирнос В.Н.** Информатика. Базовый курс / Учебник (в четырех частях). Ч. 3. Основы алгоритмизации и программирования в среде Visual C++ 2005. – Томск: В-Спектр, 2008. – 216 с.
ISBN 978-5-91191-091-4

В данной части учебника «Информатика. Базовый курс» освещаются вопросы разработки основных видов алгоритмов с последующим оформлением их на языке блок-схем. Также рассматриваются основы программирования на языке C++ в среде Visual Studio 2005. В конце каждого раздела даны задания, которые могут быть использованы при проведении практических занятий и лабораторных работ. В завершающей главе кратко рассматривается разработка приложений Windows Forms в этой же среде.

Для студентов технических специальностей вузов.

УДК 681.3.06(075)
ББК 32.973-018я7

ISBN 978-5-91191-091-4

СОДЕРЖАНИЕ

ГЛАВА 7. ОСНОВЫ АЛГОРИТМИЗАЦИИ

1. ПОНЯТИЕ ОБ АЛГОРИТМЕ

Как известно, процесс решения задачи с помощью компьютера включает в себя следующие этапы:

1. Постановка задачи
2. Построение математической модели
3. Разработка алгоритма (алгоритмизация)
4. Составление программы
5. Реализация программы на компьютере
6. Анализ результатов

Первые два этапа могут во многом оказаться решающими. В то же время однозначных рекомендаций для выполнения этих первых шагов не существует. Поэтому часто при обучении методам разработки алгоритмов и программирования используют задачи, формулировки которых заведомо освобождены от прохождения этих «неприятных» этапов. В решении таких учебных задач основная роль принадлежит *алгоритмизации*.

Понятие алгоритма относится к числу фундаментальных математических понятий. Здесь мы можем сформулировать определение алгоритма так:

Алгоритм – это однозначное, точное и полное описание последовательности элементарных действий для решения данной задачи.

Слово *алгоритм* происходит от имени арабского математика Аль-Хорезми (точнее – латинизированной формы его имени – Algorithmi), который еще в IX в. сформулировал правила выполнения четырех арифметических действий. Эти правила называли правилами Аль-Хорезми (algorithmi), а позднее просто стали называть *алгоритмом*.

Рассмотрим примеры

Пример 1.1.

Составить алгоритм перехода улицы.

Стандартным образом этот алгоритм звучит так:

1. Посмотреть налево.
2. Дойти до средины.
3. Посмотреть направо.
4. Переходить далее.

Пример 1.2.

Составить алгоритм вычисления площади данного треугольника.

1. Опустить высоту на одну из сторон.
2. Измерить длину *h* этой высоты.
3. Измерить длину *a* основания, на которое опущена высота *h*.
4. Перемножить *a* на *h* и разделить пополам.

Абу Джафар Мухаммад ибн Муса Аль-Хорезми,
скульптурный портрет (790–840 гг.)

Для однозначного и точного исполнения алгоритма, он должен удовлетворять ряду требований. Иначе говоря, обладать определенными свойствами. Приведем эти *свойства* алгоритма.

Дискретность. Это свойство алгоритма предполагает, что описываемый процесс должен быть разбит на последовательность *отдельных* шагов: только выполнив требования одного предписания, можно приступить к выполнению следующего. Это хорошо видно в предыдущих примерах – там приведена нумерация отдельных предписаний.

Понятность. Ясно, что, составляя алгоритм, нужно использовать те предписания, которые понятны предполагаемому исполнителю алгоритма. Бессмысленна, например, для нас инструкция к японскому принтеру Epson, написанная на японском языке.

Определенность. Это свойство означает: алгоритм не должен оставлять места для произвола исполнителя. После выполнения очередного предписания должно быть ясно, какое предписание является следующим. Алгоритм из примера 1.1 не является достаточно определенным, поскольку после «посмотреть налево» может быть и «переждать движущийся автомобиль» и, если его нет, то сразу «дойти до средины».

Массовость. Возможность применения алгоритма не к одной задаче, а к ряду однотипных задач и есть свойство массовости. Алгоритм примера 1.2 является массовым, поскольку применим к любому треугольнику, а пример 1.1 не содержит массового алгоритма (хотя бы потому, что не учитывает особенностей правил дорожного движения Японии и Англии).

Результативность. Смысл этого свойства в том, что алгоритм должен состоять из конечного числа шагов и при этом должен быть получен искомый результат.

Существует три основных способа написания алгоритма:

1) на естественном языке (русском, английском и т.д.),

2) на алгоритмическом языке (или языке программирования: Бейсик, Паскаль, С и т.д.),

3) на языке блок-схем.

Рассмотренные выше примеры содержали алгоритмы, написанные на *естественном языке*. Он служит основным средством общения в человеческом обществе. Разного рода правила, инструкции, рецепты – не что иное, как алгоритмы, записанные на естественном языке. Записью алгоритмов на языке программирования мы пока заниматься не будем (это делается при изучении конкретного языка программирования), а будет рассматривать подробно в следующих параграфах язык блок-схем, который активно используется при записи вычислительных алгоритмов.

Блок-схема алгоритма – *это графический способ записи алгоритма, представляющий собой систему определенным образом связанных блоков, изображаемых в виде плоских геометрических фигур.*

Элементы блок – схемы располагаются сверху вниз, линии соединения отдельных блоков показывают направление процесса обработки схемы. Каждое такое направление называют ветвью.

Перечислим типы блоков:

1. «Начало» и «конец» алгоритма изображаются овалом (рис. 7.1, *а*, *б*).

а *б* *в*

Рис. 7.1. Блоки обозначения начала, конца алгоритма и действия

2. Блок «действия» изображается прямоугольником. Внутри его указываются необходимые вычисления и присваивания результата (рис. 7.1, *в*)

3. Блок «условия» изображаются ромбом. Внутри блока записываются условия выбора направления действия алгоритма (рис. 7.2):

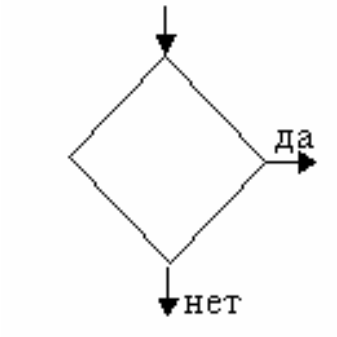


Рис. 7.2. Блок «условия»

Рис. 7.3. Блоки «ввода» и «вывода»

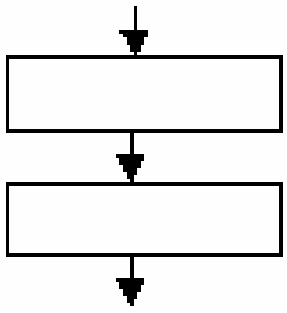
4. Блоки «ввода» и «вывода» информации изображаются параллелограммами. С их помощью вводят исходные данные задачи, выводят результат решения (рис. 7.3)

2. ОСНОВНЫЕ СТРУКТУРЫ

Из перечисленных блоков составляют структуры алгоритмов.

Структурами *называют ограниченный набор блоков и стандартных способов их соединения для выполнения типичных последовательностей действий.*

Алгоритмы «собирают» из трех основных (базовых) структур: *следование*, *развилка*, *цикл*.



2.1. Структура «следование»

Структура «следование» состоит из двух (*или более*) *блоков «действие»* (рис. 7.4):

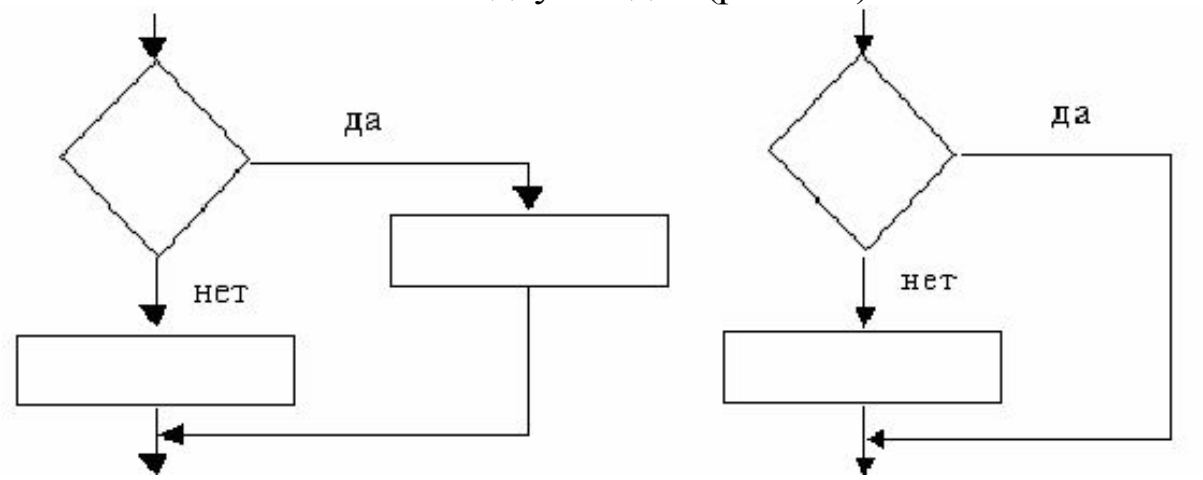
Рис. 7.4. Структура «следование»

Однако в качестве элементов этой структуры могут выступать и базовые структуры. Именно поэтому правильнее называть элементы такой структуры *функциональными блоками* и в дальнейшем под функциональными блоками будем понимать не только блок «действие», но и любую базовую структуру.

2.2. Структура «развилка»

Структура «развилка» состоит из логического элемента с проверкой некоторого условия и функциональных блоков, которые в простейшем случае есть блок «действие».

«Развилка» может быть двух видов (рис. 7.5):



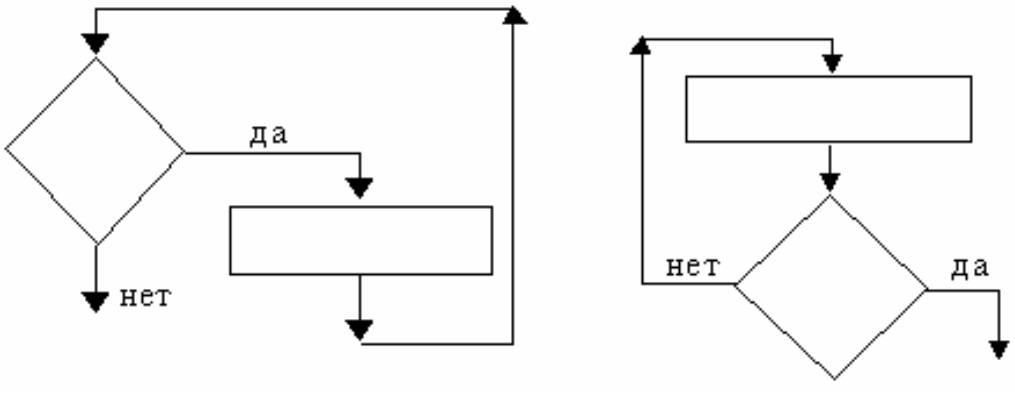
а *б*

Рис. 7.5. *а* – полная условная конструкция; *б* – неполная условная конструкция

2.3. Структура «цикл»

Структура «цикл» состоит из логического элемента с проверкой условия и функционального блока, называемого телом цикла. Ясно, что тело цикла может выполняться неоднократно.

Данная структура может быть двух видов (см. рис. 7.6, *а*, *б*):



а *б*

Рис. 7.6. *а* – цикл «ПОКА»; *б* – цикл «ДО»

В случае цикла «ПОКА» функциональный блок размещен после проверки условия, поэтому может оказаться, что тело цикла не выполниться ни разу. Однако если условие выполняется – выполняется и весь цикл. Проще говоря, цикл «ПОКА» выполняется, *пока* выполняются условие. Иное название этого цикла – цикл с *предусловием*.

В цикле «ДО» функциональный блок размещен до проверки выполнения условия, так что в этом варианте тело цикла в любом случае будет выполнено по крайней мере один раз. Условие в данном случае является условием выхода из цикла. Проще говоря, цикл «ДО» выполняется вплоть *до* наступления выполнения условия. Иное название этого цикла – цикл с *постусловием*.

3. ОСНОВНЫЕ ТИПЫ АЛГОРИТМОВ

Алгоритмы *бывают трех основных типов: линейный*, *разветвленный*, *циклический.*

3.1. Линейный алгоритм

Линейный алгоритм не содержит логических условий, *имеет одну ветвь обработки и изображается линейной последовательностью блоков.*

Проще говоря, линейный алгоритм строиться на основе структуры «следование». Условное изображение линейного алгоритма (рис. 7.7):

В качестве примера можно привести алгоритм вычисления, среднего арифметического трех чисел *a*,*b*,*c* (на рисунке изображено справа).

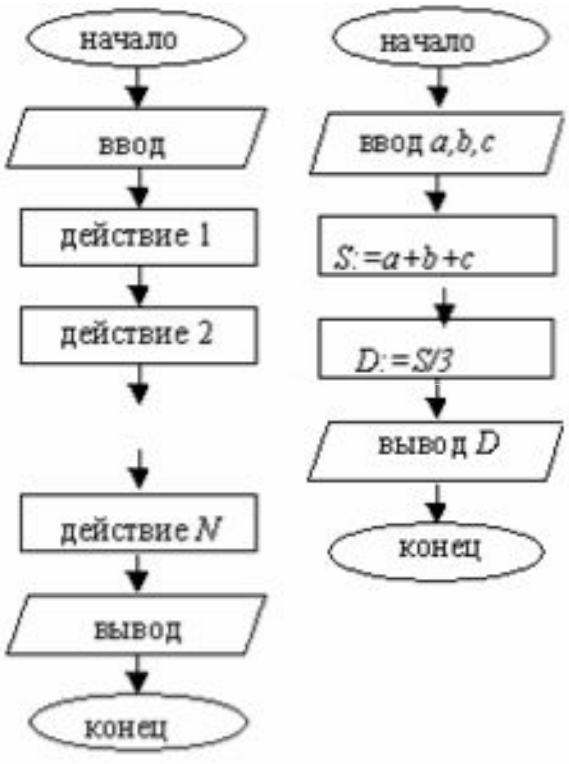
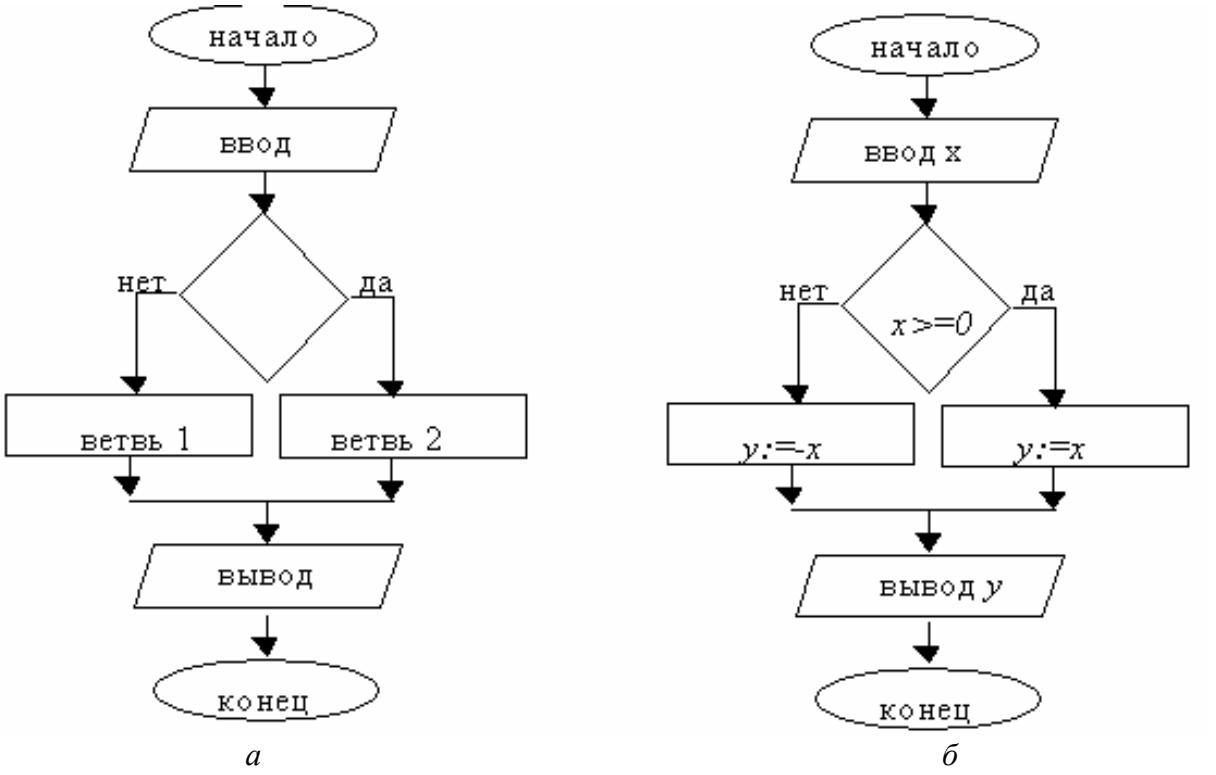


Рис. 7.7. Запись линейного алгоритма на языке блок-схем

3.2. Разветвленный алгоритм

Разветвленный алгоритм содержит одно или несколько логических условий и имеет несколько ветвей обработки. Иначе говоря, разветвленный алгоритм строится на основе структуры *«развилка».*

Условное обозначение такого алгоритма (см. рис. 7.8):



а *б*

Рис. 7.8. Запись разветвленного алгоритма на языке блок-схем

В качестве примера (на рисунке справа) рассмотрен алгоритм вычисления функции:

$$y=|x|=\begin{cases} x, \text{ если } x\geq 0 \\ -x, \text{ если } x<0 \end{cases}$$

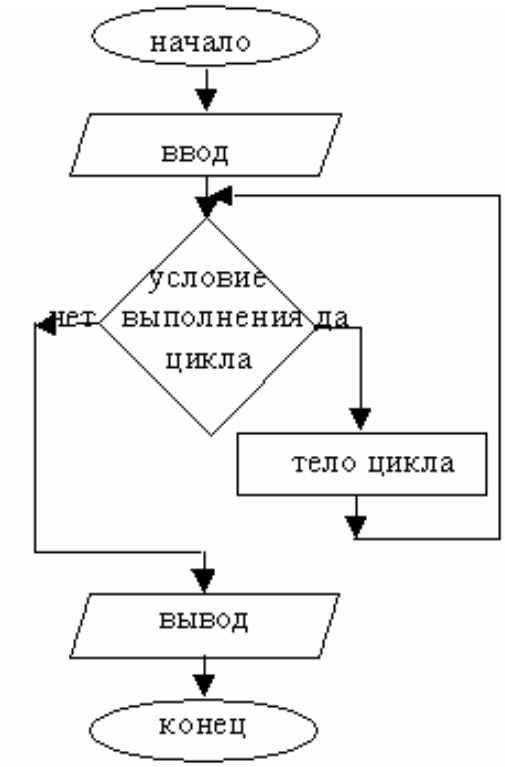
3.3. Циклический алгоритм

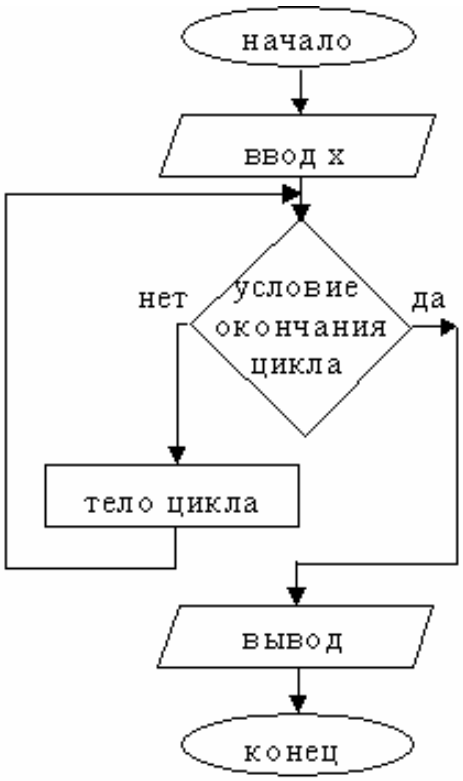
Циклический алгоритм содержит один или несколько циклов. Он строиться, таким образом, на основе структуры «цикл». Все ветви алгоритма должны, в конце концов, сойтись.

Условное обозначение циклического алгоритма (см. рис. 7.9, *а*, *б*).

В качестве примера предлагается:

- *цикл «ПОКА»* – вычислять сумму чисел 1, 1/2, 1/3 и т.д. пока не окажется, что очередное слагаемое меньше числа *E* (см. рис. 7.10);
- *цикл «ДО»* – для последовательно вводимых в цикле *N* чисел найти наибольшее (см. рис. 7.11).





а *б*

Рис. 7.9. *а* – запись циклических алгоритмов на языке блок-схем (на основе цикла «ПОКА»); *б* – запись циклических алгоритмов на языке блок-схем (на основе цикла «ДО»)

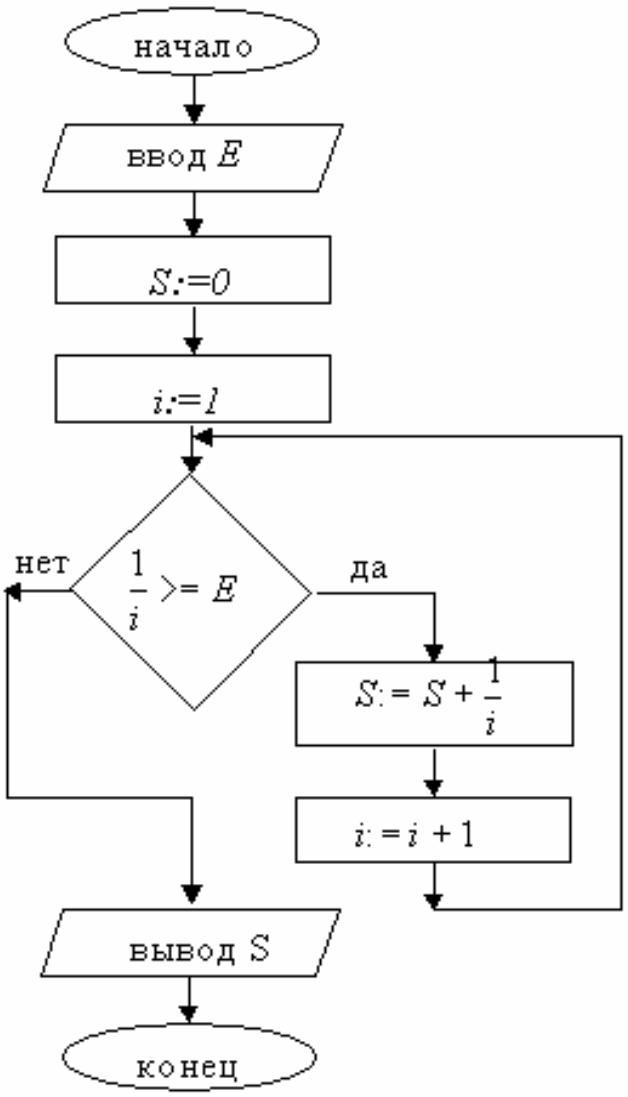


Рис. 7.10. Алгоритм вычисления суммы: $1+1/2+1/3+...$ с заданной точностью (цикл «ПОКА»)

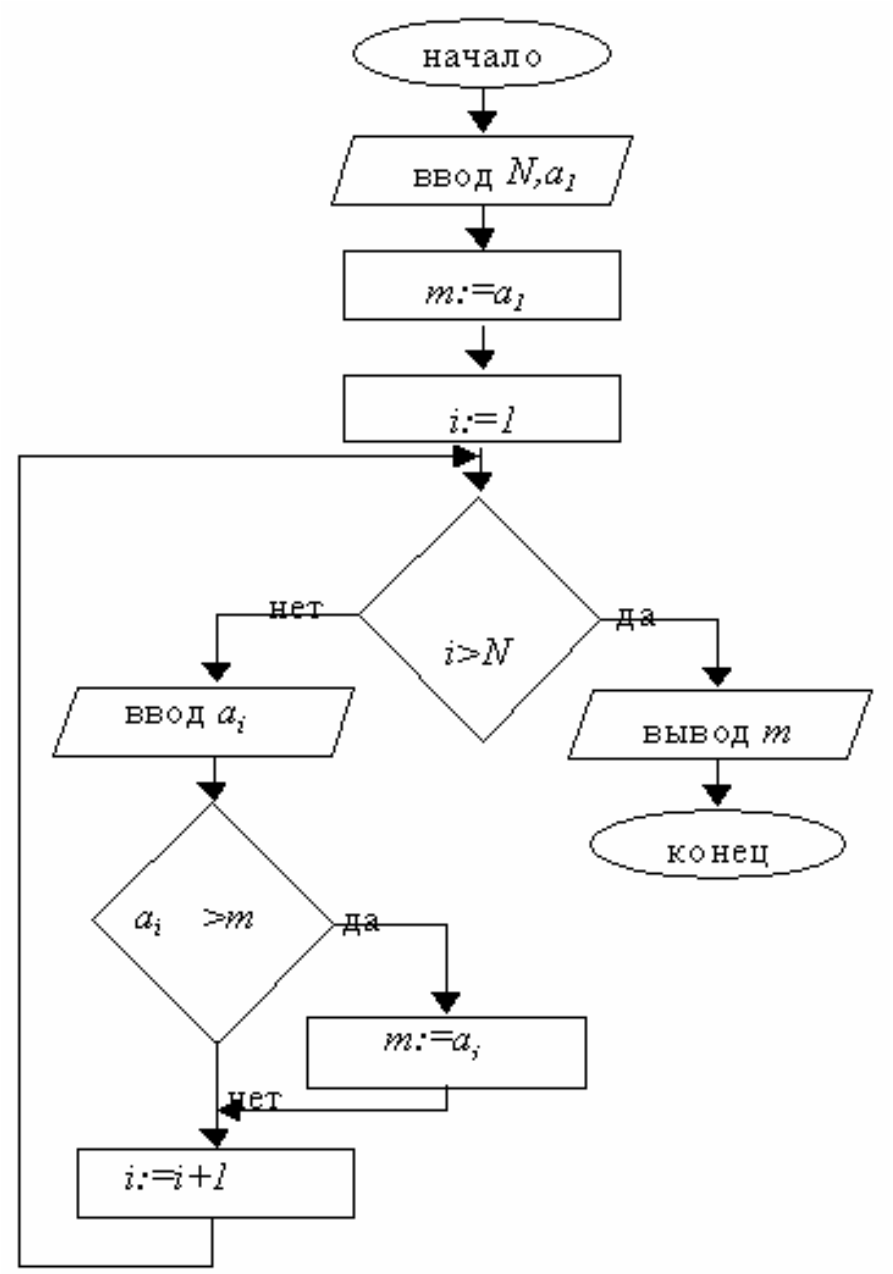
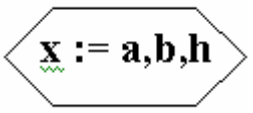


Рис. 7.11. Алгоритм поиска наибольшего из N последовательно вводимых чисел (цикл «ДО»)

В тех случаях, когда заранее известно число повторений цикла, используют частный случай цикла «ДО» – так называемый цикл «ДЛЯ».

Рассмотрим на примере вычисления значения функции $y=x^2$ при изменении интервале [a, b] с шагом h. Блок-схема алгоритма приведена на рис. 7.12**.**

Конструкция цикл «ДЛЯ» может быть сформулирована в данном случае так: «повторять ***для*** x от a до b с шагом h». Для записи цикл «ДЛЯ» даже используют специальный блок в виде продолговатого шестиугольника:



Этот блок позволяет сделать схему алгоритма более компактной и наглядной.

В свою очередь, частным случаем цикла «ДЛЯ» является цикл «повторять *n* раз» или *«цикл со счетчиком»*, который очень часто используется в вычислительных задачах. В языках программирования, как правило, такой цикл организуется при помощи специального оператора.

В качестве примера рассмотрим *вычисление N*! (*N*! = 1*2*3* …**N*). Блок-схема алгоритма приведена на рис. 7.13.

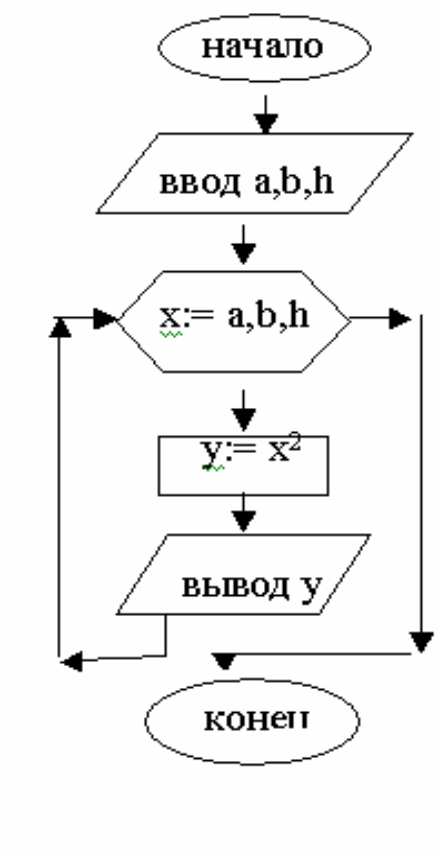


Рис. 7.12. Блок-схема алгоритма вычисления функции $y = x^2$ на интервале [*a*, *b*] с шагом *h* (цикл «Для»)

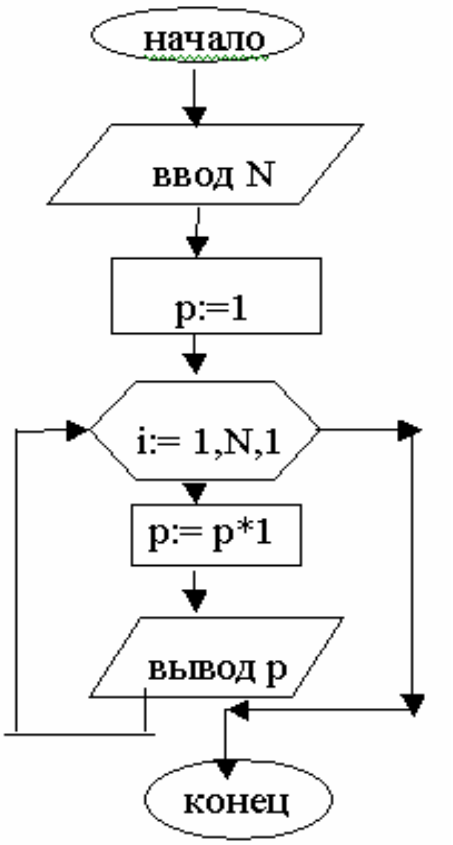


Рис. 7.13. Блок-схема алгоритма вычисления факториала числа *N* (цикл со счетчиком)

Практические задания

1. Алгоритмы линейной структуры

1.1. Вычислить площадь треугольника со сторонами *a*, *b*, *c* по формуле Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)} \quad \text{, где } p = (a+b+c)/2$$

1.2. Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.

1.3. Найти площадь равнобочной трапеции с основаниями a и b и углом при большем основании x.

1.4. Найти угол между отрезком прямой, соединяющей начало координат с точкой $A(x, y)$ и осью $0X$ (точка лежит в 1-й четверти).

1.5. Вычислить период колебания маятника длины L.

1.6. Определить время падения камня на поверхность земли с высоты h.

1.7. Три сопротивления R_1, R_2, R_3 соединены параллельно. Найти сопротивление соединения.

1.8. Смешано V_1 литров воды температуры t_1 с V_2 литрами воды температуры t_2. Найти объем и температуру образовавшейся смеси.

1.9.* Высота жидкости в цилиндрической цистерне, лежащей на боку, равна h. Найти объем жидкости, если длина цистерны равна L, а радиус основания R.

1.10.* Найти периметр и площадь пятиугольника, заданного координатами своих вершин.

2. Алгоритмы разветвляющейся структуры

2.1. Даны числа a, b, c. Проверить выполняется ли неравенство $a<b<c$.

2.2. Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (1,3).

2.3. Даны числа X, Y ($X<>Y$). Меньшее из этих двух заменить их полусуммой, а большее – их удвоенным произведением.

2.4. Найти наибольшее из трех заданных чисел.

2.5. Выяснить, существует ли треугольник с длинами сторон x, y, z (в треугольнике большая сторона меньше суммы двух других сторон).

2.6. На окружности с центром в точке (X_0, Y_0) задана дуга с координатами начальной (X_n, Y_n) и конечной (X_k ,Y_k) точек. Определить номера четвертей окружности, в которых находиться начальная и конечная точки.

2.7. Определить высоту h уровня жидкости, отчитываемой от дна резервуара, который имеет форму усеченного конуса высотой L и радиусами оснований R и r, завершенного цилиндром с радиусами основания R, если в резервуар налито V литров горючего.

2.8. Даны действительные положительные числа a, b, c, x, y. Выяснить пройдет ли кирпич с ребрами a, b, c в прямоугольное отверстие со сторонами x и y. Просовывать кирпич разрешается только так, чтобы каждое из его ребер было параллельно или перпендикулярно каждой из сторон отверстия.

2.9.* Даны действительные числа X_1, X_2, X_3, Y_1, Y_2, Y_3. Принадлежит ли начало координат треугольнику с вершинами (X_1, Y_1), (X_2, Y_2), (X_3, Y_3).

2.10.* В прямоугольнике, заданном координатами своих вершин, подсчитать количество прямых углов (если они есть).

3. Алгоритмы циклической структуры (цикл «ПОКА»)

3.1. Вычислить сумму вида: 1–1/2+1/3–1/4+... с заданной точностью E (когда очередное слагаемое станет меньше E по модулю, то суммирование прекратить).

3.2. Среди чисел 1, 1+1/2, 1+1/2+1/3, ... найти первое, большее заданного числа A.

3.3. Для заданного X в последовательности вида:

$$\sin X,\ \sin(\sin X),\ \sin(\sin(\sin X)),\ \ldots$$

найти первое число, меньшее по модулю 0.01.

3.4. Найти наименьший номер n, для которого выполняется условие: $|a_n - a_{n-1}| < 10^{-3}$, если последовательность a_n имеет вид :

$$a_{n+1} = 1/2(a_n + 2/a_{n-1}),\ a_1 = 1$$

3.5. С заданной точностью E рассчитать сумму вида:

$$1 + x/1! + x^2/2! + x^3/3! + \ldots$$

Результат сравнить со значением $y = e^x$.

Указание: *для вычисления* $n!$ *применять рекуррентную формулу* $n!=(n\text{-}1)!*n$

3.6. С заданной точностью E рассчитать сумму вида:

$$x - x^3/3! + x^5/5! + x^7/7! + \ldots$$

Результат сравнить со значением $y = \sin X$.

3.7. С заданной точностью E рассчитать сумму вида:

$$1 - x^2/2! + x^4/4! - x^6/6! + \ldots$$

Результат сравнить со значением $y = \cos X$.

3.8. С заданной точностью E рассчитать сумму:

$$1+(1/3)x/1!+(1/3)(1/3+1)x/2!+(1/3)(1/3+1)\cdot(1/3+1)\cdot(1/3+2)\cdot x/3!+\ldots$$

для $|x| < 1$. Результат сравнить со значением: $y=\sqrt{1+x}$

3.9. Для заданной функции $f(x)$ с указанной точностью E рассчитать площадь под кривой $y=f(x)$ на отрезке $a<x<b$, пользуясь формулой:

$$S=h(f(x_1)+f(x_2)+\ldots+f(x_n)),$$

где $h = (b - a)/n$, $x_i = a + i_h - h/2$ (т.е. подобрать n такое, чтобы была достигнута точность E).

3.10. Рассмотрим бесконечную последовательность X_1, X_2, ..., образованную по следующему закону: $X_1 = (u + 1)/2$

$$X_i = 1/2(x_{i-1} + u/x_{i-1}),\ i = 2, 3, \ldots$$

где u – заданное неотрицательное действительное число. Эта последовательность позволяет получить сколь угодно точные приближения числа $\sqrt{u}$. Для заданных u и n составить алгоритм приближенного вычисления квадратного корня из u с заданной точностью E.

4. Алгоритмы циклической структуры (цикл «ДО»)

4.1. Для заданного натурального n рассчитать величины:

а) $n!=1\cdot2\cdot3\cdot\ldots\cdot n$

б) $n!!=1\cdot3\cdot5\cdot\ldots\cdot n$, если n – нечетное или

$=2\cdot4\cdot6\cdot\ldots\cdot n$, если n – четное

4.2. Для заданного натурального N рассчитать сумму:

$$2\cdot1+3\cdot2\cdot1+\ldots+N\cdot(N-1)\cdot\ldots\cdot1$$

4.3. Вычислить сумму:

$$y = \cos x + \cos x^2 + ... + \cos x^{30}$$

4.4. Вычислить сумму:

$$y = \sin 1 + \sin 1{,}1 + \sin 1{,}2 + ... + \sin 2$$

4.5. Дано 10 вещественных чисел: a_1, a_2, ... a_{10}. Найти порядковый номер того из них, которое наиболее близко к какому–нибудь целому числу.

4.6. Дано 10 вещественных чисел. Определить, сколько из них принимает значение, большее заданного A.

4.7. Дано 100 чисел. Определить, сколько из них больше своих соседей, т.е. предыдущего и последующего чисел.

4.8. Вычислить K – количество точек с целочисленными координатами, попадающих в круг радиуса R с центром в начале координат.

4.9.* Для заданной функции $f(x)$ с указанным N рассчитать площадь под кривой $y = f(x)$ на отрезке $a < x <= b$, пользуясь формулой

$S = h(f(a)/2 + f(a + h) + f(a + 2h) + ... + f(b – h) + f(b)/2)$, где $h = (b – a)/N$.

4.10.* Даны целые числа x_1, y_1, x_2, y_2, ..., x_N, y_N. Выяснить, найдутся ли среди точек с координатами (x_1, y_1), (x_2, y_2), ...,(x_N, y_N) четыре таких, которые являются вершинами квадрата.

ГЛАВА 8. ОСНОВЫ ПРОГРАММИРОВАНИЯ В СРЕДЕ VISUAL C++ 2005

ВВЕДЕНИЕ

Современные системы программирования на C++ состоят из нескольких составных частей. Это такие части, как сама среда программирования, язык, стандартная библиотека С-функций и различные библиотеки С-классов.

Сразу заметим, что C++ является *объектно-ориентированным* языком. Основное отличие его от прежних, *структурных*, языков (примером таких является Турбо-Паскаль или С) является то, что он (C++) способен оперировать не только с переменными и структурами (функциями и процедурами), но и с целыми объектами. *Объекты* есть комплексы переменных и процедур (функций) по их обработке.

Как правило, чтобы выполнить программу на C++, необходимо пройти через 6 этапов: редактирование, препроцессорную (то есть предварительную) обработку, компиляцию, компоновку, загрузку и выполнение. В данном пособии мы будем рассматривать программирование в среде *Visual C++* 2005, входящую в состав Visual Studio 2005.

Первый этап представляет создание и редактирование файла с исходным текстом программы. Он может выполняться с помощью простейшего редактора текстов программ. Программист набирает в этом редакторе свою C++ программу. При необходимости он снова обращается к ней и вносит с помощью этого редактора изменения в исходный текст программы. Далее программа запоминается на диске. Имена файлов C/C++ программ оканчиваются на «c» или «cpp». Однако, пакет программ Visual C++ 2005 имеет встроенный редактор, которым также можно пользоваться.

На втором этапе компилятор начинает препроцессорную обработку текста программы, прежде чем ее компилировать. (Что же делает компилятор? Он переводит программу в машинный код. То есть в результате получаем объектный код программы, но это третий этап.). Следует знать, что в системе C++ программирования перед началом этапа самой трансляции всегда выполняется программа предварительной обработки. Что она делает? Она отыскивает так

называемые «директивы трансляции» или «директивы препроцессора», которые указывают, какие нужно выполнить преобразования перед трансляцией исходного текста программы. Обычно это включение других текстовых файлов в файл, который подлежит компиляции. Препроцессорная обработка инициируется компилятором перед тем, как программа будет преобразована в машинный код. Это позволяет забирать нужные программы-функции в текст компилируемой программы до начала процесса компоновки.

Третий этап – это **компиляция**. Как правило, программы на языке C++ содержат ссылки на различные функции, которые определены вне самой программы. Например, в стандартных библиотеках или в личных библиотеках программистов. Объектный код, созданный компилятором, содержит «дыры» на месте этих отсутствующих частей.

Четвертый этап – компоновка. Компоновщик связывает объектный код с кодами отсутствующих функций и создает, таким образом, исполняемый загрузочный модуль (без пропущенных «дыр»).

Пятый этап – загрузка. Перед выполнением программа должна быть размещена в памяти. Это делается с помощью загрузчика, который забирает загрузочный модуль программы с диска и перемещает его в память.

Наконец шестой этап – это **выполнение**. Программа редко заработает с первой попытки. Каждый из названных этапов может заканчиваться ошибкой или неудачей из-за ошибки.

Тогда программист должен вернуться к редактированию исходного текста программы. Он должен внести необходимые изменения в текст программы, предварительно его хорошо проанализировав. Затем снова пройти через все этапы работы с исходным текстом программы до получения работающего без ошибок загрузочного модуля.

1. РАЗРАБОТКА ПРОГРАММЫ

Запуск среды Visual Studio 2005 выполняется командой *Пуск/Программы/Microsoft Visual Studio* 2005*/Microsoft Visual Studio* 2005. Далее, если это первый запуск данной интегрированной среды, следует выбрать Visual C++ (напомним, что Visual Studio представляет собой единую среду, содержащую комплекс из Visual C++, Visual Basic и др. программных сред). При этом появится окно вида (рис. 8.1):

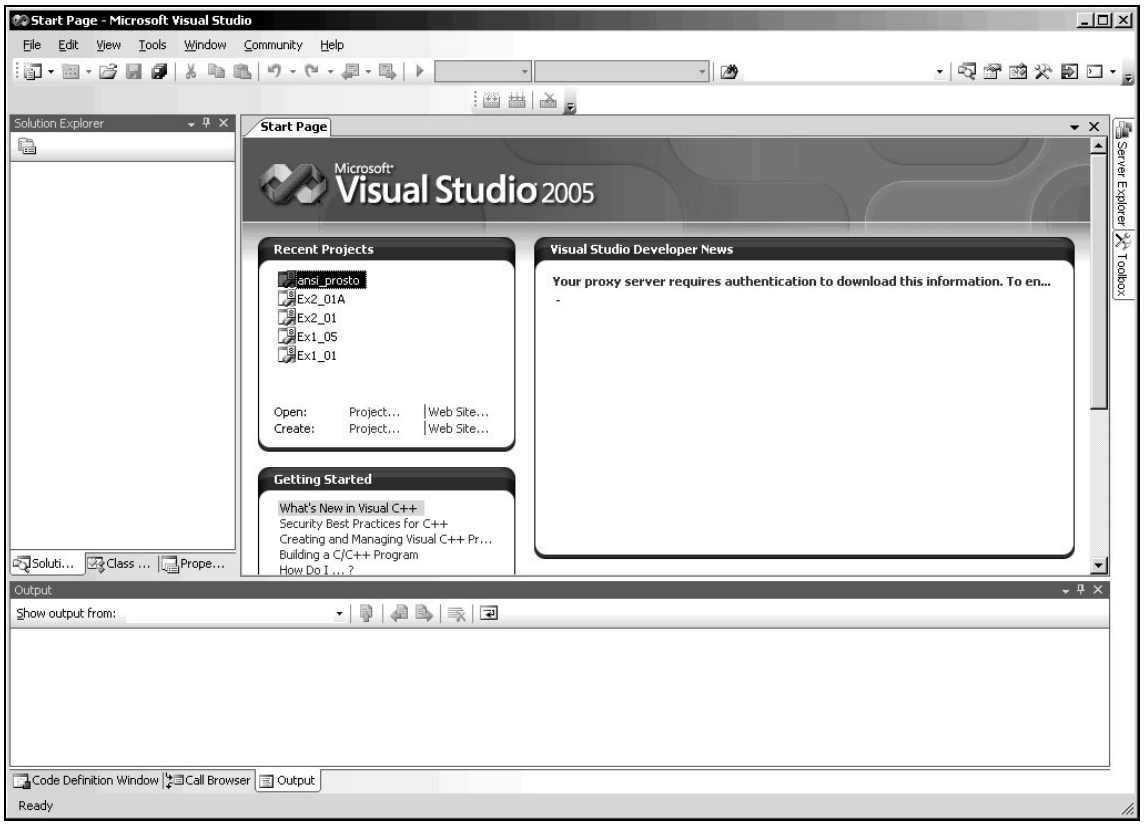


Рис. 8.1. Начальное окно среды Microsoft Visual Studio 2005

Часть окна слева на рис. 8.1 называется **окном проводника решений** (Solution Explorer), правое верхнее окно, содержащее стартовую страницу (Start Page) – это **окно редактора** (Editor), а окно в нижней части называется **окном вывода** (Output). Окно проводника решений позволяет осуществлять навигацию по программным файлам, отображать их содержимое в окне редактора, а также добавлять новые файлы к вашей программе.

Окно редактора – это место, где вы вводите и модифицируете исходный код программы. Окно вывода отображает сообщения, полученные при компиляции и компоновке вашей программы.

Первый шаг при написании программы в среде Visual C++ состоит в создании проекта. *Проект – это контейнер для всех со-*

ставляющих его программ: обычно один или более исходных файлов, содержащих ваш код и ряд вспомогательных файлов. Все файлы проекта обычно сохраняются в отдельной папке проекта.

Итак, для создания проекта в среде Visual C++ нужно дать команду *File/New/Project* (или нажать комбинацию клавиш **Ctrl-Shift-N**). В левой части возникшего окна (рис. 8.2) отображаются типы проектов, которые можно создавать. *Мы с вами пока будем создавать только консольные приложения.* Поэтому в левой части выберем **Win32**, а на правой панели выберем **Win32 Console Application**.

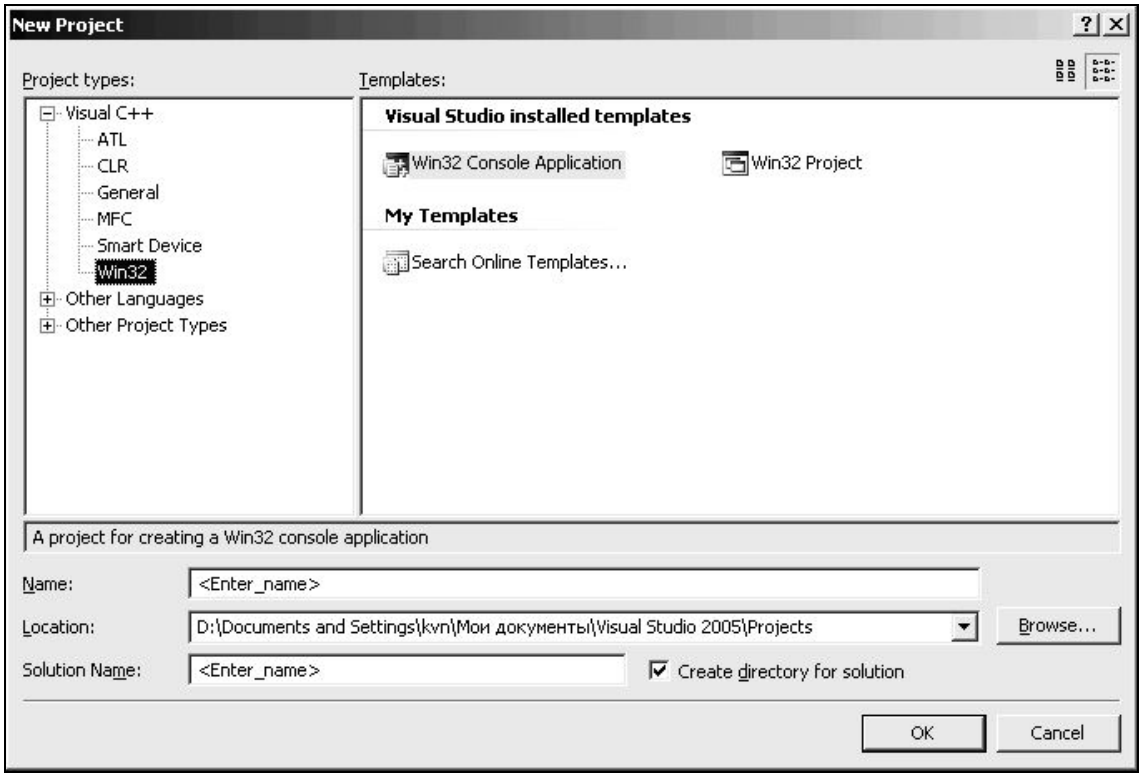


Рис. 8.2. Окно создания проекта консольного приложения

Ниже (в строке Name) ввести имя для проекта (зададим, например, **Prim1-1**) и нажать ОК. Обратите внимание, что при этом должна быть установлена «птичка» **Create directory for solution** (создать каталог для проекта). В следующем окне нажмем кнопку **Next** и в очередном окне (см. рис. 8.3) выберем **Empty project** и нажмем **Finish**.

В результате будет окно (рис. 8.4), в котором слева (в **проводнике решений**) появляется список подпапок данного проекта. Нас интересует теперь создание исходного файла. Для этого следует щелкнуть правой клавишей по папке **Source files** и выбрать ***Add/New Item***... (см. рис. 8.4) и в возникшем окне (рис. 8.5) выбрать

Code, указать **C++ File (.cpp)**, ввести имя для исходного файла (в строке **Name –** мы здесь ввели **prim1-1**) и нажать **Add**. В результате появится окно редактора, где и вводим исходный текст кода программы. Затем сохраняем его командой ***File*/*Save all*** **.**

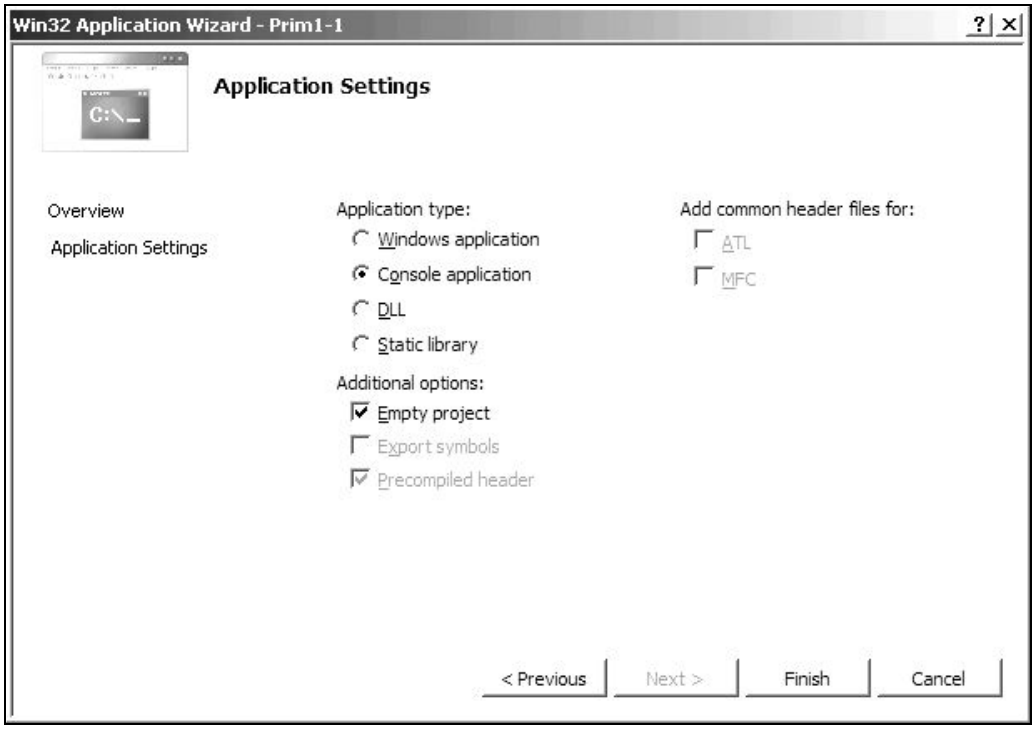


Рис. 8.3. Окно свойств приложения с выбором типа проекта

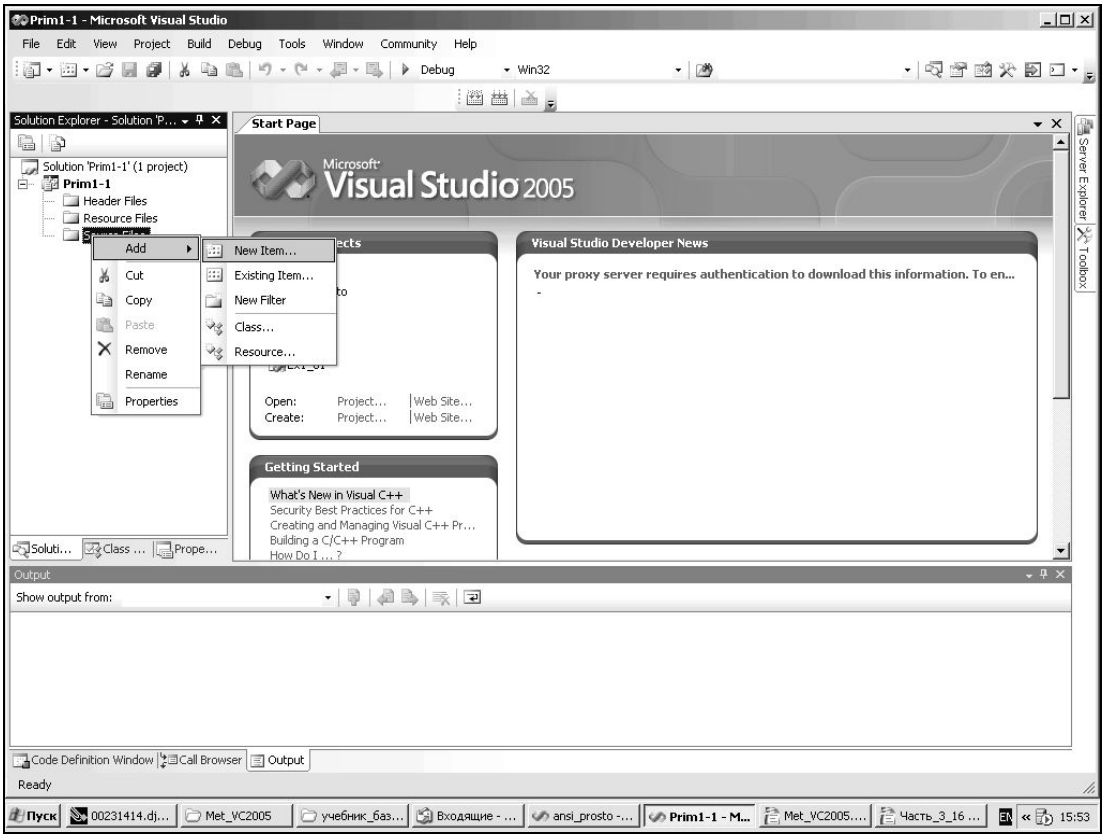


Рис. 8.4. Выбор вида исходного файла для проекта

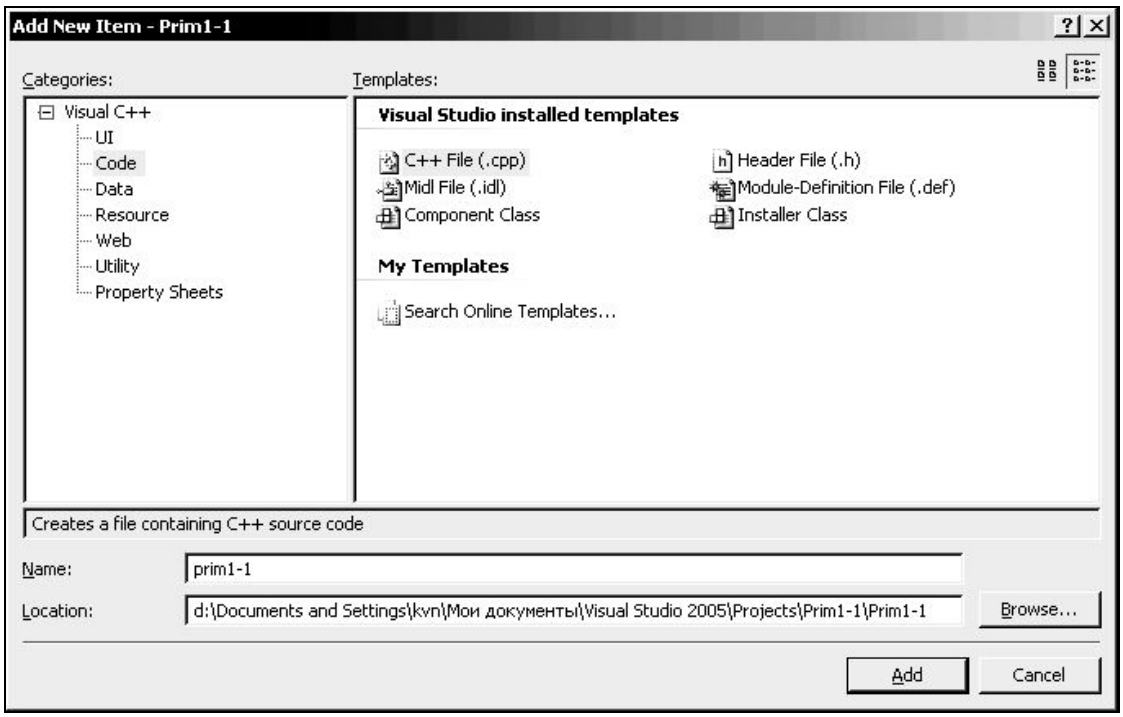


Рис. 8.5. Добавление нового исходного файла к проекту

Замечание. Если у вас еще не отображаются номера строк, дайте команду **Tools/Options**, затем разверните пункт **Text editor** и подпункт **C/C++**. Выберите у последнего **General**, а справа установите «птичку» у **Line Numbers** (см. рис. 8.6), нажмите ОК. Нумерация строк удобна потому, например, что транслятор нам будет сообщать, в какой строке (по номеру) ошибка.

Попробуем ввести текст простой программы (о содержимом каждой строки поговорим ниже). В результате окно редактора с введенным текстом программы приобретет вид (см. рис. 8.7).

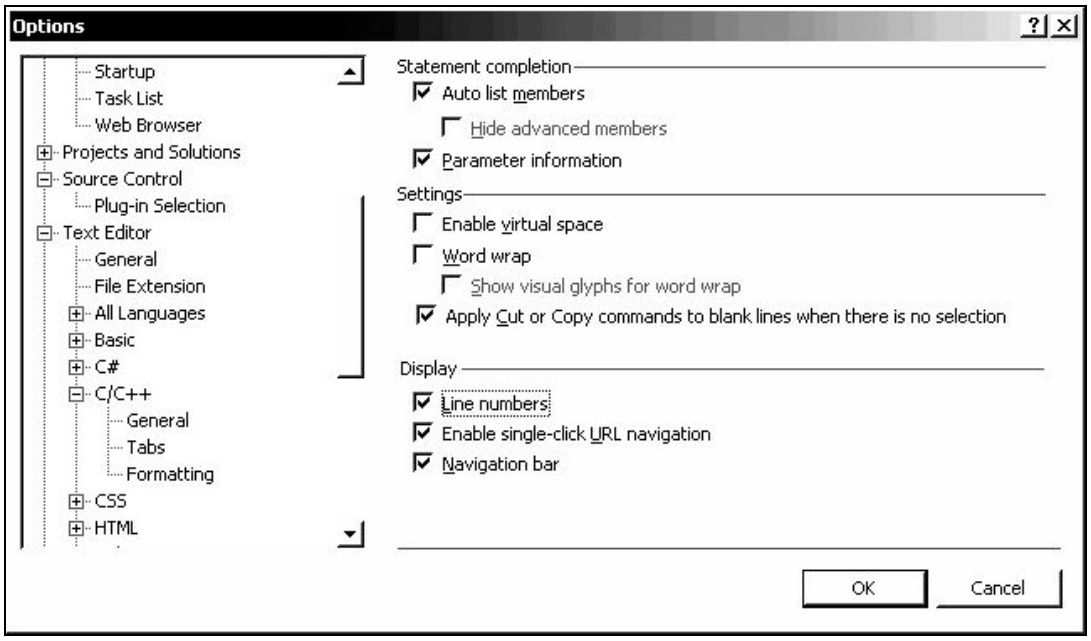


Рис. 8.6. Настройка отображения нумерации строк

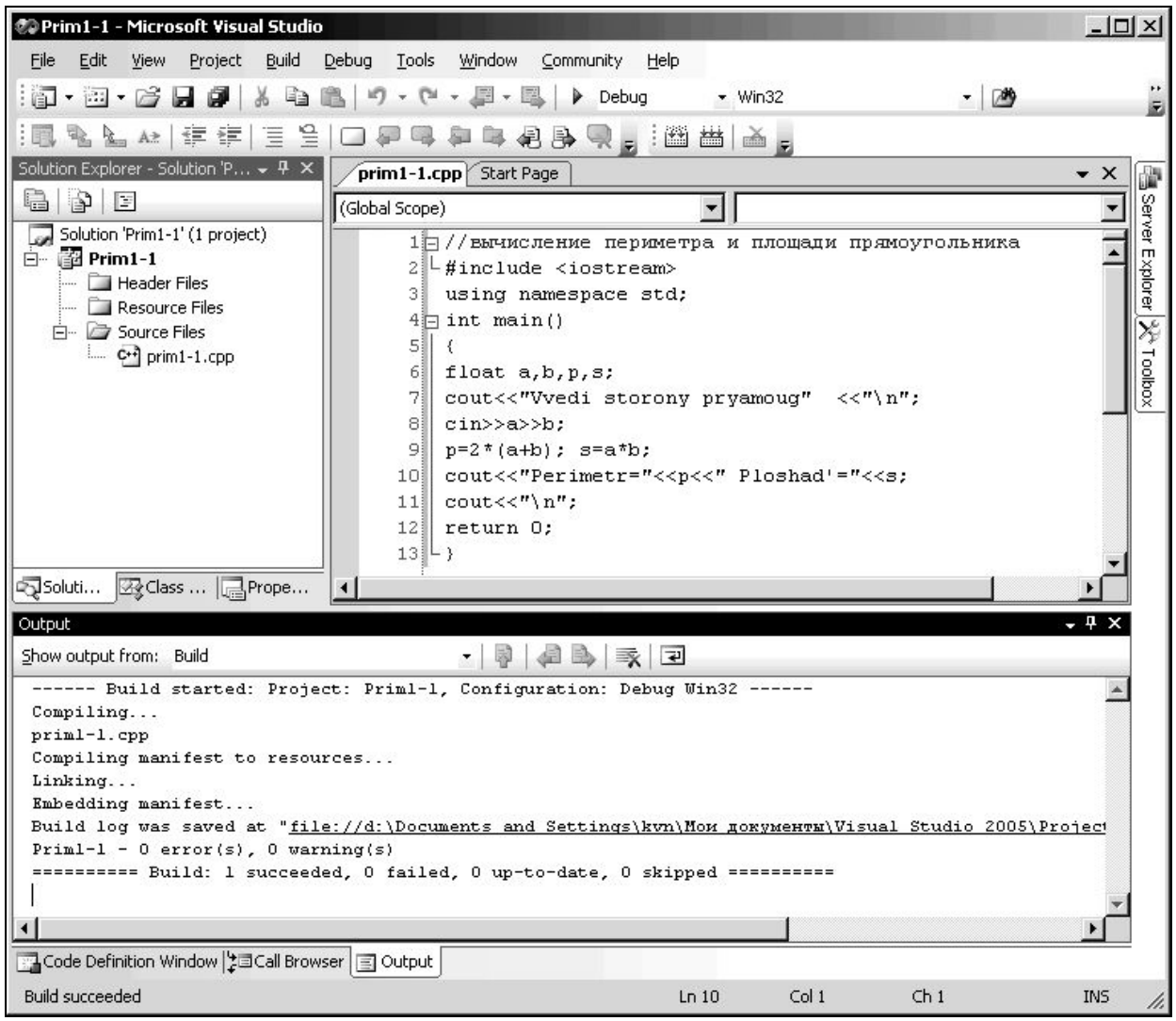


Рис. 8.7. Окно с результатами компиляции и компоновки

Далее нужно сначала провести компиляцию и компоновку – дать команду ***Build*/*Build Solution*** (или клавиша **F7**). Если все в порядке – в окне **Output** должно быть «0 failed» (см. рис. 8.7). Если же будут ошибки при компиляции, то искать ошибки в соответствии с надписями, которые появятся в нижней части окна – в окошке **Output**. Там, как правило, указывается в какой строке и какая ошибка обнаружена. После исправления ошибок, снова надо провести компиляцию и компоновку. Если же все пройдет без ошибок, то собственно запускаем на исполнение – клавишами **Ctrl-F5** (***Debug*/*Start Without Debugging***). В результате откроется «черное окно», где вводим исходные данные и получаем результат – см. рис. 8.8.

Заметим, что результаты работы программы выводятся на черном фоне, но в данном пособии мы приводим эти изображения в инвертированном цвете, чтобы не перегружать «чернотой».

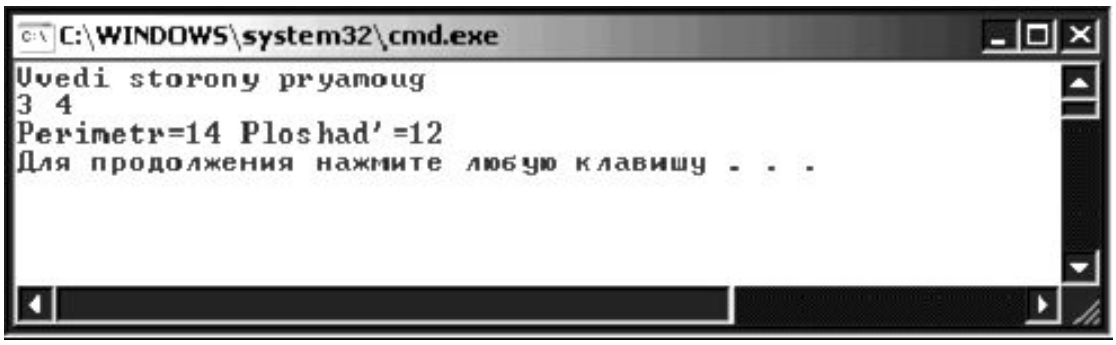


Рис. 8.8. Окно с результатами исполнения программы

Итак, **порядок разработки программы на Visual C++ 2005 (консольное приложение)**:

1) Создать проект: дать команду ***File/New/Project*** (или нажать комбинацию клавиш **Ctrl-Shift-N**)
2) В возникшем окне на левой панели выбрать **Win32** и на правой выберем **Win32 Console Application (**для создания консольного приложения), в строке **Name** введем имя проекта (при этом должна быть установлена «птичка» **Create directory for solution** создать каталог для проекта) и нажать ОК
3) в очередном окне появляется список подпапок данного проекта. Для создания исходного файла следует щелкнуть правой клавишей по папке **Source files** и выбрать ***Add/New Item***... (см. рис. 8.4)
4) в возникшем окне (рис. 8.5) выбрать **Code**, указать **C++ File (.cpp)**, ввести имя для исходного файла (в строке **Name**) и нажать **Add**.
5) в окне редактора вводим текст программы и сохраняем его командой ***File/Save all*** (или клавишами **Ctrl-Shift-S**)
6) для запуска программы на выполнение, нужно сначала провести компиляцию и компоновку – дать команду ***Build/Build Solution*** (или клавиша **F7**).
7) если все в порядке – в окне **Output** должно быть «0 failed» (см. рис. 8.7), то собственно запускаем на исполнение – клавишами **Ctrl-F5** (***Debug/Start Without Debugging***).
8) в результате откроется «черное окно», где вводим исходные данные и получаем результат – см. рис. 8.8.

2. ПЕРЕМЕННЫЕ

Ясно, что всякая программа, так или иначе, обрабатывает данные. Эти данные, как правило, размещаются в переменных. Каждая переменная имеет **имя, тип, размер и значение**. Основные типы с указанием размеров занимаемой памяти и областью значений приведены в табл. 1.

Таблица 1

Основные типы переменных в С++

тип	Тип (по-русски)	Размер памяти	Интервал допустимых значений
bool	логический	1 байт	**true** или **false**
char	символьный	1 байт	От –128 до 127 (код символа)
int	целый	4 байта	От –2 147 483 648 до 2 147 483 647
long	целый	4 байта	От –2 147 483 648 до 2 147 483 647
short	Короткий целый	2 байта	От –32 768 до 32 767
unsigned char	Беззнаковый символьный	1 байт	От 0 до 255
unsigned int	Беззнаковый целый	4 байта	От 0 до 4 294 967 295
unsigned short	Беззнаковый короткий целый	2 байта	От 0 до 65535
float	Вещественный плавающий	4 байта	От ≈3,4e – 38 до ≈3,4e+38
double	Вещественный двойной точности	8 байтов	От ≈1,7e – 308 до ≈1,7e + 308

Объявление типов переменных делается соответствующим служебным словом с последующим перечислением имен переменных: **int i, j, k, l, m** – перечисленные переменные будут целого типа и т.п. Причем, объявление типов можно совместить с присваиванием.

Присваивание значения переменным делается с помощью команды присваивания (в С++ используется знак равенства). Значения для символьных переменных заключаются в одинарные кавычки:

int i=33; float a=.156;

char f =’5’, f2=’$’, ff=’1’

Основные арифметические и логические операции, допустимые в С++, приведены в табл. 2.

Таблица 2

Основные арифметические и логические операции в С++

Арифметические операции	символы	Логические операции	символы
сложение	+	равно	= =
вычитание	–	Не равно	!=
умножение	*	больше	>
деление	/	Меньше	<
Вычисление остатка	%	Больше или равно	>=
присваивание	=	Меньше или равно	<=
отрицание	!	Логическое умножение	&&
		Логическое сложение	||

Присваивание значений переменных делается командой присваивания: *j* = 5; *i* = *j*. Кстати, в С++ можно такие операции совмещать: *i* = *j* = 5;

В С++ имеются помимо привычных и сокращенные формы записи арифметических операций. Примеры:

Операция присваивания

a+ = 3 эквивалентно a = a + 3

a– = 3 эквивалентно a = a – 3

a* = 3 эквивалентно a = a*3

a/ = 3 эквивалентно a = a/3

a% = 3 эквивалентно a = a%3

Операции инкремента

i++ постфиксная форма i=i+1;

++i префиксная форма i=i+1;

Операции декремента

i– – постфиксная форма i=i–1

– – i префиксная форма i=i–1

Операции инкремента и декремента применяются к целым числам. Различие между постфиксной и префиксной формами иллюстрируется приведенным ниже примером.

Например, пусть имеем переменные целого типа *t* и *c*. Если *c* = 5, тогда при записи *t* = ++*c* + 6 получим, что *t* равно 12.

Если же применить постфиксную форму (при том же *c* = 5): *t* = *c*++ + 6, то получим, что *t* = 11, потому что начальное значение *c* используется для вычисления выражения до того, как *c* увеличится на единицу операцией инкремента. Этот оператор эквивалентен следующим двум: *t* = *c*+ 6; ++*c*;

Эти же правила применимы и к операции декремента – – . Например, если *c* = 5, то *t* = – –*c* +6 даст значение 10 для *t*, в то время как *t* = 6 + *c*– – даст значение 11 для переменной *t*.

3. ЛИНЕЙНАЯ ПРОГРАММА

3.1. Оформление линейной программы

Программа на С++ состоит из одной или более функций. Причем, начинается выполнение программы с функции **main()**, которая, по сути, есть главный элемент программы. Каждая функция, вообще говоря, должна возвращать какой-либо результат. Поэтому и функция **main()** должна заканчивать командой возврата **return 0;,** показывающей, что программа завершена. Причем у функции (как и у переменных) указывается тип. Функция **main()** имеет тип **int**. Собственно алгоритм (исполняемая часть программы) заключается в фигурные скобки {} после выражения **int main()**. Каждая фраза алгоритма заканчивается точкой с запятой **;**.

Обычно перед функцией **main** дают так называемые директивы препроцессору. Такие директивы начинаются со знака # (произносится «хаш»). Например, директива

 # include <iostream>

подключает встроенные в С++ функции, которые обеспечивают ввод/вывод.

Для *вывода на экран* служит команда

cout << (читается «си-аут»). В ней для вывода текстовых сообщений помещают их в кавычки. Также там можно разместить управляющие последовательности для разделения строк вывода:

«**n**» – дает команду начать вывод с новой строки

‘ ‘ (пробел в одинарных кавычках) – разделяет пробелом выводимые знаки или строки. Иначе – весь вывод сливается в одну строку.

«**t**» – табуляция;

Для *ввода в программу* (в процессе ее выполнения) служит команда:

cin>> (читается «си-ин»).

В ней указывается переменная, которой и будет присвоено значение, введенное с экрана.

В программе можно размещать комментарии, которым предшествует // (двойная дробная черта).

Важной в начале исходного текста программы является директива объявления пространства имен:

using namespace std;

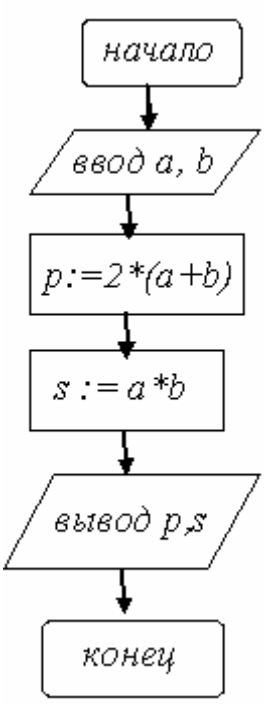
Эффект от ее применения состоит в том, что вы можете свободно применять, в частности, вышеупомянутые команды ввода-выво-

да **cout** и **cin**. И процессор при этом будет четко понимать, что это команды, а не переменные.

Изученных сведений достаточно для составления простой программы на С++. По аналогии со структурным языком назовем такую программу, исполняющуюся прямолинейно от начала и до конца – *линейной программой*. Рассмотрим простой пример.

Пример 3-1. Вычислить периметр и площадь прямоугольника, длина которого равна *a*, а ширина равна *b*.

a = 3 м, *b* = 4 м. Ответ: периметр = 14 м, площадь = 12 м^2



Приведем сначала блок-схему алгоритма для решения данной задачи.

```
//вычисление периметра и площади прямоугольника
#include <iostream>
using namespace std;
int main()
{ float a,b,p,s;
cout<<"Vvedi storony pryamoug"  <<"\n";
cin>>a>>b;
p=2*(a+b); s=a*b;
cout<<"Perimetr="<<p<<" Ploshad'="<<s;
cout<<"\n";
return 0;
}
```

Результат работы данного примера приведен на рис. 8.7 и 8.8.

Замечание. Одна из проблем при программировании в Visual C++ 2005 (при создании консольного приложения) в том, что в консольном «черном» окне не отображаются сообщения, написанные по-русски. Это связано с тем, что в Windows (где мы пишем текст программы) и в MS DOS (где отображается ее результат) используется разная кодировка русских букв. Чтобы справится с этой проблемой, нужно использовать специальную функцию `AnsiToOem`, которая преобразует текст из одной кодировки в другую, причем, надо подключить дополнительный заголовочный файл `windows.h`.

Как именно это делается, проиллюстрируем на том же примере 3.1 (нужные дополнения выделены жирным шрифтом).

```
//вычисление периметра и площади прямоугольника
#include <iostream>
#include <windows.h>
```

```
using namespace std;
int main()
{    char str[256];
float a,b,p,s;
AnsiToOem("Введи стороны прямоугольника ", str);
cout<< str <<"\n";
cin>>a>>b;
p=2*(a+b); s=a*b;
     AnsiToOem("Периметр=", str);
     cout<< str<<p;
     AnsiToOem(" Площадь=", str);
     cout<<str<<s; cout<<"\n";
return 0;
}
```

Как видим, сначала каждое русское сообщение преобразуется функцией `AnsiToOem` и помещается в строку `str` (она описана как `char str[256]`, т.е. может содержать до 256 символов). Затем уже эта строка `str` и выводится. Обратите внимание, что подключен заголовочный файл `windows.h`.

После запуска на исполнение получим результат, приведенный на рис. 8.9.

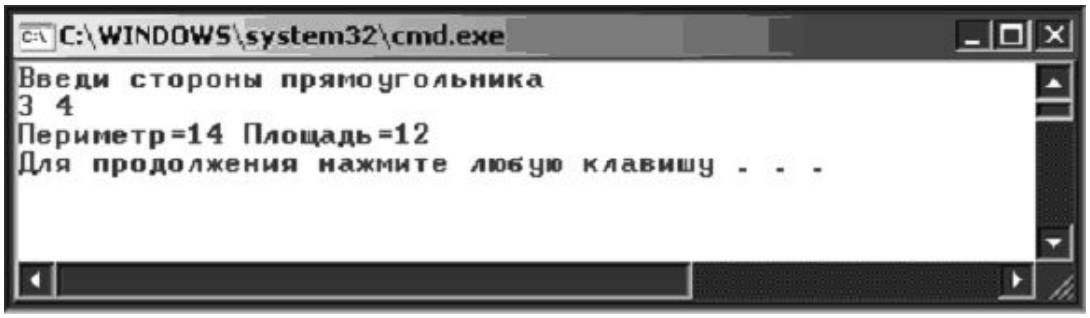


Рис. 8.9. Результат работы примера 3.1 (с выводом сообщений по-русски)

3.2. Программирование в стандартизованной среде CLR

Помимо среды Win32 в Visual C++ 2005 можно использовать среду CLR. Она является стандартизованной средой для выполнения программ, написанных на широком диапазоне высокоуровневых языков. Не вдаваясь в подробности, просто отметим, что таким образом можно создавать приложения более высокого уровня. Рассмотрим на примере.

Создадим новый проект командой ***File/New/Project***. Далее в возникшем окне диалога выбираем на левой панели **CLR** , а на пра-

вой – **CLR Console Application (**для создания консольного приложения), в строке **Name** введем имя проекта (при этом должна быть установлена «птичка» **Create directory for solution** создать каталог для проекта) и нажимаем ОК.

В очередном окне появляется список подпапок данного проекта. В правой части будет *заготовка текста исходного файла* (см. рис. 8.10) и здесь производим необходимые изменения и дополнения.

У функции **main()** появились в скобках аргументы. На них пока не обращаем внимание (но и не удаляем их!). Обратите внимание, что команда вывода здесь выглядит несколько необычно:

```
Console::WriteLine(L»Hello world»);
```

Здесь в кавычках размещается собственно выводимое сообщение. Ему обязательно предшествует буква `L`.

Доработаем эту заготовку исходного текста так, чтобы выполнить тот же пример 3.1. Пока не будем производить ввод исходных данных (ввод в среде CLR осуществляется несколько сложнее), а просто зададим исходные данные в тексте программы путем присваивания.

Для вывода чисел нужно указать *формат вывода* внутри сообщения, заключая этот формат в фигурные скобки. Причем, если выводятся несколько чисел в одной строке, то их нумеруют 0, 1 и т.д. (см. листинг ниже). Формат **F2** означает, что число вещественное (float) и будет представлено две цифры после запятой. Если потребуется выводить целое число, то указываем формат **D**. Формат **E** задают, когда вещественные числа нужно вывести в так называемой экспоненциальной форме: $6*10^{23}$ есть 6E23.

```
// prim1-1_CLR.cpp : main project file.
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{float a,b, s,p;
a=3; b=4;
Console::WriteLine (L"Заданы стороны a={0:F2} и b={1:F2}",a,b);
     p=2*(a+b); s=a*b;
     Console::WriteLine(L"Периметр={0:F2}",p);
     Console::WriteLine(L"Площадь= {0:F2}",s);
     return 0;
}
```

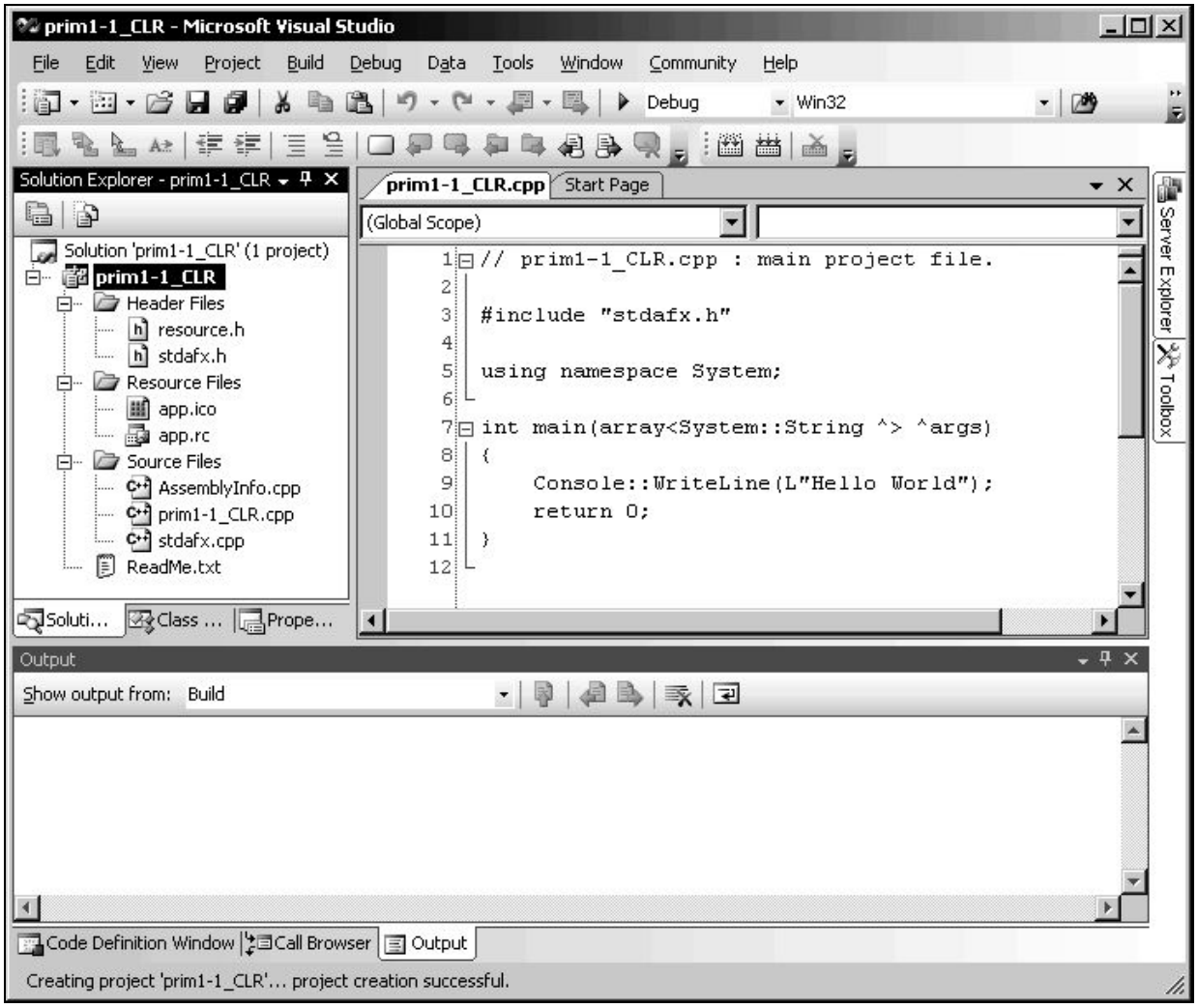


Рис. 8.10. Окно создания исходного файла в среде CLR

После компиляции и компоновки (клавишей **F7**) и запуска на исполнение (клавишами **Ctrl-F5**) получим результат, представленный на рис. 8.11.

Обратите внимание, что при этом нет никаких проблем с выводом русского текста.

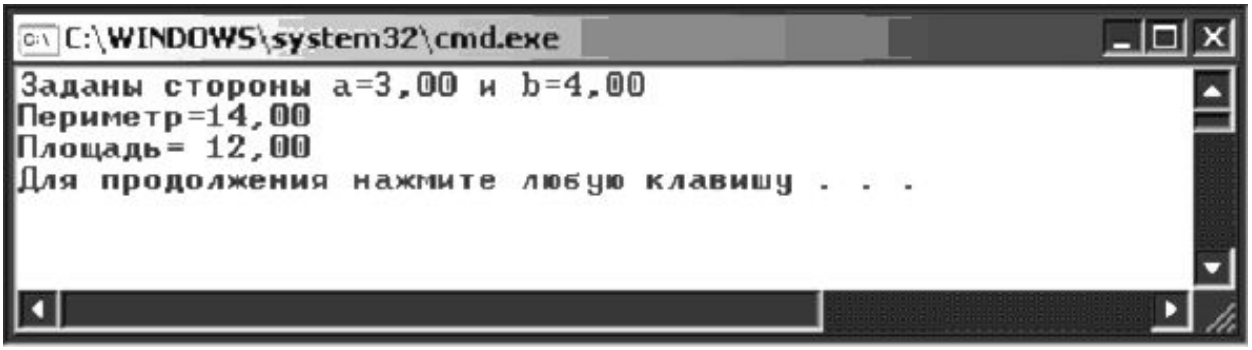


Рис. 8.11. Результат работы консольного приложения в CLR

Итак, **создание консольного приложения в CLR** делается так:

1) Создать проект: дать команду ***File/New/Project*** (или нажать комбинацию клавиш **Ctrl-Shift-N**)

2) В возникшем окне на левой панели выбрать **CLR** и на правой выберем **CLR Console Application (**для создания консольного приложения), в строке **Name** введем имя проекта (при этом должна быть установлена «птичка» **Create directory for solution** создать каталог для проекта) и нажать ОК

3) в очередном окне появляется список подпапок данного проекта. В правой части будет *заготовка текста исходного файла* (см. рис. 8.10) и здесь производим необходимые изменения и дополнения

4) в окне редактора вводим (дописываем) текст программы и сохраняем его командой ***File/Save all*** (или клавишами **Ctrl-Shift-S**)

5) для запуска программы на выполнение, нужно сначала провести компиляцию и компоновку – дать команду ***Build/Build Solution*** (или клавиша **F7**).

6) если все в порядке – в окне **Output** должно быть «0 failed», то собственно запускаем на исполнение – клавишами **Ctrl-F5** (***Debug/Start Without Debugging***).

7) в результате откроется «черное окно», где и получаем результат – см. рис. 8.11.

Практические задания

Составить программу для решения нижеприведенных задач:

а) в режиме Win32 Console Application

б) в режиме CLR Console Application, задавая данные непосредственно в тексте программы

3.1. Найти массу x литров молока, если известно, что плотность молока p кг/м^3.

пример: $x = 7$ л, $p = 1030$ кг/м^3 *Ответ:* 7,21 кг

3.2. Объем цилиндра равен V, а площадь основания – S. Какова высота цилиндра H?

пример: $V = 10$ *м*^{3} , $S = 5$ *м*^{2} *Ответ:* 2 м

3.3. Дана длина ребра куба a. Найти объем куба V и площадь его боковой поверхности S.

пример: $a = 5$ *Ответ:* $V = 125$, $S = 100$

3.4. Каков объем кислорода, содержащегося в комнате размером $a \cdot b \cdot c$, если кислород составляет 21% объема воздуха?

пример: $a = 3$, $b = 4$, $c = 5$ *Ответ:* 12,6

3.5. Найти площадь равнобочной трапеции с основаниями a и b и углом при большем основании равным x.

пример: $a = 6$, $b = 5$, $x = 45^o$ *Ответ:* 2,75

3.6. Найти угол между отрезком прямой, соединяющей начало координат с точкой $A(x,y)$, и осью OX (точка лежит в 1-й четверти).

пример: $x = 3$, $y = 4$ *Ответ:* 53,13 (градуса)

3.7. Определить время падения камня на поверхность земли с высоты h.

пример: $h = 10$ м *Ответ:* 1,4278 с

3.8. Три сопротивления R_1, R_2, R_3 соединены параллельно. Найти сопротивление соединения.

пример: $R_1 = 10$, $R_2 = 15$, $R_3 = 20$ *Ответ:* 4,62

3.9. Написать программу вычисления площади параллелограмма. Извне вводятся стороны a,b и угол между ними x.

пример: $a = 10$, $b = 15$, $x = 30°$ *Ответ:* 75

3.10. Написать программу вычисления объема прямоугольного параллелепипеда. Извне вводятся длина a, ширина b и высота c.

пример: $a = 10$, $b = 15$, $c = 20$ *Ответ:* 3000

3.11. Написать программу вычисления площади поверхности прямоугольного параллелепипеда. Извне вводятся длина a, ширина b и высота c.

пример: $a = 10$, $b = 15$, $c = 20$ *Ответ:* 1300

3.12. Написать программу вычисления объема цилиндра. Извне вводятся радиус основания R и высота цилиндра h.

пример: $R = 10$, $h = 15$ *Ответ:* 4712,39

3.13. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и карандашей. Извне вводятся цена одной тетради Ct и количество тетрадей Kt, а также цена карандаша Ck и количество карандашей Kk.

пример: $Ct = 1$, $Kt = 15$, $Ck = 0{,}2$; $Kk = 5$ *Ответ:* 16

3.14. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек к

ним. Извне вводятся цена одной тетради *Ct*, одной обложки *Cb* и количество тетрадей *Kt*.

пример: Ct = 1,2; *Kt* = 15, *Cb* = 0,2 *Ответ:* 21

3.15. Написать программу вычисления стоимости некоторого количества (по весу) яблок. Извне вводятся цена одного килограмма яблок *C* и вес яблок *V*.

пример: C = 25; *V* = 1,5 *Ответ:* 37.5

3.16. Написать программу вычисления сопротивления электрической цепи, состоящей из двух параллельно соединенных сопротивлений. Извне вводятся величина первого и второго сопротивления.

пример: $R_1 = 10$, $R_2 = 15$ *Ответ:* 6

3.17. Написать программу вычисления сопротивления электрической цепи, состоящей из двух последовательно соединенных сопротивлений. Извне вводятся величина первого и второго сопротивления.

пример: $R_1 = 10$, $R_2 = 15$ *Ответ:* 25

3.18. Написать программу вычисления силы тока в электрической цепи. Извне вводятся напряжение *U* и сопротивление *R*.

пример: U = 10, *R* = 15 *Ответ:* 0.6667

3.19. Составить программу, которая поменяет местами значения введенных переменных *x* и *y:*

а) используя дополнительную переменную,

б)* не используя дополнительной переменной.

3.20. Составить программу, которая поменяет местами значения введенных переменных *x*, *y*, *z* так, чтобы в переменной *x* оказалось значение переменной *y*, в *y* – значение переменной *z*, а в *z* – прежнее значение переменной *x*:

а) используя дополнительную переменную,

б)* не используя дополнительной переменной.

3.21. Составить программу перевода километров в мили, версты, сажени (см. табл. 3)

3.22. Составить программу перевода метров в версты, сажени, футы, аршины (см. табл. 3)

3.23. Составить программу перевода сантиметров в вершки, футы, дюймы (см. табл. 3)

3.24. Составить программу перевода килограммов в пуды, фунты, лоты, золотники (см. табл. 3)

3.25. Составить программу перевода литров в бочки, ведра, штофы, бутылки, чарки (см. табл. 3)

Таблица 3

Русская система мер

Меры массы (веса)	
1 пуд = 40 фунтов	0,016 т = 0,164 ц = 16,3805 кг
1 фунт = 32 лота = 96 золотников	0,410 кг = 409,512 г
1 лот = 3 золотника	12,7973 г
1 золотник	4,26575 г
Меры емкости (для жидкости)	
1 бочка = 40 ведер	4, 920 гл (гекталитров)
1 ведро = 10 штофов = 20 бутылок	1,2299 дкл = 12,2994 л
1 штоф = 10 чарок	1,230 л
1 чарка	0,123 л
1 бутылка	0,615 л
Меры длины	
1 миля = 7 верст	7,4676 км
1 верста = 500 саженей	1,0668 км
1 сажень = 3 аршина = 7 футов	2,1336 м
1 аршин = 16 вершков	0,711 м = 71,120 см
1 вершок	4,445 см = 44,45 мм
1 фут = 12 дюймов	0,305 м = 30,48 см
1 дюйм	2,540 см = 25,4 мм

4. ПРОГРАММА С ВЕТВЛЕНИЕМ

Опять-таки по аналогии со структурными языками будем называть программу, использующую структуру выбора, *программой с ветвлением*.

Структура выбора на C++ имеет вид:

if (***условие*** **)** ***оператор 1;*** **else** ***оператор 2***

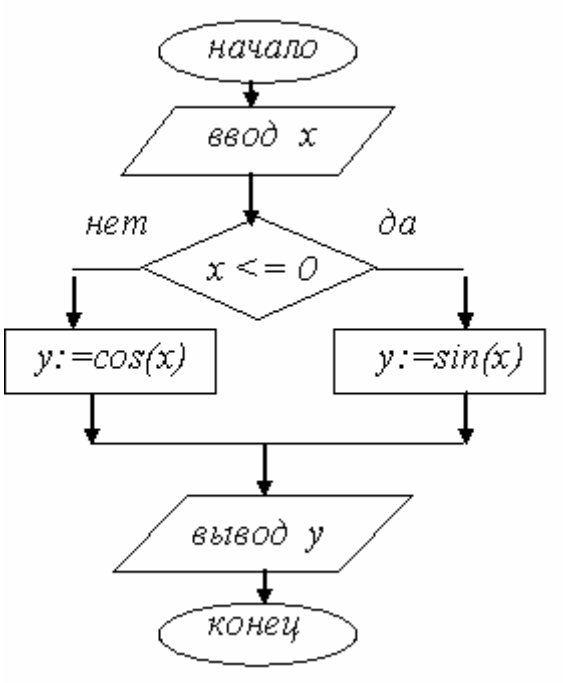
Обратим внимание, что в отличие, например, от Паскаля в таком операторе отсутствует служебное слово **then** и условие обязательно заключается в скобки. ***Оператор 1*** исполняется в случае истинности ***условия***. ***Оператор 2*** – в случае ложности ***условия***.

В C++ для использования математических функций необходимо подключать специальную библиотеку `math.h`.

Пример 4.1. Составить программу для вычисления по заданному x значения функции:

$$y = \begin{cases} \sin x, & x <= 0, \\ \cos x, & x > 0. \end{cases}$$

Приведем сначала блок-схему алгоритма для этой задачи.




```
#include <iostream>
#include <windows.h>
#include <math.h>
using namespace std;
int main()
{    double x,y;
     char str[256];
     AnsiToOem(" Введи значение x ",str);
     cout<<str; cin>>x;
     if (x<=0) y=sin(x); else y=cos(x);
     AnsiToOem("\n Функция y=",str);
     cout<<str<<y<<"\n";
return 0;}
```

Например, при $x = 0$ получим 0 ($\sin 0 = 0$), а при $x = 3{,}14159265$ (π радиан) будет получено -1 (т.к. $\cos\pi = -1$).

В тех случаях, когда требуется выполнить несколько действий (в случае истинности или в случае ложности условия), то применяют фигурные скобки:

if (*условие)* {*оператор 1_1; оператор 1_2}* else {*оператор 2_1; оператор 2_2}*

Пример 4.2. Составить программу, которая перераспределит заданные значения *x*, *y* так, что в *x* окажется большее значение, а в *y* меньшее.

```
//перестановка x,y; большее - вперед
#include <iostream>
#include <windows.h>
#include <math.h>
using namespace std;
int main()
{    double x,y,z;  char str[256];
     AnsiToOem(" Введи значение x y ",str);
     cout<<str;     cin>>x>>y;
     if (x<y) {z=x; x=y; y=z;}
     AnsiToOem("\n Итог x=",str); cout<<str<<x;
     AnsiToOem("\n Итог y=",str); cout<<str<<y<<"\n";
return 0;}
```

В результате, например, получим (рис. 8.12):

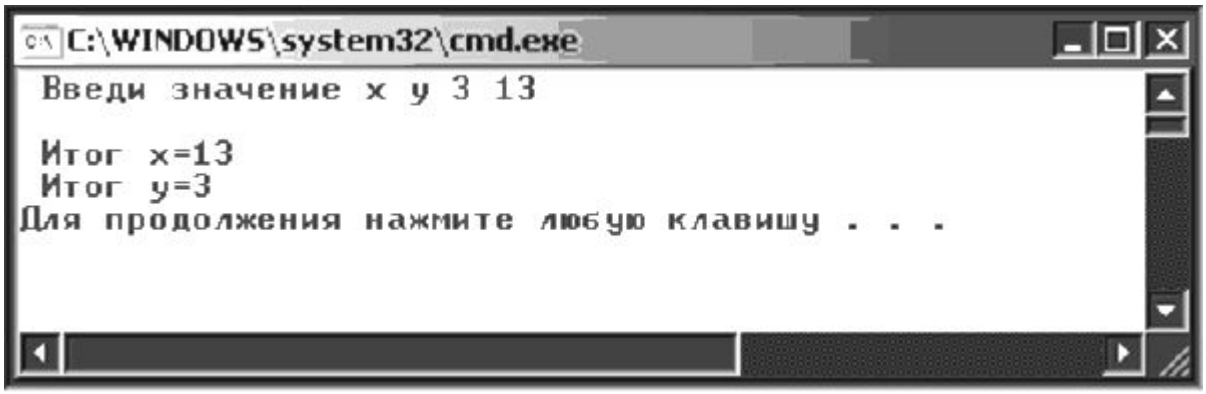


Рис. 8.12. Результат работы примера 4.2

Обратите внимание, что здесь использована сокращенная форма записи условного оператора: **if (*условие) оператор*.**

Условие может быть и весьма сложным. Здесь допускается *конъюнкция* условий (одновременное выполнение условий) – условия связываются при помощи &&, *дизъюнкция* условий (альтернативное выполнение условий) – обозначается | | и *инверсия* условий (обозначается знаком !).

Пример 4.3. Выяснить, принадлежит ли точка с координатами (x, y) кольцу с внешним радиусом r_1 и внутренним r_2.

```
// принадлежит ли точка кольцу от r2 до r1
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{ double x,y,r1,r2;  char str[256],str1[256];
 AnsiToOem("Введи x,y,r1,r2 ",str);
```

```
 cout<<str; cin>>x>>y>>r1>>r2;
 AnsiToOem("\n Точка принадлежит кольцу \n", str);
 AnsiToOem("\n Точка НЕ принадлежит кольцу \n", str1);
 if ((x*x+y*y)<r1 && (x*x +y*y)>r2) cout<<str;
                                        else cout<<str1;
return 0;}
```

Если зададим: $x = 1$, $y = 1$, $r_1 = 3$, $r_2 = 1$, получим «принадлежит», а если $x = 2$, $y = 2$, $r_1 = 2$, $r_2 = 1$, получим «не принадлежит».

Практические задания

Составить программу для решения нижеприведенных задач

а) в режиме Win32 Console Application

б) в режиме CLR Console Application, задавая данные непосредственно в тексте программы

4.1. Даны числа a, b, c . Проверить, выполняется ли неравенство $a<b<c$. Вывести об этом сообщение.

4.2. Даны три действительных числа a, b, c. Выбрать из них те, которые принадлежат интервалу (1, 3).

4.3. Даны числа x,y ($x \neq y$). Меньшее из них заменить полусуммой, а большее – их удвоенным произведением.

4.4. Найти наибольшее для трех заданных чисел a, b, c.

4.5. Выяснить, существует ли треугольник с длинами сторон x, y, z (в треугольнике большая сторона меньше суммы двух других сторон).

4.6. На окружности с центром в точке (x_0, y_0) задана дуга с координатами начальной (x_n, y_n) и конечной (x_k, y_k) точек. Определить номера четвертей окружности, в которых находятся начальная и конечная точки.

4.7. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 8.13, *а*.

4.8. Даны координаты точки (x,y). Выяснить, принадлежит ли эта точка области, указанной на рис. 8.13, *б*.

4.9. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 8.13, *в*.

4.10. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 8.13, *г*.

4.11. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 8.13, *д*.

4.12. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 8.13, *е*.

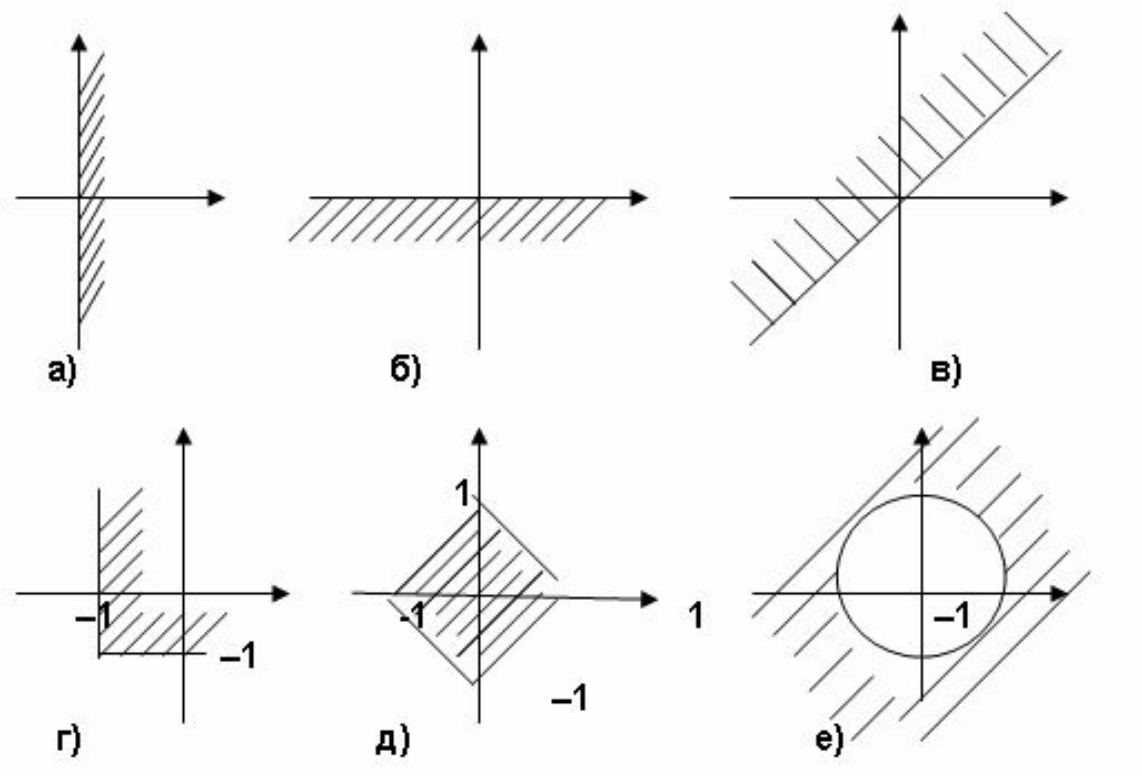


Рис. 8.13

4.13. Даны действительные положительные числа a, b, c, x, y. Выяснить, пройдет ли кирпич с ребрами a, b, c в прямоугольное отверстие со сторонами x и y. Просовывать кирпич разрешается только так, чтобы каждое из его ребер было параллельно или перпендикулярно каждой из сторон отверстия.

4.14. Даны действительные числа x_1, x_2, x_3, y_1, y_2, y_3. Выяснить, является ли треугольник с вершинами (x_1, y_1), (x_2, y_2), (x_3, y_3) прямоугольным?

4.15. Для пятиугольника, заданного координатами своих вершин, найти наибольшую и наименьшую стороны**.**

4.16. Написать программу вычисления площади кольца. Извне вводятся радиус кольца и радиус отверстия. В программе предусмотреть проверку правильности вводимых данных (радиусы положительны, причем радиус кольца больше радиуса отверстия).

4.17. Написать программу, которая переводит время из минут и секунд в секунды. Извне вводятся минуты и секунды. В программе предусмотреть проверку на правильность введенных данных (только положительные, кроме того, число минут <=60 и число секунд <=60)

4.18. Написать программу вычисления сопротивления электрической цепи, состоящей из двух сопротивлений. Извне вводятся величина первого, второго сопротивления и указывается тип соедине-

ния (например, 1 – последовательное, 2 – параллельное соединение).

4.19. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 10% предоставляется, если сумма покупки более 1000 руб. Извне вводится сумма покупки.

4.20. Написать программу вычисления стоимости разговора по телефону с учетом 20% скидки, предоставляемой по субботам и воскресеньям. Извне вводится длительность разговора (в целых минутах) и день недели цифрой (1 – понедельник, … 7 – воскресенье).

4.21. Написать программу, которая вычисляет оптимальный вес для пользователя, сравнивает его с реальным и выдает рекомендацию о необходимости поправится или похудеть. Оптимальный вес вычисляется по формуле: рост (см) – 100. Извне вводятся рост (в см) и вес (в кг).

4.22. Написать программу, которая запрашивает у пользователя номер месяца и затем выводит соответствующее название времени года. Если вводится недопустимое число (<1 или >12), должно появиться сообщение «ошибка ввода».

4.23. Написать программу, которая запрашивает у пользователя номер дня недели и выводит одно из сообщений: «рабочий день», «суббота» или «воскресенье».

4.24. Написать программу, которая запрашивает у пользователя номер дня недели и выводит название дня недели.

4.25. Написать программу, которая переводит время из часов и минут в минуты. Извне вводятся часы (целое положительное) и минуты (целое положительное и <=59). Программа должна проверять правильность введенных данных.

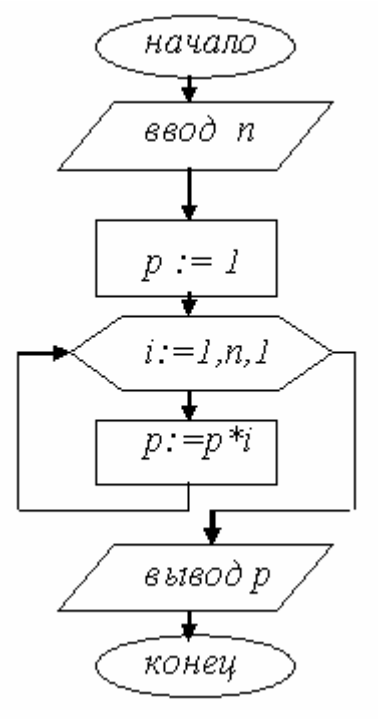
5. ЦИКЛ С ПАРАМЕТРОМ

На языке С++ циклы с параметром организуются с помощью структуры повторения

For (*парам*=нач.знач; *парам*<кон.знач; *парам*++) *тело цикла*

Отметим, что параметр может изменяться и не на единицу, а на другое (целое) число. Тогда вместо ***парам*++** следует писать, например, ***парам*+=2** (если параметр приращается на 2).

Пример 5.1. *Рассчитать для заданного N величину N!*




```
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{  unsigned long i,n, p;     char str[256], str1[256];
     AnsiToOem(" введи значение N\n",str);
     cout<<str;   cin>>n; p=1;
  for (i=1; i<n+1; i++)      p=p*i;
     AnsiToOem("факториал числа ",str);
     AnsiToOem(" равен ", str1);
   cout<<str<<n<<str1<<p<<"\n";
return 0;}
```

Заметим, что на этой задаче мы сталкиваемся с проблемой ограниченного представления целых чисел: мы сможем вычислить максимум 12! = 479 001 600 . Следующий 13! = 6 227 020 800, т.е. это число уже не помещается в интервал представления для переменных типа `unsigned long int`. Поэтому, чтобы вычислять факториал для достаточно больших чисел, следует описание `p` сменить на `double`, правда, при этом мы будем получать округленное значение факториала (тип `double` дает не более 15 достоверных значащих цифр).

Заметим, что при программировании в среде CLR существует дополнительный целочисленный тип `long long` (см. табл. 4).

В этом случае (при использовании `long long`) удастся точно вычислить до 20! =2 432 902 008 176 640 000.

Таблица 4

Дополнительные целые типы в Visual C++

тип	Тип (по-русски)	Размер памяти	Интервал допустимых значений
long long	целый	8 байт	от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
unsigned long long	целый	8 байт	От 0 до 18 446 744 073 709 551 615

Разумеется, внутри цикла можно организовывать и вложенные циклы, причем число вложений, в принципе, не ограничено.

Пример 5.2. *Получить таблицу умножения в виде таблицы Пифагора:*

```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81
```

```
//таблица умножения
#include <iostream>
using namespace std;
int main()
{int i,j,k ;
for (i=1;i<=9;i++)
  {cout<<"\n";
  for (j=1; j<=9;j++)
     {k=i*j;
     if (k<10)cout<<"  " ; else cout<<" ";
     cout<<k; }; } cout<<"\n";
return 0;}
```

Практические задания

В трех последующих задачах условие одинаковое: для заданного натурального числа n рассчитать сумму:

5.1. $\cos x + \cos x^2 + \cos x^3 + \ldots + \cos x^n$

5.2. $\sin 1 + \sin(1+0{,}1) + \sin(1+0{,}1^2) + \ldots + \sin(1+0{,}1^n)$

5.3. $1! + 2! + 3! + \ldots + n!$

В трех следующих заданиях условие одинаковое: для заданного натурального числа n рассчитать сумму и сравнить со значением y:

5.4. $1 + 2 + 3 + ... + n$ $\quad y = n(n+1)/2$

5.5. $1 + 3 + 5 + ... + (2n - 1)$ $\quad y = n^2$

5.6. $2 + 4 + 6 + ... + 2n$ $\quad y = n(n+1)$

5.7. Для заданного натурального числа n рассчитать величину $n!!$

$$n!! = \begin{cases} 1 \cdot 3 \cdot 5 \cdot ... \cdot n, & \text{если } n - \text{нечетное} \\ 2 \cdot 4 \cdot 6 \cdot ... \cdot n, & \text{если } n - \text{четное} \end{cases}$$

5.8. Дано 10 вещественных чисел: a_1, a_2, ..., a_{10}. Требуется определить, сколько из них принимает значение, большее заданного числа b.

5.9. Дано 10 вещественных чисел: a_1, a_2, ..., a_{10}. Требуется найти порядковые номера тех из них, которые отрицательны.

5.10. Вычислить K – количество точек с целочисленными координатами, попадающих в круг радиуса R с центром в начале координат.

5.11. Написать программу, которая выводит таблицу квадратов первых десяти целых положительных чисел.

5.12. Написать программу, которая выводит таблицу квадратов первых пяти целых положительных нечетных чисел.

5.13. Написать программу, которая выводит таблицу квадратов чисел от 11 до 19.

5.14. Написать программу, которая вычисляет сумму первых N целых положительных чисел.

5.15. Написать программу, которая вычисляет сумму первых N целых положительных четных чисел.

5.16. Написать программу, которая вычисляет сумму первых N целых положительных нечетных чисел.

5.17. Написать программу, которая выводит таблицу степеней двойки от нулевой до десятой степени.

5.18. Написать программу, которая выводит таблицу значений функции $y = -2{,}4x^2 + 5x - 3$ в диапазоне от –2 до 2 с шагом 0,5.

5.19. Написать программу, которая приглашает ввести последовательно 5 дробных чисел и вычисляет их среднее арифметическое.

5.20. Написать программу, которая приглашает ввести последовательно N дробных чисел и вычисляет их среднее арифметическое. Значение N вводится извне.

В пяти следующих заданиях условие одинаковое: для заданного натурального числа n рассчитать сумму и сравнить со значением y :

5.21. $1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots+\frac{(-1)^{n-1}}{2n-1}+\ldots$ $y = \pi/4$

5.22. $\frac{1}{3*5}+\frac{1}{7*9}+\frac{1}{11*13}+\ldots+\frac{1}{(4n-1)(4n+1)}+\ldots$ $y = 0{,}5 - \pi/8$

5.23. $1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots+\frac{1}{n^2}+\ldots$ $y = \pi^2/6$

5.24. $1+\cfrac{1}{2+\cfrac{3^2}{2+\cfrac{5^2}{2+\cfrac{7^2}{2+\ldots}}}}$ $y = 4/\pi$

5.25. $\frac{2*2*4*4*6*6*8*8*\ldots*2n*2n}{1*3*3*5*5*7*7*9*\ldots*(2n-1)}$ $y = \pi/2$

6. ЦИКЛ «ПОКА»

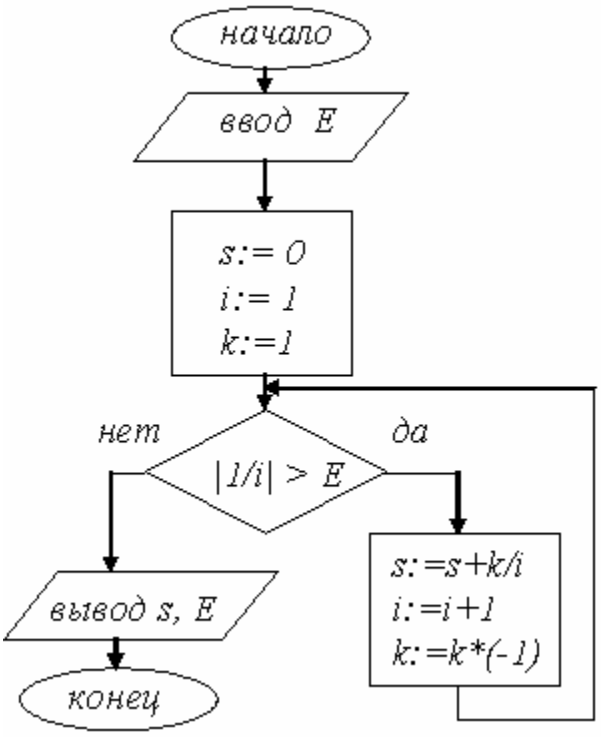
В С++ имеется также структура повторения **While**, которая служит для организации цикла «ПОКА»:

While (*условие*) *тело цикла*

Работает он так: ***тело цикла*** повторяется, пока заданное ***условие*** остается истинным.

Пример 6.1. Рассчитать сумму вида: $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots$ с заданной точностью E.

Иначе говоря, проводить суммирование, пока очередное слагаемое является большим E (по абсолютной величине).



```
//сумма бесконечного ряда
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{int i,k; double s,e;
char str[256], str1[256];
AnsiToOem("введи точность \n", str);
cout<<str; cin>>e;
s=0; i=1; k=1;
AnsiToOem(" сумма = ", str); AnsiToOem(" с точностью ",str1);
while(1.0/i>e){s=s+(double)k/i; i++; k=k*(-1);};
cout<<str<<s<<str1<<e<<"\n";
return 0;}
```

В данной программе использован известный нам алгоритм суммирования. Суть его в том, что вводится переменная (здесь это переменная *s*), задается ей нулевое значение, а затем в нее накапливается сумма. Очередное слагаемое – обратное к очередному числу натурального ряда. Однако четные (по номеру) слагаемые идут со знаком «минус», нечетные – со знаком «плюс». Для учета этого знака введена дополнительная переменная *k*, которая последовательно принимает значение то +1, то –1. Переменная *i* последовательно дает очередное натуральное число (2, 3, 4 и т.д.). Для удобства она описана как целая (`int`).

Расчет по этой программе при $E = 0{,}1$ дает значение 0,6456349, а точное значение для такой суммы (оказывается у этой бесконечной суммы результат – конечное число!) есть $ln(2) = 0{,}6931...$

Обратите внимание, что нам здесь потребовалось произвести в программе преобразование типов (**целый** к типу **double** при выполнении суммы), иначе в результате деления целого на целое получался нуль! И в условии, задающем цикл, мы сделали явное преобразование целого к вещественному – записав вместо целой единицы вещественную единицу (1.0).

Для изменения порядка работы в цикле в языке C++ применяются два оператора: **break** (для досрочного прерывания цикла) и **continue** – для пропуска следующих за ним операций с переходом в конец цикла, но из цикла выхода не производится.

Пример 6.2. *Среди К первых членов последовательности вида:* 1, 1+1/2, 1+1/2+1/3, ... *найти первый, больший заданного числа А. Если же его нет* (*среди первых К членов*), *то напечатать об этом сообщение.*

```
//поиск члена последовательности, меньшего
//заданного числа a
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{int i,k; double a,s;
char str[256],str1[256],str2[256];
   AnsiToOem("введи количество \n",str);
cout<<str; cin>>k;
   AnsiToOem("введи число a \n",str);
cout<<str; cin>>a;
s=0; i=1;
while (i<=k) { s=s+1.0/i; if (s>a) break ; i++;};
   AnsiToOem(" такого числа нет \n", str);
   AnsiToOem(" при i =", str1);
   AnsiToOem(" большее a число ",str2);
if (i>k) cout<<str;
else cout<<str1<<i<<str2<<s<<"\n";
return 0;}
```

Практические задания

6.1. Вычислить для заданного x сумму вида: $x - x^2/2 + x^3/3 - - x^4/4 + \ldots$ с заданной точностью E (когда очередное слагаемое по модулю станет меньше E, то суммирование прекратить). Результат сравнить с величиной $y = \ln(x + 1)$, $x > 0$.

6.2. Для заданного X в последовательности вида: $\sin X$, $\sin(\sin X)$), $\sin(\sin(\sin X))$, ... найти первое число, меньшее по модулю 0,01.

6.3. Найти наименьший номер n, для которого выполняется условие $| a_n - a_{n-1}| < 0{,}1$, если последовательность a_n имеет вид: $a_{n+1} = a_n + 2/a_n \quad a_1 = 1$.

6.4. С заданной точностью E рассчитать: $1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\ldots$ Результат сравнить со значением $e = 2{,}718281828\ldots$ Напомним, что запись вида $n!$ означает $1\cdot2\cdot3\cdot\ldots\cdot n$ (последовательное произведение целых чисел от 1 до n). *Указание*. Для вычисления значения $n!$ применить рекуррентную формулу: $n! = (n - 1)! \cdot n$.

6.5. Для заданного x с заданной точностью E рассчитать сумму вида: $x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots$ Результат сравнить со значением $y = \sin x$.

6.6 Для заданного x с заданной точностью E рассчитать сумму вида: $1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+...$ Результат сравнить со значением $y = \cos x$.

В приведенных ниже заданиях рассмотреть указанную последовательность в цикле и выйти из цикла, достигнув указанного условия с выдачей порядкового номера члена, при котором достигнуто условие. Если же за m оборотов цикла условие не достигнуто, напечатать об этом сообщение.

6.7. $a_{n+1} = tg a_n / 2, \quad a_1 = 0{,}1 \quad |a_n| < 10^{-3}$

6.8. $a_n = 2^n / n!, \quad |a_n| < 10^{-8}$

6.9. $a_n = 1/\sqrt[n]{n!}, \quad |a_n| < 0{,}1$

6.10. $a_{n+1} = (a_n + 1/a_n)/2, \quad a_1 = 10, \quad |1 - a_n| < 10^{-4}$

6.11. $a_{n+1} = 1{,}99 a_n - a_{n-1}, \quad a_1 = 1, a_2 = 1, \quad a_n < 0{,}9$

6.12. $a_{n+1} = (a_n + a_{n-1})/2, \quad a_1 = 1, a_2 = 10, \quad |a_n - 7| < 10^{-4}$

6.13. $a_{n+1} = \sqrt{2 + a_n}, \quad a_1 = 0, \quad |a_n - a_{n-1}| < 10^{-3}$

6.14. Написать программу, вычисляющую сумму и среднее арифметическое последовательности положительных чисел, которые вводятся с клавиатуры (длина последовательности не ограничена). Для завершения ввода нужно нажать 0.

6.15. Написать программу, которая определяет максимальное число из введенной последовательности положительных чисел (длина последовательности не ограничена). Для завершения ввода нужно нажать 0.

6.16. Написать программу, которая определяет минимальное число из введенной последовательности положительных чисел (длина последовательности не ограничена). Для завершения ввода нужно нажать 0.

6.17. Написать программу, которая задумывает число в диапазоне от 1 до 10 и предлагает пользователю угадать число за 5 попыток. В завершение должно быть сообщение о том, что пользователь угадал число или нет.

6.18. Написать программу, которая выводит на экран таблицу значений функции $y = 2x^2 - 5x - 8$ в диапазоне от – 4 до 4. Шаг изменения аргумента 0,5

6.19. Написать программу, которая вычисляет наибольший общий делитель двух целых чисел.

6.20. Написать программу для вычисления сопротивления цепи из нескольких проводников, соединенных последовательно (значения сопротивлений вводятся одно за другим и для завершения ввода нажимается 0).

6.21. Написать программу для вычисления сопротивления цепи из нескольких проводников, соединенных параллельно (значения сопротивлений вводятся одно за другим и для завершения ввода нажимается 0).

6.22. Написать программу, которая выводит на экран таблицу значений функции $y = 1/x$ в диапазоне от заданного A (вводится извне) и до 0. Шаг изменения аргумента h также вводится извне.

6.23. Написать программу, которая вычисляет число π с заданной пользователем точностью. Для этого суммируют последовательность: 1 – 1/3 + 1/5 – 1/7 + 1/9 – … до тех пор, пока очередное слагаемое (по абсолютной величине) не станет меньше заданной точности. Полученная сумма дает примерное значение числа $\pi/4$, т.е умножая полученную сумму на 4 и получим π.

6.24 Написать программу вычисления значения квадратного корня из числа $a>0$ с заданной точностью E на основе соотношения

$$x_{n+1} = \frac{x_n + a/x_n}{2} \qquad n = 1,2,3,...$$

где x_n – предыдущее, x_{n+1} – последующее приближение к корню. Начальное приближение для определенности положить равным $a/2$. Точность вычислений считается достигнутой, когда $|x_{n+1} - x_n| < E$

6.25 Спортсмен в первый день пробежал 10 км. Каждый следующий день он увеличивал дневную норму на 10 % от нормы предыдущего дня. Определить через сколько дней спортсмен:

а) будет пробегать более 20 км;

б) пробежит суммарный путь более 100 км.

7. ОДНОМЕРНЫЕ МАССИВЫ

7.1. Понятие об одномерном массиве

В программах на C++ можно использовать массивы.

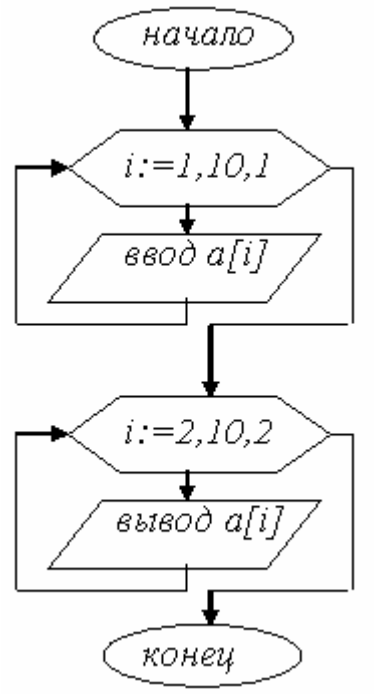
Напомним, что массив – это *упорядоченный набор величин, обозначаемых одним именем. Данные, являющиеся элементами массива, располагаются в памяти компьютера в определенном порядке, который задается индексами* (*порядковыми номерами элементов массива*).

В C++ массив, как и любая переменная, должен быть объявлен. Делается это с помощью служебного слова, указывающего тип, затем указывается имя массива и в квадратных скобках его длина. Заметим, что индексы массива ведут счет с нуля, поэтому запись вида:

```
double b[14]
```

означает, что резервируется память для 14 чисел типа `double` с именем `b` и порядковыми номерами от 0 до 13. Отдельный элемент масссива записывается с указанием имени и индекса в квадратных скобках.

Пример 7.1. *Ввести одномерный массив из* 10 *целых чисел. Вывести четные по порядковому номеру элементы этого массива.*




```
//ввод и вывод одномерного массива
#include <iostream>
#include <windows.h>
```

```
using namespace std;
int main()
{int i,a[11]; char str[256];
AnsiToOem("введи элемент номер ",str);
for (i=1;i<=10;i++)
{cout<<str<<i<<" ";cin>>a[i];};
AnsiToOem(" \n число a [",str);
for (i=2;i<=10;i+=2)
{cout<<str<<i<<"]= "<< a[i];}
cout<<"\n";
return 0;}
```

При запуске программы получим такой результат (рис. 8.14).

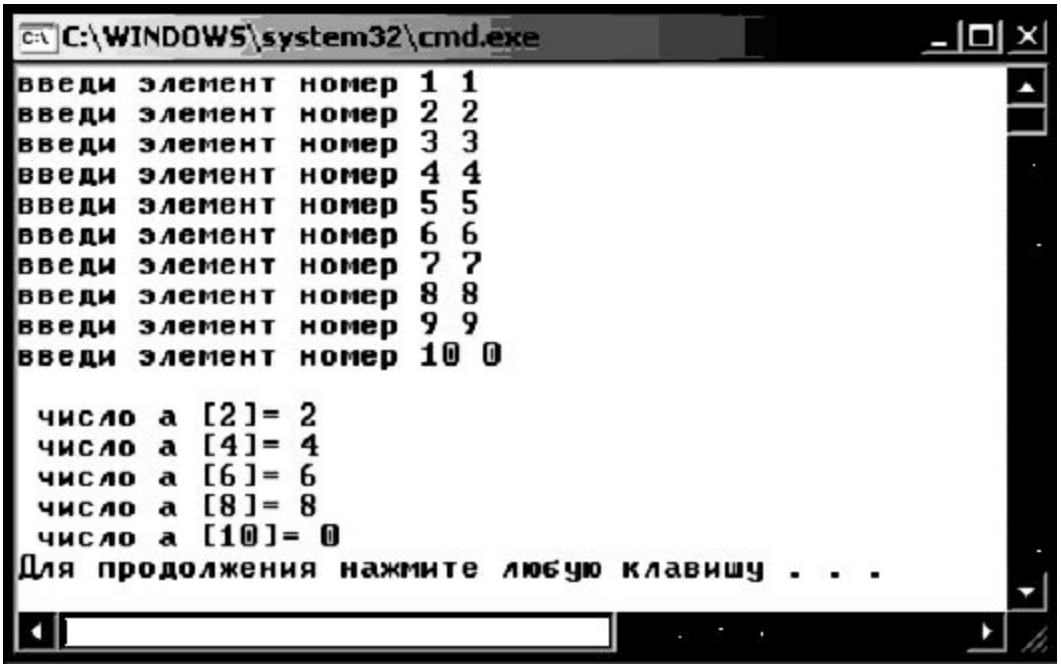


Рис. 8.14. Результат работы примера 7.1

Замечание. При работе с массивами необходимо внимательно следить за тем, чтобы не выходить за их объявленные границы. Компилятор C++ (в отличие, например, от Паскаля) не предупреждает об этой ошибке! Попытка ввести больше элементов, чем описано, приведет к неверным результатам, а попытка вывести – выведет случайный результат, находящийся в памяти.

Попробуйте в предыдущем примере описать массив **a[10]** (напомним, что при этом максимальный элемент **a[9]**, т.к. счет элементам идет с нуля), и вы не получите последнего, десятого, числа (точнее получите какое-то случайное число при выводе).

Еще раз обратим внимание, что работа с целыми переменными (и массивами) в С++ требует осторожности. Как отмечалось в примере 6.2, деление целого на целое дает в результате целое! Если требуется получать «правильный», вещественный, результат следует использовать преобразование – дописать перед нужной операцией (**float**) или (**double**). Если же требуется получать остаток от деления целого на целое, то применяют операцию **%.** Проиллюстрируем использование операции деления и вычисления остатка на простом примере.

Пример 7.2. Ввести четное и нечетное число и вычислить результаты деления их на 2 и остатки от такого деления.

```
#include<iostream>
#include <windows.h>
using namespace std;
int main()
{int n,k;  char str[256];
AnsiToOem("введи четное и нечетное",str);
     cout <<str; cin>>n>>k;
AnsiToOem(" отстаток от деления четного на 2=",str);
   cout <<str<< n%2;
AnsiToOem("\n результат деления четного на 2= ",str);
   cout<<str<<  n/2;
AnsiToOem("\n отстаток от деления нечетного на 2=",str);
   cout<<str<< k%2;
AnsiToOem("\n результат деления нечетного на 2=",str);
   cout <<str<< (float)k/2<<"\n";
   return 0;}
```

В результате получим (рис. 8.15)

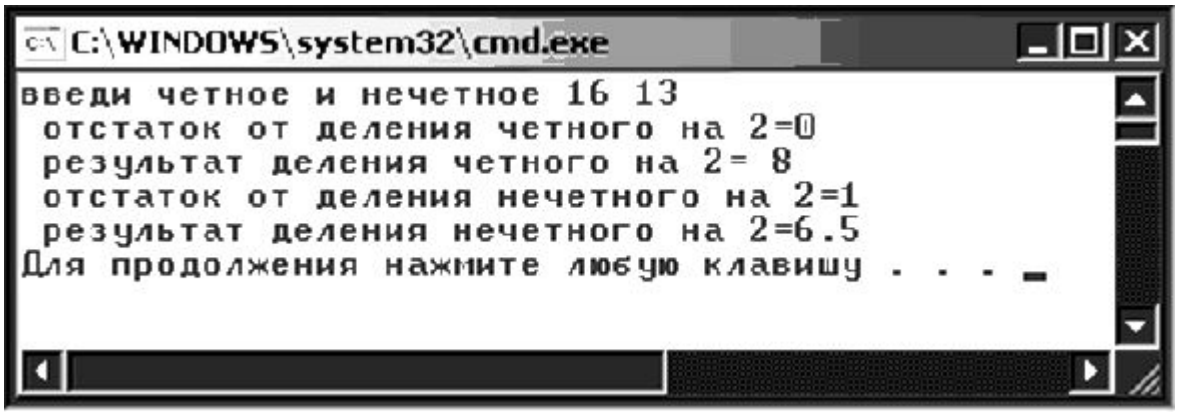


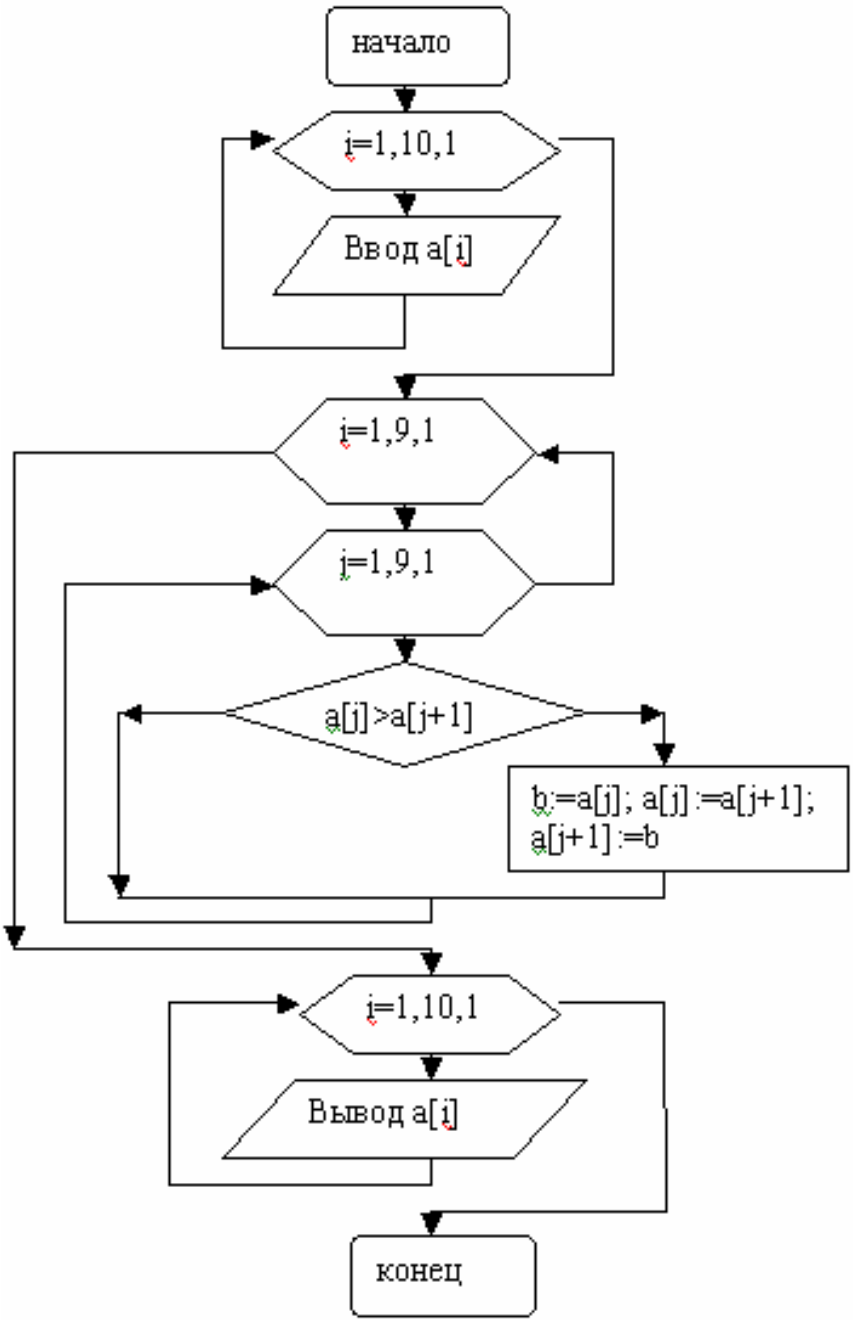
Рис. 8.15. Результат работы примера 7.2

7.2. Сортировка в одномерном массиве

Очевидно, что элементы массива, как правило, располагаются в произвольном порядке. Но во многих случаях может понадобиться расположить их, например, в порядке возрастания. Такая процедура упорядочения называется *сортировкой*.

Рассмотрим простейший способ сортировки – метод пузырька. Суть его в том, что последовательно сравниваются элементы массива и, если очередной элемент меньше предыдущего, то их меняют местами.

Пример 7.3. *Ввести одномерный массив и упорядочить его методом пузырька.*



Для ясности приведем блок-схему алгоритма такой сортировки (рис. 8.16).

Рис. 8.16. Блок-схема алгоритма сортировки массива методом пузырька

Соответственно текст программы имеет вид:

```
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{  int i,j, a[11],b;    char str[256];
     AnsiToOem("введи  a[",str);
     for (i=1; i<=10; i++)
     {    cout<<str<<i<<"] "; cin>>a[i];}
   for (i=1; i<=9; i++)
   for (j=1; j<=9; j++)
   { if (a[j]>a[j+1]){b=a[j];a[j]=a[j+1]; a[j+1]=b;}}
   for (i=1; i<=10; i++)
   { cout<<"\n a["<<i<<"] "<<a[i];}   cout<<"\n";
   return(0);}
```

Результат работы такой программы (см. рис. 8.17):

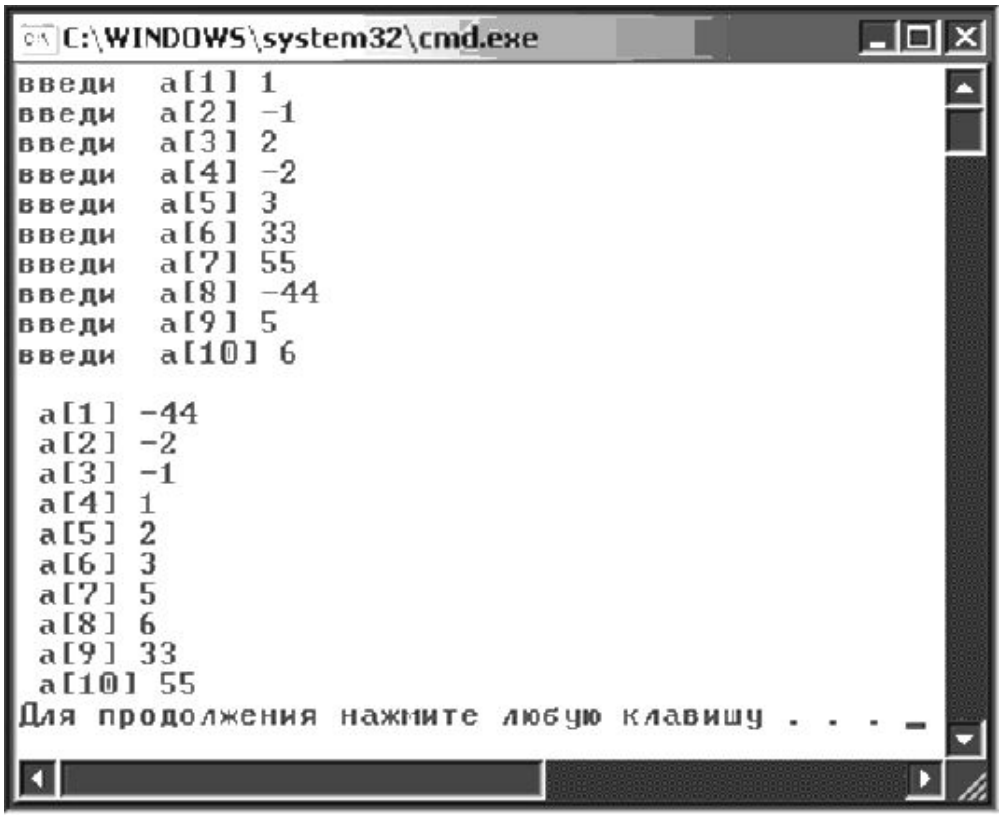


Рис. 8.17. Результат работы программы сортировки массива методом пузырька

Практические задания

7.1. Дан массив из *N* элементов. Отсортировать его по возрастанию элементов и вывести. Вывести квадраты элементов с нечетными номерами.

7.2. Дан массив из N элементов. Отсортировать его по убыванию элементов и вывести. Найти сумму элементов с четными номерами и отдельно – с нечетными номерами.

7.3. Дан массив из N элементов (натуральные числа). Отсортировать его по возрастанию элементов и вывести. В нем четные числа заменить на 0, нечетные – на -1. Вывести итоговый массив.

7.4. Дан массив из N элементов (N – четное). Отсортировать его по убыванию элементов и вывести. Распечатать его в следующем порядке: $a_1, a_N, a_2, a_{N-1}, \ldots, a_{N/2}, a_{N/2+1}$.

7.5. Дан массив из N элементов (N – четное). Отсортировать его по возрастанию элементов и вывести. В нем элементы с четными и нечетными номерами поменять местами: a_1 с a_2, a_3 с a_4 и т.д. Вывести итоговый массив.

7.6. Даны два массива, каждый из N элементов. Отсортировать первый массив по убыванию элементов и вывести. Составить третий массив, в который поочередно входят элементы из 1–го и из 2–го массивов и вывести его на печать.

7.7. Дан массив из N элементов (N – четное). Отсортировать его по возрастанию элементов и вывести. Составить вдвое меньший массив, элементами которого являются: сумма 1-го и последнего, 2-го и предпоследнего элементов исходного массива и т.д. Вывести итоговый массив.

7.8. Дан массив из N элементов (натуральные числа). Отсортировать его по убыванию элементов и вывести. Определить, количество элементов, которые являются нечетными числами.

7.9. Дан массив из N элементов (натуральные числа). Отсортировать его по возрастанию элементов и вывести. Определить количество элементов, кратных 3.

7.10. Дан массив из N элементов (натуральные числа). Отсортировать его по убыванию элементов и вывести. Определить количество элементов, имеющих четные порядковые номера и являющихся нечетными числами.

7.11. Дан массив из N элементов (натуральные числа). Отсортировать его по убыванию элементов и вывести. Определить сумму тех элементов, которые кратны 5.

7.12. Дан массив из N элементов (целые числа). Отсортировать его по возрастанию элементов и вывести. Определить сумму тех, которые нечетны и отрицательны.

7.13. Дан массив из N элементов (натуральные числа). Отсортировать его по убыванию элементов и вывести. Определить среднее арифметическое кратных 5, но не кратных 10.

7.14. Дан массив из *N* элементов (натуральные числа). Отсортировать его по возрастанию элементов и вывести. Определить произведение элементов, кратных заданному числу P.

7.15. Дан массив из *N* элементов (натуральные числа). Отсортировать его по убыванию элементов и вывести. Определить количество элементов, кратных 3, но не кратных 5.

7.16. Написать программу, которая вычисляет среднее арифметическое элементов заданного массива без учета минимального и максимального элементов массива. Заданный массив отсортировать по возрастанию и вывести.

7.17. Написать программу, которая вычисляет среднюю (за неделю) температуру воздуха. Исходные данные должны вводится во время работы программы в массив. Заданный массив должен быть отсортирован по убыванию.

7.18. Написать программу, которая проверяет, находится ли введенное с клавиатуры число в массиве. Массив также должен вводится во время работы программы, затем упорядочен по возрастанию.

7.19. Написать программу, которая проверяет, представляют ли элементы введенного с клавиатуры массива возрастающую последовательность и если нет – упорядочить элементы по убыванию.

7.20. Написать программу, которая вычисляет, сколько раз введенное с клавиатуры число встречается в заданном массиве. Исходный массив отсортировать по возрастанию и вывести.

7.21. Написать программу, которая проверяет, есть ли во введенном извне массиве элементы с одинаковыми значениями. Исходный массив отсортировать по убыванию и вывести.

7.22. Написать программу, которая определяет количество учеников в классе, чей рост превышает средний. «Расставить» учеников по росту (в порядке убывания)

7.23. Написать программу, которая вводит с клавиатуры одномерный массив из 15 целых чисел, после чего выводит количество ненулевых элементов. Перед вводом каждого элемента должна появляться подсказка с номером элемента. Введенный массив отсортировать по возрастанию и вывести.

7.24. Написать программу, которая вычисляет среднее арифметическое ненулевых элементов введенного массива из целых чисел. Введенный массив отсортировать по убыванию и вывести.

7.25. Написать программу, которая выводит минимальный элемент введенного с клавиатуры массива из целых чисел. Введенный массив отсортировать по возрастанию и вывести.

8. ДВУМЕРНЫЕ МАССИВЫ

8.1. Понятие о двумерном массиве

Массивы могут быть и многомерными, например, двумерными. В этом случае размерности записываются в двух квадратных скобках:

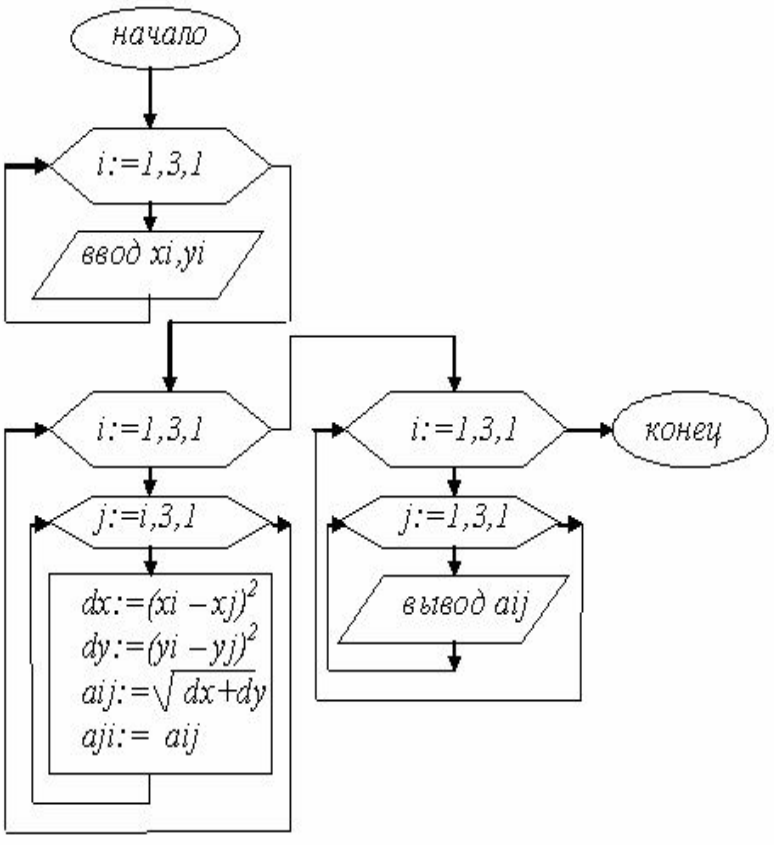
`int d[5][4]` – описан целочисленный массив из 5 строк и 4 столбцов.

Напомним, что в математике двумерный массив принято называть матрицей, при этом матрица, у которой число строк равно числу столбцов называется *квадратной*. Если у квадратной матрицы элементы симметрично относительно главной диагонали равны ($a_{21} = a_{12}$ и т.д.), то ее называют *симметричной*. Напомним, что *главная диагональ* матрицы содержит элементы с одинаковыми индексами: a_{11}, a_{22} и т.д. *Побочная диагональ* – вторая диагональ матрицы; *верхний треугольник* – элементы над главной диагональю (включая и элементы на диагонали); *нижний треугольник* – элементы под главной диагональю (включая и элементы на диагонали).

В случае, когда элементы верхнего и нижнего треугольника совпадают, то говорят, что матрица *симметрична*.

Пример 8.1. *Построить матрицу расстояний между N* ($N = 3$) *точками на плоскости, для которых заданы координаты.*

Заметим, что эта матрица будет квадратной и симметричной – расстояние от точки с номером *i* до точки с номером *j* равно расстоянию от точки *j* до точки *i*. Поэтому можно облегчить процесс построения – вычислять только элементы верхнего треугольника, а элементам нижнего треугольника просто присваивать уже вычисленные соответствующие значения. Заметим также, что элементы на главной диагонали будут равны нулю (расстояние от точки до самой себя).



```
//построение двумерного массива
#include <iostream>
#include <cmath>
#include <windows.h>
using namespace std;
int main()
{int i,j,x[3+1],y[3+1];
double a[3+1][3+1], dx,dy;
char str[256],str1[256];
AnsiToOem("введи координату X номер ",str);
AnsiToOem("введи координату Y номер ",str1);
for (i=1;i<=3;i++)
{cout<<str<<i<<" ";cin>>x[i];
 cout<<str1<<i<<" ";   cin>>y[i];};
for (i=1;i<=3;i++)
{ for (j=1;j<=3;j++)
  { dx=(double)(x[i]-x[j])*(x[i]-x[j]);
    dy=(double)(y[i]-y[j])*(y[i]-y[j]);
    a[i][j]=sqrt(dx+dy);
    a[j][i]=a[i][j];} };
for (i=1;i<=3;i++)
{cout<<" \n";
  for (j=1;j<=3;j++)
  {cout<<"  a["<<i<<"]["<<j<<"]="<< a[i][j];} }
cout<<"\n";
return 0;}
```

В результате исполнения такой программы, например, получим (рис. 8.18).

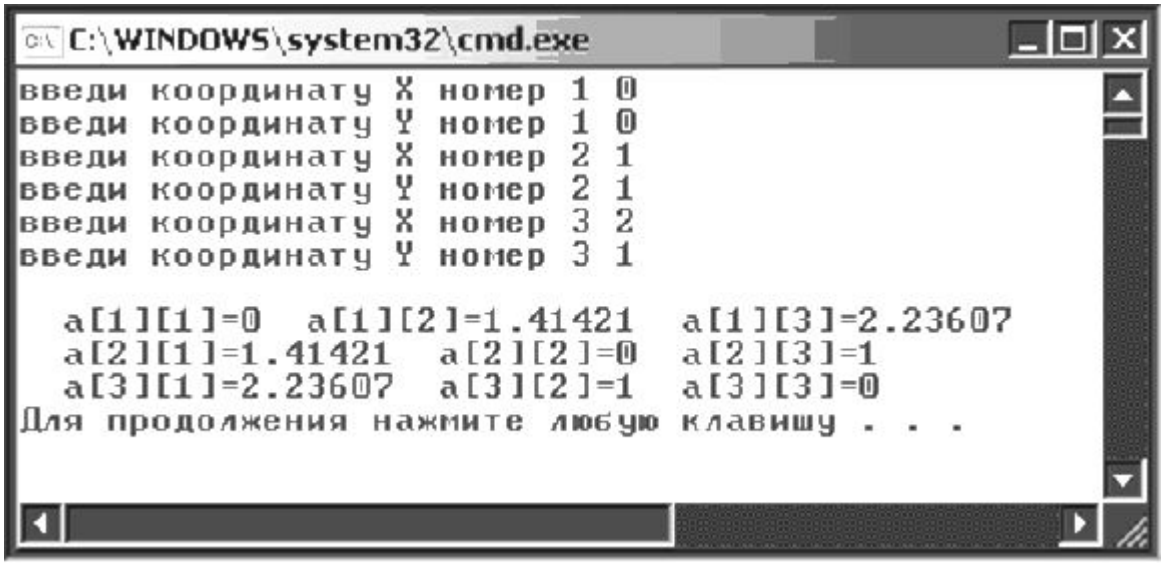


Рис. 8.18. Результат работы примера 8.1

Еще раз обращаем внимание, что хотя мы и используем массивы с размерностью на 3 элемента (берем три точки), но описываем массивы на 4 (чтобы не путаться с нулевыми по номеру элементами). Обратите также внимание, что поскольку координаты заданы в виде целых чисел, а на их основе получаем расстояние, которое является вещественным числом, то для преобразования типа от целого к вещественному (точнее – типу двойной точности) используем (`double`). Здесь для вычисления квадратного корня `sqrt()` нужно подключить `cmath` (в отличие от традиционного `math.h`), поскольку при вычислении корня, вообще говоря, возможны комплексные числа.

8.2. Датчик случайных чисел

В рассмотренном ниже примере будем использовать датчик случайных чисел. На языке C++ он реализуется функцией `rand()%RAND_MAX` – дает случайное целое число из интервала от `0` до положительного целого `RAND_MAX-1` (в качестве `RAND_MAX` можно указать сразу конкретное число). Кроме того, для запуска этого датчика случайных чисел (чтобы получать новый набор случайных чисел при каждом новом запуске программы), используют функцию `srand(time(0))`, т.е. для старта генератора случайных чисел будет использоваться системное время компьютера. Причем это время следует преобразовать в беззнаковое целое (`unsigned int`). Кроме того, требуется подключить заголовочный файл `<stdlib.h>`, содержащий эти функции. Для получения системного времени нужно подключить заголовочный файл `<time.h>`

В ***примере*** 8.2 будем впервые использовать так называемые *манипуляторы*. Они позволяют, например, выводить числа по определенному формату – скажем, отводить на вывод числа три позиции (независимо от того, что число может быть и двузначным). Для использования манипуляторов нужно подключить заголовочный файл `<iomanip>`.

Пример 8.2. *В двумерном массиве, заданном с помощью датчика случайных чисел, подсчитать количество строк, имеющих элемент, равный первому элементу строки, и указать номера этих строк.*

```
# include <iostream>
# include <stdlib.h>
# include <iomanip>
# include <windows.h>
#include <time.h>
using namespace std;
 int main()
{  const int n=5;    bool nal;
     int i, j,k,b, a[n+1][n+1], num[n+1];
     char str[256], str1[256];
// запуск генератора случайных чисел
srand((unsigned int)time(0));
   //формируем массив из случайных чисел (от 0 до 4)
   for (i=1; i<=n; i++)
   for (j=1; j<=n; j++)
   { a[i][j]=rand()%5;}
   // вывод массива
   for (i=1; i<=n; i++)
   { cout<<"\n";
   for (j=1; j<=n; j++)
   { cout<<setw(3)<< a[i][j];} }
// подсчет числа строк (по условию)
   k=0;
   for (i=1; i<=n; i++)
   { b=a[i][1]; nal=false;
     for (j=2; j<=n; j++)
      { if (a[i][j] = = b) nal=true; if nal break;}
      if (nal) {k=k+1; num[k]=i;};
   }
   AnsiToOem(" \n количество строк=",str);
   AnsiToOem(" их номера: ", str1);
   cout<<str<<k<<str1;
   for (i=1; i<=k; i++) cout<<setw(3)<<num[i];
   cout<<"\n";
   return(0);
}
```

Результат работы этой программы представлен на рис. 8.19.

Обратите внимание, что здесь использована целочисленная константа `const int n=5`. Благодаря этому, мы можем легко изменить размерность массива и количество оборотов цикла (при очередном запуске программы изменим значение этой константы в тексте программы). В программе использована также логическая переменная `nal`, которая примет значение «истина» (true), как толь-

ко найдем элемент, равный первому в строке. Это позволит сократить число оборотов цикла (не будет продолжаться поиск до конца строки).

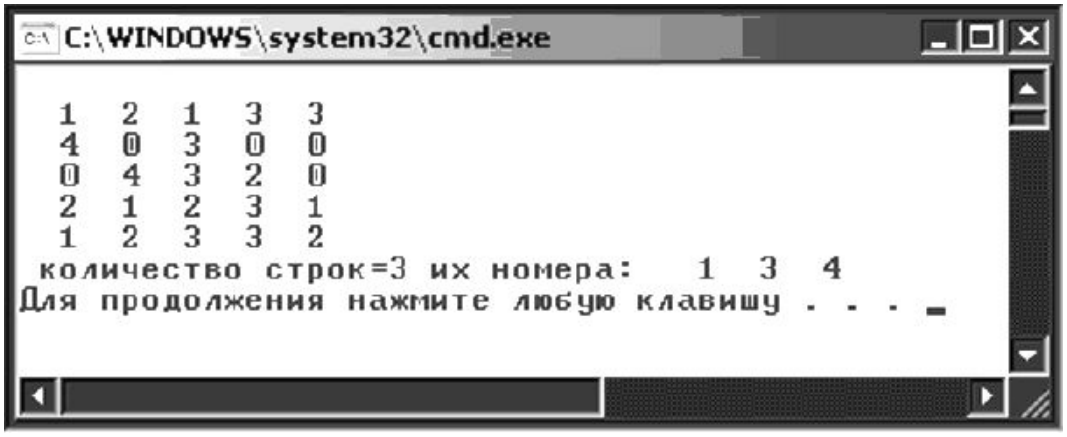


Рис. 8.19. Результат работы программы примера 8.2

Практические задания

8.1. Дана матрица 5×5. Для данного натурального *M* найти сумму тех элементов матрицы, сумма индексов которых равна *M*.

8.2. Дана матрица 5×5. Построить одномерный массив *B*(5), состоящий из сумм элементов строк.

8.3. Дана матрица 5×5. Построить одномерный массив *B*(5), состоящий из произведений элементов столбцов.

8.4. Дана матрица 5×5. Построить одномерный массив *B*(5), состоящий из наименьших значений элементов строк.

8.5. Дана матрица 5×5. Построить одномерный массив *B*(5), состоящий из средних арифметических элементов строк.

8.6. Дана матрица 5×5. Вывести ее в транспонированном виде (поменять местами строки со столбцами).

8.7. Дана матрица 5×5. Вывести ее в верхнем треугольном виде (т.е. напечатать только элементы верхнего треугольника и именно в виде треугольника).

8.8. Дана матрица 5×5. Вывести ее в нижнем треугольном виде.

8.9. Результаты соревнований по прыжкам в длину представлены в виде матрицы 5×3 (5 спортсменов по 3 попытки у каждого). Указать, какой спортсмен и в какой попытке показал наилучший результат.

8.10. Результаты соревнований по пятиборью представлены в виде матрицы 5×5 (5 спортсменов и 5 видов соревнований), в которых указаны места, занятые каждым спортсменом в данном виде. Найти лучшего спортсмена (наименьшая сумма мест).

В последующих трех задачах использовать следующее:

Таблица футбольного чемпионата задана квадратной матрицей порядка *n*, в которой все элементы, принадлежащие главной диагонали, равны нулю, а каждый элемент, не принадлежащий главной диагонали, равен 3, 1 или 0 (число очков, набранных в игре: 3 – выигрыш, 1 – ничья, 0 – проигрыш).

8.11. Найти число команд, имеющих больше побед, чем поражений.

8.12. Определить номера команд, прошедших чемпионат без поражений.

8.13. Выяснить, имеется хотя бы одна команда, выигравшая более половины игр.

8.14. Дана действительная матрица размера *n x m*, в которой не все элементы равны нулю. Получить новую матрицу путём деления всех элементов данной матрицы на её наибольший по модулю элемент.

8.15. Дана действительная квадратная матрица порядка 12. Заменить нулями все её элементы, расположенные на на главной диагонали и выше неё.

8.16. Дана действительная матрица размера *n*×*m*. Найти сумму наибольших значений элементов её строк.

8.17. В данной действительной квадратной матрице порядка *n* найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.

8.18. В данной действительной матрице размера 6×9 поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением. Предполагается, что эти элементы единственны.

8.19. Дана действительная матрица размера *n*×*m*, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.

8.20. Дана целочисленная квадратная матрица порядка 8. Найти наименьшее из значений элементов столбца, который обладает наибольшей суммой модулей элементов. Если таких столбцов несколько, то взять первый из них.

8.21. Составить программу формирования двумерного массива *n*×*n* по следующему правилу: элементы главной диагонали приравнять 1, ниже главной диагонали – 0, а выше – сумме индексов.

8.22. Составить программу формирования двумерного массива из предложенного одномерного так, чтобы первая строка содержала четные по номеру элементы исходного массива, а вторая – нечетные. Предусмотреть случай нечетного количества элементов массива.

8.23. Составить программу формирования двумерного массива из предложенного одномерного, разделив его на два столбца.

8.24. Составить программу формирования двумерного массива из предложенного одномерного, разделив его на две строки.

8.25. Дан двумерный числовой массив. Составить программу обмена местами заданных двух его строк.

9. ФУНКЦИИ

9.1. Понятие о пользовательских функциях

В C++ можно создавать пользовательские функции. Собственно говоря, мы с самого начала создавали функции: **main()** – не что иное, как главная функция пользователя.

В библиотеке C++ имеется немало встроенных функций. Они размещены в отдельных заголовочных файлах (тех самых, которые подключаются с помощью **#include**). Заголовочный файл имеет имя и расширение **.h** Мы уже использовали заголовочные файлы:

iostream.h – содержит функции для ввода-вывода

math.h – содержит математические функции.

Для *создания функции пользователя* (и ее последующего исполнения) необходимо ее описать:

1) задать прототип функции :

***тип имя_функции* (*параметры*);**

это делается там же, где описываются переменные, т.е. до заголовка **main(),** обратим внимание, что в конце записи прототипа точка с запятой ставится обязательно;

2) собственно описать функцию

***тип имя_функции*(*параметры*)**

{ тело функции }

Такое описание делается вне тела другой функции, в том числе и функции **main()** (как правило, за закрывающей фигурной скобкой главной функции).

В конце описания (после фигурной скобки) точка с запятой не ставится. Функция может иметь только одно возвращаемое значение, которое определяется служебным словом **return (*возвращаемое_значение*),** которое размещается в конце функции (перед закрывающей фигурной скобкой).

Внутри функции могут быть описаны свои локальные переменные.

Пример 9.1. Вычислить площадь пятиугольника, у которого известны длины сторон и двух диагоналей (см. рис. 8.20).

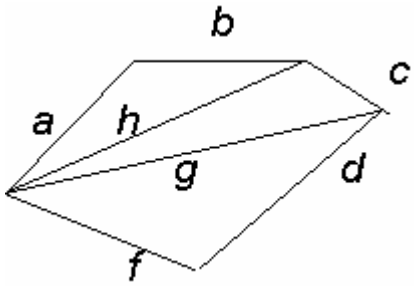
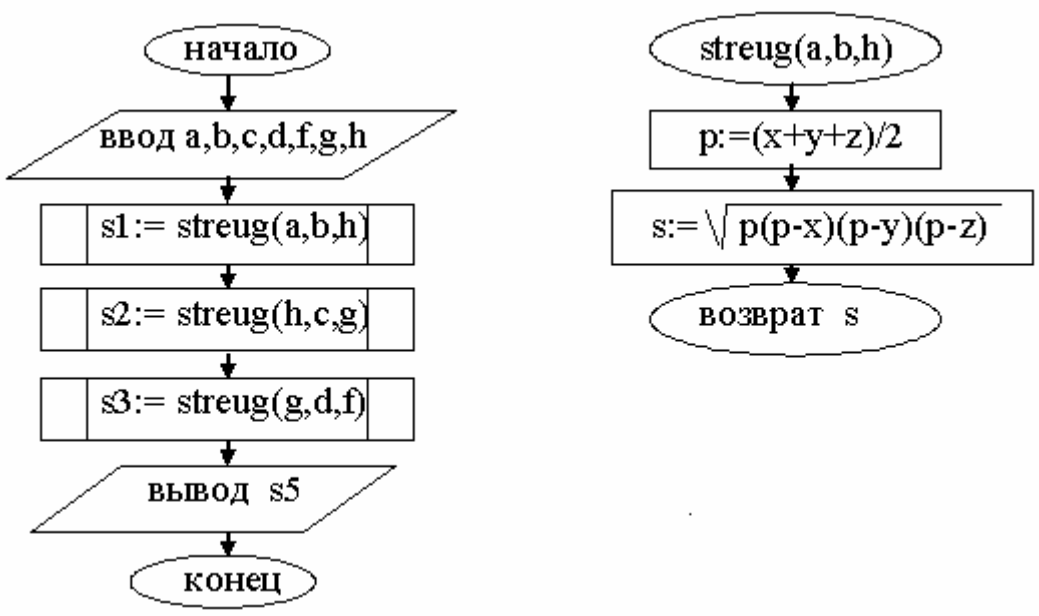


Рис. 8.20. Пятиугольник со сторонами *a*, *b*, *c*, *d*, *f* и диагоналями *h*,*g*.

Очевидно, что площадь такого пятиугольника – сумма площадей треугольников, для каждого из которых известны длины сторон. Для вычисления площади каждого треугольника (условно обозначим его стороны за *x*,*y*,*z*) применим формулу Герона:

$$S=\sqrt{p(p-x)(p-y)(p-z)}\,, \qquad \text{где } p=(x+y+z)/2\,.$$




```
// площадь 5-угольника через 3-угольники
#include <iostream>
#include <cmath>
#include <windows.h>
using namespace std;
float streug(float x,float y,float z);
int main()
{    float a,b,c,d,f,g,h,s1,s2,s3,s5; char str[256];
```

```
     AnsiToOem("\n введи a,b,c,d,f,g,h ",str);
  cout<<str;  cin>>a>>b>>c>>d>>f>>g>>h;
     s1=streug(a,b,h); s2=streug(h,c,g);
   s3=streug(g,d,f);    s5=s1+s2+s3;
  cout<<"\n"; cout<<s5<<"\n";
return 0;}
float streug(float x,float y,float z)
//ф-я для вычисления площади 3-ника по его сторонам
   {   float p,s;
       p=(x+y+z)/2;  s=sqrt(p*(p-x)*(p-y)*(p-z));
       return (s); }
```

Если провести расчеты по этой программе с такими данными: $a = 1{,}5$; $b = 1$; $c = 2$; $d = 2{,}5$; $f = 1$; $g = 2{,}5$; $h = 2$, то получим площадь 5-угольника = 3,90 (примерно).

9.2. Рекурсия

В C++ допустима *рекурсия* – обращение функции к самой себе. Проиллюстрируем использование рекурсии на примере.

Пример 9.2. Рассчитать число сочетаний из N элементов по M:

$$C_N^M = \frac{N!}{M!(N-M)!} \qquad \text{для } (N > M).$$

По такой формуле можно рассчитать, например, каким числом способов можно рассадить учеников класса (скажем, $N = 30$) по двое ($M = 2$).

Здесь потребуется трижды вычислять факториал для разных чисел. Вычисление факториала и оформим рекурсивной функцией.

```
// число сочетаний из N по M
#include <iostream>
#include <cmath>
#include <windows.h>
using namespace std;
int n,m; double c;
float fact(int n);
int main()
{    char str[256], str1[256];
     AnsiToOem("\n  введи N и M ", str);
     cout<<str; cin>>n>>m;
     c=floor(fact(n)/(fact(m)*fact(n-m)) );
     AnsiToOem("\n число сочетаний из ", str);
     AnsiToOem(" по ", str1);
```

```
     cout<<str<<n<<str1<<m<<" = "<<c<<"\n";
   return (0); }
float fact(int n)
{if (n==0) return(1);else return (n*fact(n-1));}
```

Как видно, внутри функции `fact` имеется обращение к ней же, но со значением параметра, меньшим на 1. Поэтому, если, например, задать $n = 5$, то внутри функции произойдет обращение к ней же, но с $n = 4$, затем с $n = 3$, $n = 2$, $n = 1$, $n = 0$. Для $n = 0$ функция примет значение 1 и процесс пойдет в обратную сторону, последовательно выполняя умножение на 1, на 2, на 3, 4, 5. В результате и получим значение 5! .

Функцию факториала лучше оформить функцией вещественного типа (хотя факториал – целое число) во избежание переполнения при вычислениях. Однако, поскольку число сочетаний это строго целое число, то для его окончательного вычисления применили функцию округления вещественного до целого `floor` (она выдает наибольшее целое, не превосходящее данное число). Заметим, кстати, что для правильного округления вещественного числа ***x*** нужно указывать `floor(`x`+0.5)`

Если запустить программу и задать $n = 9$, $m = 5$, то получим в ответе 126.

Пример 9.3. Провести сортировку одномерного массива последовательным поиском максимума и перестановкой его в конец, причем применить рекурсивную процедуру сортировки.

```
#include <iostream>
#include <windows.h>
using namespace std;
const int n=5; float a[n+1];
int fmax(float a[], int nach, int kon);
void sort (float a[], int nach, int kon);
int main()
{    int i; char str[256];
     AnsiToOem("\n введи a[", str);
     for (i=1; i<=n; i++)
   { cout<<str<<i<<"] "; cin>>a[i];}
   sort (a,1,n);
   for (i=1; i<=n; i++)
   {cout<<"  a["<<i<<"]="<<a[i];} cout<<"\n";
   return(0);}
int fmax(float a[], int nach, int kon)
```

```
{//поиск максимума в массиве и запоминание номера
 // макс. элемента
     int k, kmax; float max;
     kmax=nach; max=a[nach];
   for (k=nach; k<=kon; k++)
   {if (a[k]>max) {max=a[k]; kmax=k;}}
   return kmax;}
void sort (float a[], int nach, int kon)
{//сортировка в массиве перестановкой максимума
 // в конец рекурсивным образом
     float b; int nmax,kons;      kons=kon;
   if (nach<=kon)
   { nmax=fmax(a,nach, kons);
     b=a[nmax]; a[nmax]=a[kon]; a[kon]=b;
      sort(a, nach, kons-1);}  }
```

Результат работы такой программы на рис. 8.21.

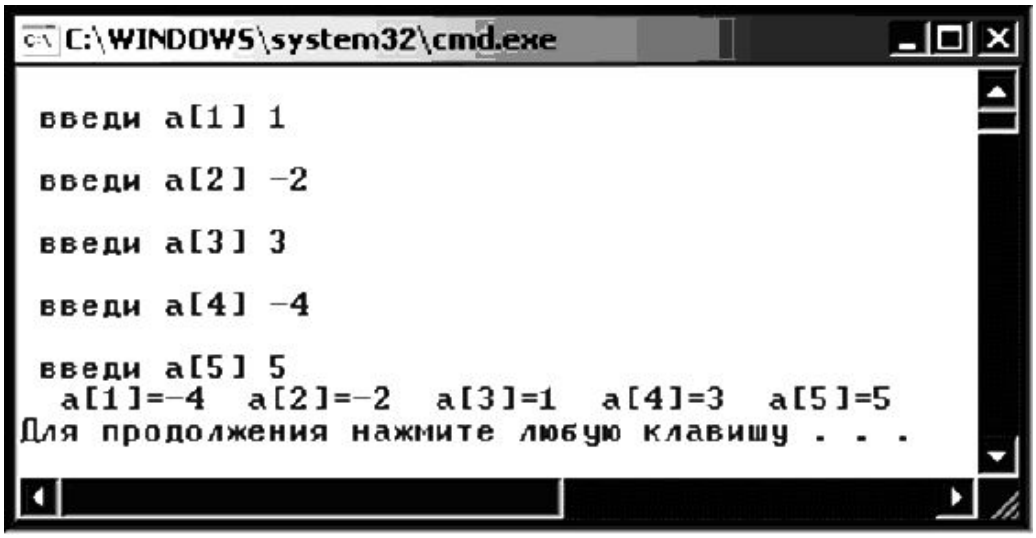


Рис. 8.21. Результат работы программы примера 9.3

9.3. Вызов функции из функции

Рассмотрим пример с использованием *нескольких функций*. Особенностью языка С++ является невозможность для функции определять внутри своего тела другую функцию. То есть не допускаются вложенные определения. Но вполне допускается вызов функции из другой функции.

Пример 9.4. Рассчитать треугольник Паскаля[1] вида:

[1] Треугольник Паскаля образуется по правилу: по краям каждой строки стоят единицы, а каждое из остальных чисел равно сумме двух стоящих над ним чисел предыдущей строки.

$$C_0^0$$

$$C_1^0 \quad C_1^1$$

$$C_2^0 \quad C_2^1 \quad C_2^2$$

$$C_3^0 \quad C_3^1 \quad C_3^2 \quad C_3^3$$

и т.д. Здесь используется число сочетаний из n элементов по m, формула для расчета которого приведена выше. Оформим расчет числа сочетаний функцией, а вычисление факториала – внутренней функцией, причем, как и ранее, оформим вычисление факториала рекурсивным образом.

```
#include <cmath>
#include <iomanip>
#include <windows.h>
using namespace std;
int n,m,r,i,j; double c;
float fact(int n);  int soch(int n,int m);
int main()
{    char str[256];
AnsiToOem("\n  введи размер треугольника Паскаля", str);
     cout<<str; cin>>r;
 for (i=0;i<=r;i++)
     {cout<<"\n";
         for (j=0;j<=i;j++)
          {c=soch(i,j);cout<<setw(r*(r-i+j+1))<<c;};
          cout<<"\n";   }
 return (0);}
float fact(int n)
{if (n==0) return(1); else return (n*fact(n-1)); }
int soch(int n,int m)
{ int c= floor(fact(n)/(fact(m)*fact(n-m)) );
  return(c);}
```

В данном примере нам потребовался форматированный вывод. Для его организации в C++ применяются так называемые *манипуляторы* (для их использования нужно подключить заголовочный файл **iomanip**). Конкретно мы использовали манипулятор `setw(int)` – который задает ширину поля (количество позиций, отведенных для вывода переменной) с выравниванием по правому краю.

Если запустить данную программу и задать размер треугольника равным 3, то получим (рис. 8.22):

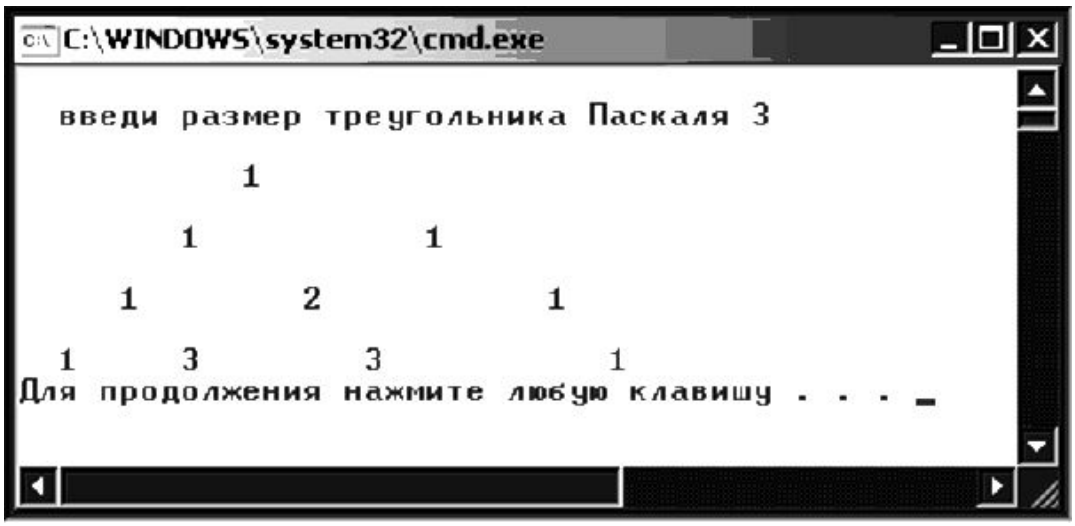


Рис. 8.22. Результат работы программы примера 9.4

9.4. Функция типа void и глобальные переменные

Но как быть, когда нам потребуется вызвать функцию, которая должна получить нам *несколько значений в результате*. В Паскале мы организовывали так называемые *процедуры*. Здесь для этого проще всего использовать так называемые функции типа **void**. Такие функции не содержат оператора **return** и поэтому как бы ничего не передают наружу. Однако в C++ существует понятие *глобальной переменной*, т.е. такой переменной, которая доступна в любой функции данной программы. Если, таким образом, обратится к функции типа **void** просто по имени (без использования присваивания) и внутри нее пересчитать глобальные переменные, то после этого в основной программе значения глобальных переменных изменятся. Напомним, что глобальные переменные описываются до открытия главной функции **main**().

> ***Замечание***. Более строго ограничение на *возврат из функции только одного значения* можно обойти с помощью *указателей*, но работа с ними будет рассмотрена нами в разделе 12.2 данного пособия.

Пример 9.5. Даны три тройки чисел: $a1$, $b1$, $c1$; $a2$, $b2$, $c2$; $a3$, $b3$, $c3$. Найти в каждой из них наибольшее и наименьшее, а затем наибольшее и наименьшее среди наибольших и отдельно – среди наименьших. (Здесь и организуем функцию, в которой получим «на выходе» два значения – максимум и минимум для заданной тройки чисел.)

```
#include <iostream>
#include <iomanip>
#include <windows.h>
```

```
using namespace std;
void maxmin(double x,double y,double z);
double mx, mn;// глобальные переменные (из maxmin)
int main()
{double max1,min1,max2,min2,max3,min3,mxmax,mnmax,
mxmin, mnmin, a1,b1,c1, a2,b2,c2, a3,b3,c3;
 char str[256], str1[256], str2[256];
 AnsiToOem("\n введи a1,b1,c1 ", str);
cout<<str; cin>>a1>>b1>>c1;
AnsiToOem("\n введи a2,b2,c2 ", str1);
cout<<str1; cin>>a2>>b2>>c2;
AnsiToOem("\n введи a3,b3,c3 ", str2);
cout<<str2; cin>>a3>>b3>>c3;
maxmin(a1,b1,c1);   max1=mx; min1=mn;
cout<<"\n max1,min1"<<setw(5)<<max1<<setw(5) <<min1;
maxmin(a2,b2,c2);   max2=mx; min2=mn;
cout<<"\n max2,min2"<<setw(5)<<max2<<setw(5) <<min2;
maxmin(a3,b3,c3);   max3=mx; min3=mn;
cout<<"\n max3,min3"<<setw(5)<<max3<<setw(5) <<min3;
maxmin(max1,max2,max3);   mxmax=mx; mnmax=mn;
cout<<"\n maxmax,minmax "<<setw(5)<<mxmax<<setw(5) <<mnmax;
maxmin(min1,min2,min3);   mxmin=mx; mnmin=mn;
cout<<"\n maxmin,minmin "<<setw(5)<<mxmin<<setw(5) <<mnmin;
cout<<"\n";
   return (0);}
void maxmin(double x,double y,double z)
     {if ((x>y)&&(x>z)) mx=x; if ((y>x)&&(y>z)) mx=y;
      if ((z>x)&&(z>y)) mx=z;if ((x<y)&&(x<z)) mn=x;
      if ((y<x)&&(y<z)) mn=y;if ((z<x)&&(z<y)) mn=z;
 }
```

В результате, например, получим (рис. 8.23):

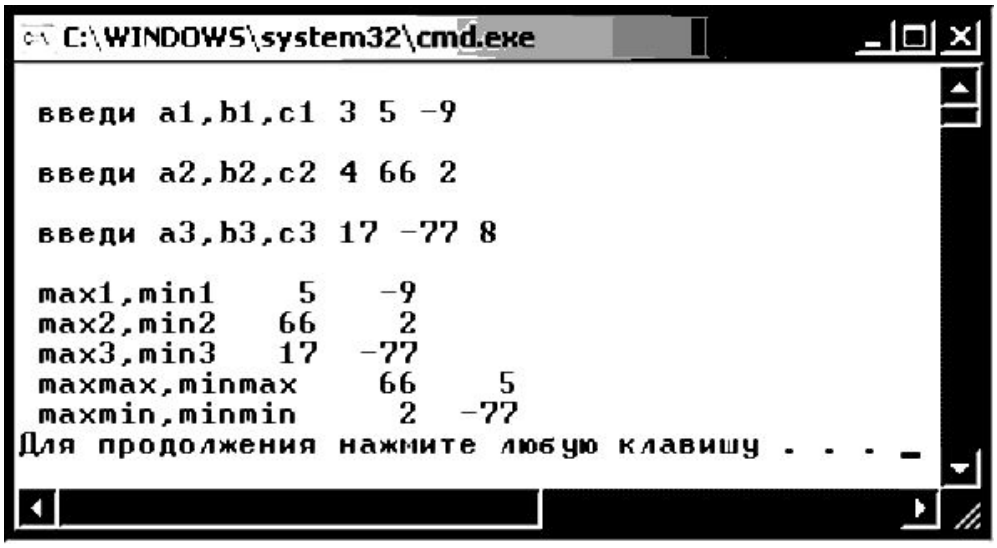


Рис. 8.23. Результат работы примера 9.5

9.5. Передача в функцию имени функции

Как передать в функцию имя функции? Точно так же, как любую другую величину: в списке параметров перед именем параметра указать его тип. Значит, специально нужно будет определить собственный тип для функции. А правильнее задать синоним с помощью ключевого слова **typedef**:

```
typedef double (*Pfun) (double, double)
```

(в данном случае описали функциональный тип `Pfun` для функции от двух параметров)

Пример 9.6. Создадим программу с функцией для вывода таблицы значений произвольной функции в заданных пределах и с указанным шагом. И соответственно нужную функцию будем туда передавать (в функцию вывода).

```
// передача в функцию имени функции
// для построения таблицы переданной функции
#include <iostream>
#include <math.h>
#include <iomanip>
#include <windows.h>
using namespace std;
typedef double(*Pfun)(double);
void print_tabl(Pfun fun, double xn,  double xk, double dx);
double cosh(double x);
int main()
{    double xn, xk, dx;
     char str[256];
     AnsiToOem("введите xn, xk, dx ",str);
        cout<<str; cin>>xn>>xk>>dx;
          print_tabl(cosh, xn, xk, dx);
return(0);
}
void print_tabl(Pfun fun, double xn, double xk,  double dx)
{double x;
 cout<<"-------------------------------------\n";
 cout<<"|      X      |          Y           |\n" ;
 cout<<"-------------------------------------|\n";
 for (x= xn; x<=xk; x+=dx)
 cout<<"\n"<<setw(15)<< x <<setw(15)<<fun(x);
 cout<<"\n-------------------------------------|\n";
 }
 double cosh(double x)
 { double  y;
  y=(exp(x)+exp(-x))/2;
     return(y)   ;  }
```

В результате выполнения программы получим, например, такой результат (рис. 8.24)

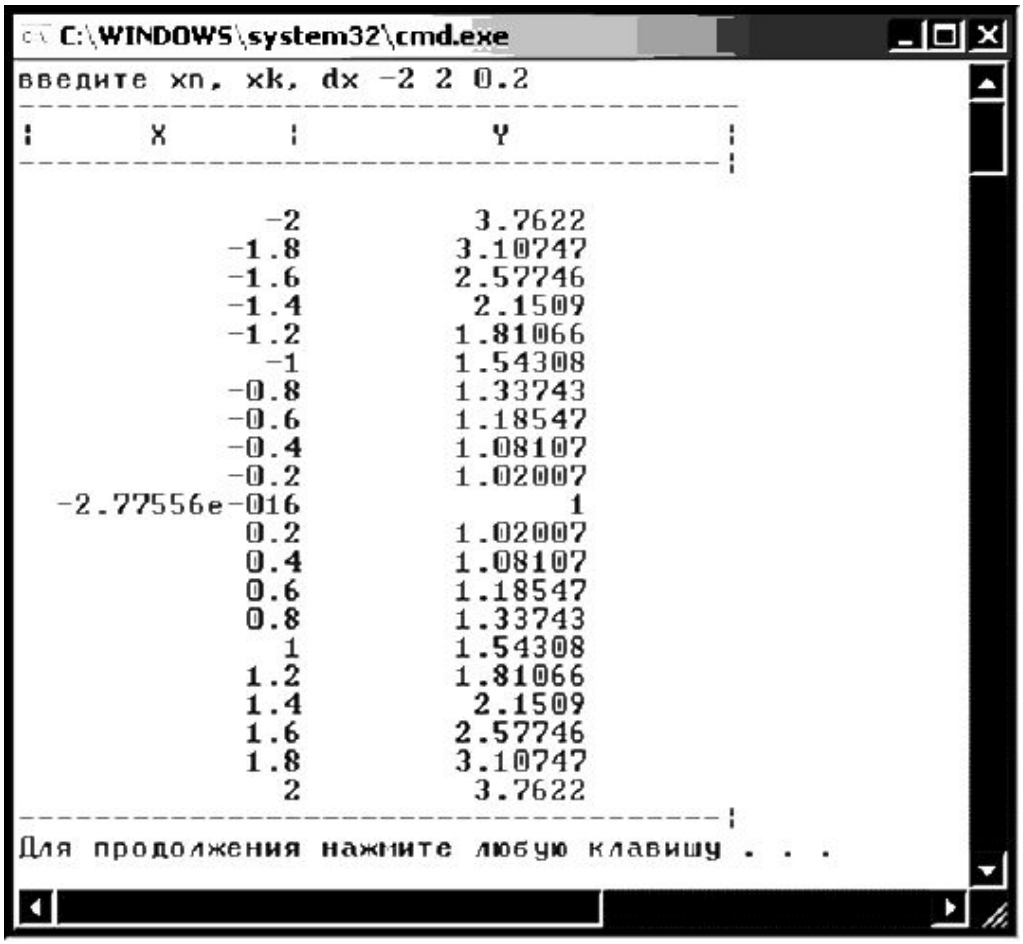


Рис. 8.24. Результат работы примера 9.6

Практические задания

9.1. Даны числа a, b, c, d. Получить $x = \max(a, b)$, $y = \max(c, d)$, $z = \max(x, y)$. Вычисление $\max(k, m)$ (большего из двух чисел k, m) оформить функцией.

9.2. Оформить функцию step(x, n) от вещественного x и целого n, вычисляющую (через последовательное умножение) x^n и проверить ее в работе.

9.3. Даны координаты вершин двух треугольников. Определить, какой из них имеет большую площадь. Вычисление площади треугольника по координатам оформить функцией.

9.4. Даны четыре натуральных числа. Найти наименьшее общее кратное (НОК) для этих четырех чисел. Поиск НОК двух чисел оформить функцией.

9.5. Даны координаты n точек на плоскости в виде масивов X, Y. Найти наиболее и наименее удаленные точки. Вычисление расстояния между парой точек оформить функцией.

9.6. Даны отрезки *a*, *b*, *c*, *d*. Для каждой тройки этих отрезков, из которых можно построить треугольник, напечатать площадь данного треугольника. Вычисление площади треугольника по заданным сторонам *x*, *y*, *z* с проверкой существования такого треугольника оформить функцией.

9.7. Рассчитать набор из *n* чисел Фибоначчи: 1, 1, 2, 3, 5, 8, 13, 21... (т.е. $f(n) = f(n-1) + f(n-2)$). Для вычисления очередного числа Фибоначчи оформить рекурсивную функцию.

9.8. Дано число *n*. Получить все простые делители этого числа. Процедуру распознания простого числа оформить отдельно. (Напомним, что простым называется число, которое не имеет целочисленных делителей, кроме единицы и самого себя).

9.9. Составить программу, которая число, заданное в десятичной системе счисления, переведет в: *а*) двоичную систему счисления; *б*) восьмеричную. Перевод в каждую из систем счисления оформить отдельной функцией.

9.10. Составить программу, которая позволит вычислить *N*-ю степень числа *X*, пользуясь формулой:

$$X^N = \begin{cases} 1, & \text{при} \quad N = 0, \\ 1/X^{|N|} & \text{при} \quad N < 0, \\ X \cdot X^{N-1} & \text{при} \quad N > 0. \end{cases}$$

Оформить вычисление такой степени в виде рекурсивной функции.

9.11. Написать функцию, которая сравнивает два целых числа и возвращает результат сравнения в виде одного из знаков: >, < или =. Проверить ее в работе, написав программу с ее использованием.

9.12. Написать функцию, которая вычисляет сопротивление цепи состоящей из двух проводников. Параметрами ее являются значения сопротивлений, а также тип соединения – последовательное или параллельное (цифрами 1 или 2). Проверить ее в работе, написав программу с ее использованием.

9.13. Написать функцию для вычисления заданного процента введенного числа. У функции два параметра – процент и само число. Проверить ее в работе, написав программу с ее использованием.

9.14. Написать функцию, которая вычисляет доход по вкладу. Исходными данными для функции являются: величина вклада, процентная ставка (годовых) и срок вклада (количество дней). Проверить ее в работе, написав программу с ее использованием.

9.15. Написать подпрограмму, которая возвращает преобразованную к верхнему регистру строку, полученную в качестве аргу-

мента (т.е. вместо малых букв выводит строку заглавными буквами). Проверить ее в работе, написав программу с ее использованием.

9.16. Написать функцию решения квадратного уравнения. Исходными данными для подпрограммы должны быть коэффициенты уравнения, а выдавать подпрограмма должна сообщение о том, есть корни или нет корней, их количество и их значение. Проверять вводимые данные (коэффициент при x^2 не должен быть нулем). Проверить ее в работе, написав программу с ее использованием.

9.17. Написать функцию, которая выводит на экран строку, состоящую из звездочек. Длина строки (количество звездочек) задается извне. Проверить ее в работе, написав программу с ее использованием.

9.18. Дано натуральное число n. Среди чисел 1, 2, ..., n найти все те, которые можно представить в виде суммы квадратов двух натуральных чисел. (Определить функцию, позволяющую распознавать полные квадраты.)

9.19. Даны действительные числа $x1$, $y1$, $x2$, $y2$, ..., $x10$, $y10$. Найти периметр десятиугольника, вершины которого имеют соответственно координаты ($x1$, $y1$), ($x2$, $y2$), ..., ($x10$, $y10$). (Определить функцию вычисления расстояния между двумя точками, заданными своими координатами.)

9.20. Дано натуральное число n. Выяснить, имеются ли среди чисел n, $n + 1$, ..., $2n$ близнецы, т.е. простые числа, разность между которыми равна двум. (Определить процедуру, позволяющую распознавать простые числа.)

В ***пяти следующих*** задачах потребуется сократить обыкновенную дробь результата. Процедуру сокращения дроби оформить в виде отдельной функции (напомним, что для этого нужно искать наибольший общий делитель для числителя и знаменателя дроби).

9.21. Составить программу для сложения двух обыкновенных дробей *a/b + c/d*.

9.22. Составить программу для вычитания двух обыкновенных дробей *a/b – c/d*.

9.23. Составить программу для умножения двух обыкновенных дробей *a/b * c/d*.

9.24. Составить программу для деления двух обыкновенных дробей *a/b : c/d*.

9.25. Составить программу для возведения в степень обыкновенной дроби.

10. СОБСТВЕННАЯ БИБЛИОТЕКА ПРОГРАММИСТА

Созданные пользователем функции целесообразно объединить в отдельную собственную библиотеку программиста. Для этого создадим (в своей папке!) на диске специальную папку под названием **SUBPROG**. В ней и будем размещать заголовочные файлы, содержащие функции пользователя.

Всякий заголовочный файл оформляется директивами препроцессору:

#infdef *имя_файла*
#define *имя_файла*
…
#endif

Это сочетание директив предотвращает повторные включения директив-компонентов. ***Имя_файла*** может быть любым, но не следует использовать его еще в каком-либо качестве в своих программах. Таким образом будет создан заголовочный файл ***имя_файла*.h** Такой файл содержит прототипы соответствующих функций, константы и т.п. Кроме того, там могут содержаться директивы на включение в программу тех заголовочных файлов из библиотеки компилятора, использование которых неизбежно в любой задаче (ввод/вывод, математические и т.п.)

В отличие от встроенных в компилятор, собственные списки и функции включаются с указанием пути к файлу, в который они записаны, и в обычных двойных кавычках.

Будем каждую библиотечную функцию записывать в своем отдельном программном файле. Имя файла и имя записанной в него функции могут и различаться, но желательно, чтобы они совпадали. Расширения таких файлов – как и для любой программы C++ – .cpp.

10.1. Перегрузка функций

C++ позволяет определить несколько функций с одним и тем же именем, но с различным набором аргументов – это называется *перегрузкой функций*. Приведем пример, иллюстрирующий, зачем нужна перегрузка.

В библиотеке стандартных функций (см. Приложение 1) имеется три функции для определения абсолютной величины числа трех типов:

```
int abs(int)
long labs(long)
double fabs(double)
```

Гораздо удобнее было бы вызывать одну функцию для выполнения всех вариантов.

Для осуществления перегрузки необходимо записать три варианта прототипа новой функции и три варианта описания функции в форме функции **abs1**. Теперь при вызове функции **abs1** компилятор определит тип переменной и исполнит нужный вариант. Итак, перегруженная функция **abs1** имеет вид:

```
int abs1(int);
double abs1(double);
long abs1(long);

int abs1(int i);
{ return abs(i); }

double abs1(double a);
{ return fabs(a); }

long abs1(long i);
{ return labs(i); }
```

А программа с ее использованием будет выглядеть примерно так:

```
#include <iostream>
#include <math.h>
using namespace std;
int x=-7;double z=-1.56789e2; long y=-33156;
int abs1(int);
double abs1(double);
long abs1(long);
main()
{ cout<<"\n"<<abs1(x)<<"\n"<<abs1(y)<<"\n"<<abs1(z);
  cout<<”\n”; return (0); }
int abs1(int x)
{ return abs(x); }
double abs1(double z)
{ return fabs(z); }
long abs1(long y)
{ return labs(y); }
```

Когда запустим на исполнение, то получим (рис. 8.25):

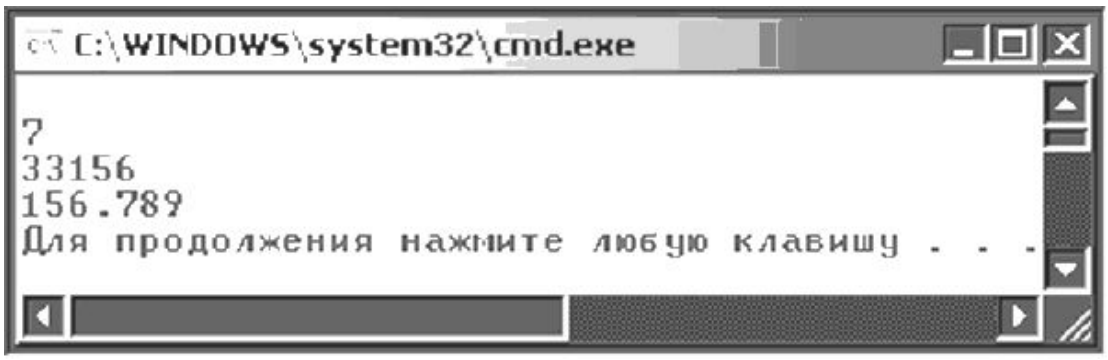


Рис. 8.25. Результат работы программы примера

11. ПЕРЕЧИСЛИМЫЙ ТИП

11.1. Понятие о перечислимом типе

В C++ имеется *перечислимый* тип. Его называют **enum**. При описании такого типа задаются наборы констант (в фигурных скобках). При этом идентификаторам-константам присваиваются значения натурального ряда от нуля. Если значение одного из идентификаторов задается явно, то последующим присваиваются номера по возрастанию через 1. Рассмотрим на примере.

Пример 11.1. Подсчитать число часов рабочей недели, если известно, что суббота и воскресенье – выходные, а в пятницу рабочий день длится 8 часов (в остальные дни 8.25 часа).

```
#include <iostream>
#include <windows.h>
using namespace std;
float t; enum dni
{Monday=1, Tuesday, Wednesday, Thursday, Friday, Saturday, Sun-
day} ;
     int main()
     {dni d;   t=0; char str[256];
       AnsiToOem(" сумма часов раб.недели=", str);
       for (d=Monday; d<=Friday; d=dni(d+1))
          { if (d == Friday) t=t+8; else t=t+8.25; } ;
        cout << str <<t<<"\n";
     return(0); }
```

Обратите внимание на необычную форму записи приращения параметра цикла: `d=dni(d+1)`. К сожалению, операция инкремента `d++` для перечислимого типа не работает. Как обойти эту проблему, рассмотрим в следующем разделе («12. Указатели»).

11.2. Множественный выбор

В C++ имеется *структура множественного выбора* **switch** (переключатель). Переключатель начинается с заголовка, определяющего имя метки. Тело переключателя заключено в фигурные скобки. Текст тела переключателя разделен метками **case**. Двоеточие – признак метки. Рассмотрим использование структуры множественного выбора на следующем примере.

Пример 11.2. Дан перечислимый тип *muns* (*названия месяцев года*), описаны переменные *int d*1, *d*2*; muns m*1, *m*2. Извне вводится дата в виде двух чисел:номер дня и номер месяца. Проверить, предшествует ли (в рамках года) дате *d*2, *m*2 дата *d*1, *m*1. Вывести соответствующее сообщение.

```
#include <iostream>
#include <windows.h>
using namespace std;
int s;  float t;  int nm1,nm2;
int main()
{    enum muns
{jan=1,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec};
int d1,d2;
muns m1,m2;
char str[256], str1[256];
AnsiToOem("введи день и номер месяца первой даты ", str);
cout<<str; cin>>d1>>nm1;
AnsiToOem(" введи день и номер месяца второй даты ", str);
cout<<str; cin>>d2>>nm2;
   switch (nm1)
   { case 1 :      m1=jan; break;
     case 2 :      m1=feb; break;
     case 3 :      m1=mar; break;
     case 4 :      m1=apr; break;
     case 5 :      m1=may; break;
     case 6 :      m1=jun; break;
     case 7 :      m1=jul; break;
     case 8 :      m1=aug; break;
     case 9 :      m1=sep; break;
     case 10 :     m1=oct; break;
     case 11 :     m1=nov;  break;
     case 12 :    m1=dec;  break; }
    switch (nm2)
   { case 1 :     m2=jan; break;
```

```
      case 2 :     m2=feb; break;
      case 3 :     m2=mar;  break;
      case 4 :     m2=apr; break;
      case 5 :     m2=may; break;
      case 6 :     m2=jun; break;
      case 7 :     m2=jul; break;
      case 8 :     m2=aug; break;
      case 9 :     m2=sep;  break;
      case 10 :    m2=oct; break;
      case 11 :    m2=nov;  break;
      case 12 :    m2=dec; break;};
     AnsiToOem(" предшествует \n", str);
     AnsiToOem("не предшествует \n", str1);
   if (m1<m2) cout<<str; else
   if ((m1==m2)&&(d1<d2))cout<<str; else cout<<str1;
return(0); }
```

12. УКАЗАТЕЛИ

12.1. Понятие об указателях

Указатель – это целочисленная переменная, содержащая адрес (номер ячейки памяти) другой переменной.

Описание:

тип *имя , т.е перед именем указателя ставится знак *

Например, `float a, *aP;`

Так как указатель есть адрес, а адреса распределяет сам компьютер, то указатель не может быть инициализирован непосредственно. Операция *адресации*, т.е. присваивание объявленному указателю адреса переменной, производится, например, следующим образом:

```
float a, *aP;
aP=&a;
```

Знак & перед именем переменной возвращает ее адрес, или, как говорят, дает *ссылку*.

Для того чтобы получить значение переменной, производится операция разыменования. Знак * перед указателем возвращает значение переменной, на которую указатель ссылается. Например,

```
float a1, a=6.0, *aP;
aP=&a; a1=*aP;
```

Здесь в результате `a1` примет значение `6.0`

Операции адресации и разыменования взаимно компенсируют друг друга при их применении к указателю последовательно и в любом порядке:

`&*aP;` эквивалентно `*&aP;`

Пример 12.1. Проведем манипулирование данными с помощью указателей: переменной *a* присваивается значение 5, указателю *aP* присваивается ее адрес, затем по адресу указателю присваивается значение 7. В результате оказывается, что значение переменной *a* изменилось на 7, хотя непосредственно к ней никто не обращался. И далее изменим непосредственно значение *a* на 9 и при обращении к ней, как непосредственно, так и через указатели увидим, что значение изменилось на 9.

```
//  использование указателей
#include <iostream>
#include <windows.h>
using namespace std;
int a, *aP; char str[256];
int main()
{    a=5; cout<<"\n a= "<<a;
     aP=&a;  cout<<"\n *aP= "<<*aP;
     AnsiToOem("\n присваиваем *aP значение 7 ", str);
   cout<<str;
   *aP=7;  cout<<"\n *aP= "<<*aP;  cout<<" a= "<<a;
   AnsiToOem("\n присваиваем a значение 9 ", str);
   cout<<str;
   a=9;    cout<<"\n a= "<<a;  cout<<" *aP= "<<*aP;
   cout<<"\n";
   return (0); }
```

В результате получим (рис. 8.26):

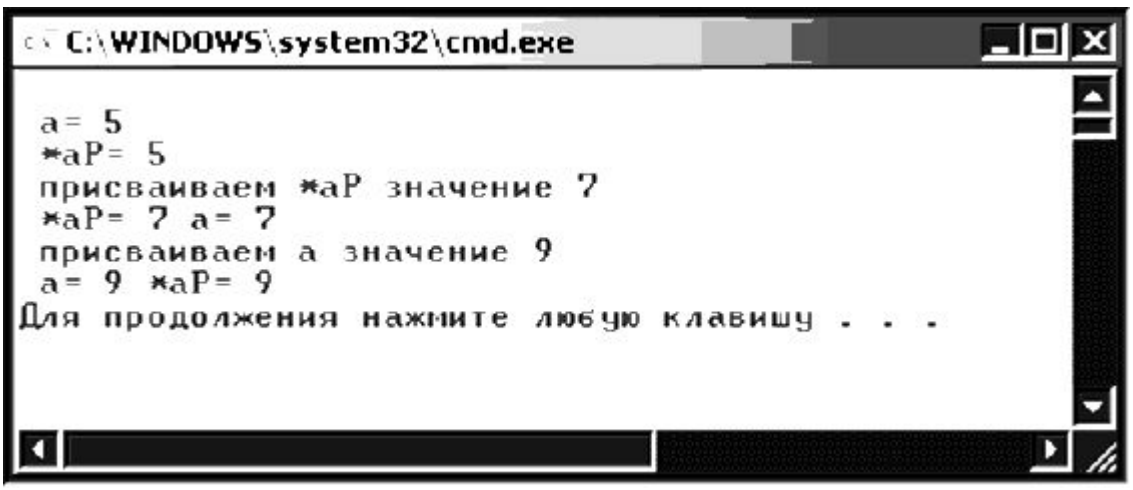


Рис. 8.26. Результат работы программы примера 12.1

12.2. Указатели и функции

Как известно, при обращении к функции параметры, передаваемые функции, являются локальными переменными данной функции. Изменения, внесенные в аргументы во время выполнения функции не влияют на переменные, значения которых передаются функции. (Это одна из причин, почему функция по результатам работы выдает *только одно значение*.)

Проиллюстрируем это на примере.

Пример 12.2. Создать функцию swap(*x*, *y*), которая меняет значения переменных *x*, *y* местами и проверить значения переменных до обращения к функции, внутри функции и после обращения к функции.

```
// передача параметров в функцию (по значению)
#include <iostream>
#include <iomanip>
#include <windows.h>
using namespace std;
char str[256];
void swap(int x, int y);
int main()
{    int x,y;  x=5, y=10;
     AnsiToOem("\n x,y до обращения", str);
     cout<<str <<setw(3)<<x<<setw(3) <<y;  swap(x,y);
     AnsiToOem("\n x,y после обращения", str);
     cout<<str<<setw(3)<<x<<setw(3) <<y<<"\n";
   return(0);}
void swap(int x, int y)
{    int z;
AnsiToOem("\n x,y внутри функции до преобразования", str);
cout<<str<<setw(3)<<x<<setw(3)<<y; z=x; x=y; y=z;
 AnsiToOem("\n x,y внутри функции после преобразования", str);
 cout<<str <<setw(3)<<x<<setw(3)<<y;
    }
```

В результате исполнения программы получим (рис. 8.27):

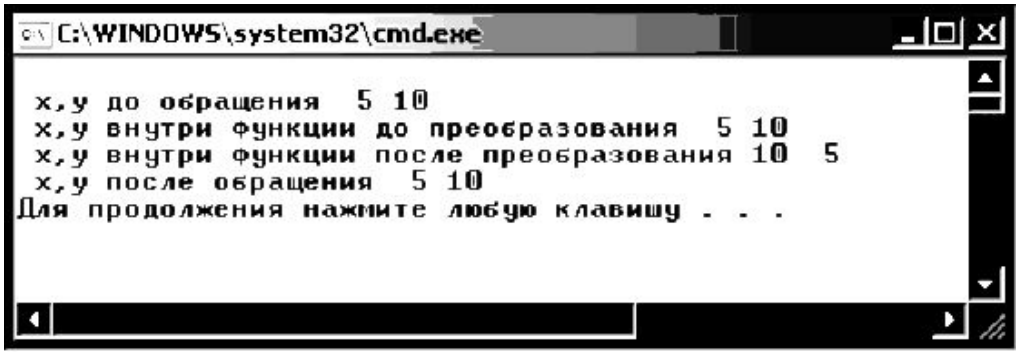


Рис. 8.27. Результат работы программы примера 12.2

Как видим, несмотря на то, что внутри функции значения переменных поменялись, по окончании работы, в основной программе они оказались неизменными.

С данной проблемой можно справиться, используя передачу аргументов по ссылке с помощью указателей. При передаче указателя на самом деле передается лишь адрес объекта, а, следовательно, функция получает возможность манипулировать значением, находящимся по этому адресу.

```
//передача параметров (по ссылке с пом. указателей)
#include <iostream>
#include <iomanip>
#include <windows.h>
using namespace std;
char str[256];
void swap(int *x, int *y);
int main()
{    int x,y;
     x=5, y=10;
     AnsiToOem("\n x,y до обращения ",str);
        cout<<str<<setw(3)<<x<<setw(3) <<y;
     swap(&x,&y);
     AnsiToOem("\n x,y после обращ-я ",str);
        cout<<str<<setw(3)<<x<<setw(3) <<y<<"\n";
   return(0);}
void swap(int *px, int *py)
{int z;
AnsiToOem("\n x,y внутри функции до преобразования",str);
cout<<str <<setw(3)<<*px<<setw(3)<<*py;
z=*px; *px=*py; *py=z;
AnsiToOem("\n x,y внутри функции после преобразования ",str);
cout<<str <<setw(3)<<*px<<setw(3)<<*py;
    }
```

В результате получаем (рис. 8.28):

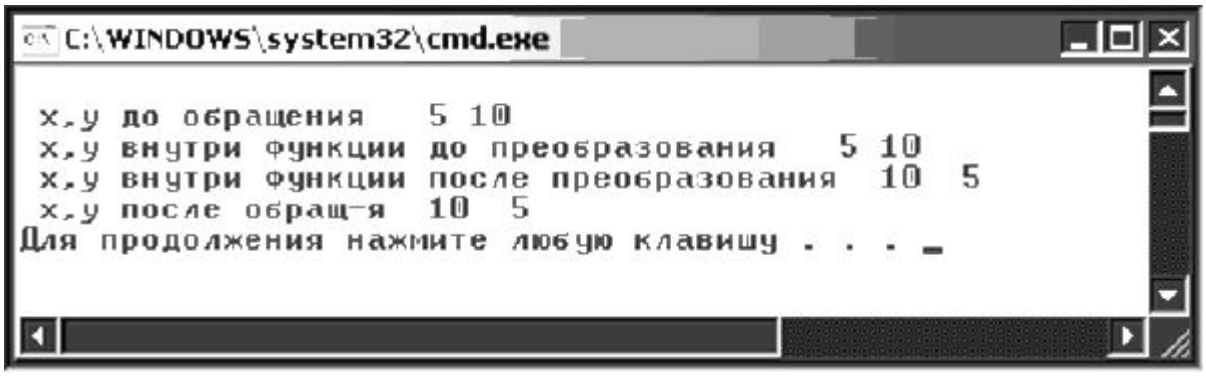


Рис. 8.28. Результат работы примера 12.2 (с указателями)

Но можно в таких случаях обойтись и просто *ссылками*. В этом случае удается скрыть от пользователя функции подробности ее реализации. И к тому же исчезает необходимость многократно возвращать значения указателей.

```
// передача параметров в функцию (по ссылке)
#include <iostream>
#include <iomanip>
#include <windows.h>
using namespace std;
char str[256];
void swap(int &x, int &y);
int main()
{    int x,y; x=5, y=10;
        AnsiToOem(" \n x,y до обращения ",str);
        cout<<str <<setw(3)<<x<<setw(3)<<y;
        swap(x,y);
        AnsiToOem(" \n x,y после обращения ",str);
        cout<<str<<setw(3)<<x<<setw(3) <<y<<"\n";
return(0);}
void swap(int &rx, int &ry)
{    int z;
AnsiToOem("\n x,y внутри функции до преобразования ",str);
        cout<<str <<setw(3) <<rx<<setw(3) <<ry;
        z=rx; rx=ry; ry=z;
AnsiToOem("\n x,y внутри функции после преобразования ",str);
        cout<<str <<setw(3)<<rx <<setw(3) <<ry;  }
```

В результате имеем (рис. 8.29):

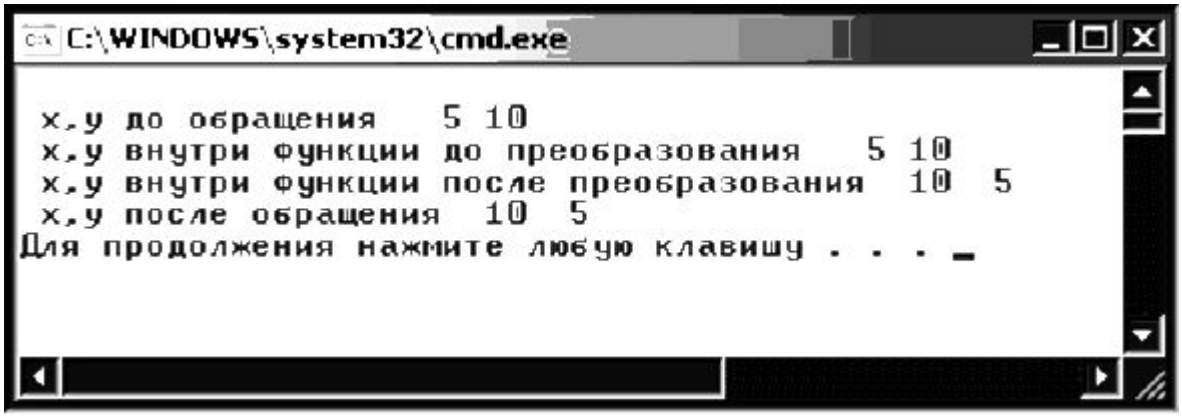


Рис. 8.29. Результат работы программы примера 12.2
(с использованием ссылок)

12.3. Указатели и динамические массивы

Если до начала разработки программы неизвестно, сколько в массиве элементов, в программе следует использовать *динамические массивы*. Память под них выделяется с помощью операции **new** в динамической области памяти во время выполнения программы. Адрес начала массива хранится в переменной, называемой указателем.

Например,

```
int n=10;
    int *a = new int[n]
```

Заметим, что обнуление памяти при ее выделении не происходит. Инициализировать динамический массив нельзя. Обратится к элементу динамического массива можно как обычно `a[3]`, так и с использованием указателя `*(a+3)`. Дело в том, что в переменной указателе хранится адрес начала массива. Для получения адреса третьего элемента к этому адресу прибавляется смещение 3. Операция сложения с константой для указателей учитывает размер адресуемых элементов, т.е. на самом деле индекс умножается на длину элемента массива:

```
a + 3*sizeof(int)
```

Затем с помощью операции * (разадресации) выполняется выборка значения из указанной области памяти.

Если динамический массив в какой-то момент работы программы перестает быть нужным, и мы собираемся впоследствии использовать эту память повторно, необходимо освободить ее с помощью операции `delete[]`, например

```
delete [] a;
```

размерность массива при этом не указывается.

Пример 12.3. Рассчитать сумму элементов правее последнего отрицательного.

```
// сумма элементов правее последнего отрицательного
// элемента (с использованием динамич. массива)
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{ int  n;
  char str[256], str1[256];
  AnsiToOem("введи размерность массива ",str);
```

```
     cout<<str; cin>>n;
     float *a=new float[n];
     int i, notr;
     AnsiToOem("вводите элементы массива ",str);
     cout<<str;
     AnsiToOem("\n  сумма= ",str);
     AnsiToOem("\n отрицательных нет ",str1);
     for (i=0; i<n; i++) cin>>a[i];
     notr=-1;
      for (i=0; i<n; i++) if (a[i]<0) notr=i;
      if (notr!=-1)
     {float sum=0;
      for (i=notr+1;i<n;i++)sum=sum+a[i];
      cout<<str<<sum;}
     else cout<<str1;
      cout<<"\n";
return(0);
}
```

12.4. Указатели и перегрузка операций

Если мы вводим данные нового типа (таковым является, например, перечислимый тип), то желательно сделать так, чтобы стандартные операции для них имели тот же внешний вид. Такого рода переобозначение операций (для новых типов данных) называют *перегрузкой операций*. Для уяснения этого вспомните также, как мы ввели перегрузку функций (см. раздел 10.1). Общая форма для перегрузки операций имеет вид:

***тип* operator@ (*параметр*)**

Здесь ***тип*** – нужный тип переменной, **operator** – служебное слово, **@** – символическое обозначение нужной операции.

Указатели могут быть использованы и для *перегрузки операций*. Вспомним, пример 11.1, где мы не смогли использовать операцию инкремента. Достаточно ввести перегрузку для операции ‘**++**’ для перечислимого типа и проблема исчезнет. Такая перегрузка имеет вид:

```
enum dni{Monday=1, Tuesday, Wednesday, Thursday, Friday,
Saturday, Sunday} ;
dni operator++(dni &m)
{ return (m=dni(m+1));}
```

Тогда окончательно текст программы примера 11.1 примет вид:

```
// перечислимый тип - часы рабочей недели
#include <iostream>
```

```
#include <windows.h>
using namespace std;
float t;
enum dni{Monday=1, Tuesday, Wednesday, Thursday, Friday, Satur-
day, Sunday} ;
dni operator++(dni &m)
{ return (m=dni(m+1));}
int main()
{dni d;
     t=0;
     char str[256];
     AnsiToOem(" сумма часов раб.недели=", str);
  for (d=Monday; d<=Friday; d++)
     { if (d == Friday) t=t+8; else t=t+8.25; } ;
   cout << str <<t<<"\n";
return(0); }
```

13. ОБРАБОТКА СИМВОЛЬНЫХ СТРОК

13.1. Символьные переменные

Напомним, что для описания символьных переменных служит тип `char`, соответственно массив символов должен при описании рядом с именем иметь в квадратных скобках указание на размер.

Но для начала рассмотрим работу просто с отдельными символами. Значение символьной переменной может быть задано как с помощью команды присваивания (обратите внимание, что отдельный символ заключается в апострофы), так и с помощью команды ввода `cin`.

Поскольку мы уже освоили работу с функциями и указателями, то для организации вывода русских сообщений можем разработать отдельную функцию символьного ссылочного типа

```
char* Rus(const char* str)
{    AnsiToOem(str,str_rus);
return str_rus;}
```

У этой функции на входе задается символьная константа `str` (тот самый русский текст), которая внутри преобразуется с помощью той самой функции `AnsiToOem` и преобразованный текст помещается в `str_rus`. Благодаря этому, выводить русский текст в основной программе можно просто вызовом этой функции (см. пример 13.1)

Пример 13.1. Среди последовательно вводимых N символов подсчитать число восклицательных знаков.

```
// подсчет числа символов !
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
int main()
{    char c,c1; int sum=0,i,n;
     c1='!'; cout<<Rus(" \n введи значение N \n");
        cin>>n;
     for (i=1; i<=n; i++)
        {cout<<Rus(" \n введи очередной символ "); cin>>c;
          if (c==c1) sum++; }
     cout<<Rus(" количество знаков ")<<c1;
     cout<<Rus(" равно ")<<sum<<"\n";
   return (0); }
char* Rus(const char* str)
{    AnsiToOem(str,str_rus);
     return str_rus;}
```

Символы в C++ также имеют коды. Для получения цифрового кода символа достаточно применить операцию перевода в целые: `(int) c`, где `c` – символьная переменная. Для получения по коду самого символа достаточно применить операцию по переводу целых в символьные `(char) k`, где `k` – целое число (код символа).

В рассмотренном ниже примере будем также использовать датчик случайных чисел, который мы уже использовали в разделе 8. Напомним, что на языке C++ он реализуется функцией `rand()%RAND_MAX` – дает случайное целое число из интервала от `0` до положительного целого `RAND_MAX-1` (в качестве `RAND_MAX` можно указать сразу конкретное число). Кроме того, для запуска этого датчика случайных чисел (чтобы получать новый набор случайных чисел при каждом новом запуске программы), используют функцию `srand(time(0))`, т.е. для старта генератора случайных чисел будет использоваться системное время компьютера. Причем это время следует преобразовать в беззнаковое целое (`unsigned int`). Кроме того, требуется подключить заголовочный файл `<stdlib.h>`, содержащий эти функции. Для получения системного времени нужно подключить заголовочный файл `<time.h>`

Наиболее употребительные символы имеют коды в интервале от 33 до 122, поэтому в программе предусмотрено дополнительно ограничение на получение случайных чисел из интервала от 33 до 123.

Здесь мы также впервые использует команду **goto *метка***. Метка в C++ задается как и любой идентификатор (название метки начинается с буквы) и не требует описания (в отличие от Паскаля). Признаком метки является двоеточие сразу после нее.

Пример 13.2. Составить программу «клавиатурный тренажер» (случайным образом выводится символ, который следует нажать на клавиатуре).

```
// клавиатурный тренажер (распознание символа по коду)
#include <iostream>
#include <stdlib.h>
#include <windows.h>
#include <time.h>
using namespace std;
char* Rus (const char* str);
char str_rus[256];
int kod,n, r_max, r_min; char c,d;
int main ()
{srand( (unsigned)time(0) );
One:
r_min=33; r_max=123;
kod=(double)rand()/(RAND_MAX+1)*(r_max-r_min)+r_min;
c=(char)kod; cout<<Rus(" нажми ")<<c<<"\n";
cin>>d;
if ((int)d==kod) cout<<Rus(" правильно \n ");
        else cout<<Rus(" нет \n");
goto One;
return (0);}
char* Rus (const char* str)
{    AnsiToOem(str, str_rus);
return str_rus;}
```

Данная программа будет работать непрерывно («зациклили» оператором перехода `goto One`). Чтобы прервать ее работу достаточно нажать **Ctrl-C**.

Замечание. Коды символов *кириллицы* задаются в C++ отрицательными числами!

Для демонстрации кодов русских букв рекомендуем набрать следующую программку (см. пример 13.3).

13.2. Символьные строки (как массивы символов)

К сожалению, в C++ отсутствует тип «строка» (хотя для создания строк здесь имеются более мощные средства). Мы сначала будем рассматривать строки как массивы символов.

Пример 13.3. Распечатать коды русских букв (предварительно записав их в виде массива символов).

```
// вывод кодов русских букв
#include <iostream>
#include <iomanip>
#include <windows.h>
using namespace std;
char* Rus (const char* str);
char str_rus[256],salf[65],sal;
int main()
{  int i; char rusalf[65]=
     "АаБбВвГгДдЕеЁёЖжЗзИиЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦцЧчШшЩщЪъЫыЬьЭэ
ЮюЯя";
     cout<< Rus(" \n КОДЫ РУССКИХ БУКВ ")<<"\n";
     AnsiToOem(rusalf, salf);
     for (i=0; i<=64; i++)
// для вывода  ровными столбиками по 8 применяется if
   { sal=salf[i];
       cout<<setw(5)<<sal<<setw(3)<<(int)rusalf[i];
          if (i%8 == 7)cout<<"\n";}
return (0);}
char* Rus (const char* str)
{    AnsiToOem(str, str_rus);
return str_rus;}
```

При запуске программки получим (рис. 8.30):

Рис. 8.30. Результат работы программы примера 13.3

В рассмотренном примере строка текста (в виде массива символов) заполнялась внутри программки путем присваивания. Значительно важнее было бы научиться *вводить значение строки извне* с помощью той же команды `cin`. К сожалению, если при вводе в строке будет пробел, то `cin` воспримет его как конец строки и остановит запись в буфер. Кроме того, если пользователь введет больше символов, чем описано, то оператор `cin` отбросит последние символы. Для преодоления этого организуют так называемый *метод* `cin.get()`. В нем указывается строка для ввода и ее максимальная длина. В данном случае указываем 79, хотя строка описана на 80 символов потому, что символы на самом деле нумеруются с 0 и последний символ массива всегда «\0» (null).

Пример 13.4. Введенную строку текста разбить на отдельные слова, т.е. вывести каждое слово в отдельную строку.

```
// ввод строки текста и разбиение на слова
#include <iostream>
#include <windows.h>
#include <string>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
int main()
{char s[80];int n,i;
 cout <<Rus("\n введи строку текста \n");
 cin.get(s,79);
 n=strlen(s);cout<<Rus("\n текст разбит на слова:\n");
 for (i=0; i<n; i++)
 { if (s[i] ==' ') cout<<"\n"; else cout<<s[i]; }
 cout<<"\n";
return (0);  }
char* Rus(const char* str)
{AnsiToOem(str, str_rus);
return str_rus;}
```

Здесь мы использовали специальную функцию `strlen(s)`, которая позволяет получить реальную длину строки `s` (сколько именно символов было введено).

13.3. Обработка массивов строк

В некоторых случаях для ввода строк правильнее использовать метод `cin.getline(s,n)`. Отличия его от метода `cin.get(s,n)` в следующем:

- Метод **getline** считывает из входного потока n-1 символов или менее (если символ перевода строки встретится раньше) и записывает их в строковую переменную S.
- Символ перевода строки (\n встречается, когда нажимает Enter) также считывается (удаляется) из входного потока, но не записывается в строковую переменную, вместо него размещается завершающий 0.
- Если в строке исходных данных более n-1 символов. Следующий ввод будет выполняться из той же строки, начиная с первого несчитанного символа
- Метод **get** работает аналогично, но оставляет в потоке символ перевода строки. В строковую переменную добавляется завершающий 0.

Для обработки строк существует ряд специальных функций:

- функция **strcpy**(s1,s2) – она копирует в строку S1 содержимое строки S2
- функция **strcmp**(s1,s2) – она сравнивает содержимое строк s1 и s2: если они равны, то функция дает 0, если s1<s2, то отрицательное число и если s1>s2, то положительное
- функция **_itoa**(I,s,r) – преобразует целое I в строку s, пользуясь системой счисления с основанием r

Все эти функции находятся в заголовочном файле **string.h .**

Пример 13.5. Дан список из 10 фамилий. Найти однофамильцев и, если они есть, указать их порядковые номера в списке.

```
/ поиск однофамильцев в списке из 10 фамилий
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
const int n=10; int i,j, k;
char fam [n+1] [20], s[20],s2[20],s3[20];
int main()
```

```
{for (i=1;i<=n;i++)
 { cout<<Rus("введи фамилию № ")<<i<<" ";
   cin.getline(s,19); strcpy(fam[i],s);}  k=0;
 for (i=1;i<n;i++)
 {  strcpy(s,fam[i]);
  for (j=i+1;j<=n;j++)
  {  strcpy(s2,fam[j]);
     if (strcmp(s, s2)==0)   {  cout<<s;
k++; _itoa(k,s3,2); strcpy(fam[j],s3); //затирание уже найденных
однофамильцев
      cout<<" "<<i<<" "<<j;cout<<endl;  } }}
return(0);}
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus;}
```

Результат работы программы на рис. 8.31.

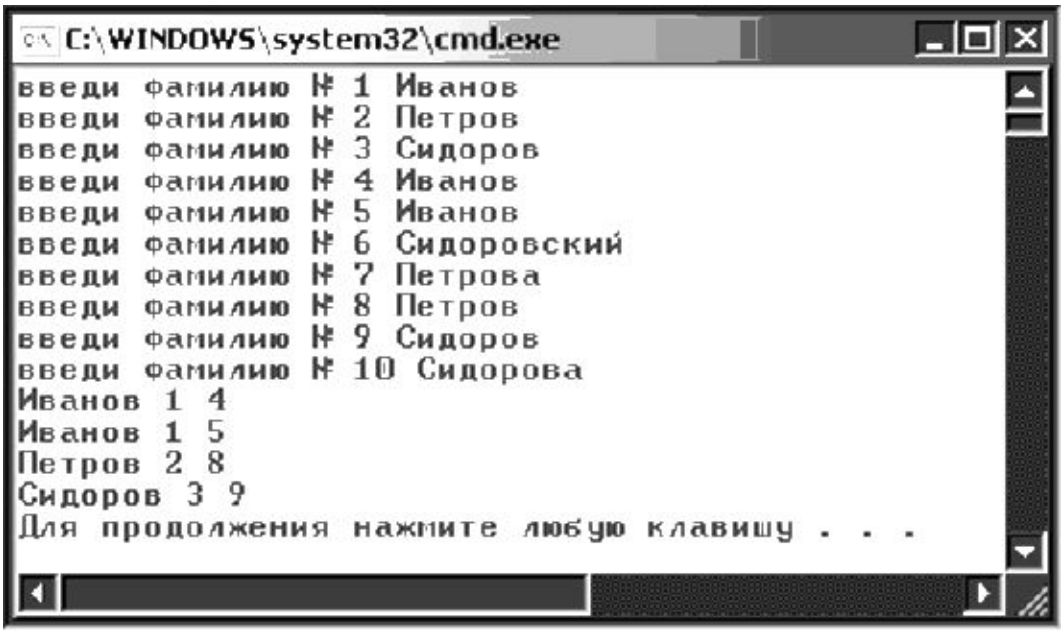


Рис. 8.31. Результат работы программы примера 13.5

Рассмотрим, как реализуется задача сортировки в массиве строк.

Пример 13.6. Дан список из 10 фамилий. Упорядочить их по алфавиту.

```
// упорядочение фамилий в списке из 10 фамилий
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
```

```
char str_rus[256];
const int n=10; int i,j, k; char fam [n+1] [20],
s[20],s2[20],s3[20];
int main()
{for (i=1;i<=n;i++)
{cout<<Rus("введи фамилию №")<<i<<" ";
 cin.getline(s,19);
 strcpy(fam[i],s);
 }  k=0;
 for (i=1;i<n;i++)
 { for (j=1;j<n;j++)
   {  strcpy(s,fam[j]);
     strcpy(s2,fam[j+1]);
          if (strcmp(s, s2)>0)
    { strcpy(s3,fam[j]);
strcpy(fam[j],fam[j+1]);strcpy(fam[j+1],s3);
    } }}
  cout<<"\n";
  for (i=1;i<=n;i++)
{cout<<Rus("теперь фамилия №")<<i<<" "<<fam[i]<<endl;}
return(0); }
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus;
}
```

Результат работы на рис. 8.32.

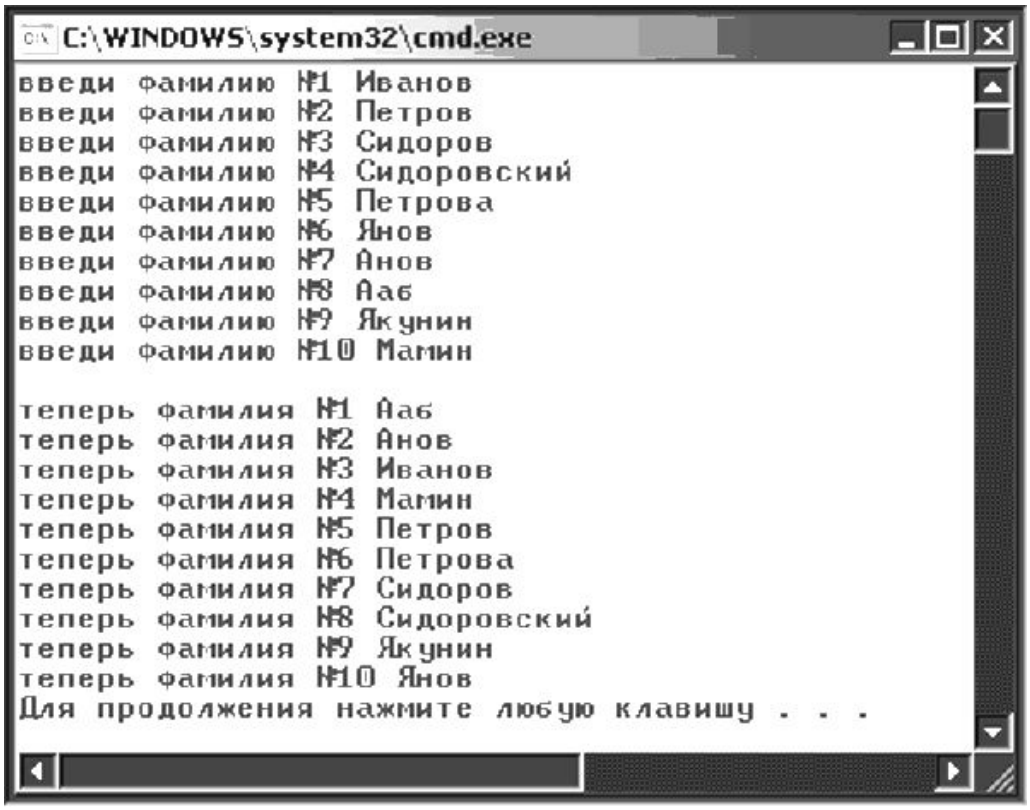


Рис. 8.32. Результат работы программы примера 13.6

Пример 13.7. Дан список из 10 фамилий. Упорядочить их по алфавиту, используя динамические массивы.

```
//алфавитное упорядочение фамилий с использованием
// динамического массива
#include <iostream>
#include <string.h>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
int main()
{ int  i,j,n,m; char s[20],s2[20], s3[20];
     cout<<Rus("введите количество фамилий "); cin>>n;
//объявление переменной типа "указатель на указатель" и
// выделение памяти под массив указателей на число фамилий
     char **fam = new char* [n+1];
     cout<<Rus("введи макс. длину фамилии "); cin>>m;
// ниже - цикл для выделения памяти под каждую фамилию
// на максимально указанную длину фамилии
     for (i=1;i<=n;i++) fam[i]= new char[m];
//цикл ввода фамилий
     for (i=1; i<=n; i++){ cout<<Rus("введи ")<<i;
     cout<<Rus("-ю фамилию "); cin>>fam[i];}
// упорядочение фамилий путем сравнения соседних
// элементов и перестановки их
     for (i=1; i<n; i++)  for (j=1; j<n; j++)
     { if (strcmp(fam[j],fam[j+1])>0)
          {strcpy(s3,fam[j]); strcpy(fam[j],fam[j+1]);
      strcpy(fam[j+1],s3);}    }
for (i=1;i<=n; i++)
cout<<Rus("теперь фам №")<<i<<" "<<fam[i]<<endl;
return(0); }
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);  return str_rus; }
```

Результат работы на рис. 8.33.

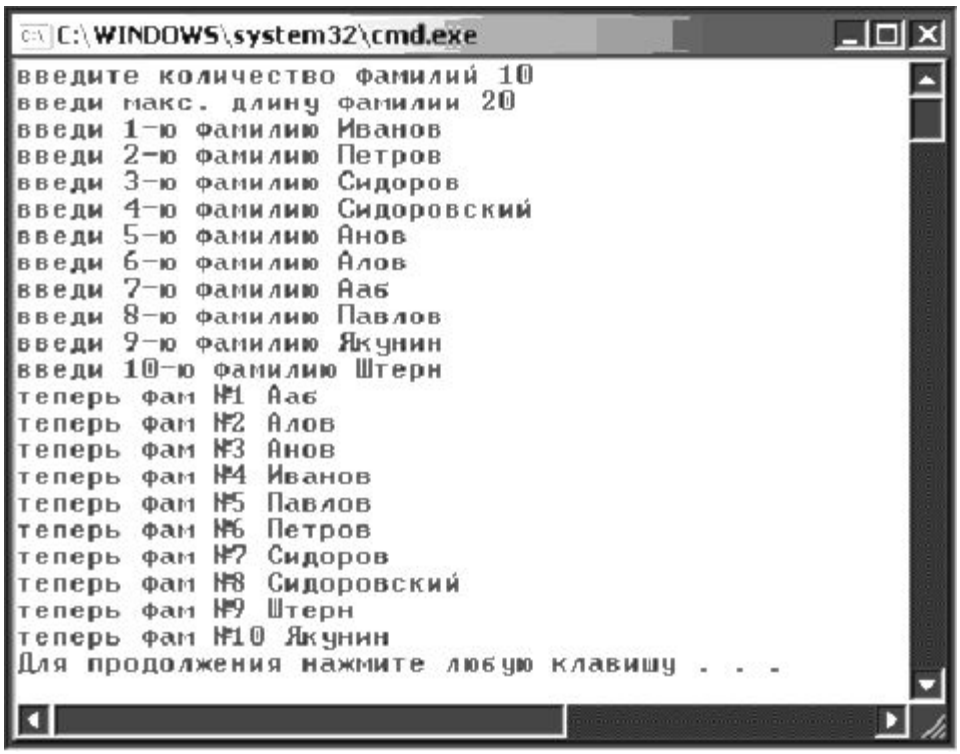


Рис. 8.33. Результат работы программы примера 13.7

Практические задания

13.1. В символьную переменную вводится цифра. Вывести следующую и предыдущую цифры, считая, что за 9 следует 0, а, соответственно, нулю предшествует девятка.

13.2. Вывести в одну строку нечетные (по порядковому номеру) буквы латинского алфавита: *a c e g ...*

13.3. Дан текст, заканчивающийся точкой. Является ли этот текст правильной записью целого числа (возможно, со знаком).

13.4. Дан текст, заканчивающийся точкой. Напечатать этот текст, удалив из него все цифры и знаки «+» или «–».

13.5. Составить программу, которая определит, является ли заданное слово перевертышем (например, «кок», «шалаш» являются).

13.6. Составить программу, которая проверяет правописание «жи – ши» (т.е. если в заданном тексте после «ш» или «ж» встретится «ы», то программа должна выдавать сообщение об ошибке).

13.7. Составить программу, которая проверяет правописание «ча–ща» (т.е. если в заданном тексте после «ч» или «щ» встретится «я», то программа должна выдавать сообщение об ошибке).

13.8. Составить программу, которая введенное слово напечатает в обратном порядке (например, слово «упал» превратится в «лапу»).

13.9. Составить программу, которая во введенном слове удалит каждую вторую букву (т.е., например, для слова «символ» напечатает «смо»).

13.10. Составить программу, которая во введенном слове удалит заданную букву (например, если задано слово «трактор» и буква «т», то получится «ракор»).

13.11. Написать программу, которая в введенной с клавиатуры строке преобразует строчные (малые) буквы русского алфавита в прописные (заглавные).

13.12. Написать программу, которая проверяет, является ли введенная с клавиатуры строка двоичным числом.

13.13. Написать программу, которая проверяет, является ли введенная с клавиатуры строка 16-тиричным числом.

13.14. Написать программу, которая проверяет, является ли введенная с клавиатуры строка дробным числом.

В пяти последующих задачах одинаковое условие: Дана строка из *N* символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами.

13.15. Подсчитать количество слов в данной последовательности.

13.16. Найти количество слов, начинающихся с буквы б.

13.17. Найти количество слов, оканчивающихся на букву а.

13.18. Найти количество слов, у которых первый и последний символы совпадают между собой.

13.19. Преобразовать данную последовательность, заменяя всякое вхождение слова это на слово то.

13.20. Найти длину самого короткого слова.

13.21. Написать программу, которая в словах с дефисом меняет местами части до и после дефиса.

13.22. Написать программу вывода на экран списка символов, из которых образован заданный текст.

13.23. Написать программу, определяющую в заданном тексте для каждой буквы алфавита количество ее употребления.

13.24. Написать программу, определяющую в заданном тексте для каждой буквы алфавита частоту ее употребления (отношение количества употреблений буквы к количеству всех букв в тексте)

13.25. Дан список из слов различной длины. Составить программу упорядочения слов по их длине.

14. СТРУКТУРЫ

Одна из главных особенностей языка C++ – это возможность создавать данные новых типов. Причем, данные нового типа могут быть *комбинацией данных различного типа*. Такие данные называют *структурами*.

Описание структуры делается так**:**

Struct ***имя структуры***
{ ***тип и имя элемента структуры*** 1***;***
тип и имя элемента структуры 2***;***
…
тип и имя элемента структуры 2};

А затем новый структурный тип, как обычно, можно использовать для объявления новых переменных. Доступ к отдельным элементам структуры осуществляется операцией **.** (точка) для объекта и операцией **–>** (стрелка) для указателя на объект.

Например, мы создадим структуру типа **date**:

```
struct  date
{int day;
 enum muns {jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec};
 int year; }
```

И затем объявим переменную **den** типа **date**. Тогда элементы такой структурной переменной будут обозначаться: **den.day, den.muns, den.year**.

Пример 14.1. Даны два класса учащихся. Определить, являются ли они параллельными.

```
#include <iostream>
#include <iomanip>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
int main()
{struct clas {int god;
                              char bukwa;};
 clas x,y;
 cout<<Rus("\n введи номера классов ");
 cin>>x.god>>y.god;
 cout<<Rus("\n введи буквы классов ");
 cin>>x.bukwa>>y.bukwa;
  if ((x.god == y.god) && (x.bukwa != y.bukwa))
```

```
     cout <<Rus(" ЯВЛЯЕТСЯ ПАРАЛЛЕЛЬНЫМИ ");
       else cout <<Rus(" не параллельны ") ;
     cout<<"\n"; return 0;}
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus;}
```

Заметим, что в качестве элемента структуры может быть в свою очередь структура. Кроме того, из структур можно формировать и массивы.

Пример 14.2. Список учащихся состоит из 10 записей. Каждая запись содержит фамилию и дату рождения ученика. Найти тех, кто родился в один и тот же день (год рождения может и отличаться).

```
// ученики, родившиеся в один день
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
struct date {int day;
          int muns;
          int year;};
struct child {char fam[15];
          date den; };
int main()
{ child spis[11]; const int n=10; int i,k;
for (i=1; i<=n; i++)
{ cout<<Rus("\n введи фамилию ученика ")<<i<<' ';
cin>> spis[i].fam;
cout<<Rus("введи день, месяц и год рождения ученика ") <<i<<" ";
cin>>spis[i].den.day>>spis[i].den.muns>>spis[i].den.year;};
cout<<Rus("\n РОДИЛИСЬ В ОДИН ДЕНЬ \n");
for (i=1; i<=n-1; i++)
for (k=i+1; k<=n; k++)
if  ((spis[i].den.day==spis[k].den.day) &&
     (spis[i].den.muns==spis[k].den.muns))
{cout<<spis[i].den.day <<" "<<spis[i].den.muns<<"   ";
 cout<<spis[i].fam<<"  "<< spis[k].fam; cout<<"\n";}
return 0; }
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus;}
```

При запуске получим примерно такой результат (рис. 8.34).

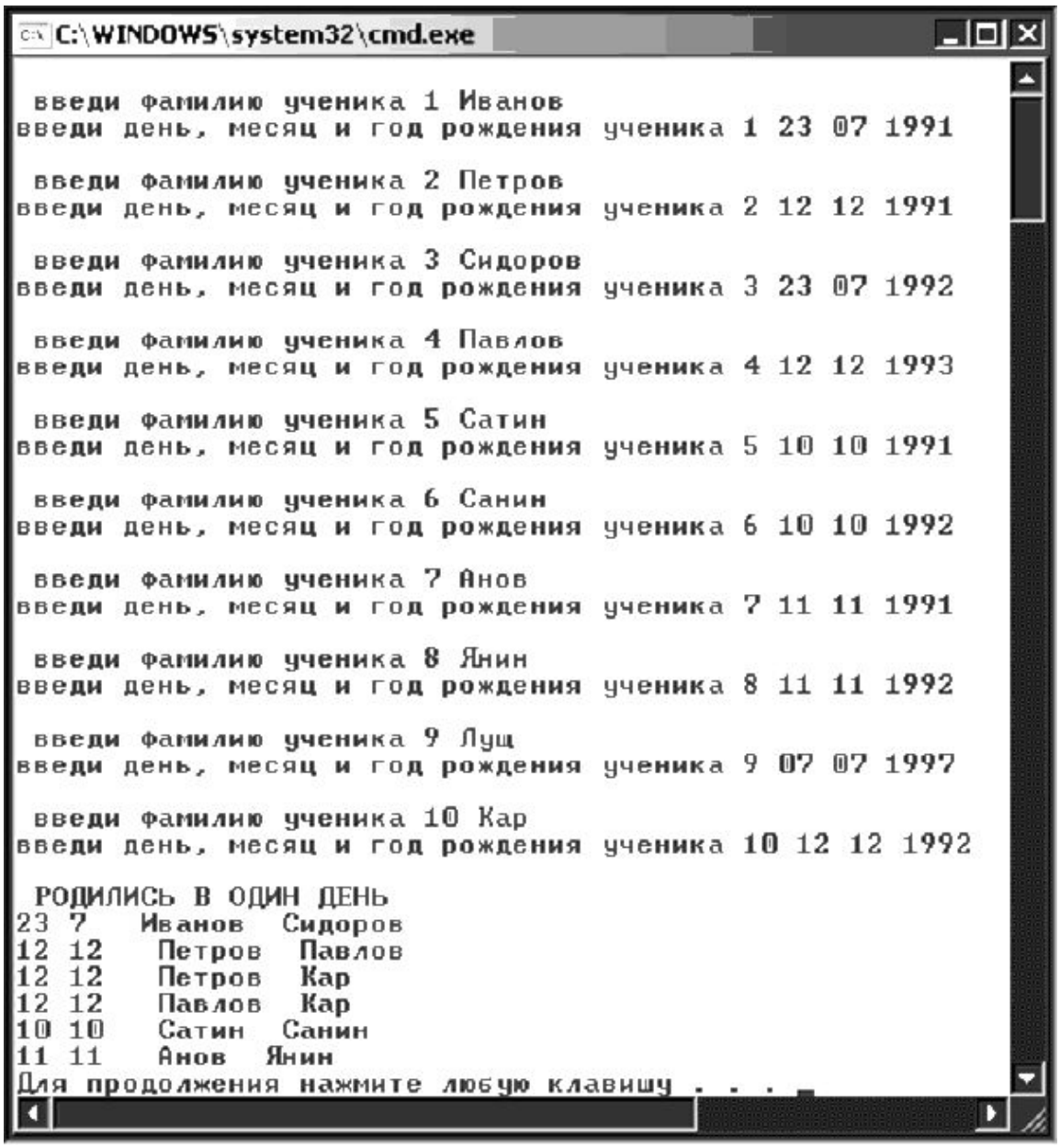


Рис. 8.34. Результат работы программы примера 14.2

В качестве одного из элементов структуры может быть *массив*. Это продемонстрируем на следующем примере. Кроме того, над структурными переменными можно выполнять операции как над едиными объектами (присваивать одной структурной переменной другую, сравнивать их и т.д.), но только в том случае, если структурные переменные однотипны, т.е. описаны на основе одного структурного типа.

Пример 14.3. Список учащихся состоит из 10 записей. Каждая запись содержит фамилию ученика, класс и поля – оценки по семи предметам. Вывести фамилии и класс тех учеников, кто имеет пятерки по всем предметам и сформировать из них отдельный список – список отличников.

```
// поиск отличников в списке
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
     struct clas{   int god;
                                    char bukwa;};
      struct chil {  char fam[15];
                              clas cl;
                     int ocenky[8];};
     chil spis[11], spisotl[11];  int k,s,i,j;
int main()
{for (i=1; i<=10; i++)
{cout<<Rus("\n введи фамилию ученика, номер и букву его класса");
 cin>>spis[i].fam >> spis[i].cl.god >> spis[i].cl.bukwa;
  for (j=1; j<=7; j++)
 {cout<<Rus("введи оценку ")<<j<<" ";
  cin>>spis[i].ocenky[j];};  cout<<"i="<<i; };
k=0;
for (i=1; i<=10; i++)
{ s=0;
  for (j=1; j<=7; j++) if (spis[i].ocenky[j]==5)  s=s+1;
  if (s==7) { k=k+1; spisotl[k]=spis[i]; } };
cout<<Rus("\n СПИСОК ОТЛИЧНИКОВ ");
for (i=1; i<=k; i++)
{cout<<"\n"<<spisotl[i].fam<<" "<<spisotl[i].cl.god<<" "
   <<spisotl[i].cl.bukwa;}
return 0;}
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus;
}
```

Практические задания

В задачах 14.1–14.4 одинаковое начало условия задачи:

Дан список учащихся из 10 записей. Каждая запись имеет поля: фамилия, имя, номер класса, буква класса.

14.1. Найти однофамильцев, обучающихся в одном классе.

14.2. Найти тезок (одинаковые имена), обучающихся в одном классе.

14.3. Найти двух учащихся, у которых совпадают имя и фамилия.

14.4. Вывести фамилию и первую букву имени для всех учеников указанного извне класса.

В задачах 14.5–14.8 одинаковое начало условия задачи:

Багаж пассажира характеризуется количеством вещей (целый тип) и общим весом вещей (вещественный тип). Дан список из сведений о багаже 10 пассажиров.

14.5. Найти багаж, средний вес одной вещи, в котором отличается не более, чем на 0,3 кг от общего среднего веса одной вещи по всему списку.

14.6. Найти число пассажиров, имеющих более двух вещей.

14.7. Число пассажиров, количество вещей которых превосходит среднее число вещей по всему списку.

14.8. Выяснить, имеется ли пассажир, багаж которого состоит из одной вещи весом менее 30 кг.

В задачах 14.9–14.10 одинаковое начало условия задачи:

Список книг состоит из 10 записей. Запись содержит поля: фамилия автора (тип string), название книги (тип строка), год издания (тип целый).

14.9. Найти названия книг данного автора, изданных с 1960 г.

14.10. Определить, имеются ли книги с названием «Информатика» и если да, то сообщить фамилии авторов, год издания этих книг.

В задачах 14.11–14.16 одинаковое начало условия задачи:

Дана структура, задающая дату вида:

```
Struct date {int day;
             int month;
             int year;};
```

Пользуясь таким структурным типом, составить программу, определяющую:

14.11. дату следующего (относительно сегодняшнего) дня;

14.12. дату предыдущего дня;

14.13. дату, которая наступит через m дней;

14.14. дату, которая была за m дней до сегодняшнего дня;

14.15. число суток, прошедших от заданной даты d1 до даты d2;

14.16. день недели, выпадающий на дату *d*1, если известно, что в первый день нашей эры был понедельник;

В задачах 14.17–14.18 одинаковое начало условия задачи:

Дана структура, задающая время, вида:

```
struct time { int hour;
   int min;
   int sec;};
```

Пользуясь таким структурным типом, составить программу:

14.17. определяющую предшествует ли время t1 времени t2 (в пределах суток);

14.18. присваивающую праметру t1 время, на 1 сек большее времени t (учесть смену суток)

В задачах 14.19–14.20 одинаковое начало условия задачи:

Дан список учащихся из 10 записей. Каждая запись имеет поля: фамилия, имя, номер класса (только 8-е и 9-е классы)

14.19. вывести отдельно учеников 9-х классов;

14.20. выяснить, на сколько человек в 8-х классах больше, чем в 9-х

В задачах 14.21–14.25 одинаковое начало условия задачи:

Дан список учащихся из 10 записей. Каждая запись имеет поля: фамилия, имя, номер класса, оценки по трем предметам.

14.21. вывести отдельно отличников для указанного извне класса;

14.22. вывести отдельно двоечников (хотя бы одна двойка) для указанного извне класса;

14.23. вывести отдельно «хорошистов» (оценки 4 и 5) для указанного извне класса;

14.24. вывести отдельно «троечников» (имеется хотя бы одна тройка, но нет двоек) для указанного извне класса:

14.25. вывести список всех учеников с указанием среднего балла для каждого.

15. КЛАССЫ

15.1. Понятие класса

Фундаментальное нововведение языка C++ – понятие *класс*. Если структура объединяет разнородные по типам переменные, то ***класс*** *– объединяет переменные и функции*, *предназначенные для работы с ними*. Здесь, пожалуй, впервые мы сталкиваемся с элементами объектно-ориентированного программирования. Можно сказать, что класс – это нового типа объект, который имеет *свойства* (переменные-члены класса) и *методы* (функции-члены класса). Определив класс, мы потом можем, как обычно, объявить определенный *экземпляр*, т.е. описать объект на основе заданного класса. Объявление класса производится с помощью служебного слова

class с указанием имени и перечислением его членов в фигурных скобках. Например,

```
class DayOfYear
     {public:
        void output();// прототип функции-члена
               int month;
        int day; };
```

Обратите внимание, что при описании указывается только прототип функции, а саму функцию надо затем описать с указанием имени класса и через :: имя самой функции :

```
void dayOfYear :: output()
{cout << " месяц " << month;
 cout << " день " << day << endl;   }
```

Впрочем, допускается описание функции-члена тут же, при описании класса.

Обращение в программе к членам класса делается также через точку (подобно тому, как это делалось для элементов структуры).

Отметим, что члены класса могут быть *открытыми* (public) и *закрытыми* (private). При этом закрытые члены могут использоваться только другими элементами класса, а к открытым элементам класса разрешен доступ из программы через объекты соответствующего типа (так же, как к элементам структуры – через точку). При попытке вызвать из программы закрытый элемент класса, компилятор укажет на ошибку. Заметим, что по умолчанию члены класса являются закрытыми (private).

Пример 15.1. Создать программу, которая запрашивает у пользователя сегодняшнее число и дату дня рождения, а затем сообщает: «Поздравляем с днем рождения!», если дата совпадает с днем рождения или просто сообщает «Счастливого дня!», если сегодня не ваш день рождения.

```
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
class DayOfYear
     {  public:
     void output(); // прототип функции-члена
     int month;
     int day;
     };
int main()
```

```
{  DayOfYear today, birthday;
   cout <<Rus(" Введите текущую дату: \n");
   cout << Rus(" ВВЕДИТЕ МЕСЯЦ: ");
   cin >> today.month;
   cout << Rus(" ВВЕДИТЕ ДЕНЬ МЕСЯЦА: ");
   cin >> today.day;
   cout << Rus(" ВВЕДИТЕ ВАШ ДЕНЬ РОЖДЕНИЯ: \n");
   cout << Rus(" ВВЕДИТЕ МЕСЯЦ: ");
   cin >> birthday.month;
   cout << Rus(" ВВЕДИТЕ ДЕНЬ МЕСЯЦА: ");
   cin >> birthday.day;
   cout << Rus("СЕГОДНЯ ");
   today.output();
   cout << Rus("ВАШ ДЕНЬ РОЖДЕНИЯ  ");
   birthday.output();
   if ((today.month = = birthday.month)
     && (today.day == birthday.day) )
      cout << Rus(" ПОЗРАВЛЯЕМ С ДНЕМ РОЖДЕНИЯ ! \n");
      else cout << Rus("ЖЕЛАЕМ УДАЧНОГО ДНЯ \n");
   return 0; }
void DayOfYear :: output()
     {    cout << Rus(" МЕСЯЦ ") << month;
          cout << Rus(" ДЕНЬ " )<< day << endl;    }
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus; }
```

15.2. Открытые и закрытые члены класса

Рассмотренный пример реализован с открытыми членами класса. Закрытые члены могут понадобиться, когда мы захотим создать более универсальный класс – такой, чтобы при изменении описания элементов класса, изменения в основной программе не вносились. Например, если мы изменим описание элемента **month** с целого **int** на строковый **char(3)** – месяц обозначается тремя буквами – тогда переменная-член **month** исчезнет и нужно менять, по сути, всю программу. После объявления переменной-члена закрытой, не существует другого способа изменить ее значения (или обратиться к нему), кроме использования одной из предназначенных для этого функций-членов.

Реализуем тот же пример, но с использованием закрытых членов.

Пример 15.2. Создать программу, которая запрашивает у пользователя сегодняшнее число, а затем сообщает: «С ДНЕМ

РОЖДЕНИЯ, Иоганн Себастьян!», если дата совпадает с днем рождения И.С. Баха или просто сообщает «Желаем удачного дня, Иоганн Себастьян», если сегодня не день рождения И.С. Баха.

```
//  пример с закрытыми членами класса
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
class DayOfYear
     { public:
       void input();
       void output();
       void set(int new_month, int new_day);
       int get_month(); //возвращает значение месяца
       int get_day(); //возвращает день месяца
       private:
          int month;
               int day;
     };
int main()
{  DayOfYear today, bach_birthday;
     cout << Rus(" Введите текущую дату: \n");
     today.input();
cout << Rus("СЕГОДНЯ ");
today.output();
bach_birthday.set(3,21);
cout <<Rus("День рождения И.С. Баха ");
bach_birthday.output();
  if (today.get_month() == bach_birthday.get_month()
     && today.get_day() == bach_birthday.get_day() )
  cout <<Rus("С ДНЕМ РОЖДЕНИЯ, Иоганн Себастьян ! \n");
  else cout<<Rus("Желаем удачного дня, Иоганн Себастьян \n");
return 0;}
   void DayOfYear :: input()
{  cout << Rus(" ВВЕДИТЕ МЕСЯЦ: ");   cin >> month;
   cout << Rus(" ВВЕДИТЕ ДЕНЬ МЕСЯЦА: ");
   cin >> day;}
   void DayOfYear :: output()
     {    cout << Rus(" МЕСЯЦ ") << month;
          cout << Rus(" ДЕНЬ ") << day <<"\n";     }
void DayOfYear :: set(int new_month, int new_day)
{    month=new_month;
     day=new_day;}
int DayOfYear ::get_month()
```

```
{    return month; }
int DayOfYear ::get_day()
{    return day; }
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus;
}
```

При запуске программы и вводе месяца и дня рождения И.С. Баха получим (рис. 8.35):

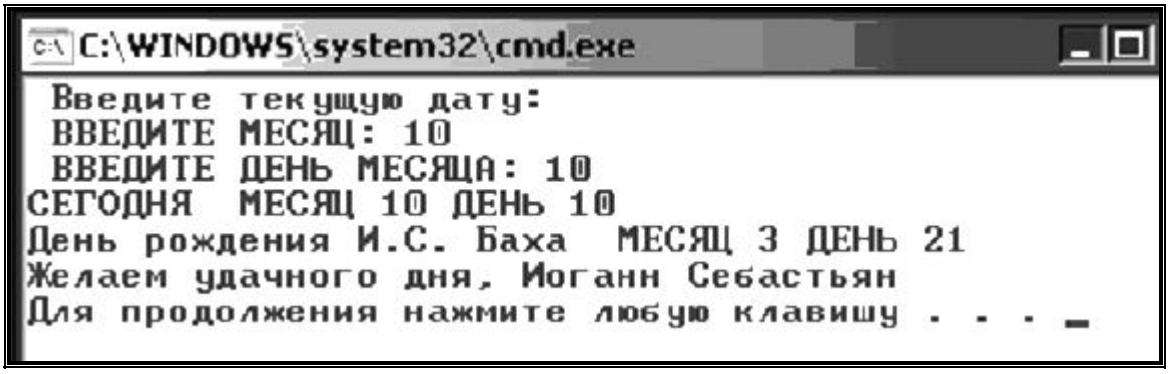


Рис. 8.35. Результат работы программы примера 15.2

Рассмотрим пример с использованием указателей.

Пример 15.3. *Создать программу для расчета суммы двух чисел.*

```
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
class Sum
{  private:
     int x,y,s;
   public:
      void getx(int x1){x=x1;}
      void gety(int y1){y=y1;}
      void summa();   };
int main()
{ Sum z,*b=&z;
   int x2,y2;
   cout <<Rus("\n Введи 1-е слагаемое:");
   cin >>x2;
   cout <<Rus("\n Введи 2-е слагаемое:");
   cin >>y2;
   z.getx(x2);
   z.gety(y2);
```

```
   b->summa();
   cout<<"\n";
   return 0; }
void Sum :: summa()
{    s=x+y;
     cout<<Rus("\n сумма ")<<x;
     cout<<Rus(" и " )<<y;
     cout<<Rus(" равна:")<<s; }
 char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus; }
```

15.3. Конструкторы и деструкторы

Объявление класса не производит выделения памяти. Это происходит потом, когда мы, на основе объявленного класса, проводим описание переменной.

Поэтому при объявлении класса нельзя сразу производить инициализацию (присваивание данных) для объекта. Но если в определение класса включить функцию-член специального вида, известную как *конструктор*, то она автоматически вызывается при объявлении объекта данного класса. Конструктор определяется так же, как и любая другая функция-член; существует лишь два отличия:

1) имя конструктора должно совпадать с именем класса,

2) определение конструктора не может иметь возвращаемого значения. Более того, в начале прототипа функции или в ее заголовке не должно присутствовать никакого типа, даже `void`.

Например, для класса `Sum` описание конструктора может выглядеть так:

```
Sum (int x2=0, int y2)
{    x=x2;
     y=y2; }
```

Далее конструктор `Sum()` можно вызвать двумя способами: `Sum A=Sum(1,2)` и `Sum A(1,2)`. В обоих случаях создается объект `A`, элементы данных `x` и `y` которого получают начальные значения 1 и 2 соответственно. А можно было и так: `Sum A(,2)`, т.к. первый параметр задан сразу при описании конструктора (`x2=0`). В одном классе может быть несколько конструкторов.

Внутри класса могут быть и так называемые *деструкторы*. Деструкторы уничтожают объекты класса и освобождают занимаемую ими память. Деструктор представляет собой метод с именем, совпадающий с именем класса, перед которым стоит символ ~ (тильда). Деструктор не должен иметь ни параметров, ни типа возвращаемого значения. Например, для класса `Sum` деструктор имеет вид: `~Sum () {}`.

Пример 15.4. Создать программу для расчета произведения вида $s = a \cdot b + c \cdot k + a \cdot c$

```
// применение конструктора и деструктора
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
class  Pro
{    private:
          int x,y,z;
     public:
   // прототипы методов
     Pro(int,int); // конструктор
      int putx(); //доступ к x
      int puty(); //доступ к y
      int putz(); //доступ к z
      void proizv(); // произведение
      ~Pro(); // деструктор
};
// описание методов
Pro::Pro(int x1,int y1){x=x1; y=y1;}
int Pro::putx(){return x;}
int Pro::puty(){return y;}
int Pro::putz(){return z;}
void Pro::proizv(){z=x*y;}
Pro::~Pro(){}
int main()
{int s,a,b,c,k;
 cout << Rus("\n введите a,b,c,k \n");
 cin>>a>>b>>c>>k;
  Pro D=Pro(a,b);  //создание и инициализация объекта D
  Pro E(c,k);      //создание и инициализация объекта E
  Pro F(a,c);      //создание и инициализация объекта F
  D.proizv();      // получение произведения a*b
  E.proizv();      // получение произведения c*k
  F.proizv();      // получение произведения a*c
```

```
   cout <<"\n D.a="<<D.putx();
   cout <<"\t D.b="<<D.puty();
 cout <<"\t D.z="<<D.putz();
 s=D.putz()+E.putz()+F.putz();
 cout<<"\n  s="<<s<<"\n";
   F.Pro::~Pro(); // уничтожение объекта F
    E.Pro::~Pro(); // уничтожение объекта E
     D.Pro::~Pro(); // уничтожение объекта D
}
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus;}
```

При запуске получим примерно следующее (рис. 8.36).

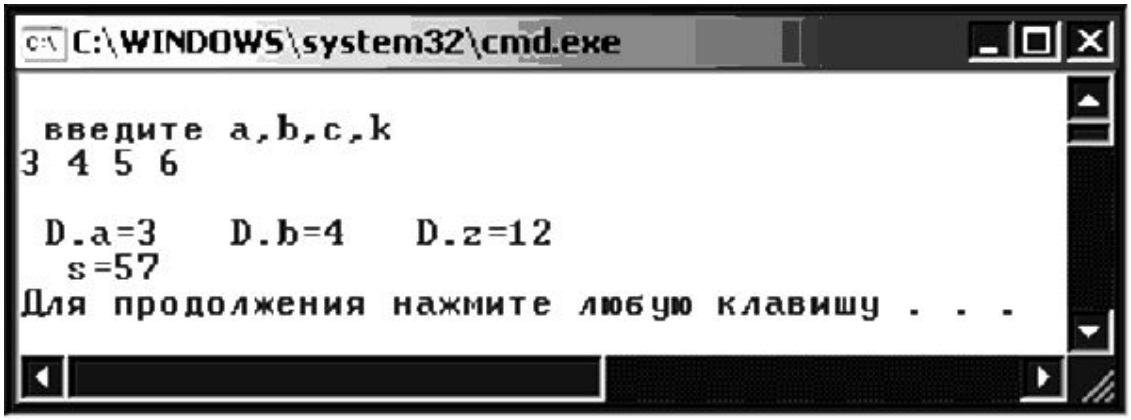


Рис. 8.36. Результат работы программы примера 15.4

Здесь создан класс `Pro`, элементами которого являются сомножители `x,y` и переменная `z` (результат их произведения). Для доступа к ним имеются три метода `putx()`, `puty()`, `putz()`, а для получения произведения `proizv()`. Кроме того, для класса определены конструктор `Pro` и деструктор `~Pro` (в классе описаны их прототипы).

Практические задания

Раздел А

Для всех рассматриваемых ниже заданий (раздел А) разработать класс, содержащий двое членов (назовем их `first`, `second`) и следующие методы:

– ввод с клавиатуры `Read`;

– вывод на экран `Display`;

– метод, указанный в задании

Для полной ясности приведем решение следующего задания:

Поле `first` – вещественное число; поле `second` – целое число, показатель степени. Реализовать метод `power( )` – возведение числа `first` в степень `second` .

```
        #include <iostream>
        #include <windows.h>
        char* Rus(const char* text);
        using namespace std;
class stepen
        { public:
          float first;
          int second;
          void power();
          void Display();
          void Read ();
};
        int n; float rez;
int main()
{    stepen a;
          a.Read();
          a.power();
          a.Display();
          return 0;
}
        char bufRus[256];
        char* Rus(const char* text)
        { AnsiToOem(text, bufRus);  return bufRus;}

void stepen::Read()
   {cout<<Rus("Введи число first "); cin>>first;
   cout<<Rus("Введи число second "); cin>>second;   }
void stepen::Display()
   {cout<<Rus("\n Результат=")<<rez<<endl;      }
void stepen::power()
   { rez=1;
     for (int i=1; i<=second; i++) {rez=rez*first;} }
```

А15.1. Элемент a_i арифметической прогрессии вычисляется по формуле: $a_{i+1} = a_i + d$, $i = 0,1,2,$ … Поле `first` – вещественное число, первый элемент прогрессии a_0; поле `second` – разность прогрессии, `d`. Определить метод `element_i()` – для вычисления заданного элемента прогрессии.

A15.2. Поле `first` – вещественное число; поле `second` – вещественное число, показатель степени. Реализовать метод `power( )` – возведение числа `first` в степень `second`.

A15.3. Поле `first` – целое положительное число, числитель; поле `second` – целое положительное число, знаменатель. Реализовать метод `ipart( )` – выделение целой части дроби `first/second`.

A15.4. Поле `first` – целое положительное число, номинал купюры; номинал может принимать значения 1, 2, 5, 10, 50, 100, 500, 1000, 5000. Поле `second` – целое положительное число, количество купюр данного достоинства. Реализовать метод `summa( )` – вычисление денежной суммы.

A15.5. Поле `first` – вещественное положительное число, цена товара; поле `second` – целое положительное число, количество единиц товара. Реализовать метод `cost( )` – вычисление стоимости товара.

A15.6. Поле `first` – целое положительное число, калорийность 100 г продукта; поле `second` – вещественное положительное число, масса продукта в килограммах. Реализовать метод `Power( )` – вычисление общей калорийности продукта.

A15.7. Поле `first` – вещественное число, левая граница диапазона; поле `second` – вещественное число, левая граница диапазона. Реализовать метод `rangecheck( )` – проверку заданного числа на принадлежность диапазону.

A15.8. Поле `first` – целое число, левая граница диапазона, включается в диапазон; поле `second` – целое число, правая граница диапазона, не включается в диапазон. Пара чисел представляет полуоткрытый интервал [`first`, `second`). Реализовать метод `rangecheck( )` – проверку заданного числа на принадлежность диапазону.

A15.9. Поле `first` – целое положительное число, часы; поле `second` – целое положительное число, минуты. Реализовать метод `minutes( )` – приведение времени в минуты.

A15.10. Линейное уравнение `y = Ax + B`. Поле `first` – вещественное число, коэффициент `A`; поле `second` - вещественное число, коэффициент `B`. Реализовать метод `function( )` – вычисление для заданного `x` значение функции `y`.

A15.11. Линейное уравнение `y = Ax + B`. Поле `first` – вещественное число, коэффициент `A`; поле `second` - вещественное число, коэффициент `B`. Реализовать метод `function( )` – вычисление корня линейного уравнения.

A15.12. Поле `first` – вещественное число, координата *x* точки на плоскости; поле `second` – вещественное число, координата *y* точки на плоскости. Реализовать метод `distance( )` – расстояние точки от начала координат.

A15.13. Поле `first` – вещественное число, катет *a* прямоугольного треугольника; поле `second` – вещественное число, катет *b* прямоугольного треугольника. Реализовать метод `hypotenuse()` – вычисление гипотенузы.

A15.14. Поле `first` – вещественное положительное число, оклад; поле `second` – целое положительное число, количество отработанных дней в месяце. Реализовать метод `summa ( )` – вычисление начисленной зарплаты для заданного месяца: `оклад/ дни_месяца * отработанные_дни`.

A15.15. Поле `first` – целое положительное число, продолжительность телефонного разговора в минутах; поле `second` – вещественно положительное число, стоимость одной минуты в рублях. Реализовать метод `cost( )` – вычисление общей стоимости разговора.

A15.16. Поле `first` – вещественное число, целая часть числа; поле `second` – положительное вещественное число, дробная часть числа. Реализовать метод `multiply( )` – умножение на произвольное вещественное число типа `double`.

A15.17. Поле `first` – целое положительное число, числитель; поле `second` – целое положительное число, знаменатель. Реализовать метод `nesokr( )` – приведение дроби `first/second` к несократимому виду.

A15.18. Поле `first` – целое число, целая часть числа; поле `second` – положительное целое число, дробная часть числа. Реализовать метод `multiply( )` – умножение на произвольное целое число типа `int`.

A15.19. Число сочетаний из `n` объектов по `k` объектов (`k<n`) вычисляется по формуле: `C(n,k)=n!/(n - k)!*k!)`. Поле `first` – целое положительное число, `k`; поле `second` – положительное целое число, `n`. Реализовать метод `combination( )` – вычисление `C(n,k)`.

A15.20. Элемент a_j геометрической прогрессии вычисляется по формуле: $a_j = a_0 * r, j = 0,1,2, \ldots$ Поле `first` – вещественное число, первый элемент прогрессии a_0; поле `second` – знаменатель прогрессии, `r`. Определить метод `elementj( )` – для вычисления заданного элемента прогрессии.

A15.21. Поле `first` – целое число, целая часть числа, записанного в восьмеричной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в восьмеричной системе счисления. Реализовать метод `add8( )` – сложение чисел в восьмеричной системе.

A15.22. Поле `first` – целое число, целая часть числа, записанного в восьмеричной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в восьмеричной системе счисления. Реализовать метод `mult8( )` – умножение чисел в восьмеричной системе.

A15.23. Поле `first` – целое число, целая часть числа, записанного в двоичной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в двоичной системе счисления. Реализовать метод `add2( )` – сложение чисел в двоичной системе.

A15.24. Поле `first` – целое число, целая часть числа, записанного в двоичной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в двоичной системе счисления. Реализовать метод `mult2( )` – умножение чисел в двоичной системе.

A15.25. Поле `first` – целое число, целая часть числа, записанного в 16-ричной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в 16-ричной системе счисления. Реализовать метод `mult2( )` – сложение чисел в 16-ричной системе.

Раздел Б

Для всех рассматриваемых ниже заданий (раздел Б) разработать класс с нужным числом членов и содержащий, помимо указанных в задании, следующие методы:

– ввод с клавиатуры `vvod`;

– вывод на экран `vyvod`;

Применить созданный класс для решения конкретных задач.

Для полной ясности приведем решение следующего примера:

Комплексное число представляется парой действительных чисел (a,b), *где* a – *действительная часть*, b – *мнимая часть:* a+b*i, *здесь* i – *мнимая единица*, i = sqrt(–1) . *Реализовать класс* Complex() *для работы с комплексными числами. Обязательно должны присутствовать операции:*

Сложение add, (a,b) + (c,d) = (a+c, b+d)*;*

Вычитание sub, (a,b) – (c,d) = (a – b, c – d)*;*

Умножение mul, (a,b) * (c,d) = (ac – bd, ad+bc)*;*

Деление div, (a,b) / (c,d) = (ac+bd, bc – ad)/ ($c^2 + d^2$)*;*

Сравнения equ, (a,b) = (c,d) , *если* (a=c) *и* (b=d)*;*

Сопряженное число conj(a,b) = (a, –b).

```
//арифм. операции с комплексными числами
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* text);
class Complex
{ public:
  float a,b;
void vvod ()
 {cout<<Rus("\n Введи действ.часть ");cin>>a;
  cout<<Rus("Введи мнимую часть "); cin>>b;}
void vyvod ()
  {char sig='+'; if (b<0) sig='  ';
   cout<<Rus("\n комплексное число=")<<a<<sig<<b<<"i";}
void mul(Complex c1, Complex c2)
  { a= c1.a*c2.a - c1.b*c2.b;
    b=c1.a*c2.b+c1.b*c2.a;     }
void add(Complex c1, Complex c2)
  { a=c1.a+c2.a; b=c1.b+c2.b; }
void sub(Complex c1, Complex c2)
  { a=c1.a-c2.a; b=c1.b-c2.b; }
void div(Complex c1, Complex c2)
  { a=(c1.a*c2.a+c1.b*c2.b)/(c2.a*c2.a+c2.b*c2.b);
    b=(c1.b*c2.a-c1.a*c2.b)/(c2.a*c2.a+c2.b*c2.b);}
void conj(Complex c)
  { a=c.a; b=-c.b;   }
};
int equ(Complex c1, Complex c2);
int equ(Complex c1, Complex c2)
  { if ((c1.a= =c2.a)&&(c1.b= =c2.b)) return (1);
```

```
        else return (0); }
char bufRus[256];
char* Rus(const char* text)
  { CharToOem(text,bufRus); return bufRus;}
int main()
{    Complex c1,c2,c; int r;
   cout<<"\n Vvodi 1-e chislo ";
        c1.vvod();
   cout<<"Vvodi 2-e chislo ";
        c2.vvod();
   cout<<"\n 1-e chislo=";
        c1.vyvod();
   cout<<" \n 2-e chislo="; c2.vyvod(); cout<<endl;
        c.add(c1,c2);
   cout<<"\n Rez slojen: ";    c.vyvod();
        c.sub(c1,c2);
   cout<<"\n Rez vychit: ";    c.vyvod();
        c.mul(c1,c2);
   cout<<"\n Rez umnojen: ";   c.vyvod();
        c.div(c1,c2);
   cout<<"\n Rez delen: ";     c.vyvod();
        r=equ(c1,c2);
   if (r= =1) cout<<"\n c1=c2 ";
   else cout<<"\n c1 <> c2 ";
        c.conj(c1);
   cout<<"\n Rez sopryaj c1: "; c.vyvod();
        cout<<"\n";
return 0;}
```

Б15.1. Создать класс `EngMer` для работы с английскими мерами длины: фунтами и дюймами, при этом учтем, что 1 фунт = 12 дюймов. Длина объекта будет задаваться парой чисел (фунты и дюймы) и нужно реализовать: сложение и вычитание длин, умножение и деление длин, сравнение длин.

Б15.2. Создать класс `vector3D`, задаваемый тройкой координат. Обязательно должны быть реализованы: сложение и вычитание векторов, скалярное произведение векторов, умножение на скаляр, сравнение векторов, вычисление длины вектора, сравнение длины векторов.

Б15.3. Создать класс `EngMoney` для работы с устаревшей денежной системой Великобритании. В ней использовались фунты, шиллинги и пенсы. При этом: 1 фунт = 20 шиллингов, 1 шиллинг = 12 пенсов. Денежные суммы будут задаваться в фунтах, шиллингах и

пенсах и результат выдаваться также в этих величинах. Должны быть реализованы: сложение и вычитание, умножение и деление, сравнение сумм.

Б15.4. Создать класс `Money` для работы с денежными суммами. Число должно быть представлено двумя полями: типа `long int` для рублей и типа `int` – для копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операция сравнения.

Б15.5. Создать класс `Triangle` для представления треугольника. Поля данных должны включать углы и стороны. Требуется реализовать операции: получения и изменения полей данных, вычисления площади, вычисления периметра, вычисления высот, а также определения вида прямоугольника (равносторонний, равнобедренный или прямоугольный).

Б15.6. Создать класс `Angle` для работы с углами на плоскости, задаваемыми величинами в градусах и минутах. Обязательно должны быть реализованы: перевод в радианы, приведение к диапазону 0–360, увеличение и уменьшение угла на заданную величину, получение синуса, сравнение углов.

Б15.7. Создать класс `Point` для работы с точками на плоскости. координаты точки – декартовы. Обязательно должны быть реализованы: перемещение точки по оси *X*, перемещение по оси *Y*, определение расстояния до начала координат, расстояние между двумя точками.

Б15.8. Рациональная (несократимая) дробь представляется парой целых чисел (*a*,*b*), где *a* – числитель, *b* – знаменатель. Создать `класс Rational` для работы с рациональными дробями. Обязательно должны быть реализованы операции: сложения, вычитания, умножения, деления и сравнения дробей, причем результат должен приводится в виде несократимой дроби (т.е. результат нужно сокращать).

Б15.9. Создать класс `Date` для работы с датами в формате «год.месяц.день». Дата представляется структурой с тремя полями типа `unsigned int`: для года, месяца и дня. Класс должен включать операции: вычисление даты через заданное количество дней от указанной, вычитание заданного количества дней из даты, определение високосного года, присвоение и получение отдельных частей

(год, месяц, день), сравнение дат (равно, до, после), вычисление количества дней между датами.

Б15.10. Создать класс `Time` для работы со временем в формате «час:минута:секунда». Обязательными операциями являются: вычисление разницы между двумя моментами времени в секундах, сложение времени и заданного количества секунд, вычитание из времени заданного количества секунд, сравнение моментов времени, перевод в секунды, перевод в минуты (с округлением до целой минуты).

Б15.11. В морской навигации координаты точки измеряются в градусах и минутах широты и долготы. Один градус равен 60 минутам (ранее минуту делили на 60 секунд, но сейчас минуту делят на обычные десятичные доли). Долгота измеряется от 0 до 180 градусов восточнее (E) или западнее (W) Гринвича. Широта принимает значения от 0 до 90 градусов севернее (N) или южнее (S) экватора. Создать класс `Koord`, включающий следующие три поля: типа `int` для числа градусов, типа `float` для числа минут и типа char для указания направления (N,S, W или E). Объект этого класса может содержать значение как широты, так и долготы. Предусмотреть ввод координат точки, указания полушария для указанной точки, вычисления расстояния между двумя точками. Учитывать, что по широте 1 градус равен 111 км, а по долготе 1 градус равен 111 км * cos(широты).

Б15.12. Реализовать класс `Account`, представляющий собой банковский счет. В классе должны быть реализованы 4 поля: фамилия владельца, номер счета, процент начисления и сумма в рублях. Необходимо выполнять следующие операции: сменить владельца счета, снять некоторую сумму со счета, положить деньги на счет, начислить проценты, перевести сумму в доллары, перевести сумму в евро, получить сумму прописью (преобразовать в числительное).

Б15.13. Номиналы российских рублей могут принимать значения 1, 2, 5, 10, 50, 100, 500, 1000, 5000. Копейки представить как 0,01 (1 копейка), 0,05 (5 копеек), 0,1 (10 копеек), 0,5 (50 копеек). Создать класс `Money` для работы с денежными суммами. Сумма должна быть представлена полями-номиналами, значениями которых должны быть количества купюр данного достоинства. Реализовать сложение сумм, вычитание сумм, деление сумм, деление суммы на дробное число, умножение на дробное число и операцию

сравнения. Дробная часть (копейки) при выводе на экран должны быть отделены от целой части запятой.

Б15.14. Реализовать класс `Bankomat`, моделирующий работу банкомата. В классе должны содержаться поля для хранения идентификационного номера банкомата, информации о текущей сумме денег, оставшейся в банкомате, минимальной и максимальной суммах, которые позволяется снять клиенту в один день. Сумма денег представляется полями-номиналами 10–1000 (см. задание Б15.13). Реализовать метод загрузки купюр в банкомат, и метод снятия определенной суммы денег. Метод снятия денег должен проверять на корректность снимаемую сумму (не должна быть меньше минимально допустимой и больше максимально допустимой). Должен быть метод для отображения оставшейся в банкомате суммы.

Б15.15. Создать класс `Fraction` для работы с дробными числами. Число должно быть представлено двумя полями: целая часть – длинное целое со знаком, дробная часть – беззнаковое короткое целое. Реализовать арифметические операции сложения, вычитания, умножения и операции сравнения.

Б15.16. Создать класс `Goods` (товар). В классе должны быть представлены поля: наименование товара, дата оформления, цена товара, количество единиц товара, номер накладной, по которой товар поступил на склад. Реализовать методы изменения цены товара, изменения количества товара (увеличения и уменьшения), вычисления стоимости товара. Должен быть метод для отображения стоимости товара в виде строки.

Б15.17. Создать класс `BitString` для работы с 64-битовыми строками. Битовая строка должна быть представлена двумя полями типа unsigned long. Должны быть реализованы все традиционные операции для работы с битами: and, or, xor, not. Реализовать сдвиг влево shiftLeft и сдвиг вправо shiftRight на заданное количество битов.

Б15.18. Создать класс `LongLong` для работы с целыми числами из 64 бит. Число должно быть представлено двумя полями: long – старшая часть, unsigned long – младшая часть. Должны быть реализованы арифметические операции (сложение, вычитание, умножение и деление) и сравнение.

Б15.19. Создать класс `Payment` (зарплата). В классе должны быть представлены поля: фамилия-имя-отчество, оклад, год поступления на работу, процент надбавки, подоходный налог, количество отработанных дней в месяце, количество рабочих дней в месяце,

начисленная и удержанная сумма. Реализовать методы: вычисления начисленной суммы, вычисления удержанной суммы, вычисления суммы, выдававаемой на руки, вычисления стажа. Стаж вычисляется как полное количество лет, прошедших от года поступления на работу, до текущего года. Начисления представляют собой сумму, начисленную за отработанные дни, и надбавки. Удержания представляют собой отчисления в пенсионный фонд (1% от начисленной суммы) и подоходный налог (13%).

Б15.20. Создать класс `Ship`, который будет содержать данные об учетном номере корабля и его координатах. Для хранения координат используйте поля типа `Koord` (см. задание Б15.11). Разработайте методы для ввода данных о корабле, о выводе его координат (с указанием полушария), метод для вычисления расстояния между кораблями (см. задание Б15.11).

Б15.21. Создать класс `Binary1`, который будет содержать число в двоичной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода двоичных чисел (с дробной частью!), вывода двоичных чисел, методы для вычисления суммы и произведения двоичных чисел.

Б15.22. Создать класс `Binary2`, который будет содержать число в двоичной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода двоичных чисел (с дробной частью!), вывода двоичных чисел, методы для вычисления разности и деления двоичных чисел.

Б15.23. Создать класс `Hexadec1`, который будет содержать число в 16-ричной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода 16-ричных чисел (с дробной частью!), вывода 16-ричных чисел, методы для вычисления суммы и произведения 16-ричных чисел.

Б15.24. Создать класс `Hexadec2`, который будет содержать число в 16-ричной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода 16-ричных чисел (с дробной частью!), вывода 16-ричных чисел, методы для вычисления разности и деления 16-ричных чисел.

Б15.25. Создать класс `Oct1`, который будет содержать число в 8-ричной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода 8-ричных чисел (с дробной частью!), вывода 8-ричных чисел, методы для вычисления суммы и произведения 8-ричных чисел.

16. ФАЙЛЫ

В C++ ***файл***, как и в других языках, представляет собой *поименованную совокупность данных*, *находящуюся на внешнем устройстве и имеющую определенные атрибуты* (*характеристики*). Правило именования файлов определяются операционной системой. Для организации работы с файлами необходимо усвоить такое понятие как ***поток*** – *абстрактный канал связи*, *который создается в программе для обмена данными с файлами*. Это понятие введено, чтобы можно было не учитывать детали физической организации канала связи между источником и приемником информации.

Для того, чтобы в C++ начать работу с файлами, находящимися на логических дисках, необходимо «создать» соответствующие потоки. Средства C++ позволяют работать с *текстовыми* и *бинарными* файлами после того, как в программе установлена их взаимосвязь с текстовыми и бинарными потоками соответственно. В данном пособии мы ограничимся рассмотрением работы с текстовыми файлами.

16.1. Работа с текстовыми файлами

Файл, рассматриваемый как последовательность строк символов, разделенных *непробельными* символами, называется ***текстовым***. Кроме символа «пробел» пробельными символами являются специальные символы табуляции, новая строка, возврат каретки и перевод формата (страницы): *'\t'*, *'\v'*, *'\n'*, *'\r'*, *'\f'*. Среди пробельных символов выделяются символы новой строки (*'\n'*, *'\r'*), которые используются для стандартного разделения совокупностей строк на файловые строки. Строки символов в текстовых файлах представляют собой последовательности непробельных символов, которые могут интерпретироваться при вводе как данные определенного вида, представленные в некотором формате. Например, последовательность 125, ограниченная с двух сторон пробельными символами, может быть преобразована при вводе в вещественное число типа **double** или целое число типа **int**.

Пример 16.1. Создать программу, которая на диске C создаст файл numbers.txt и запишет в него 5 введенных пользователем целых чисел.

(Просмотрите затем созданный файл в обычном текстовом редакторе и убедитесь, что каждое число находится в отдельной строке).

Для работы с файлами на *запись* нужно сделать следующее:

1) подключить заголовочный файл **fstream**: **#include <fstream>**

2) создать объект класса **ofstream** для управления потоком вывода: **ofstream fout**

3) поставить этот объект в соответствие определенному файлу (с помощью метода **open**()): **fout.open(«c:\\numbers2.txt «)**

4) использовать этот объект так же, как используется объект **cout** (только вывод теперь будет в файл, а не на экран): **fout<< «запись»** – это запись в файл слова «запись».

5) Для закрытия работы с файлом используют метод **close()** : **fout.close ();**

```
// запись в текстовый файл
#include <fstream>
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
int main()
{    ofstream fout;
     int i; int n;
     fout.open("C:\\numbers2.txt ");
   for (i=1;i<=5;i++)
   { cout<<Rus("ВВЕДИ ")<<i;
          cout<<Rus(" ЧИСЛО "); cin>> n;
          fout<<n<<"\n"; }
     fout.close ();
return 0;}
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus;
}
```

При *чтении* из файла действуют аналогично, но только в пункте 2-м создать объект класса **ifstream**: **ifstream fin**; поставить этот объект в соответствие определенному файлу:

fin.open(«c:\\numbers2.txt «);

и для чтения из файла использовать этот объект как объект **cin** : для ввода вместо **cin** писать

fin >>numb

Пример 16.2. Создать программу, которая считает информацию из созданного ранее на диске С файла numbers.txt, выведет ее на экран и подсчитает среднее арифметическое этих чисел.

```
// чтение из текстового файла
#include <fstream>
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
int main()
{    ifstream fin;
     int i,n,sum,sr; sum=0; n=0;
   const int len = 81  ;  int a;
     fin.open("C:\\numbers2.txt ");
   for (i=1; i<=5; i++)
     {fin>>a;
      cout<<Rus("\n ЧИСЛО")<<i<<" = "<<a;
     sum=sum+(int)a; n++; }
   fin.close();
   sr=(float) sum/n;
     cout<<Rus("\n количество чисел в файле = ")<<n;
   cout<<Rus("\n среднее арифметическое их = ")<<sr;
   cout<<"\n";
return 0;}
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus;}
```

Для работы в стиле С++ могут быть заданы режимы обращения к файлам (они также задаются при открытии файлов). Мы их пока не записывали, поскольку они нередко открываются по умолчанию (для **ifstream** будет режим *считывания*, для **ofstream** – режим *записи*). Здесь они записываются несколько более громоздко (см. табл. 6).

Таблица 6

Режимы работы с файлами (в духе C++)

Значение режима	Описание режима	Позиция ввода-вывода
ios : : in	Считывание, для потока **ifstream** устанавливается по умолчанию	в начало файла (по умолчанию), но возможно позиционирование в любое место файла
ios : : out	Запись поверх имеющейся (старая запись стирается без предупреждения); для потока **ofstream** устанавливается по умолчанию	в начало файла (по умолчанию), но возможно позиционирование в любое место файла
ios : : ate	Запись и считывание	в конец файла (по умолчанию), но возможно позиционирование в любое место файла
ios : : app	Только для записи ; Добавление в конец файла после уже имеющейся записи	в конец файла

16.2. Работа со структурами в файлах

Рассмотрим, как можно записывать/считывать целые структуры в файлы. Правила работы со структурами мы рассмотрели в разделе 14.

Пример 16.3. В текстовом файле хранится база отдела кадров предприятия. На предприятии 10 сотрудников. Каждая строка файла содержит запись об одном сотруднике. Формат записи: фамилия и инициалы (30 поз., фамилия должна начинаться с первой позиции), год рождения (5 поз.), оклад (10 поз.). Написать программу, которая по заданной фамилии выводит на экран сведения о сотруднике, подсчитывая средний оклад всех запрошенных сотрудников.

Очевидно, что при считывании данных из файла их следует в программе разместить в массив (типа структуры). Поскольку далее будем вводить фамилию сотрудника, то для хранения фамилии будем использовать строку символов (на те же 30 поз.). Для подсчета среднего оклада понадобится цикл, в котором будем вычисляться сумма окладов задаваемых сотрудников и их количество. Для простоты условимся, что для выхода из цикла (где обрабатываются сведения о сотрудниках) вместо фамилии будем вводить слово end.

Итак, алгоритм решения задачи:

1. Считать из файла в массив сведения о сотрудниках;
2. Организовать цикл вывода сведений о сотруднике:

- ввести с клавиатуры фамилию,
- выполнить поиск сотрудников в массиве,
- увеличить суммарный оклад и счетчик количества сотрудников,
- вывести сведения о сотруднике или сообщение об их отсутствии.

3. Вывести средний оклад

Разумеется, следует подготовить текстовый файл `dbase.txt` с нужными данными, т.е. создать текстовый файл с нужным количеством строк, причем каждая строка должна иметь нужную длину и содержимое. Рекомендуется создавать такой файл в DOS-овском текстовом файле (например, в редакторе, встроенном в файловый менеджер FAR). В этом случае не будет проблем с кодировкой русских букв. Но если заполняете файл `dbase.txt` в windows-редакторе Блокнот или Word, то для отображения русских букв надо раскомментировать строки с функциями AnsiToOem и OemToAnsi (см. листинг ниже).

Например, создадим такой файл:

```
Мамин                    1989 10000.00
Папин                    1979 11000.00
Papin                    1969 12000.00
Mamin                    1959 13000.00
Маминский                1949 14000.00
Dedov                    1939 15000.00
Pradedov                 1929 16000.00
Ионов                    1919 17000.00
Яковлев                  1909 18000.00
Elenin                   1899 19000.00
```

В рассмотренной ниже программе в цикле `while (!fin.eof())` происходит построчное считывание из файла в строку `buf` и заполнение очередного элемента массива `dbase` (типа структуры). Здесь `eof()` – метод для проверки конца файла (он дает логическое значение). Для формирования полей структуры используются:

- функция копирования строк `strncpy(p,a,n)` – она копирует `n` символов из строки `a` в строку `s`

- функция `atoi(str)` – преобразует символьное представление целого числа в целое число,
- функция `atof(str)` – преобразует символьное представление вещественного числа в вещественное число.

Обратите внимание, что завершающий нуль-символ в поле Фамилия заносится «вручную», поскольку функция `strncpy(p,a,n)` делает это только в том случае, если строка-источник короче строки-приемника. В функцию `atoi` передается адрес начала подстроки, в которой находится год рождения. В цикл добавлен контрольный вывод на экран считанной строки (в виде сформированных полей структуры). В конец цикла для выхода вставлен `if (i>=l_dbase) break;` , поскольку в C++ после чтения из файла последнего элемента условие конца файла *не возникает*. Оно возникает при следующем чтении, когда программа пытается считать данные за последним элементом в файле.

Цикл поиска сотрудников по фамилии `while (true)` организован как бесконечный с принудительным выходом (при вводе строки «`end`»). В этом цикле используется для сравнения функция поиска подстроки `strstr(s1,s2)` – она ищет вхождение строки `s2` в строку `s1` и принимает соответствующее логическое значение (если входит, то `true`). Функция `strcmp(s1,s2)` просто проверяет равенство строки `s1` строке `s2`, она также принимает логическое значение `(false, true)`.

```
#include <fstream>
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
int main()
{const int l_name=30, l_year=5, l_pay=10,
l_buf=l_name+l_year+l_pay;
struct Man
{    int birth_year;
          char name[l_name+1];
          float pay; };
const int l_dbase=10;
```

```
Man dbase [l_dbase+1];
char buf[l_buf+1];
char sName[l_name+1];
ifstream fin;
fin.open( "c:\\dbase.txt", ios::in );
if (!fin){cout<<Rus(" Ошибка открытия файла "); return 1;}
int i=0;
while (!fin.eof())
{    fin.getline(buf,l_buf);
     strncpy(dbase[i].name, buf, l_name);
     dbase[i].name[l_name]= '\0';
     dbase[i].birth_year=atoi(&buf[l_name]);
     dbase[i].pay=atof(&buf[l_name+l_year]);
 cout<<dbase[i].name<<dbase[i].pay
   <<"    "<<dbase[i].birth_year<<"\n";
 i++; if (i>=l_dbase) break; }
int n_record =i, n_man=0;
float mean_pay=0;
while (true)
{    cout<<Rus(" Введите фамилию или слово end: ");
  cin>>sName;
     //OemToAnsi(sName, sName);
     if (strcmp(sName, "end")== 0) break;
     bool not_found=true;
     for (i=0; i<n_record; i++)
     {    if (strstr(dbase[i].name, sName))
               if (dbase[i].name[strlen(sName)]==' ')
               {    //AnsiToOem(sName, sName);
                    strcpy(sName, dbase[i].name);
                    //AnsiToOem(sName, sName);
                    n_man++; mean_pay+=dbase[i].pay;
                    not_found=false;              }
          }
if (not_found) cout<<Rus("такого сотрудника нет")<< endl;
}         if (n_man>0)
     cout<<Rus(" Средний оклад: ")<<mean_pay/n_man<<"\n";
return 0;}
char* Rus(const char* str)
     { AnsiToOem(str, str_rus);
     return str_rus;}
```

Результат работы программы будет иметь, например, такой вид (рис. 8.37):

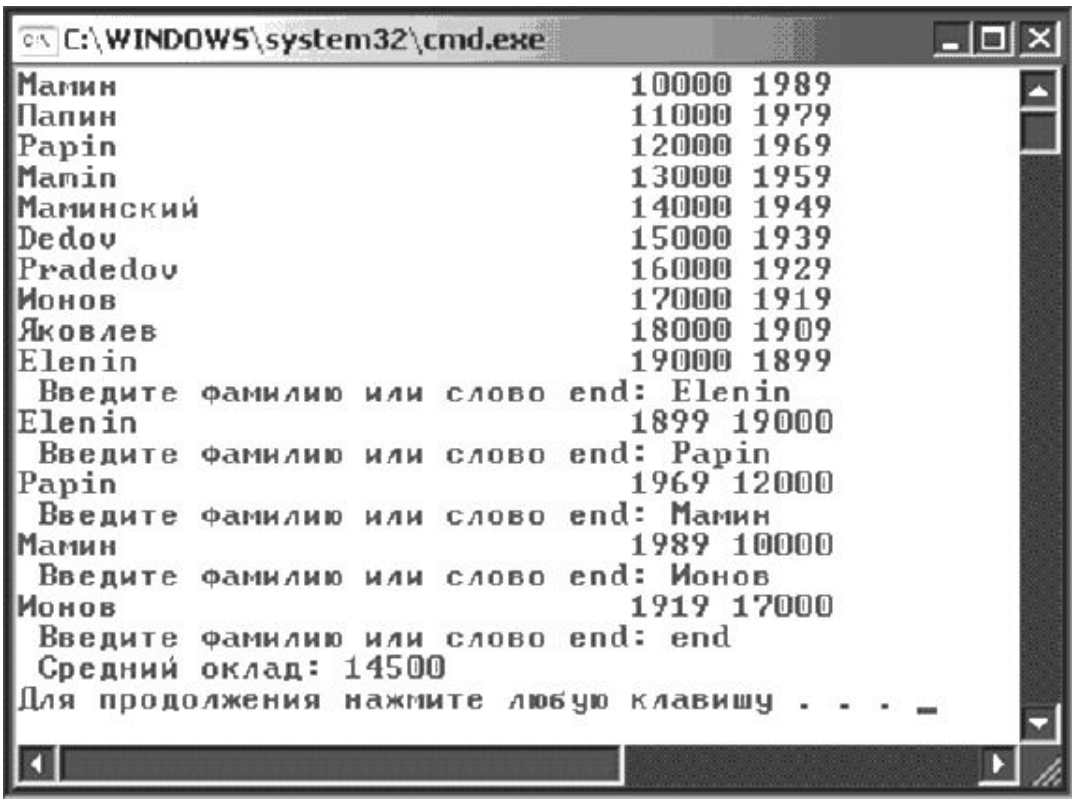


Рис. 8.37. Результат работы программы примера 16.3

Пример 16.4. Считать данные из файла (использовавшегося в примере 16.3), упорядочить их по году рождения и записать в другой файл.

Для сортировки здесь будем использовать простой алгоритм: выбирается наименьший элемент массива и меняется местами с первым элементом, затем рассматриваются элементы, начиная со второго, наименьший из них меняется местами со вторым и т.д.

Элементами массива в данной задаче являются структуры. Для структур одного типа определена операция присваивания, поэтому обмен двух элементов массива структур выглядит точно так же, как для основных типов данных.

```
#include <fstream>
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <windows.h>
using namespace std;
char* Rus(const char* str);
char str_rus[256];
int main()
{const int l_name=30, l_year=5, l_pay=10,
     l_buf=l_name+l_year+l_pay;
```

```
struct Man
{     int birth_year;
      char name[l_name+1];
      float pay; };
const int l_dbase=10;
Man dbase [l_dbase+1];
char buf[l_buf+1];
char sName[l_name+1];
ifstream fin;
fin.open("c:\\dbase.txt", ios::in );
if (!fin){cout<<Rus(" Ошибка открытия файла "); return 1;}
int i=0;
while (!fin.eof())
{     fin.getline(buf,l_buf);
      strncpy(dbase[i].name, buf, l_name);
      dbase[i].name[l_name]= '\0';
      dbase[i].birth_year=atoi(&buf[l_name]);
      dbase[i].pay=atof(&buf[l_name+l_year]);
      cout<<dbase[i].name<<dbase[i].pay
       <<" "<<dbase[i].birth_year<<"\n";
 i++;  if (i>=l_dbase) break; }
int n_record=i;
fin.close();
ofstream fout;fout.open("c:\\dbase2.txt", ios::out);
if (!fout) {cout<<Rus(" Ошибка открытия файла "); return 1;}
for (i=0; i<n_record-1; i++)
{// принимаем за наименьший первый из элементов
 int imin=i;
 // поиск номера минимального элемента из неупорядоченных
 for (int j=i+1; j<n_record; j++)
 if (dbase[j].birth_year<dbase[imin].birth_year) imin=j;
 // обмен двух элементов массива структур
 Man a=dbase[i]; dbase[i]=dbase[imin]; dbase[imin]=a;}
//запись в файл отсортированной базы данных
for (i=0; i<n_record; i++)
{     fout<<dbase[i].name<<dbase[i].birth_year
    <<" "<<dbase[i].pay<<endl;}
fout.close();
cout<<Rus("Сортировка базы данных завершена")<< endl;
//вывод на экран отсортированной базы данных
for (i=0; i<n_record; i++)
{     cout<<dbase[i].name<<dbase[i].birth_year<<"
"<<dbase[i].pay<<endl;}
return 0;}
char* Rus(const char* str)
```

```
{ AnsiToOem(str, str_rus);
  return str_rus;}
```

Результат работы программы, например, такой (рис. 8.38).

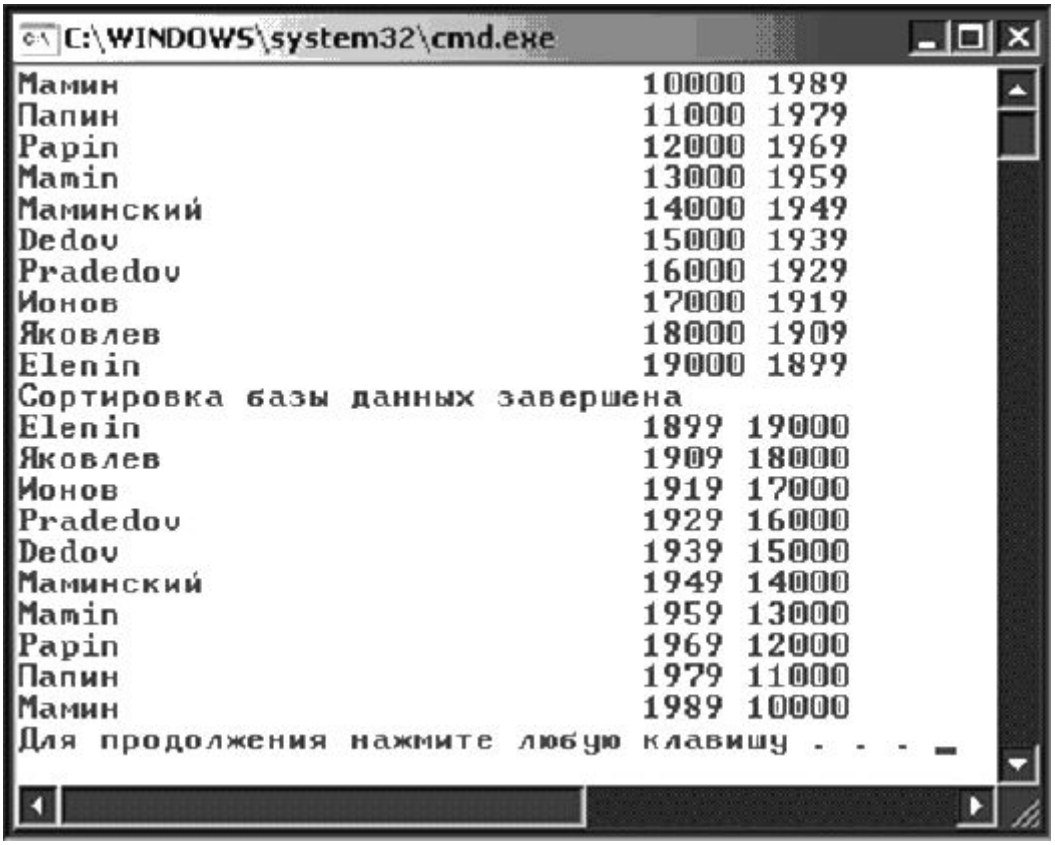


Рис. 8.38. Результат работы программы примера 16.4

16.3. Работа с классами в файлах

Рассмотрим, как работать с *классами* (вместо структур) при чтении (или записи) в файл (из файла). Рассмотрим тот же пример 16.3, но пользуясь технологией работы с классами.

Пример 16.5. В текстовом файле хранится база отдела кадров предприятия. На предприятии 10 сотрудников. Каждая строка файла содержит запись об одном сотруднике. Формат записи: фамилия и инициалы (30 позиций, фамилия должна начинаться с первой позиции), год рождения (5 позиций), оклад (10 позиций). Написать программу, которая по заданной фамилии выводит на экран сведения о сотруднике, подсчитывая средний оклад всех запрошенных сотрудников.

Напомним, что в программе, предложенной для решения примера 16.3 для хранения сведений об одном сотруднике была использована следующая структура `Man`:

```
struct Man
{ char name[l_name + 1];
int birth_year;
float pay; };
```

Начнем с того, что преобразуем эту структуру в *класс*, так как мы предполагаем, что наш новый тип будет обладать более сложным поведением, чем просто чтение и запись его полей:

```
class Man
{ char name[l_name + 1];
int birth_year;
float pay;};
```

Замечание здесь в том, что все поля класса по умолчанию – закрытые (private). Так что если клиентская функция `main()` объявит объект `Man man`, а потом попытается обратиться к какому-либо его полю, например: `man.pay =value`, то компилятор быстро пресечет это безобразие, отказавшись компилировать программу. Поэтому в состав класса надо добавить методы доступа к его полям. Эти методы должны быть общедоступными, или открытыми (public).

Однако предварительно вглядимся внимательнее в определения полей. В решении примера 16.3 поле `name` объявлено как статический массив длиной `l_name + 1`. Это не очень гибкое решение. Мы хотели бы, чтобы наш класс `Man` можно было использовать в будущем в разных приложениях, например для случая больших длин поля `name`. Решение состоит в использовании *динамического* массива символов с требуемой длиной. Поэтому заменим поле `char name[l_name + 1]` на поле `char* pName`. Сразу возникает вопрос: кто и где будет выделять память под этот массив? Для этого в состав класса необходимо включить метод, который обеспечил бы выделение памяти под указанный динамический массив при создании объекта (переменной типа `Man`). Напомним, что метод, который автоматически вызывается при создании экземпляра класса, называется *конструктором*. Компилятор безошибочно находит этот метод среди прочих методов класса, поскольку его имя всегда совпадает с именем класса.

Парным конструктору является другой метод, называемый *деструктором*, который автоматически вызывается перед уничтожением объекта. Имя деструктора отличается от имени конструктора только наличием предваряющего символа ~ (тильда). Ясно, что если в конструкторе была выделена динамическая память, то в деструкторе нужно побеспокоиться об ее освобождении. Напомним, что

объект, созданный как локальная переменная в некотором блоке { }, уничтожается, когда при выполнении достигнут конец блока. Если же объект создан с помощью операции `new`, например:

```
Man* pMan = new Man;
```

то для его уничтожения применяется операция `delete`, например: `delete pMan`.

Итак, наш класс принимает следующий вид:

```
class Man
{public:
     Man(int lName = 30)
     { pName = new char[lName + 1]; } // конструктор
      ~Man() { delete [] pName; } // деструктор
private:
     char* pName;
     int birth_year;
     float pay; };
```

Обратим ваше внимание на одну *синтаксическую* деталь – объявление класса должно обязательно завершаться точкой с запятой (;). Если вы забудете это сделать, то получите от компилятора длинный список маловразумительных сообщений о чем угодно, но только не об истинной ошибке.

Рассмотрим теперь одну важную *семантическую* деталь: в конструкторе класса параметр `lName` имеет значение по умолчанию (30). Если все параметры конструктора имеют значения по умолчанию или если конструктор вовсе не имеет параметров, он называется конструктором по умолчанию. Зачем понадобилось специальное название для такой разновидности конструктора? Разве это не просто удобство для клиента – передать некоторые значения по умолчанию одному из методов класса? Нет! Конструктор – это особый метод, а конструктор по умолчанию имеет несколько специальных областей применения.

Во-первых, такой конструктор используется, если компилятор встречает определение массива объектов, например: `Man man[25]`. Здесь объявлен массив из 25 объектов типа `Man`, и каждый объект этого массива создан путем вызова конструктора по умолчанию! Поэтому если вы забудете снабдить класс конструктором по умолчанию, то вы не сможете объявлять массивы объектов этого класса.

В данном случае мы описали методы внутри описания самого класса (как встроенные функции), но это можно (а чаще – нужно) делать вне описания класса, как показано ниже:

```
class Man
{public:
     Man(int lName = 30); // конструктор
     ~Man(); // деструктор
private:
     char* pName;
     int birth_year;
     float pay; };
// (реализация класса)
Man::Man(int IName) { pName = new char[lName + 1]; }
Man::~Man() { delete [] pName: }
```

При внешнем определении метода перед его именем указывается имя класса, за которым следует операция доступа к области видимости `::` . Выбор способа определения метода зависит в основном от его размера: короткие методы можно определить как встроенные, что может привести к более эффективному коду.

Продолжим процесс проектирования интерфейса нашего класса. Какие методы нужно добавить в класс? На этом этапе очень полезно задаться следующим вопросом: какие обязанности должны быть возложены на класс `Man`?

Первую обязанность мы уже реализовали: объект класса хранит сведения о сотруднике. Чтобы воспользоваться этими сведениями, клиент должен иметь возможность получить эти сведения, изменить их и вывести на экран. Кроме этого, для поиска сотрудника желательно иметь возможность сравнивать его имя с заданным.

Начнем с методов, обеспечивающих доступ к полям класса. Для считывания значений полей добавим методы `GetName()`, `GetBirthYear()`,`GetPay()`. Очевидно, что аргументы здесь не нужны, а возвращаемое значение совпадает с типом поля.

Для записи значений полей добавим методы `SetName()`, `SetBirthYear()`, `SetPay()`.

Чтобы определиться с содержимым этих методов, надо представить себе, как они будут вызываться клиентом. Напомним, что в примере 16.3 фрагмент ввода исходных данных был реализован следующим образом (здесь мы используем идентификатор `man` вместо идентификатора `dbase`):

```
int i = 0;
while (fin.getline(buf, l_buf))
{ strncpy(man[1].name, but, l_name);
 man[i].name[l_name] = '\0';
 man[i].birth_year = atoi(&buf[l_name]);
 man[i].pay = atof(&buf[l__name + l_year]);
 i++; }
```

Как видно из этого фрагмента, в теле цикла `while` на i-й итерации выполняются следующие действия:

- очередная строка читается из файла `fin` в символьный массив `buf`;
- из массива `buf` в поле `name` структуры `man[i]` копируются первые `l_name` символов;
- из буфера `buf` извлекается очередная порция символов, отличных от пробела, преобразуется к типу `int` и записывается в поле `birth_year` структуры `man[i];`
- извлекается следующая порция символов, отличных от пробела, преобразуется к типу `float` и записывается в поле `pay` структуры `man[i].`

Если чисто механически скопировать такое разделение обязанностей между клиентом (`main`) и сервером (объект класса `Man`), то мы получим в теле цикла код вида:

```
man[i].SetName(buf);
man[i].SetBirthYear(atoi(&buf[l_name]));
man[i].SetPay(atof(&buf[l_name + l_year])):
i++;
```

Вряд ли такое решение можно признать удачным. Глаз сопровождающего программиста наверняка «споткнется» на аргументах вида `atoi (&buf[l_name])`. Ведь подобные выражения фактически являются сущностями типа «как делать», а не сущностями типа «что делать»! Иными словами, крайне нежелательно нагружать клиента избыточной информацией. Второстепенные детали, или детали реализации, должны быть упрятаны внутрь класса. Поэтому в данной задаче более удачной является оформление метода `void SetBirthYear(const char* fromBuf).` Аналогичные размышления можно повторить и для метода `SetPay()`.

Разобравшись с методами доступа, перейдем теперь к основной части алгоритма функции `main()`, решающей подзадачу поиска

сотрудника в базе по заданной фамилии и вывода сведений о нем. Напомним, что в ***примере*** 16.3 фрагмент поиска сотрудника с заданным именем был реализован с помощью следующего цикла:

```
for (i = 0; i < n_record; ++i)
{    if (strstr(man[i].name. name)) // 1
  if (man[i].name[strlen(name)] = = ' ' )
 { strcpy(name. man[i].name);
cout <<name<<man[i].birth_year<<' '<<man[i].pay<< "\n"; }
}
```

Опять задумаемся над вопросом: какая информация в этом решении избыточна для клиента? Здесь решаются две подзадачи: сравнение имени в очередной прочитанной записи с заданным именем `name` и вывод информации в поток `cout`.

Поручим решение первой подзадачи методу `CompareName(const char* name)`, спрятав в него подробности решения, а решение второй подзадачи – методу `Print ()`.

Проектирование интерфейса класса `Man` завершено. Давайте посмотрим на текст программы, в которой реализован и использован описаный выше класс `Man`.

```
#include <iostream>
#include <cstring>
#include <windows.h>
#include <fstream>
const char filename[] = "c:\\dbase.txt";
using namespace std;
char* Rus(const char* str);
char str_rus[256];

const int l_name = 30;
const int l_year = 5;
const int l_pay = 10;
const int l_buf = l_name + l_year + l_pay;
class Man
{public:
    Man(int lName = 30);
    ~Man();
    bool  CompareName(const char*) const;
    int   GetBirthYear() const { return birth_year; }
    float GetPay() const { return pay; }
    char* GetName() const { return pName; }
    void  Print() const;
    void  SetBirthYear(const char*);
    void  SetName(const char*);
```

```
    void  SetPay(const char*);
private:
    char* pName;
    int   birth_year;
    float pay;
};

Man::Man(int lName)
{   cout << "Constructor is working" << endl;
    pName = new char[lName + 1];}
Man::~Man()
{   cout << "Destructor is working" << endl;
    delete [] pName;}
void Man::SetName(const char* fromBuf)
{   strncpy(pName, fromBuf, l_name);
    pName[l_name] = 0;}
void Man::SetBirthYear(const char* fromBuf)
{    birth_year = atoi(fromBuf + l_name);}
void Man::SetPay(const char* fromBuf)
{    pay = atof(fromBuf + l_name + l_year);}
bool Man::CompareName(const char* name) const
{if ((strstr(pName, name))&&(pName[strlen(name)]= =' '))
    return true;    else        return false; }
void Man::Print() const
{ cout << pName << birth_year << ' ' << pay << "\n";}

int main()
{ const int maxn_record = 10;
 Man man[maxn_record+1];
 char buf [l_buf + 1];
 char name[l_name + 1];
 ifstream fin(filename);
 if (!fin)
{cout <<Rus("Нет файла ")<<filename<<endl; return 1;}
  int i = 0;
  while (fin.getline(buf, l_buf))
 { if (i > maxn_record)
   { cout << Rus("Слишком длинный файл"); return 1;}
        man[i].SetName(buf);
        man[i].SetBirthYear(buf);
        man[i].SetPay(buf);
        i++;  }
 int n_record = i, n_man = 0;
 float mean_pay = 0;
 while (true)
{  cout << Rus("Введите фамилию или слово end: ");
   cin >> name;
   if (0 = = strcmp(name, "end")) break;
```

```
   bool not_found = true;
   for (i = 0; i < n_record; ++i)
   { if (man[i].CompareName(name))
        {       man[i].Print();
              n_man++; mean_pay += man[i].GetPay();
              not_found = false;
              break;       }    }
 if (not_found) cout<<Rus("Такого сотрудника нет") << endl; }
 if (n_man) cout<< Rus ("Средний оклад:") << mean_pay /n_man << endl;
    return 0; }
char* Rus(const char* str)
{ AnsiToOem(str, str_rus);
  return str_rus; }
```

Вставим дополнительно отладочную печать в конструкторе и деструкторе. Вывод сообщений типа «Constructor is working», «Destructor is working» очень помогает на начальном этапе освоения классов.

Результат работы такой программы смотри на рис. 8.39:

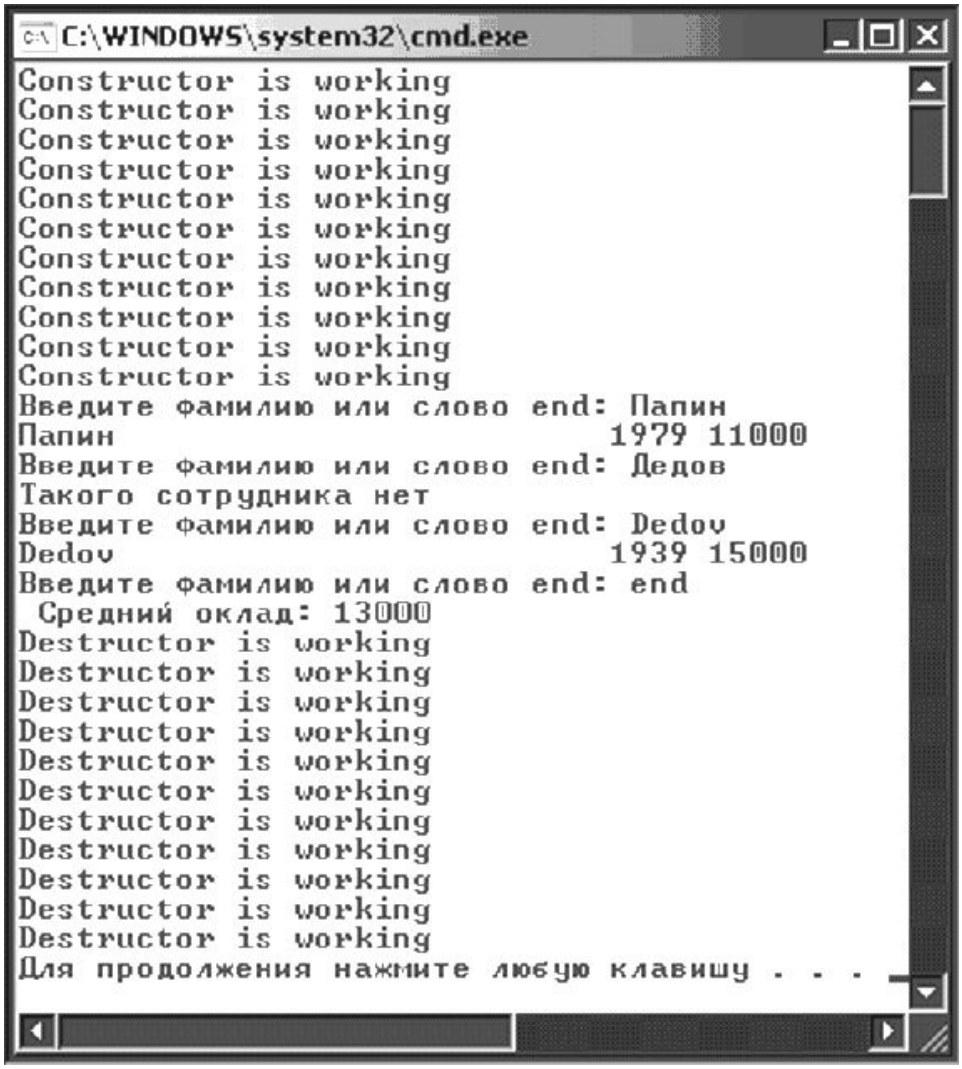


Рис. 8.39. Результат работы программы примера 16.5

Практические задания

Раздел А

Следующие задания требуется решить с использованием текстовых файлов. Во всех заданиях предусмотреть в программе:

- формирование текстового файла, записав в него 20 случайных чисел от –10 до +10, по одному на строке.
- Используя созданный выше файл как входной, сформировать выходной файл по указанному для каждого варианта правилу.

А16.1. Записать выходной файл, прибавив к каждому числу последнее число файла

А16.2. Записать выходной файл, разделив каждое число на полусумму первого отрицательного и 10-го числа файла

А16.3. Записать выходной файл, вычтя из каждого числа наибольшее число файла

А16.4. Записать выходной файл, умножив каждое число на минимальное число из файла.

А16.5. Записать выходной файл, разделив все нечетные по абсолютной величине числа на среднее арифметическое

А16.6. Записать выходной файл, вычтя из каждого числа сумму чисел файла.

А16.7. Записать выходной файл, умножив каждое третье число на удвоенную сумму первого и последнего отрицательных чисел.

А16.8. Записать выходной файл, добавив к каждому числу первое нечетное по абсолютной величине число файла.

А16.9. Записать выходной файл, умножив каждое четное число на первое отрицательное число файла.

А16.10. Записать выходной файл, добавив к каждому числу половину последнего отрицательного числа файла.

А16.11. Записать выходной файл, разделив все числа на половину максимального числа.

А16.12. Записать выходной файл, заменив все нули средним арифметическим.

А16.13. Записать выходной файл, заменив все положительные числа квадратом минимума.

А16.14. Записать выходной файл, добавив к каждому числу полусумму всех отрицательных чисел.

А16.15. Записать выходной файл, добавив к каждому числу среднее арифметическое наименьшего по абсолютной величине и наибольшего из чисел файла.

А16.16. Записать выходной файл, разделив каждое число на минимум и добавив максимум.

А16.17. Записать выходной файл, заменив все положительные числа на максимум.

А16.18. Записать выходной файл, заменив каждое четное число по абсолютной величине на разность максимума и минимума.

А16.19. Записать выходной файл, заменив каждое второе отрицательное число половиной максимума.

А16.20. Записать выходной файл, заменив каждое третье положительное число средним арифметическим отрицательных чисел.

А16.21. Записать выходной файл, заменив каждое положительное число средним арифметическим положительных чисел.

А16.22. Записать выходной файл, заменив каждое положительное число средним арифметическим отрицательных чисел.

А16.23. Записать выходной файл, заменив каждое отрицательное число средним арифметическим отрицательных чисел.

А16.24. Записать выходной файл, заменив каждое отрицательное число средним арифметическим положительных чисел.

А16.25. Записать выходной файл, заменив каждое положительное число этим же числом, разделенным на максимальное из всех чисел.

Раздел Б

Следующие задания требуется решить с использованием классов. При этом обязательно оформить методы для выполнения каждого из действий: по вводу данных, выводу их в файл, чтению данных из файла и выводу их на экран, сортировке данных.

Б16-1. Дана структура с именем STUDENT, состоящая из полей:

- Фамилия и инициалы,
- Номер группы,
- Успеваемость (массив из пяти элементов)

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных в массив, состоящий из 10 структур типа STUDENT, и занесение их в файл данных
- Чтение данных из файла и вывод их на экран
- вывод на экран фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4 (если таких нет – вывести об этом сообщение)

•Список студентов должен быть упорядочен по возрастанию номера группы.

Б16-2. Дана структура с именем STUDENT, состоящая из полей:

•Фамилия и инициалы,

•Номер группы,

•Успеваемость (массив из пяти элементов)

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 10 структур типа STUDENT, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5 (если таких нет – вывести об этом сообщение)

•Список студентов должен быть упорядочен по убыванию среднего балла.

Б16-3. Дана структура с именем STUDENT, состоящая из полей:

•Фамилия и инициалы,

•Номер группы,

•Успеваемость (массив из пяти элементов)

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 10 структур типа STUDENT, и занесение их в файл данных

•Чтение данных из файла и и вывод их на экран

•вывод на экран фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2 (если таких нет – вывести об этом сообщение)

•Список студентов должен быть упорядочен по алфавиту фамилий.

Б16-4.Дана структура с именем AEROFLOT, состоящая из полей:

•Название пункта назначения рейса,

•Номер рейса,

•Тип самолета

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 7 элементов типа AEROFLOT, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием,

введенным с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по возрастанию номера рейса.

Б16-5.Дана структура с именем AEROFLOT, состоящая из полей:

•Название пункта назначения рейса,

•Номер рейса,

•Тип самолета

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 7 элементов типа AEROFLOT, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту названий пунктов назначения.

Б16-6. Дана структура с именем WORKER, состоящая из полей:

•Фамилия и инициалы работника,

•Название занимаемой должности,

•Год поступления на работу

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 10 элементов типа WORKER, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту фамилий.

Б16-7. Дана структура с именем TRAIN, состоящая из полей:

•Название пункта назначения,

•Номер поезда,

•Время отправления

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа TRAIN, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту пунктов назначения.

Б16-8. Дана структура с именем TRAIN, состоящая из полей:

•Название пункта назначения,

•Номер поезда,

•Время отправления

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 6 элементов типа TRAIN, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о поездах, отправляющихся в пункт, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по времени отправления поезда.

Б16-9. Дана структура с именем TRAIN, состоящая из полей:

•Название пункта назначения,

•Номер поезда,

•Время отправления

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа TRAIN, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о поезде, номер которого введен с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по номерам поездов.

Б16-10. Дана структура с именем MARSH, состоящая из полей:

•Название начального пункта назначения,

•Название конечного пункта назначения,

•Номер маршрута

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа MARSH, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о маршруте, номер которого введен с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по номерам маршрутов.

Б16-11. Дана структура с именем MARSH, состоящая из полей:

•Название начального пункта назначения,

•Название конечного пункта назначения,

•Номер маршрута

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа MARSH, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о маршрутах, которые начинаются или заканчиваются в пункте, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по номерам маршрутов.

Б16-12. Дана структура с именем NOTE, состоящая из полей:

•Фамилия, имя,

•Номер телефона,

•Дата рождения (массив из трех чисел)

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа NOTE, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о человеке, номер телефона которого введено с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по датам рождения.

Б16-13. Дана структура с именем NOTE, состоящая из полей:

•Фамилия, имя,

•Номер телефона,

•Дата рождения (массив из трех чисел)

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа NOTE, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту.

Б16-14. Дана структура с именем NOTE, состоящая из полей:

•Фамилия, имя,

•Номер телефона,

•Дата рождения (массив из трех чисел)

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа NOTE, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по двум первым цифрам номера телефона.

Б16-15. Дана структура с именем ZNAK, состоящая из полей:

•Фамилия, имя,

•Знак Зодиака,

•Дата рождения (массив из трех чисел)

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ZNAK, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по датам рождения.

Б16-16. Дана структура с именем ZNAK, состоящая из полей:

•Фамилия, имя,

•Знак Зодиака,

•Дата рождения (массив из трех чисел)

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ZNAK, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о людях, родившихся под знаком, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по датам рождения.

Б16-17. Дана структура с именем ZNAK, состоящая из полей:

•Фамилия, имя,

•Знак Зодиака,

•Дата рождения (массив из трех чисел)

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ZNAK, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по знакам Зодиака.

Б16-18. Дана структура с именем PRICE, состоящая из полей:

•Название товара,

•Название магазина, в котором продается товар,

•Стоимость товара в руб.

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа PRICE, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о товаре, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту названий товара.

Б16-19. Дана структура с именем PRICE, состоящая из полей:

•Название товара,

•Название магазина, в котором продается товар,

•Стоимость товара в руб.

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа PRICE, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту названий магазинов.

Б16-20. Дана структура с именем ORDER, состоящая из полей:

•Расчетный счет плательщика,

•Расчетный счет получателя,

•Перечисляемая сумма в руб.

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ORDER, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по расчетным счетам плательщиков.

Б16-21. Дана структура с именем BOOKS, состоящая из полей:

•Автор книги,

•Название книги,

•Место издания

•Издательство

•Год издания

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа BOOKS, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран книг автора, который введен с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту названий издательств.

Б16-22. Дана структура с именем BOOKS, состоящая из полей:

•Автор книги,

•Название книги,

•Место издания

•Издательство

•Год издания

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа BOOKS, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран книг издательства, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту фамилий авторов.

Б16-23. Дана структура с именем BOOKS, состоящая из полей:

•Автор книги,

•Название книги,

•Место издания

•Издательство

•Год издания

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа BOOKS, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран книг, изданных после года, который введен с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту названий книг.

Б16-24. Дана структура с именем BOOKS, состоящая из полей:

•Автор книги,

•Название книги,

•Место издания

•Издательство

•Год издания

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа BOOKS, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран книг, изданных в городе, который введен с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по годам издания.

Б16-25.Дана структура с именем SOTRUD, состоящая из полей:

•Фамилия, имя сотрудника

•Должность,

•Год рождения

•Год поступления на работу

Написать программу, которая выполняет следующие действия:

•Ввод с клавиатуры данных в массив, состоящий из 8 элементов типа SOTRUD, и занесение их в файл данных

•Чтение данных из файла и вывод их на экран

•вывод на экран сведений сотрудников, которые имеют стаж работы, больший введенного с клавиатуры (если таких нет – вывести об этом сообщение)

•Список должен быть упорядочен по алфавиту фамилий.

ПРИЛОЖЕНИЯ

Приложение 1. Список библиотечных функций

Приведенный ниже список содержит основные функции с указанием их типа и типов аргументов (без указания имени аргумента). При использовании функции следует добавлять **.h** к имени заголовочного файла.

Математические функции

Прототип	Описание	Заголовочный файл
int abs(int)	Абсолютное значение	stdlib
long labs(long)	Абсолютное значение	stdlib
double fabs(double)	Абсолютное значение	math
double sqrt(double)	Квадратный корень	math complex
double pow(double, double)	Первый аргумент в степени второго	math complex
double exp(double x)	Функция $\mathbf{e^x}$ **(**где x – аргумент)	math complex
double log(double)	Натуральный логарифм (**ln**)	math
double log10(double)	Логарифм по основанию 10 (**lg**)	math
double ceil(double)	Наименьшее целое большее или равное аргументу	math
double floor(double)	Наибольшее целое меньшее или равное аргументу	math
int rand % n	Случайное число от 0 до n-1	stdlib
void srand(unsigned int)	Реинициализирует генератор случайных чисел	stdlib
double acos(double)	арккосинус	math
double asin(double)	арксинус	math
double atan(double)	арктангенс	math
double cos(double)	косинус	math
double cosh(double)	косинус гиперболический	math
double sin(double)	синус	math
double sinh(double)	синус гиперболический	math
double tan(double)	тангенс	math
double tanh(double)	тангенс гиперболический	math

Строковые функции (для работы с символьными массивами)

Прототип	Описание	Заголовочный файл
char* strcat(char *Dest, const char *Source)	Добавляет строку Source в строку Dest и возвращает указатель на Dest	string
char* strchr(const char* string, int ch)	Выполняет поиск символа с кодом ch слева направо в строке string; возвращает указатель на первое вхождение символа. Если символ не обнаружен, возвращает NULL (нулевой указатель)	string
char* strstr(const char* string, const char* Search)	Выполняет поиск строки ch слева направо в строке string; возвращает указатель на первое вхождение символа. Если символ не обнаружен, возвращает NULL (нулевой указатель)	string
char* strcpy(char *Dest, const char *Source)	Копирует строку Source в строку Dest и возвращает указатель на Dest	string
char* strncpy(char *Dest, const char *Source, size_t count)	Копирует не более чем count символов из строки Source в строку Dest и возвращает указатель на Dest	string
int strcmp(const char* string1, const char* string2)	Сравнивает строки string1, string2 лексикографически: возвращает –1, если string1< string2; 0, если равны; +1, если string1> string2	string
double atof (const char* str)	Преобразует строку str в вещественное число	stdlib
int atoi (const char* str)	Преобразует строку str в целое число	stdlib

Приложение 2. План лабораторных работ

Лабораторная работа 1. *Линейная и разветвленная программы*

Изучить пп. 1, 2, 3 и 4 мет. пособия.

Создать отдельную папку для сохранения своих программ на Visual C++.

Проверить в работе приведенные там примеры.

Выполнить задания 3.х и 4.х, где х – номер варианта, указанный преподавателем.

Лабораторная работа 2. *Циклические программы.*

Изучить пп. 5 и 6 мет.пособия.

Проверить в работе приведенные там примеры.

Выполнить задания 5.х и 6.х, где х – номер варианта, указанный преподавателем.

Лабораторная работа 3. *Массивы.*

Изучить пп. 7 и 8 мет.пособия.

Проверить в работе приведенные там примеры.

Выполнить задания 7.х и 8.х, где х – номер варианта, указанный преподавателем.

Лабораторная работа 4. *Функции . Перечислимый тип. Указатели.*

Изучить пп. 9, 10, 11 и 12 мет.пособия.

Проверить в работе приведенные там примеры.

Выполнить задание 9.х , где х – номер варианта, указанный преподавателем.

Лабораторная работа 5. *Обработка символьных строк и структур.*

Изучить пп. 13 и 14 мет.пособия.

Проверить в работе приведенные там примеры.

Выполнить задания 13.х и 14.х, где х – номер варианта, указанный преподавателем.

Лабораторная работа 6. *Классы.*

Изучить п. 15 мет. пособия.

Проверить в работе приведенные там примеры.

Выполнить задания А15.х и Б15.х, где х – номер варианта, указанный преподавателем.

Лабораторная работа 7. *Файлы.*

Изучить п. 16 мет.пособия.

Проверить в работе приведенные там примеры.

Выполнить задание А16.х , где х – номер варианта, указанный преподавателем.

Лабораторная работа 8. *Обработка файлов на С.*

Изучить п. 16 мет.пособия.

Проверить в работе приведенные там примеры.

Выполнить инд. задание Б16.х, где х – номер варианта, указанный преподавателем.

ГЛАВА 9. ПРИЛОЖЕНИЯ WINDOWS FORMS

ВВЕДЕНИЕ

Windows Forms – это набор средств для создания windows-приложений, выполняющихся в среде CLR (Common Language Runtime – общеязыковая исполняющая среда). **Форма** – это окно, служащее основой окна приложения или диалогового окна, в которое можно добавлять другие элементы управления, предназначенные для взаимодействия с пользователем. Пакет Visual C++ 2005 поставляется со стандартным набором более 60 элементов управления, которые можно использовать в формах. Многие стандартные элементы управления, такие как `Button`, представляющие кнопки, предназначенные для обработки щелчков мыши на них, или `TextBox`, которые позволяют вводить текст, реализуют простые функции взаимодействия с пользователем. Другие являются **контейнерами** – то есть могут содержать другие элементы управления. Например, `GroupBox` может содержать другие элементы управления вроде `Button` или `TextBox`, и его функция состоит просто в группировании элементов управления.

Форма и используемые с ней элементы управления представлены классом C++/CLI. Каждый класс обладает набором свойств, которые определяют поведение и внешний вид элемента управления или формы. Определение свойств элементов управления может выполняться:

- интерактивно при построении графического интерфейса пользователя с помощью средств IDE (Integrated Developement Environment – интегрированная среда разработки)
- изменение значений свойств может производиться также во время выполнения с помощью добавляемых в программу функций,
- либо с помощью кода, добавляемого в существующие функции.

При создании проекта приложения создается как окно приложения Windows Forms, построенное на основе класса Form, так и весь код, обеспечивающий отображение этого окна приложения. После создания проекта Windows Forms разработка приложения сводится к выполнению четырех отдельных операций:

Интерактивное создание графического интерфейса пользователя на вкладке `Form Design` (Конструктор формы), отображаемой

в панели Editor (Редактор), путем выбора элементов управления в окне Toolbox (Панель инструментов) и их помещения в форму. На этом этапе можно также создавать дополнительные окна форм.

Изменение свойств элементов управления и форм в окне Properties (Свойства) в соответствии с потребностями приложения.

Обработчики событий щелчков для элементов управления можно создавать, дважды щелкая на элементе управления на вкладке Form Design. В окне Properties элемента управления в качестве его обработчика события можно также определять существующую функцию.

Для удовлетворения потребностей приложения можно изменять и расширять классы, автоматически создаваемые в результате взаимодействия с вкладкой Form Design.

1. РАЗРАБОТКА ПРИЛОЖЕНИЯ

Как обычно, запускаем среду Visual Studio 2005 командой ***Пуск/Программы/Microsoft Visual Studio*** 2005***/ Microsoft Visual Studio*** 2005. Далее, приступаем к созданию проекта. Для создания проекта, как обычно, нужно дать команду ***File/New/Project*** (или нажать комбинацию клавиш **Ctrl-Shift-N**). В левой части возникшего окна (рис. 9.1) отображаются типы проектов, которые можно создавать. В данном случае в левой части выберем **CLR**, а на правой панели выберем **Windows Forms Application**.

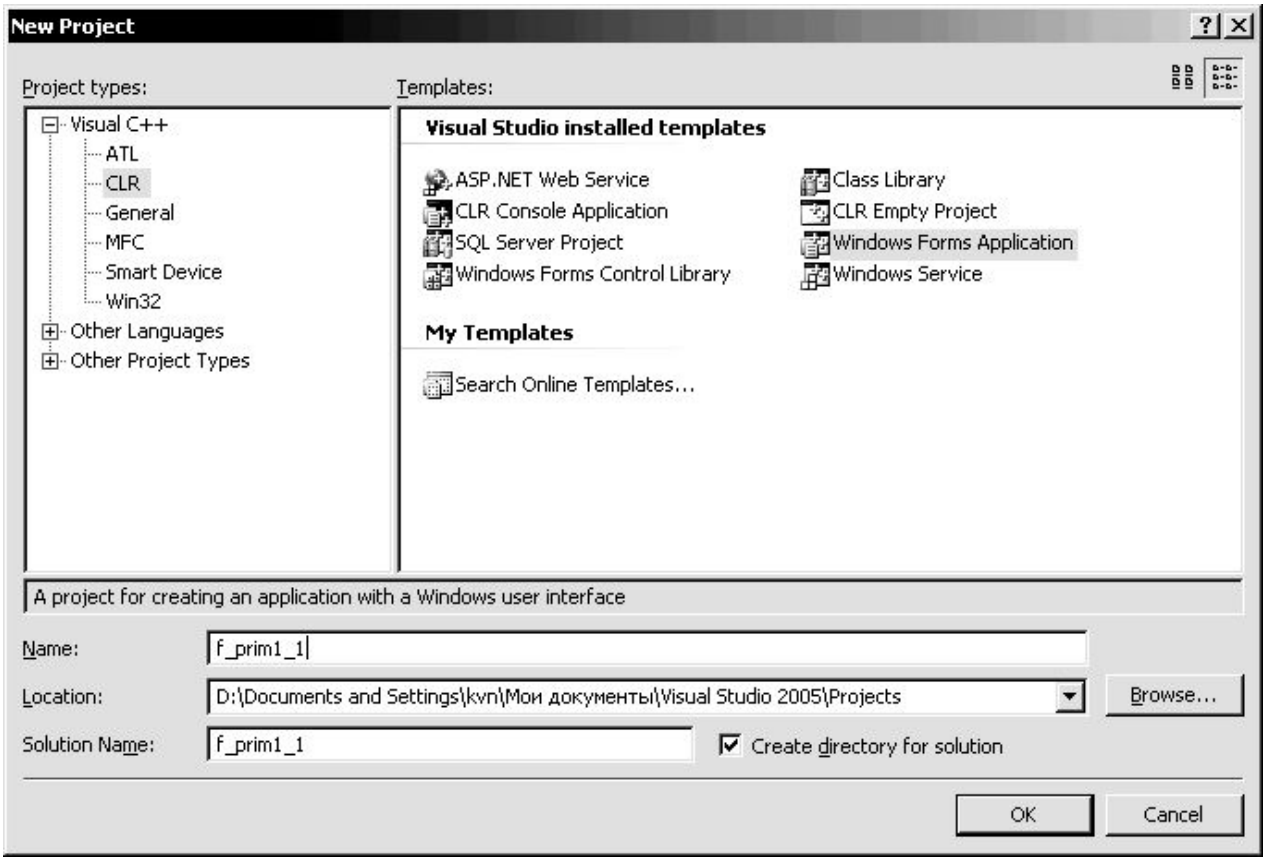


Рис. 9.1. Окно создания приложения Windows Forms

Ниже (в строке Name) ввести имя для проекта (зададим, например, **f_prim1_1**) и нажать ОК. Обратите внимание, что при этом должна быть установлена «птичка» **Create directory for solution** (создать каталог для проекта). В результате мастер создания приложений сгенерирует код приложения формы Windows и отобразит окно конструктора, содержащее форму, как она будет отображена приложением (см. рис. 9.2)

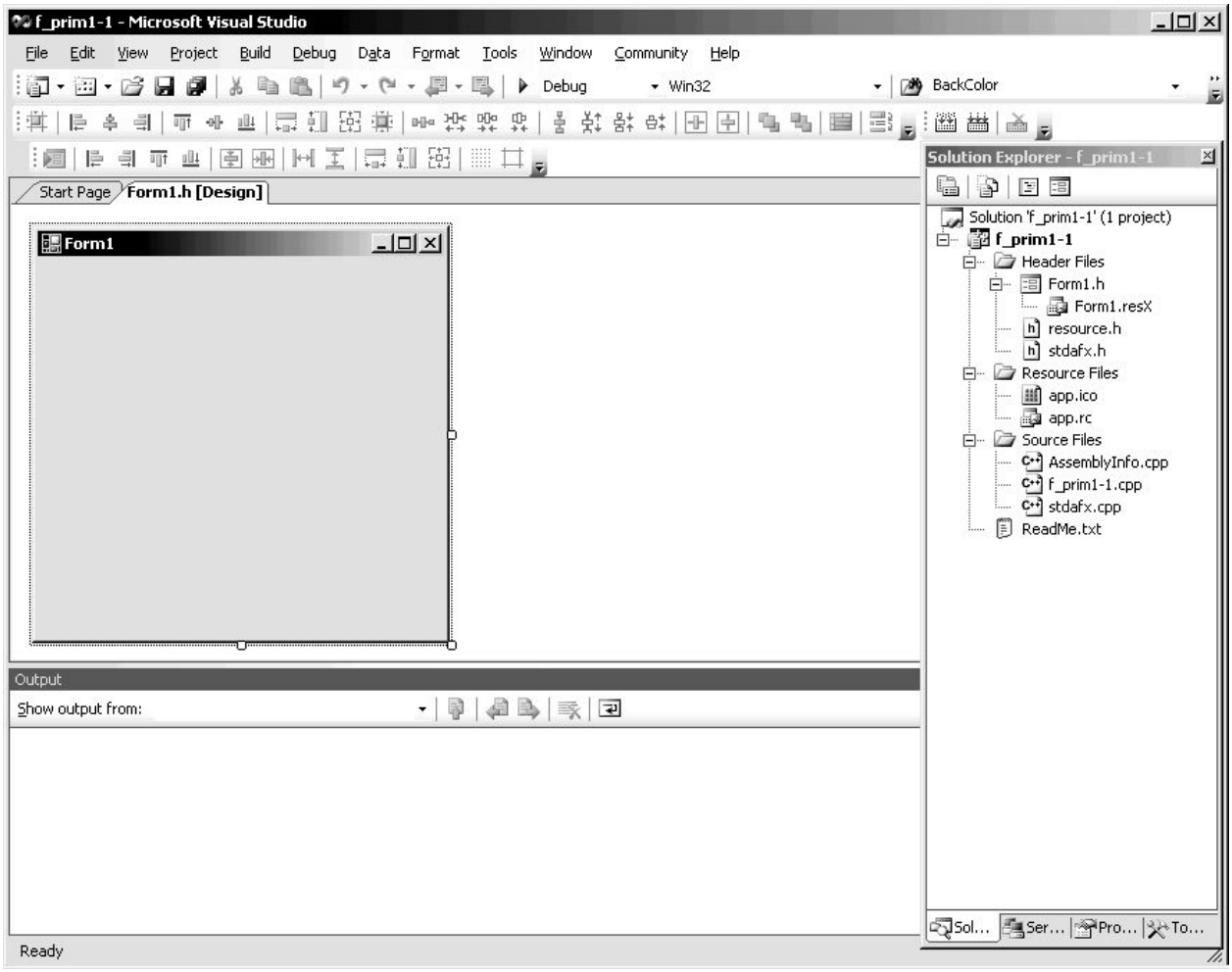


Рис. 9.2. Окно конструктора с пустой формой

Оказывается, что даже такое пустое приложение уже имеет код, хотя мы еще не создавали никаких описаний, функций и т.п. Для открытия окна кода достаточно щелкнуть по форме правой клавишей и мыши и в контекстном меню выбрать `View Code`. При этом появится вкладка `Form1.h`, где и будет располагаться код.

Код определяет класс Form1, представляющий окно приложения. Прежде всего, следует отметить, что он определен в собственном пространстве имен:

```
namespace f_prim11
{    using namespace System;
 using namespace System::ComponentModel;
 using namespace System::Collections;
 using namespace System::Windows::Forms;
 using namespace System::Data;
 using namespace System::Drawing;
```

Компиляция проекта приводит к созданию новой сборки, код которой относится к пространству имен, совпадающему с именем

проекта (в данном случае `f_prim11`). Пространство имен позволяет различать типы с одинаковыми именами в различных сборках, поскольку каждое имя типа уточняется именем конкретного пространства имен.

Существуют шесть рекомендаций к использованию пространств имен библиотеки .NET, охватывающих функциональные возможности, которые, скорее всего, потребуются в приложениях (см. табл. 9.1)

Таблица 9.1

Пространство имен библиотеки .NET

Пространство имен	Содержимое
System	Содержит классы, которые определяют типы данных, используемые во всех приложениях CLR. Оно содержит также классы событий и обработчиков событий, исключения и классы, поддерживающие функции общего применения.
System::ComponentModel	Содержит классы, которые поддерживают работу компонентов графического интерфейса пользователя в приложениях CLR
System::Collections	Содержит классы коллекций, предназначенные для всевозможной организации данных, в том числе классы определения списков, очередей и стеков
System::Windows::Forms	Содержит классы, которые поддерживают использование в пиложении средств Windows Forms
System::Data	Содержит классы, которые поддерживают набор компонентов ADO.NET, используемый для доступа и обновления источников данных
System:: Drawing	Содержит классы, которые поддерживают основные графические операции, подобные рисованию на форме или компоненте

Далее в окне кода приводятся строки вида:

```
public ref class Form1: public System::Windows::Forms::Form
     {
     public:
          Form1(void)
          {    InitializeComponent();
               //
```

```
            //TODO: Add the constructor code here
            //
        }

    protected:
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        ~Form1()
        {    if (components)
            {          delete components;              }
        }
```

Класс `Form1` – производный от класса `Form`, который определен в пространстве имен `System::Windows::Forms`. Класс `Form` представляет окно приложения или диалогового окна, а класс `Form1`, который определяет окно для пространства имен `f_prim11`, наследует все члены класса `Form`.

Этому разделу на странице кода предшествует такой комментарий:

```
/// WARNING: If you change the name of this
//  class, you will need to change the
/// 'Resource File Name' property for the
//  managed resource compiler tool
/// associated with all .resx files this class //  de-
pends on.
/// Otherwise, the designers will not be able //  to in-
teract properly with localized
/// resources associated with this form.
```

Раздел в конце класса `Form1` содержит определение функции `InitializeComponent()`. Эта функция вызывается конструктором для определения окна приложения и любых компонентов, добавляемых в форму. Согласно приведенным комментариям, этот раздел кода не следует модифицировать с помощью редактора кода, поскольку он обновляется автоматически при интерактивном изменении окна приложения. При использовании средств `Form Design` важно соблюдать рекомендации этого комментария и не изменять вручную автоматически сгенерированный код. В противном случае рано или поздно это приведет к возникновению ошибок. Конечно, весь код для приложения `Windows Forms` можно создать с нуля, но применение возможностей `Form Design` для ин-

терактивного создания графического интерфейса пользователя приложения значительно ускоряет работу и снижает вероятность ошибок. Тем не менее, это не означает, что не нужно знать, как работает этот код.

Вначале код функции `InitializeComponent()` выглядит следующим образом:

```
void InitializeComponent(void)
          {     this->components =
 gcnew System::ComponentModel::Container();
                this->Size = System::Drawing::Size(300,300);
                this->Text = L"Form1";
                this->Padding = System::Windows::Forms::Padding(0);
                this->AutoScaleMode =
 System::Windows::Forms::AutoScaleMode::Font;          }
```

Вначале `components` класса `Form1` унаследован от базового класса, и его задача – отслеживание компонентов, постепенно добавляемых в форму. Первый оператор сохраняет в члене `components` дескриптор объекта `Container`, представляющего коллекцию, которая хранит компоненты графического интерфейса пользователя в списке. Каждый новый компонент, добавляемый в форму с помощью средств `Form Design`, добавляется в этот объект `Container`.

Остальные операторы функции `InitializeComponent()` определяют свойства объекта **Form1**. Ни одно из этих свойств не следует изменять непосредственно в коде, но их значения можно выбирать посредством окна `Properties` (см. рис. 9.3) формы.

Если переключится обратно на вкладку `Form1.h[Design]` в окне редактора и щелкнуть правой клавишей мыши, выбрать `Properties`, то и увидим окно свойств формы (рис. 9.3).

Чтобы получить представление о доступных возможностях, имеет смысл просмотреть список свойств формы. Щелчок на любом из них приводит к отображению описания в нижней части окна.

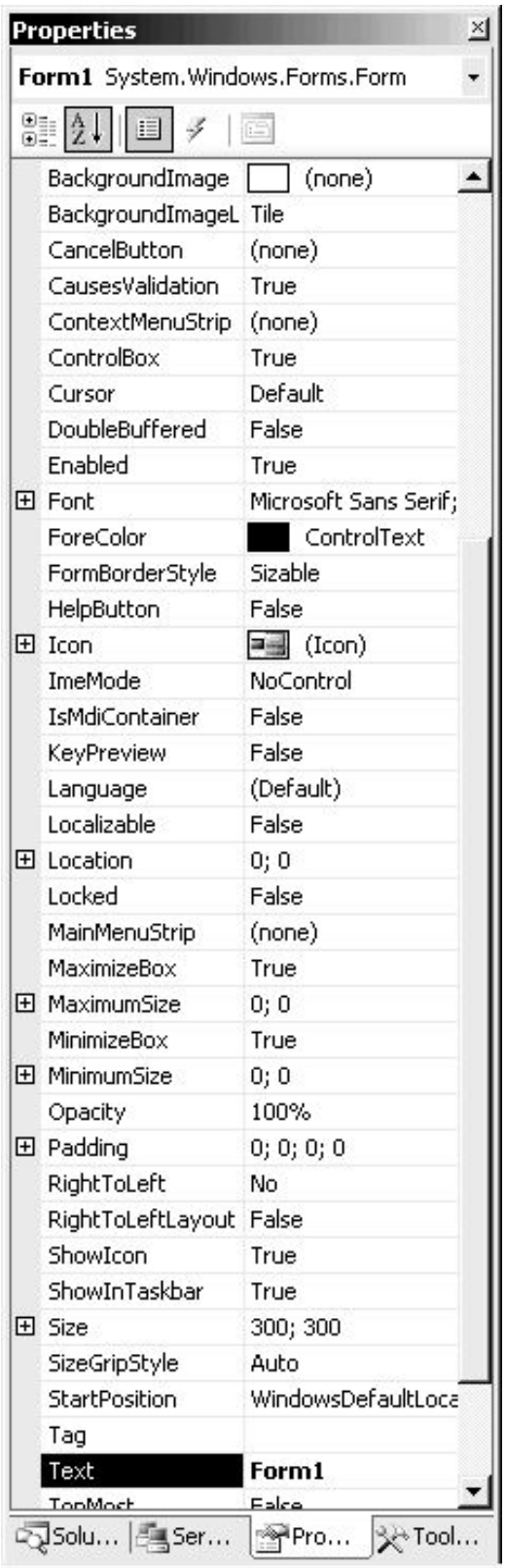
Например, если сменить значение свойства **Text** (вместо *Form*1 написать, например, «Первый проект» и нажать клавишу Enter), то изменится надпись в заголовке окна формы на «Первый проект».

Свойства, слева от которых отображен символ +, обладают несколькими значениями, и щелчок на этом символе отображает эти значения. Например, если выберем свойство **Size** (щелкнем по +), то

появятся свойства **Width** (ширина) и **Height** (высота). Там можно задать другие числовые значения.

Если вернуться к вкладке `Form1.h` в окне Editor, то увидим, что код функции `InitializeCom-ponent()` изменился, отражая внесенные изменения свойств.

Рассмотрим порядок разработки конкретного приложения.



Пример 1-1. *Создать проект с тремя кнопками:*

1) для вывода надписи на форме,

2) для смена цвета формы (на красный и обратно),

3) для закрытия формы (см. рис. 9.4).

Для размещения кнопок на форме включим окошко `Toolbox` – см. рис. 9.5, *а* (командой ***View/Toolbox*** или клавишами **Ctrl+Alt+X**). Выберем там кнопку `Button` (щелкнем по ней левой клавишей мыши) и затем щелкнем (также левой кнопкой) в нужном месте формы. Повторим это еще два раза, т.к. нужно три кнопки. Затем меняем надписи на них. Для этого щелкнем правой клавишей мыши, например, по первой кнопке и выберем пункт Properties. При этом откроется окошко `Properties` (свойства) для данного объекта (рис 9.5, *б*) и здесь свойство **Text** сменим на «Вывод надписи». Проделаем аналогичную операцию для двух других кнопок.

Рис. 9.3. Окно Properties объекта Form1

Рис. 9.4. Форма для проекта примера 1-1.

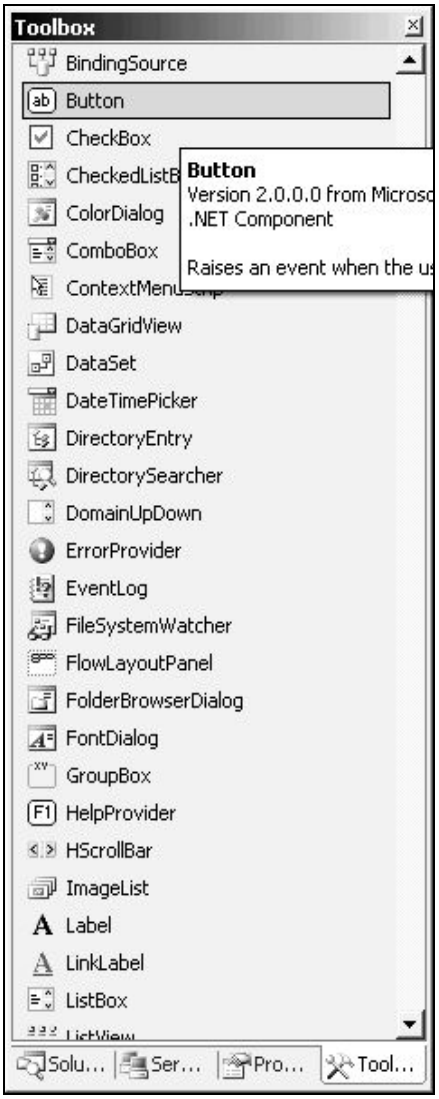


Рис. 9.5, *а*. Панель инструментов Toolbox с выбранным Button

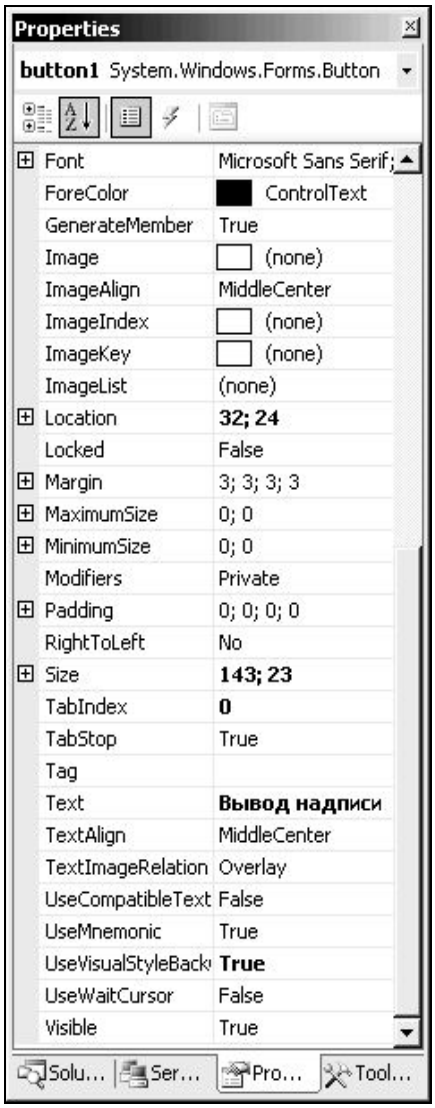


Рис. 9.5, *б*. Панель свойств объектов Properties для кнопки Button

Затем разместим на форме объект `Label` (из `Toolbox`) – он нужен для вывода надписи на форму. Его разместим справа от кнопки «Вывод надписи». Его свойство **Text** просто очистим.

Обратите внимание еще на одно важное свойство объекта – свойство **Name** (имя). По умолчанию у кнопок свойство **Name** будет **button1**, **button2**, **button3**. У надписи имя будет **label1**. Эти имена можно поменять, но пока в этом нет необходимости.

Теперь для каждой кнопки будем программировать *событие* (event). Для этого выделим первую кнопку и в окошке Properties перейдем на вкладку *Events* – щелкнем по значку «молния» (см. рис. 1.5***б***) и там, в строке **Click** (событие при щелчке мышью по данной кнопке), дважды щелкаем мышью – событие будет названо **button1Click**. Откроется окно для ввода кода программы (рис. 9.6). Обратите внимание, что заголовок функции, а также открывающая и закрывающая скобки появились автоматически. И здесь вводим всего одну строку:

```
label1->Text=«Кнопка работает!»;
```

Аналогично создадим событие для кнопки «Смена цвета формы» (будем менять цвет формы на красный **Red** и обратно – на исходный **ButtonFace**):

```
if (this->BackColor == Color::Red )
        this->BackColor= SystemColors::ButtonFace;
            else this->BackColor= Color::Red;
```

И так же создадим событие для кнопки «Закрыть»:

```
this->Close();
```

Результат окна с кодом приведен на рис. 9.6.

Обратите внимание, что здесь активно используются понятия объектно-ориентированного программирования. Указываются классы `Color, SystemColors` и используется `this` – указание на текущий активный объект (в данном случае на форму).

Теперь проект следует сохранить (***File/Save All…*** или щелкнуть по кнопке ). Для запуска программы на выполнение, нужно сначала провести компиляцию и компоновку – дать команду ***Build/Build Solution*** (или клавиша **F7**). Если все в порядке – в окне **Output** должно быть «0 failed» (см. рис. 9.6), то собственно запускаем на исполнение – клавишами **Ctrl-F5** (***Debug/Start Without Debugging***). Если в процессе создания программы не было ошибок,

то мы получим действующее приложение. Можно будет пощелкать по кнопкам и убедиться в их действии:

- при щелчке по кнопке «Вывод надписи» появится на форме надпись: Кнопка работает! (в том месте, где мы разместили объект **Label**);
- при щелчке по кнопке «Смена цвета формы» форма поменяет цвет на красный, а при повторном щелчке по этой же кнопке цвет вернется – станет серым;
- при щелчке по кнопке «Закрыть» приложение закрывается, и мы вернемся в среду.

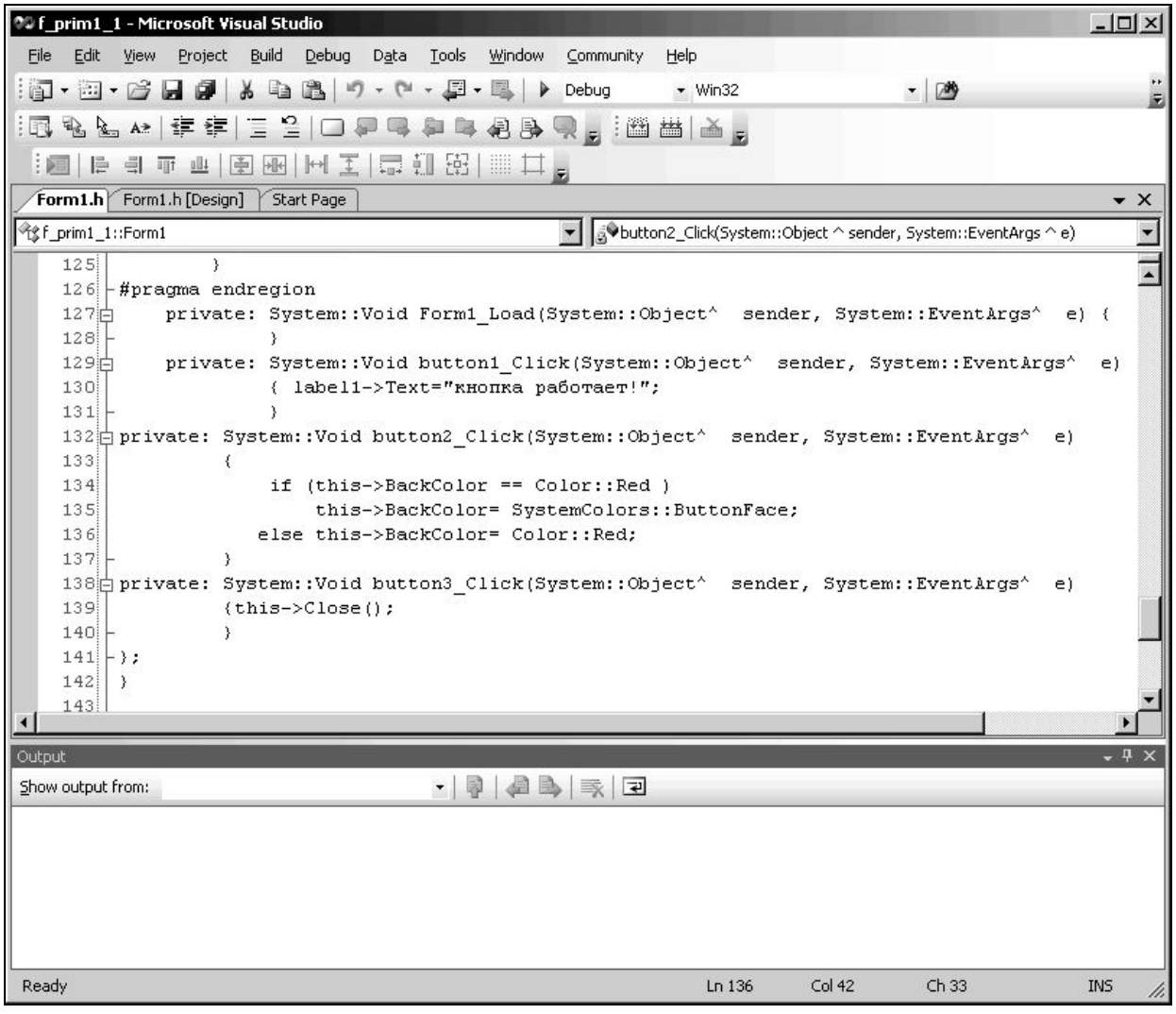


Рис. 9.6 Окно кода программы примера 1-1.

2. ОКНО ВВОДА ТЕКСТА Textbox и РАДИОКНОПКА RadioButton. БОЛЕЕ СЛОЖНЫЙ ПРОЕКТ

Среди компонентов на панели *Toolbox* есть еще две наиболее часто использующиеся – окно ввода текста **Textbox** и радиокнопка-переключатель **RadioButton**. **Textbox** позволяет вводить и редактировать текст и использовать его в процессе работы приложения. **RadioButton** предназначена для переключений в процессе работы приложения, для выбора вариантов работы приложения и т.д. У нее основное свойство – это свойство *Checked* логического типа (значение **false** или **true**).

Пример 2-1. Составить проект для начисления зар.платы, исходя из задаваемого на форме оклада, процента надбавки и предусмотреть варианты расчета – с премией (с указанием %) или без премии. (Не забудем, что в конце необходимо учесть подоходный налог в 13%) .

Вид приложения см. на рис. 9.7 *а*.

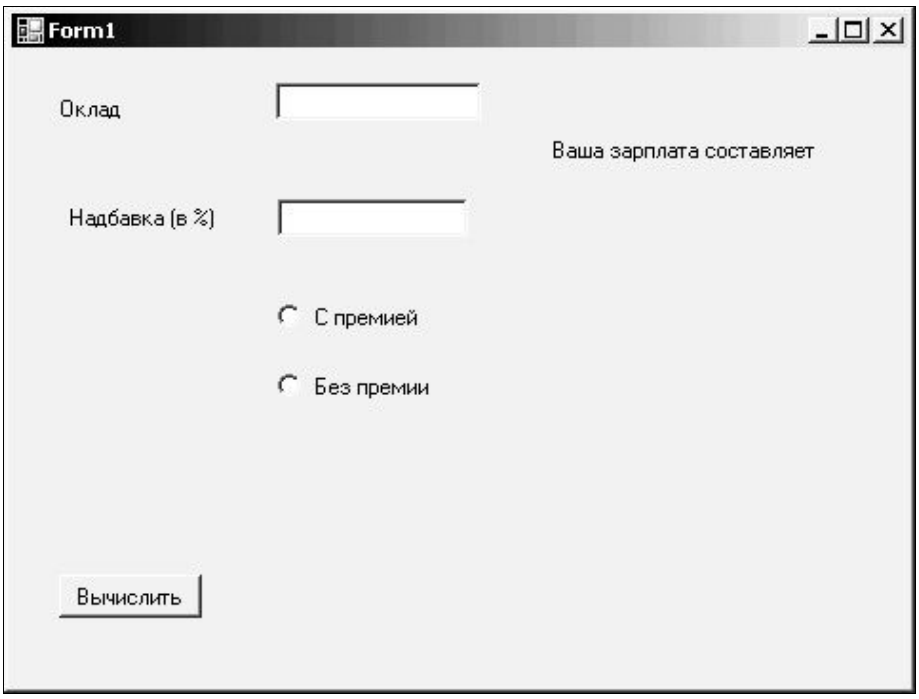


Рис. 9.7 *а*. Вид формы приложения для начисления зар.платы

Итак, создадим новый проект (***File/New/Project***), выберем **CLR** и **Windows Forms Applications**. Разместим на нем нужные элементы управления (см. рис. 9.7 *б*).

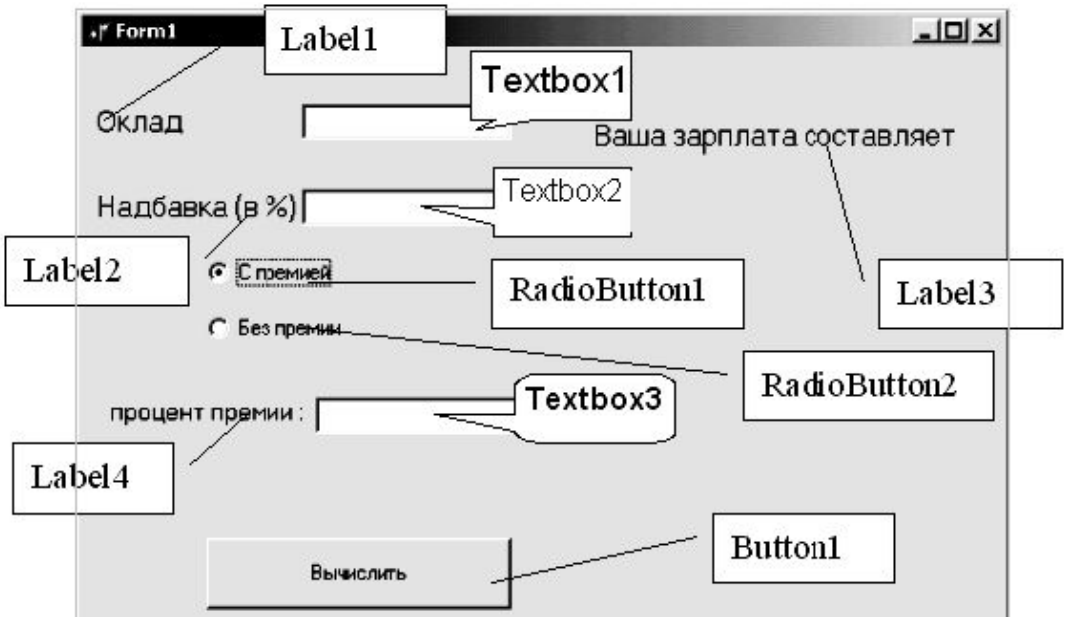
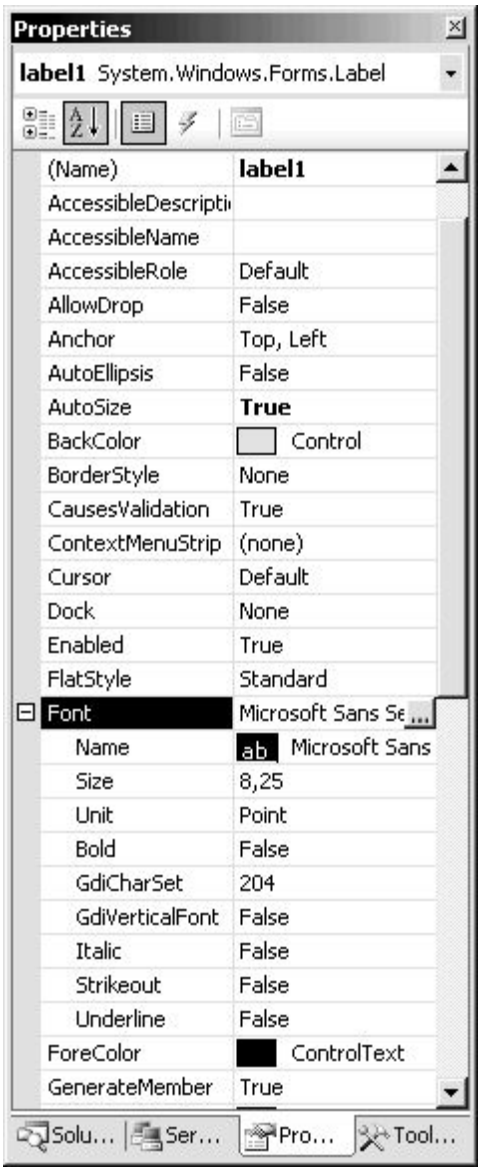


Рис. 9.7 *б*. Вид формы приложения для начисления зар.платы при наличии премии



Для объектов **Label** сменим надписи и сделаем их более крупными. Для этого выберем свойство *Font* для этих надписей (см. рис. 9.8, *а*), щелкнем там по многоточию (…), появится очередное окно (см. рис. 9.8, *б*) и там выберем больший размер шрифта (а можно выбрать и тип шрифта).

Окошки **Textbox** просто очистим – выберем у них свойство *Text* и очистим его. Поменяем надписи (*Text*) у кнопок **RadioButton** и **Button**.

Теперь приступим к созданию кода – соответствующих функций обработки событий для того или иного объекта.

Рис. 9.8, *а*. Свойство Font в окне Properties для объекта Label1

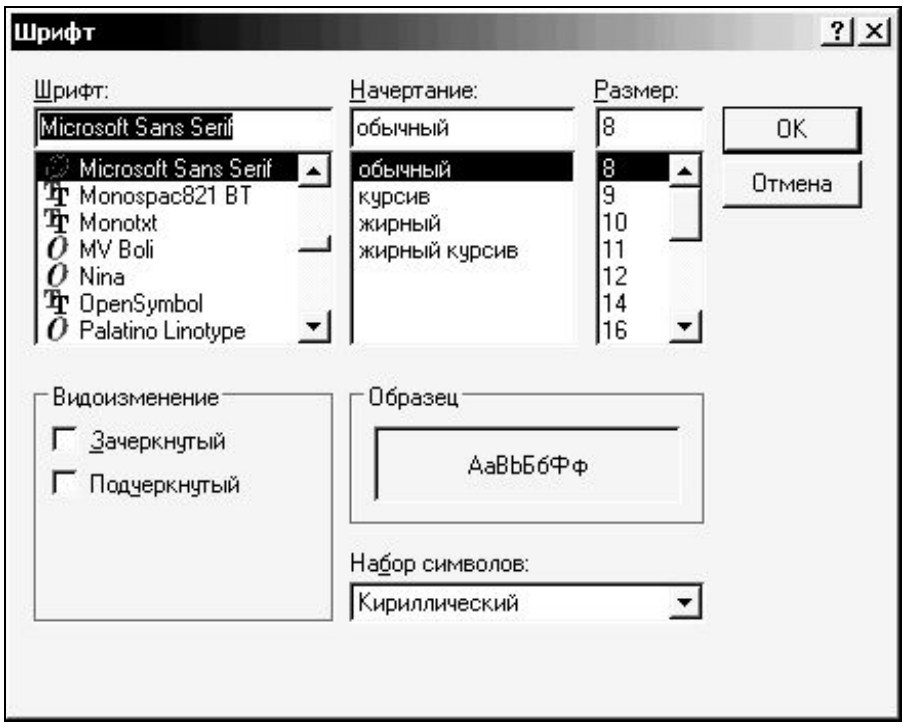


Рис. 9.8, *б*. Выбор типа и размера шрифта (после нажатия … в строке Font)

Сначала позаботимся о том, чтобы при запуске приложения надпись «процент премии» и окошко **Textbox** для ввода этого процента премии *не были видны* (они потом появятся только при выборе радиокнопки «С премией»). Для этого создадим событие *Load* для формы – выберем форму (щелкнем по чистому месту формы) и в **Properties** перейдем на вкладку **Events** и в строке *Load* щелкнем два раза. Откроется окно кода с процедурой Form1_Load. Введем там такие строки:

```
textBox3->Visible=false;
    label4->Visible=false;
```

Здесь мы использовали свойство *Visible*, которое принимает два значения: **false** (объект невидим) и **true** (объект видим).

Помните: событие *Load* срабатывает в момент запуска приложения.

Начнем программировать событие *Click* для кнопки «Вычислить». Очевидно, что сначала нужно будет считать данные из окошек **Textbox**, а там информация дается в текстовом виде. Поэтому нужно будет текстовую информацию (`Textbox->Text`) преобразовать в число, в данном случае в вещественное число. Делается это следующим образом:

```
float::Parse(textBox1->Text)
```

Обратно – для вывода результата на форму – нужно преобразовать вещественное число в строку с помощью метода `ToString()`.

Итак, выберем на форме кнопку и выберем событие *Click*. Введем туда следующий код программы:

```
private: System::Void button1_Click (System:: Object^
sender, System::EventArgs^  e)
  {
  float okl; // переменная для величины оклада
  float proc; // переменная для процента надбавки
  float zar; // переменная для зарплаты
  float procprem; //переменная для процента премии
  okl=float::Parse(textBox1->Text);
  proc=float::Parse(textBox2->Text);
  zar= okl+okl*proc/100 ;
  if (radioButton1->Checked)
  // если выбрано "с премией"
       { procprem=float::Parse(textBox3->Text);
         zar=zar+zar*procprem/100; }
  zar=zar - 0.13*zar;
  label3->Text=
  label3->Text+"\n"+zar.ToString()+" руб.";          }
```

Чтобы при выборе радиокнопки «С премией» появлялись надпись и окошко для ввода процента премии, нужно для **RadioButton2** создать событие *Click* вида:

```
if (radioButton1->Checked)
    {textBox3->Visible=true;
                    label4->Visible=true; }
```

Однако если снова выберем «Без премии», то нужно чтобы надпись и окошко для ввода процента премии исчезали. Это получится, если создать событие *Click* для кнопки **RadioButton1** вида:

```
if (radioButton2->Checked)
    {textBox3->Visible=false;
                    label4->Visible=false; }
```

> ***Замечание***. Приложения, содержащие окошко редактирования **Textbox**, следует усовершенствовать таким образом, чтобы предусмотреть «защиту от дурака» – установить ограничение на ввод данных определенного вида. Для этого используют событие *KeyPress* данного объекта **textbox**. Например, если при выполнении проекта в окошко **textbox1** должны вводится только числа, то такое событие имеет вид:

```
private: System::Void textBox1_KeyPress (System:: Object^
sender, System:: Windows::Forms:: KeyPressEventArgs^  e)
     { if (! ((e->KeyChar>='0') && (e->KeyChar<='9')
          || (e->KeyChar==',')) ) e->KeyChar=Char(0); } };
```

Обратите внимание, у функции для этого события имеется параметр `e->KeyChar`, контролируя который мы и задаем ограничение на ввод значений. В данном случае: если нажимаемая клавиша не лежит в интервале от 0 до 9 и не является десятичной запятой, то символ не отображается (Char(0)).

3. ДИНАМИЧЕСКИЕ ССЫЛКИ НА ОБЪЕКТЫ

3.1 Понятие о динамических ссылках.

Помимо обычных непосредственных ссылок на объекты допустимы и так называемые динамические ссылки на объекты. Обычная ссылка, например, на свойство *TabIndex* кнопки **Button** имеет вид: `Button->TabIndex`. Аналогичная динамическая ссылка имеет вид `(Button^)sender->TabIndex` . В чем преимущество (и необходимость) динамических ссылок? В том, что можно подразумевать не один объект, а целую группу объектов. В результате можно назначить одну и ту же функцию в качестве события для целой группы объектов. Продемонстрируем на примере.

Пример 3-1. Организовать форму с семью кнопками, в зависимости от нажатия на которые будет меняться цвет формы (см. рис. 9.9). Причем для реализации события Click для всех кнопок использовать одну и ту же процедуру.

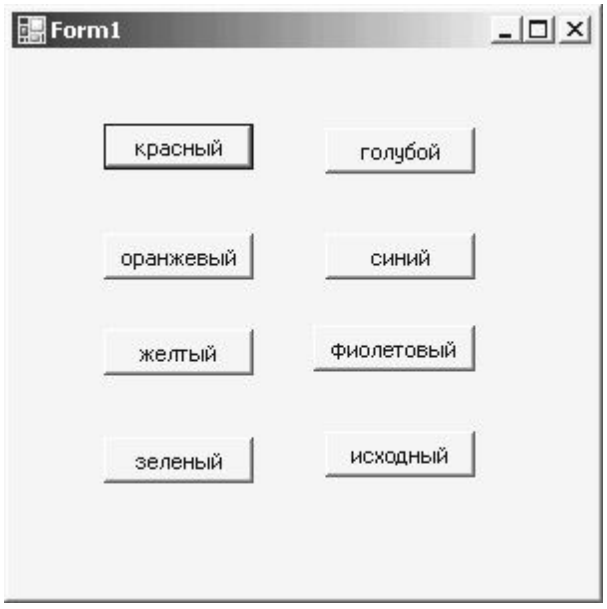


Рис. 9.9

Итак, здесь потребуется разместить семь кнопок, сменить их названия и, главное, у каждой из них указать свой номер и свойства *TabIndex*: для кнопки «красный» зададим 1, для кнопки «оранжевый» 2 и т.д. Затем создадим событие для кнопки «красный» следующего вида:

```
switch (((Button^ )sender)->TabIndex)
     {     case 0:    this->BackColor=Color::Red; break;
           case 1: this->BackColor=Color::Orange; break;
           case 2: this->BackColor=Color::Yellow; break;
           case 3: this->BackColor=Color::Green; break;
           case 4: this->BackColor=Color::SkyBlue; break;
           case 5: this->BackColor=Color::Blue; break;
           case 6: this->BackColor=Color::Violet; break;
           case 7: this->BackColor=SystemColors::Control; break;
     }
```

Далее в окне *Properties* перейдем на вкладку *Events* и укажем там, в строке *Click* (везде, для каждой кнопки!) ту же процедуру **button1Click**.

Остается запустить приложение (естественно, при этом сохранить проект и программу) и проверить кнопки в действии.

3.2. Программа «Калькулятор»

Пример 3-2. Создать программу «Калькулятор вида (рис. 9.10). Обеспечить выполнение нужных операций с отображением задаваемых цифр и результата на индикаторе.

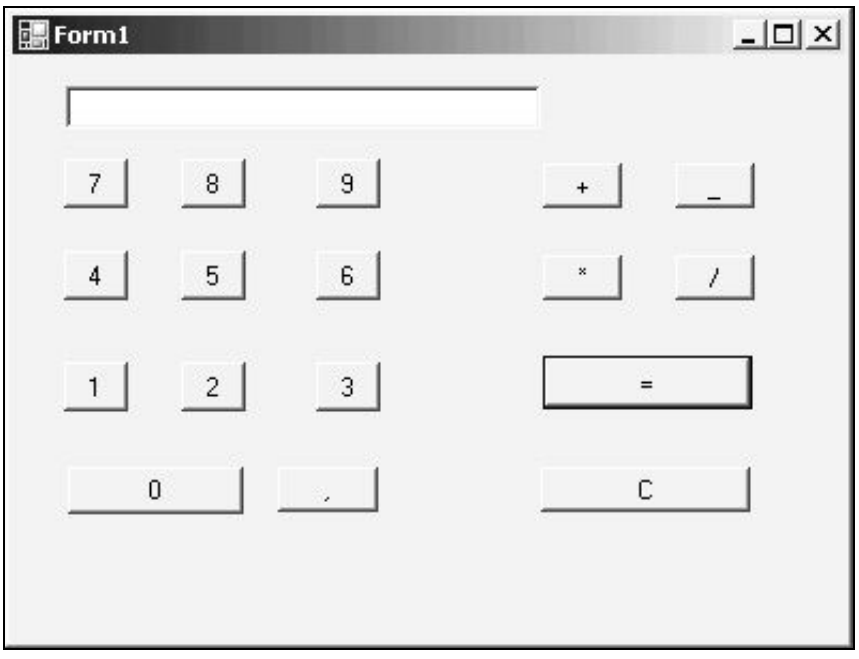


Рис. 9.10

Разместим соответствующие цифровые кнопки. Выполним надписи. Для каждой из них укажем свойство *TabIndex*, равным цифре на кнопке.

В окне кода после строк:

```
public ref class Form1 : public System::Windows:: Forms:: Form
    {   public:
```

опишем глобальные переменные:

```
        int f;   // признак для проверки
    // f=0 — вводится первый операнд операции
    // f=1 – вводится второй
int oper;  // код операции:
    //0 - "=", 1- "+", 2 - "-", 3 - "*", 4 -"/"
float accum; // результат операций
```

Создадим для кнопки «1» для события *Click* функцию следующего вида:

```
private: System::Void button9_Click(System::Object^
sender, System:: EventArgs^  e)
    // функция отображения символа цифры
{   Char Ch;  // символ цифры
 // преобразование цифры в ее символ
 // т.е. код 0 есть 48, код 1 есть 49 и т.д.
Ch=Char(((Button^)sender)->TabIndex+48);
switch ( ((Button^)sender)->TabIndex)
 {case 0:
 case 1:
 case 2:
 case 3:
 case 4:
 case 5:
 case 6:
 case 7:
 case 8:
 case 9: if (f = =0) {textBox1->Text=Ch.ToString();f=1;}
     else textBox1->Text=textBox1->Text+Ch.ToString(); }
 f=1;  }
```

Обратите внимание, что мы сразу предусмотрели и все другие цифры. Поэтому можем назначить эту же процедуру и всем остальным цифровым кнопкам, т.е. перейдем на вкладку *Events* и, последовательно выделяя каждую кнопку, событию *Click* для нее назначим ту же функцию `button1Click`.

Аналогично разместим кнопки для операций и для кнопки очистки **C**. Для кнопок операций также укажем свойство *TabIndex* следующим образом: 0 – для «=», 1 – «+», 2 – «-», 3 – «*», 4 – «/».

Создадим процедуру, например, для кнопки «+»*:*

```
private: System::Void button12_Click (System::Object^  sender, System::
EventArgs^  e)
   { // функция выполнения операции
      float numb;
      numb=(float::Parse)(textBox1->Text);
      if (f==0) oper=((Button^)sender)->TabIndex;
      else
      {  switch(oper)
         { case 0: accum=numb; break;
           case 1: accum=accum+numb; break;
           case 2: accum=accum-numb; break;
           case 3: accum=accum*numb; break;
           case 4: accum=accum/numb; break;
           }
           oper=((Button^)sender)->TabIndex;
           f=0;
           textBox1->Text=(accum).ToString();
        }
   }
```

И также назначим эту функцию для остальных кнопок с операциями.

Создадим отдельно функцию для *кнопки* «*C*»*:*

```
private: System::Void button17_Click(System::Object^  sender,
System::EventArgs^  e)
{     textBox1->Text="0"; oper=0; accum=0;   }
```

И создадим для кнопки «,» такую функцию:

```
private: System::Void button11_Click(System::Object^
sender, System::EventArgs^ e)
{    textBox1->Text=textBox1->Text+»,»;      }
```

Кроме того, создадим функцию, срабатывающую в момент создания формы:

```
private: System::Void Form1_Load(System::Object^ sender,
System::EventArgs^ e)
              {    textBox1->Text=««;
               oper=0;          }
```

Работающий калькулятор готов!

4. ИСПОЛЬЗОВАНИЕ ТАЙМЕРА. КОМПОНЕНТ CHECKBOX

4.1 Таймер

До сих пор рассматривали компоненты, которые непосредственно видны на форме и выполняют видимую роль.

Однако, есть еще и компоненты, которые выполняют свою работу, не будучи видными, для постороннего глаза. Одним из таких компонентов является **Timer (таймер)**. В его задачу входит обрабатывать какое-либо сообщение через определенные интервалы времени. Хотя этот компонент и невидим в процессе работы программы, у него тоже есть свои события и свойства. Вернее, событие одно – **Tick**, которое определяет, что должна делать программа, когда истечет заданный интервал.

Самое важное свойство таймера – **Interval** (Интервал). Оно указывает, когда (через какое время) в следующий раз таймер должен сработать. Промежуток времени задается в тысячных долях секунды – миллисекундах. По умолчанию, свойство **Interval** содержит число 1000, следовательно, таймер будет срабатывать каждую секунду.

Рассмотрим пример.

Пример 4.1. Создать программу-игру «Прыгающая кнопка» (рис. 9.11): кнопка будет прыгать по поверхности формы с заданным интервалом и следует попасть по ней (щелкнуть на ней мышью). Предусмотреть возможность ускорить и замедлить движение кнопки.

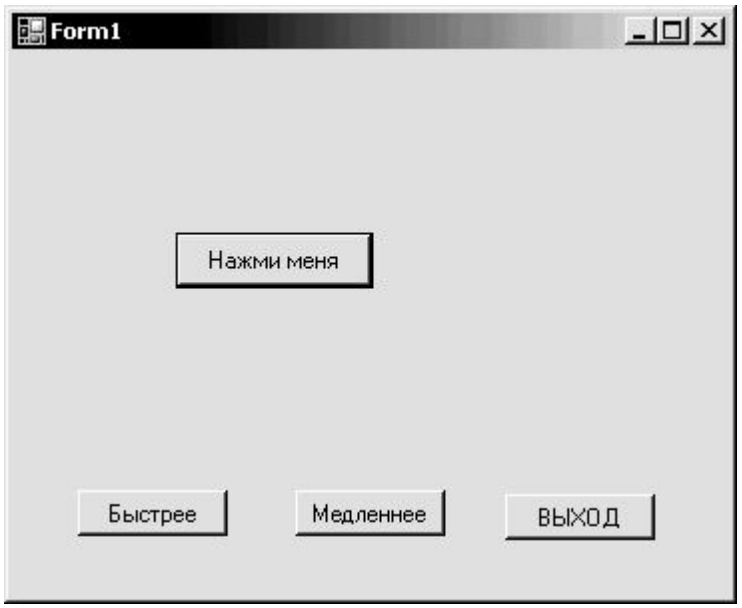


Рис. 9.11

Итак, создадим новый проект и разместим, прежде всего, объект **Timer**. Обратите внимание, что он разместится не на самой форме, а сразу под ней. Поместим также нужное число кнопок и сменим на них надписи. Давайте сделаем так, чтобы через каждые пол-секунды кнопка прыгала куда-нибудь. А задача пользователя – ее поймать. Свойство **Interval** объекта **Timer** сделайте равным 500.

Затем дважды щелкните по значку **Timer**, чтобы открыть обработчик события. Начинаем писать код:

```
private: System::Void timer1_Tick(System::Object^
sender, System::EventArgs^  e)
             {int x,y;
        // кнопка может прыгать по всей длине формы
       x=rand()%((this->Size.Width)-100);;
       // кнопка может прыгать по всей ширине формы
       y=rand()%(this->Size.Height-100);
         button1->Location=Point(x, y);
                     }
```

Обратите внимание, что положение кнопки на форме (координаты ее левой верхней точки *x*,*y*) задаются с помощью датчика случайных чисел. Дополнительный сдвиг на 100 предусмотрели, чтобы кнопка своими краями не выскакивала за пределы формы.

Когда нужно закончить игру? Когда пользователь щелкнет по кнопке. То есть для кнопки произойдет событие **Click**. Его и будем программировать. Щелкните два раза по кнопке **button1**, чтобы создать процедуру обработки щелчка. Введите следующий код:

```
private:    System::Void    button1_Click(System::Object^
sender, System::EventArgs^  e)
          {   timer1->Enabled=false;
              button1->Text="ÓÐÀ!" ;
          }
```

Осталось доработать кнопку **Выход**, а также кнопку **Медленнее** и кнопку **Быстрее**, при нажатии на которые увеличивается или уменьшается интервал таймера. Функции события **Click** для них имеют вид:

```
private: System::Void button2_Click(System::Object^
sender, System::EventArgs^  e)
          { //кнопка «Быстрее»
              timer1->Interval=timer1->Interval/2;
          }
private: System::Void button3_Click(System::Object^
sender, System::EventArgs^  e)
          { //кнопка «Медленнее»
              timer1->Interval=timer1->Interval*2;
          }
private: System::Void button4_Click(System::Object^
sender, System::EventArgs^  e)
```

```
{ //кнопка «Выход»
   this->Close();          }
```

И остается проверить в работе.

Пример 4.2. Создадим электронные часы, на которых будет отображаться текущее время, дата и день недели (рис. 9.12).

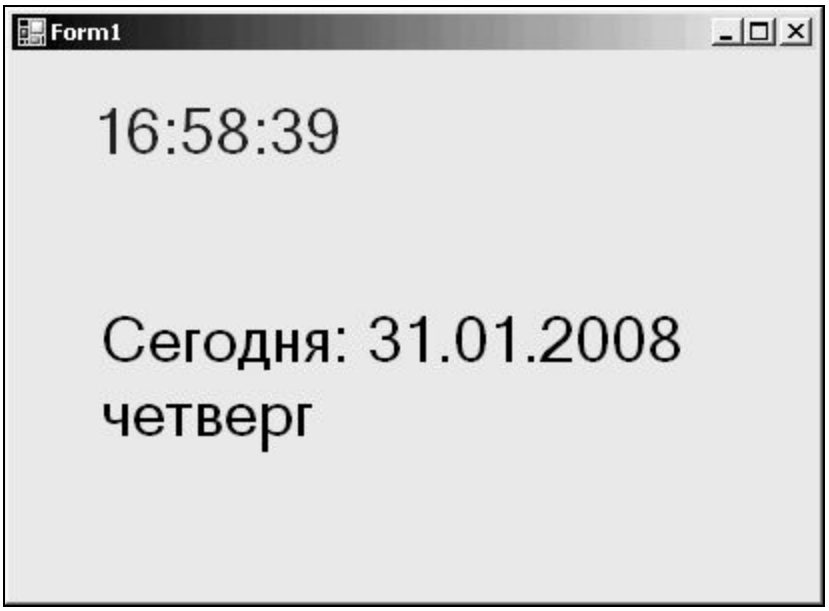


Рис. 9.12

Создадим новый проект и разместим на форме два объекта **Label**. сразу зададим шрифт побольше для этих надписей (например, 24 кегля). Для этого, напомним, нужно выделить этот объект и в окне Инспектора объектов выбрать свойство Font, щелкнуть там по многоточию и в возникшем окне выбрать тип и размер шрифта. Поместим также объект **Timer**.

Далее нужно создать события. Одно событие для формы **Form_Load** (выбрать форму и на вкладке *Events* выбрать Load):

```
private: System::Void Form1_Load(System::Object^ sender,
System::EventArgs^  e)
{ int dn; String^ named;
timer1->Interval = 1000;//период сигналов таймера 1с
  timer1->Enabled = true; // пуск таймера
  //получить дату
label2->Text=
"Сегодня: " +  DateTime::Now.Today.ToShortDateString() ;
  // вывести день недели
dn= (int)DateTime::Now.DayOfWeek ;
  switch ( dn)
  {case 0: named="воскресенье"; break;
```

```
   case 1: named="понедельник"; break;
   case 2: named="вторник";    break;
   case 3: named="среда";     break;
   case 4: named="четверг";   break;
   case 5: named="пятница";   break;
   case 6: named="суббота";   break;
   }
  label2->Text+=" \n"+named;
             }
```

Второе событие создадим для **Timer** – событие Tick:

```
private: System::Void timer1_Tick(System::Object^  sender,
System::EventArgs^  e)
     { // получить системное время
     label1->Text = DateTime::Now.ToLongTimeString();
     }
```

Обратите внимание, что для получения даты и времени служит класс `DateTime` и его элемент `Now`. Для получения сегодняшней даты применяют метод `Today`. А для перевода даты в строку (причем в сокращенном представлении даты) применяют метод `ToShortDateString()`. Для перевода времени в строку используют метод `ToLongTimeString()` Для получения номера дня недели применяют метод `DayOfWeek`, он дает номер дня недели в «американском» виде – у них счет дней недели идет с воскресенья: 0 – воскресенье, 1 – понедельник и т.д. Далее, используя структуру множественной развилки, получаем название дня недели.

4.2. Компонент CheckBox

Данный компонент (а это просто окошко, в котором ставят «галочку») удобно использовать в том случае, когда требуется ответить: да или нет. У этого компонента есть свойство **Checked**, которое принимает значение **true** (истина) или **false** (ложь) в зависимости от того стоит галочка на этом компоненте (**true**) или нет (**false**).

Пример 4.3. Создать приложение вида (рис. 9.13).В нем кнопка «Нажми меня» служит для закрытия данного окна, но при условии, что стоит «галочка» напротив «Разрешить закрытие программы». Если же поставить галочку напротив «Отключить кнопку», то кнопка вовсе становится недоступной.

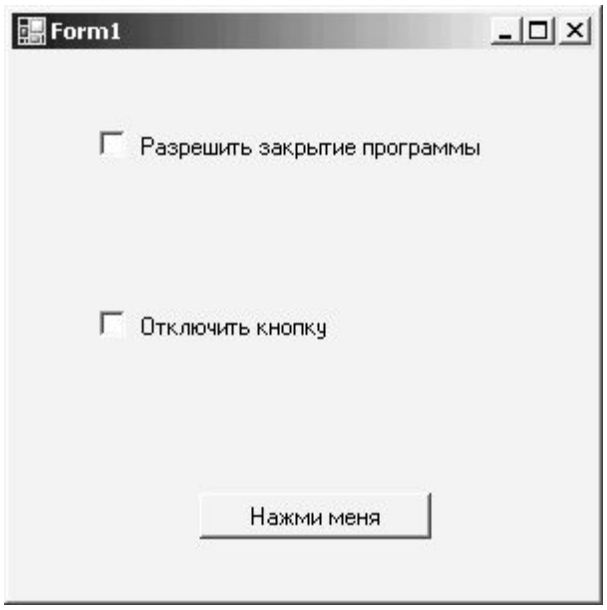


Рис. 9.13

Создадим новое приложение и оформим такие функции:

```
private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e)
                {// кнопка «Нажми меня»
                    if (checkBox1->Checked) this->Close();
}
private: System::Void checkBox2_CheckedChanged (System::Object^
sender, System::EventArgs^  e)
                { // «Отключить кнопку»
               button1->Enabled=!checkBox2->Checked;  }
```

Пример 4.4. Создать приложение вида (рис. 9.14). Такое приложение позволяет рассчитать затраты для поездки на дачу на автомобиле. Как видим, нужно будет вводить расстояние и цену на бензин, а также потребление бензина (на 100 км пути). Здесь же потребуется использовать компонент **CheckBox**, а значит – если там стоит «галочка» – полученные затраты будут удваиваться.

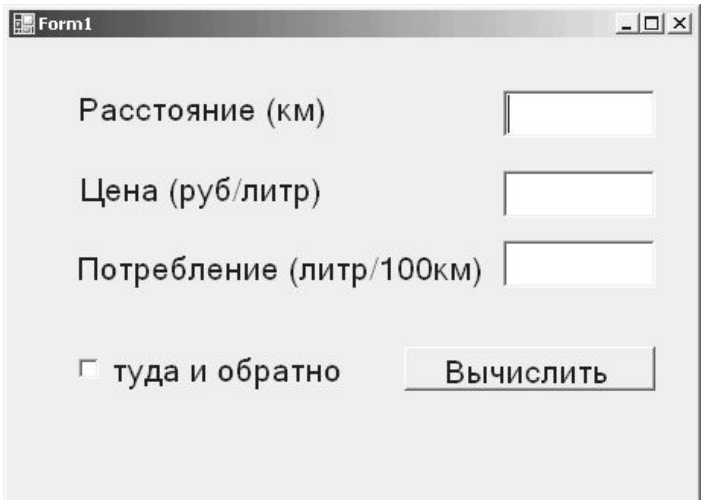


Рис. 9.14

Здесь нужно, по сути, разработать только функцию для щелчка по кнопке «Вычислить»:

```
private: System::Void button1_Click(System::Object^
sender, System::EventArgs^  e)
    {  // кнопка "Вычислить"
```

```
    float r,c,p,stoim;
    r=float::Parse(textBox1->Text)  ;
    c=float::Parse(textBox2->Text);
    p=float::Parse(textBox3->Text);
    stoim=r/100*p*c;
    if (checkBox1->Checked)  stoim=stoim*2;
    label4->Text="Стоимость вашей поездки на дачу \n
составляет:" + (stoim).ToString()+" рублей";
             }
```

5. СПИСКИ ВЫБОРА И ПОЛОСЫ ПРОКРУТКИ. ГРАФИЧЕСКИЕ КОМПОНЕНТЫ В C++Builder

5.1. Список выбора ListBox

Этот компонент находится на вкладке *Standart*. Используется для выбора одного или нескольких параметров из списка. Доступ к строкам списка – через свойство *Items* строкового типа *TString*, при этом номер строки определяется свойством *ItemIndex* (счет идет от нуля, т.е. первый элемент списка имеет нулевой индекс).

Пример 5-1. Создать приложение со списком цветов. При выборе каждого из них – в поле **Edit** появляется имя выбранного цвета и меняется цвет формы на выбранный (см. рис. 9.15).

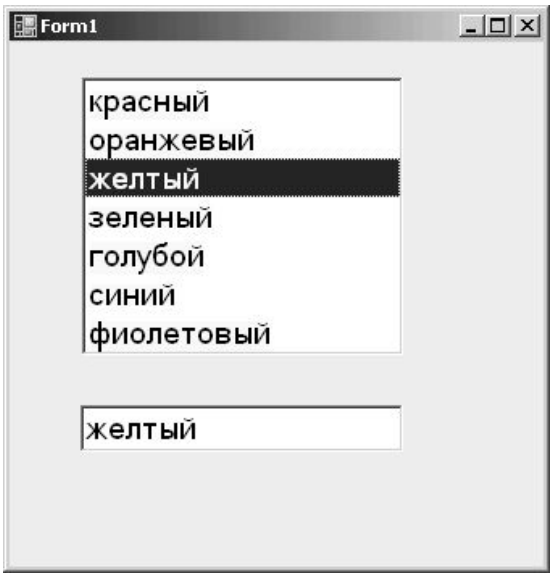


Рис. 9.15

Разместим **ListBox** и **TextBox**. Дважды щелкаем на свойстве *Items* компонента **ListBox** и вводим список цветов: красный, оранжевый…

Создадим событие `SelectedIndexChanged` для него:

```
private: System::Void listBox1_SelectedIndexChanged
(System::Object^  sender, System::EventArgs^  e)
{textBox1->Text=listBox1->SelectedItem->ToString() ;
                  //Strings[listBox1->SelectedIndex];
switch ( listBox1->SelectedIndex )
{case   0: this->BackColor=Color::Red; break;
 case    1: this->BackColor=Color::Orange; break;
```

```
case    2: this->BackColor=Color::Yellow; break;
case    3: this->BackColor=Color::Green; break;
case    4: this->BackColor=Color::SkyBlue; break;
case    5: this->BackColor=Color::Blue; break;
case    6: this->BackColor=Color::Violet; break;}   }
```

5.2. Полосы прокрутки

Полосы прокрутки **vScrollBar (**вертикальная полоса прокрутки) и **hScrollBar (**горизонтальная полоса прокрутки) используются для прокручивания информации в окне. Значение свойства полосы прокрутки *Value* меняется (по умолчанию) от 0 (Min) до 100 (Max), но может быть изменено пользователем.

Пример 5-2. Создать приложение с двумя полосами прокрутки с выбором чисел от 1 до 10 и вычислением суммы и произведения выбранных таким образом чисел (см. рис. 9.16).

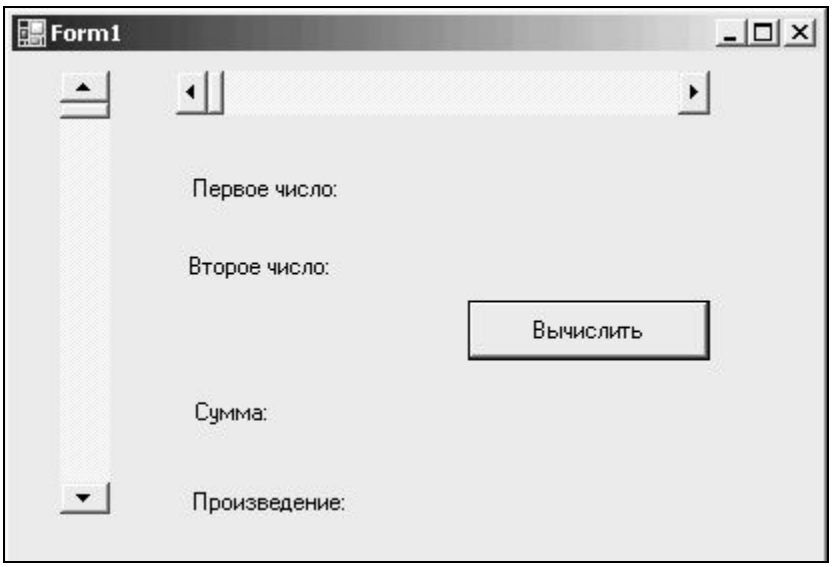


Рис. 9.16

Разместим на форме две полосы: горизонтальную и вертикальную. Зададим у каждой из них *Min* равным 1, а *Max* равным 10. Рекомендуется также изменить свойство *LargeChange* (сделать равным 1). Разместим также четыре надписи **Label**. Создадим для первой полосы событие `Scroll`:

```
private: System::Void hScrollBar1_Scroll(System::Object^
sender, System::Windows::Forms::ScrollEventArgs^  e)
{   label1->Text=
```

```
    "Первое число: "+(hScrollBar1->Value).ToString();
}
```

Аналогично создаем функцию события `Scroll` и для второй полосы прокрутки:

```
private: System::Void vScrollBar1_Scroll
(System::Object^ sender, System::Windows::
Forms::ScrollEventArgs^ e)
{ label2->Text=
«Второе число: «+(vScrollBar1->Value).ToString();
     }
```

И для кнопки «Вычислить» событие **Click** :

```
private: System::Void button1_Click(System::
Object^ sender, System::EventArgs^ e)
{ int a,b,s,p;
 a=hScrollBar1->Value;
 b=vScrollBar1->Value;
 s=a+b;
 p=a*b;
 label3->Text=«Сумма = « + s.ToString();
 label4->Text=«Произведение = « + p.ToString();
     }
```

Остается проверить приложение в работе.

5.3. Графика

Для рисования графических объектов на форме служит событие для формы `Paint`. При использовании этого события обычно следует указать

- включение графического режима командой вида:
  `e = this->CreateGraphics()`
- настроить «карандаш» для рисования `Pen` (его цвет и толщину),
- при рисовании закрашенных фигур указывают тип закраски `SolidBrush` и ее цвет,
- при выводе текста в это графическое окно настраивают шрифт и т.п.

Для собственно вывода графических объектов служат специальные методы. Например:

- для рисования линии:

```
e->DrawLine(myPen, 0, 0, 200, 300);
```

- для рисования эллипса:

```
e->DrawEllipse(myPen,Rectangle(0,0,200, 300));
```

- для рисования прямоугольника:

```
e->DrawRectangle(myPen, Rectangle(0, 0, 200, 300));
```

- для рисования закрашенного прямоугольника:

```
e->FillRectangle(myBrush, Rectangle(0, 0, 200, 300));
```

- для вывода текста на рисунке:

```
e->DrawString(s,myFont,myBrush, 10, 10);
```

Здесь `myPen` – заданный вид карандаша, `myBrush` – заданный вид закраски, `Rectangle` – заданный прямоугольник (первые два числа – координаты левой верхней точки прямоугольника на форме, вторые – это его длина и высота). При выводе текста `s` – содержит строку текста, `myFont` – заданный тип шрифта.

Пример 5-3. Нарисовать флаг России и сделать снизу надпись (см. рис. 9.17).

Рис. 9.17

```
// ФЛАГ РОССИИ
    private: System::Void Form1_Paint(System::Object^
sender, System::Windows::Forms::PaintEventArgs^ e)
             {
Pen^ myPen = gcnew Pen(Color::Black,3);// цвет и толщина
пера
SolidBrush^ myBrush =
    gcnew SolidBrush(Color::Black); //цвет закраски
    System::Drawing::Font^ myFont =
```

```
gcnew System::Drawing::Font(«Arial»,16);//тип и размер
шрифта
        String^ myString= « Р о с с и я «;
        int x=10, y=10, //начальная точка
        L=200, H=40; //ширина флага и высота полосы
// задание размера прямоугольника (одной полосы флага)
            Rectangle rect = Rectangle(x,y,L,H);
    //рисуем белую полосу флага
            e->Graphics->DrawRectangle(myPen,rect);
            myBrush->Color=Color::White;
            e->Graphics->FillRectangle(myBrush, rect);
    //рисуем синую полосу флага
            rect = Rectangle(x,y+H,L,H);
            e->Graphics->DrawRectangle(myPen,rect);
            myBrush->Color=Color::Blue;
            e->Graphics->FillRectangle(myBrush, rect);
    //рисуем красную полосу флага
            rect = Rectangle(x,y+2*H,L,H);
            e->Graphics->DrawRectangle(myPen,rect);
            myBrush->Color=Color::Red;
            e->Graphics->FillRectangle(myBrush, rect);
    //вывод надписи(сначала красным, потом черным цветом)
e->Graphics->DrawString(myString, myFont, myBrush,
50,200);
            myBrush->Color=Color::Black;
e->Graphics->DrawString(myString, myFont, my-
Brush,51,201);
             }
```

6. РАБОТА С ТЕКСТОВЫМИ ФАЙЛАМИ.

6.1. Чтение и запись текстового файла

В среде Visual C++ (Windows Forms) имеется возможность обращаться к внешним файлам: отображать информацию из файла на форме, помещать информацию с формы в файл и т.п. Мы рассмотрим работу с текстовыми файлами в простейшем варианте – с помощью специальных компонентов.

Для работы с файлами будем использовать окна диалога. В Windows Forms имеются специальные элементы управления, орга-

низующие диалоги именно для открытия и сохранения файлов: `openFileDialog` и `saveFileDialog`.

Пример 6-1. Создать приложение, которое выводит в поле richTextBox, размещенное на форме, содержимое текстового файла, а также позволяет отредактировать этот текст (в том числе и изменить шрифт) и сохранить в файл. (см. рис. 9.18).

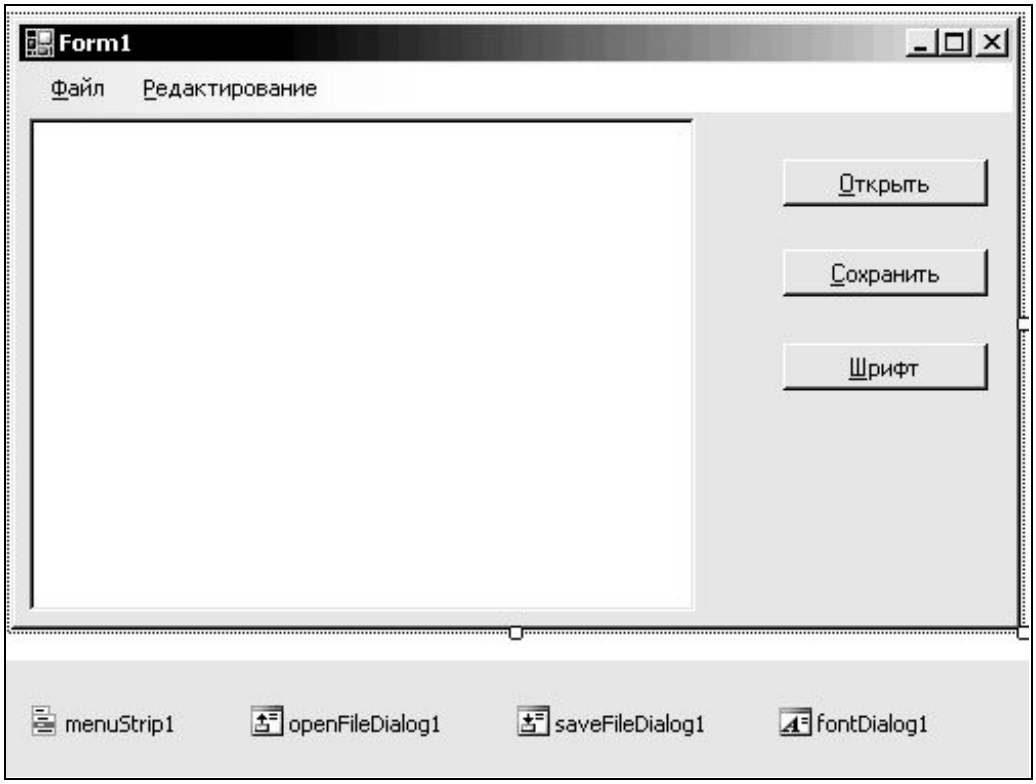


Рис. 9.18

При реализации этого проекта попутно освоим работу по созданию *меню* на форме и работу с *окнами диалога*.

Как видим, нам нужно разместить элемент управления `richTextBox` (куда будет помещаться текст из файла и где его будем редактировать), три кнопки: `Открыть` (файл), `Сохранить` (файл), `Шрифт` (изменить шрифт текста). Разместив эти элементы, сразу их настроим: сменим надписи на кнопках (свойство `Text` у каждой кнопки), поле `richTextBox` растянем на всю оставшуюся часть формы. Отметим, что содержимое этого поля хранится в свойстве `Lines`, которое имеет вид массива: первая строка в окошке `richTextBox` может быть получена так:

```
richTextBox1->Lines[0],
```

вторая так:

```
richTextBox1->Lines[1],
```

т.е. обратите внимание, что нумерация строк идет с нуля.

Кроме того, разместим элементы: menuStrip – для организации меню на форме, openFileDialog, saveFileDialog – для организации диалогов при открытии и сохранении файлов, а также fontDialog – для организации диалога при смене шрифта. Мы эти элементы, как обычно, перетаскиваем на форму, но они будут находиться внизу, под самой формой (см. рис. 6.1). Для окон диалога проведем настройку. Выберем openFileDialog1 на форме и у свойства Filter установим **richTextFile|*.rtf.** Аналогично сделаем и у saveFileDialog1. В результате в окне диалога будут высвечиваться только файлы указанного типа. Обратим внимание, что данное окно richTextBox1 предназначено для работы именно с файлами типа **rtf**.

Далее нужно настроить *меню*: задать его пункты и подпункты. Для этого щелкнем по значку menuStrip1 и вводим меню вида:

&Файл	&Редактирование
&Открыть	&Шрифт
&Закрыть	&Очистка

Значок «&» здесь помещаем для того, чтобы можно было вызывать соответствующий пункт с помощью клавиши на клавиатуре. В данном случае, пункт «Файл» будет вызываться клавишей «Ф», подпункт «Открыть» клавишей «О» и т.д.

Теперь разработаем события для всех этих пунктов. Дважды щелкаем по подпункту «Открыть» и оформляем событие вида:

```
private:   System::Void   открытьToolStripMenuItem_Click
(System::Object^ sender, System::EventArgs^ e)
{openFileDialog1->ShowDialog();
richTextBox1->LoadFile(openFileDialog1->FileName);}
```

Аналогично создадим событие и для пункта «Сохранить» :

```
private:  System::Void  сохранитьToolStripMenuItem_Click
(System::Object^ sender, System::EventArgs^ e)
{saveFileDialog1->ShowDialog();
richTextBox1->SaveFile(saveFileDialog1->FileName); }
```

Для подпункта «Шрифт» событие имеет вид:

```
private: System::Void шрифтToolStripMenuItem_Click (Sys-
tem::Object^ sender, System::EventArgs^ e)
{fontDialog1->ShowDialog();
richTextBox1->Font=fontDialog1->Font;  }
```

Для подпункта «Очистить»:

```
private:System::Void     очиститьToolStripMenuItem_Click
(System::Object^ sender, System::EventArgs^ e)
         {richTextBox1->Clear();   }
```

Заметим, что на форме имеются кнопки с аналогичными действиями. Поэтому кнопке «Открыть» назначим то же событие `открытьToolStripMenuItem_Click`. Напомним, что для этого достаточно выделив эту кнопку, перейти в окошко `Properties` и переключится на вкладку `Events` (щелкнуть по значку «молния»), а там, в строке **Click,** щелкаем и выбираем нужное событие.

Аналогично кнопке «Сохранить» назначим событие `сохранитьToolStripMenuItem_Click` , а кнопке «Шрифт» событие `шрифтToolStripMenuItem_Click`.

Проект готов. Остается проверить в работе. Как обычно, сохраняем его, проводим сборку (Build) и отправляем на выполнение.

После выбора пункта «Файл» и подпункта «Открыть» появится окно диалога вида (рис. 9.19).

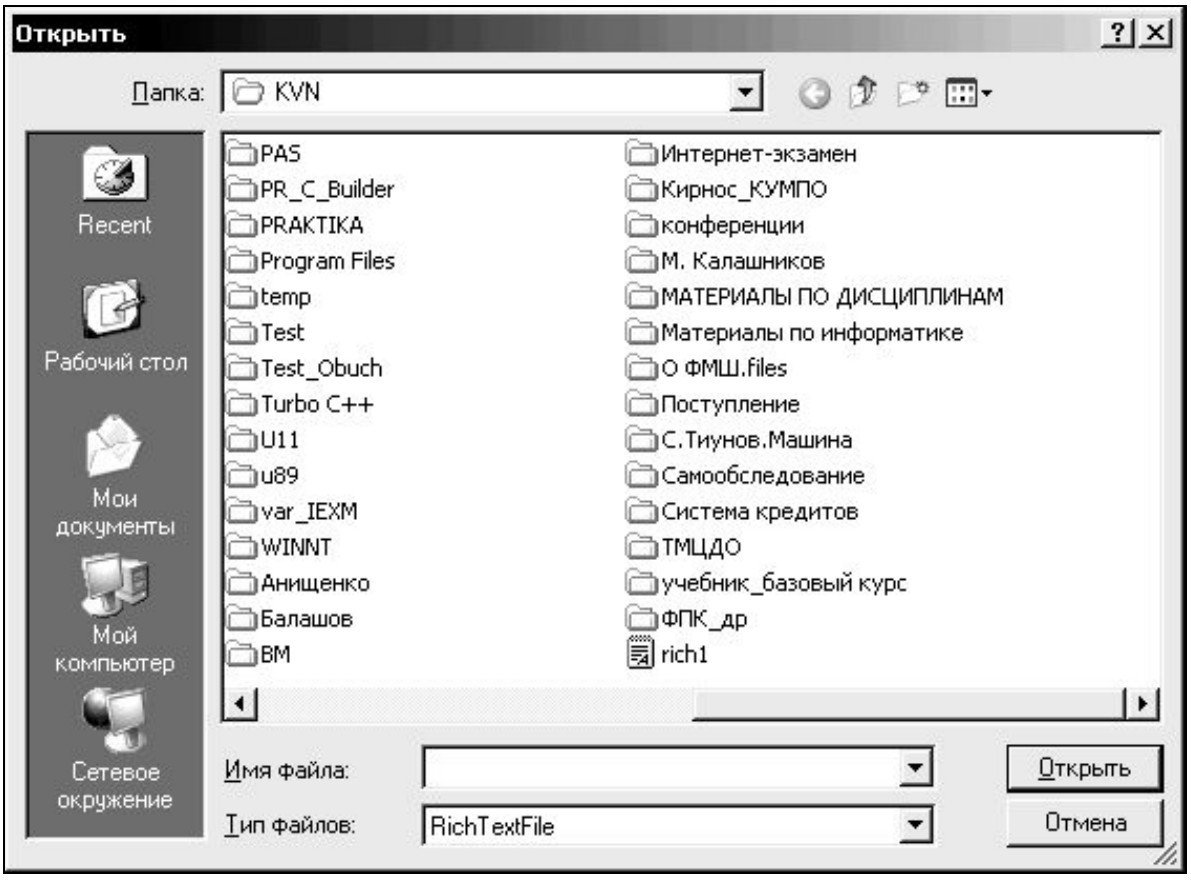


Рис. 9.19

Как видно, отображается единственный файл заданного нами типа – его имя **rich1**. Выбрав его и щелкнув по кнопке **Открыть**,

мы и получим его содержимое на форме в окне richTextBox (см. рис. 9.20).

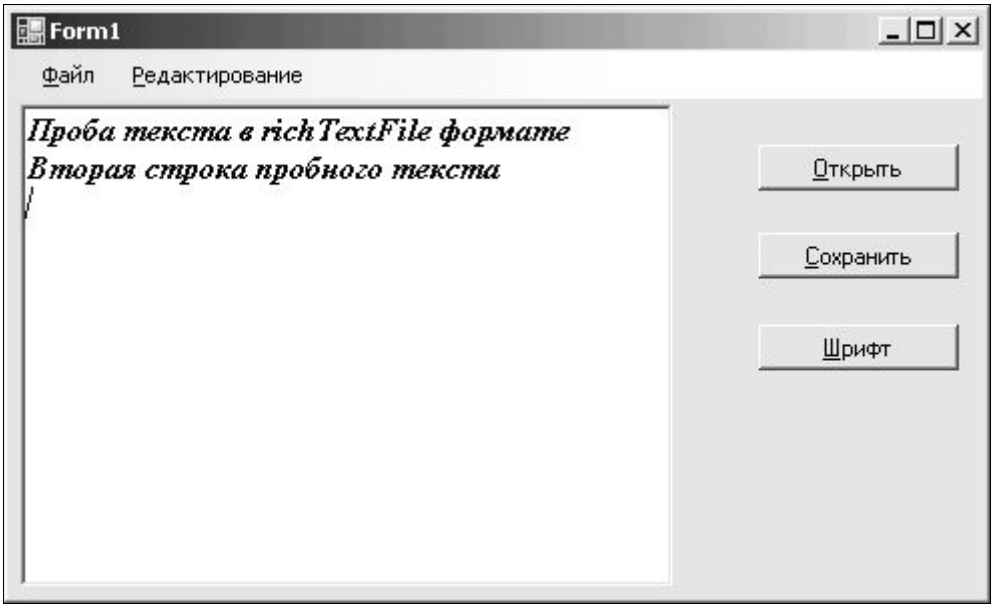


Рис. 9.20

В данном окне можем дописать текст и попробовать изменить шрифт. Щелкнем по пункту «Редактирование» и подпункту «Шрифт» – получим окно вида (рис. 9.21)

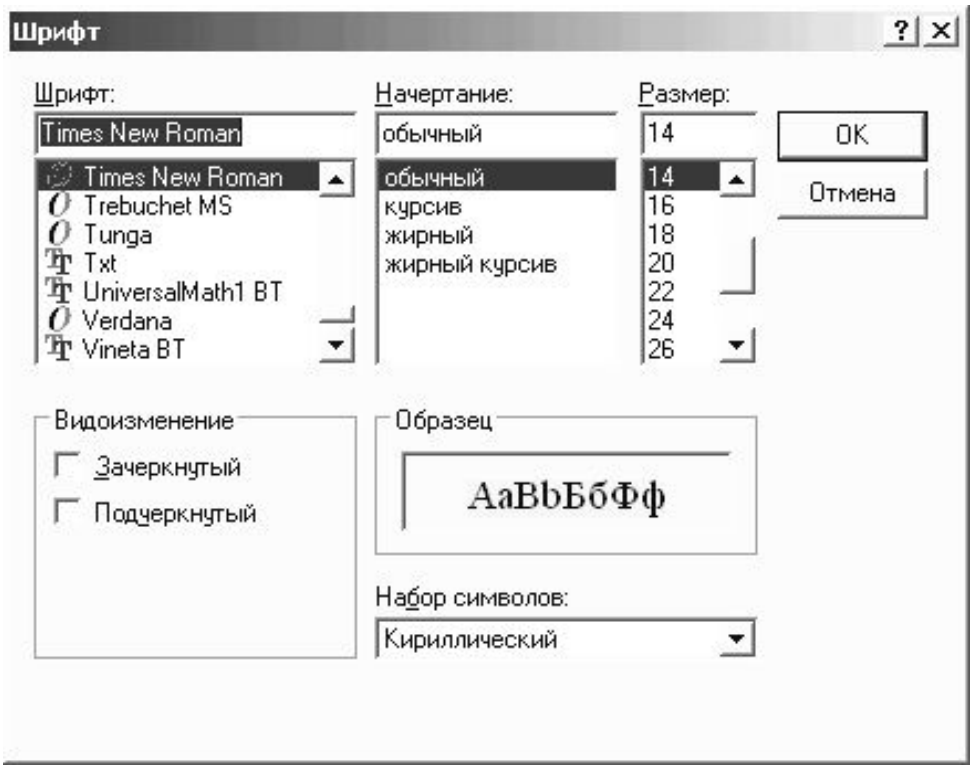


Рис. 9.21

В результате (когда щелкнем ОК в данном окошке), получим на форме результат, например, такого вида (см. рис. 9.22**)**.

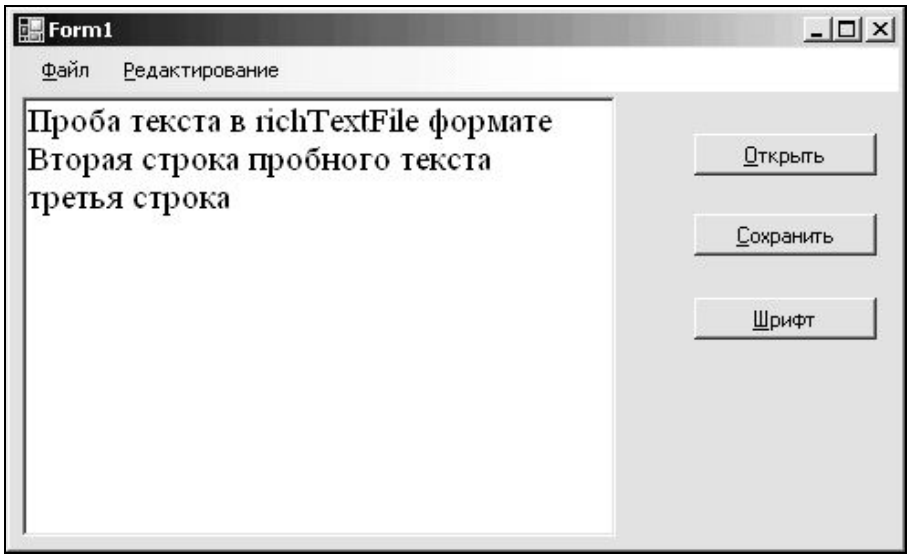


Рис. 9.22

Теперь для сохранения файла достаточно вызвать меню ***Файл/Сохранить***, откроется окно диалога для сохранения (см. рис. 9.23), указать имя для сохраняемого файла (или выбрать прежнее имя) и щелкнуть ОК. Измененный файл записан.

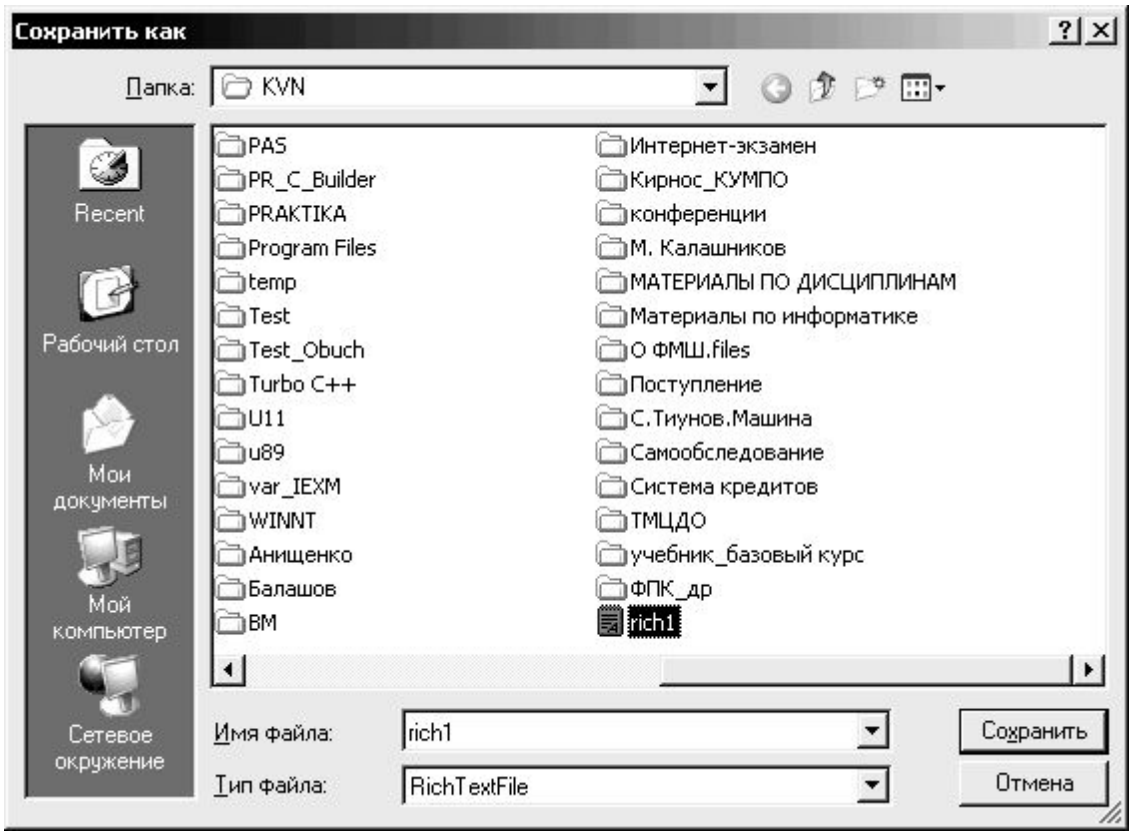


Рис. 9.23

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Раздел 1. Кнопки, метки и окна редактирования

1.1. Написать программу, которая вычисляет силу тока в электрической цепи. Рекомендуемый вид формы приведен на рис. И-1. Программа должна быть спроектирована таким образом, чтобы кнопка **Вычислить** была доступна только в том случае, если пользователь ввел величину сопротивления.

Пример: $U = 10$, $R = 15$. Ответ: 0,6667

Рис. И-1

Рис. И-2

1.2. Написать программу, которая вычисляет силу тока в электрической цепи (рис. И-2). Цепь состоит из двух параллельно соединенных сопротивлений. Рекомендуемый вид формы:

Пример: $R_1 = 10$, $R_2 = 15$. Ответ: 6

1.3. Написать программу, которая вычисляет силу тока в электрической цепи. Цепь состоит из двух последовательно соединенных сопротивлений. Рекомендуемый вид формы см. выше.

Пример: $R_1 = 10$, $R_2 = 15$. Ответ: 25

1.4. Найти массу x литров молока, если известно, что плотность молока p кг/м^3.

Пример: $x = 7$ л, $p = 1030$кг/м^3. Ответ: 7,21 кг

1.5. Объем цилиндра равен V, а площадь основания – S. Какова высота цилиндра H?

Пример: $V = 10$м3 , $S = 5$м^2. Ответ: 2 м

1.6. Дана длина ребра куба a. Найти объем куба V и площадь его боковой поверхности S.

Пример: $a = 5$ Ответ: $V = 125$, $S = 100$

1.7. Каков объем кислорода, содержащегося в комнате размером $a \cdot b \cdot c$, если кислород составляет 21% объема воздуха?

Пример: $a = 3$, $b = 4$, $c = 5$.Ответ: 12,6

1.8. Найти площадь равнобочной трапеции с основаниями a и b и углом при большем основании равным x.

Пример: $a = 6$, $b = 5$, $x = 45°$. Ответ: 2,75

1.9. Найти угол между отрезком прямой, соединяющей начало координат с точкой $A(x, y)$, и осью OX (точка лежит в 1-й четверти).

Пример: $x = 3$, $y = 4$. Ответ: 53,13°

1.10. Определить время падения камня на поверхность земли с высоты h.

Пример: h=10 м. Ответ: 1,4278 с

1.11. Три сопротивления R_1, R_2, R_3 соединены параллельно. Найти сопротивление соединения.

Пример: $R_1 = 10$, $R_2 = 15$, $R_3 = 20$. Ответ: 4,62

1.12. Написать программу вычисления площади параллелограмма. Извне вводятся стороны a,b и угол между ними x.

Пример: $a = 10$, $b = 15$, $x = 30°$. Ответ: 75

1.13. Написать программу вычисления объема прямоугольного параллелепипеда. Извне вводятся длина a, ширина b и высота c.

Пример: $a = 10$, $b = 15$, $c = 20$. Ответ: 3000

1.14. Написать программу вычисления площади поверхности прямоугольного параллелепипеда. Извне вводятся длина a, ширина b и высота c.

Пример: $a = 10$, $b = 15$, $c = 20$. Ответ: 1300

1.15. Написать программу вычисления объема цилиндра. Извне вводятся радиус основания R и высота цилиндра h.

Пример: $R = 10$, $h = 15$. Ответ: 4712,39

1.16. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и карандашей. Извне вводятся цена одной тетради Ct и количество тетрадей Kt, а также цена карандаша Ck и количество карандашей Kk.

Пример: $Ct = 1$, $Kt = 15$, $Ck = 0.2$, $Kk = 5$. Ответ: 16

1.17. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек к

ним. Извне вводятся цена одной тетради *Ct*, одной обложки *Cb* и количество тетрадей *Kt*.

Пример: *Ct* = 1.2, *Kt* = 15, *Cb* = 0.2. Ответ: 21

1.18. Написать программу вычисления стоимости некоторого количества (по весу) яблок. Извне вводятся цена одного килограмма яблок *C* и вес яблок *V*.

Пример: *C* = 25, *V* = 1.5. Ответ: 37.5

1.19. Написать программу вычисления периметра и площади треугольника, заданного длинами сторон.

Пример: *a* = 3, *b* = 4, *c* = 5. Ответ: *P* = 12, *S* = 6

1.20. Написать программу вычисления периметра и площади треугольника, заданного координатами вершин.

Пример: $x1 = 0$, $y1 = 0$, $x2 = 0$, $y2 = 3$, $x3 = 4$, $y4 = 0$. Ответ: *P* = 12, *S* = 6

Раздел 2. Радиокнопки

2.1. Написать программу, которая вычисляет сопротивление электрической цепи, состоящей из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Рекомендуемый вид формы дан на рис. И-3.

Рис. И-3

2.2. Написать программу, которая вычисляет силу тока в электрической цепи (используя закон Ома). Цепь состоит из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Рекомендуемый вид формы приведен на рис. И-4.

Рис. И-4

2.3. Написать программу, которая находит максимум, либо минимум для двух задаваемых чисел

2.4. Написать программу, которая находит максимум, либо минимум для трех задаваемых чисел

2.5. Написать программу, которая находит среднее арифметическое, либо среднее геометрическое для двух задаваемых чисел

2.6. Написать программу, которая указывает знак значения функции sin в зависимости от выбранной на форме четверти координатной плоскости

2.7. Написать программу, которая указывает знак значения функции cos в зависимости от выбранной на форме четверти координатной плоскости.

2.8. Написать программу, которая вычисляет стоимость разговора по междугороднему телефону в зависимости от указанных минут длительности и выбранного времени (день или ночь). Ночью – скидка в 30%

2.9. Написать программу, которая вычисляет стоимость разговора по междугороднему телефону в зависимости от указанных минут длительности и выбранного вида дня недели (рабочие или выходные дни). В выходной – скидка в 20%.

2.10. Написать программу, которая в зависимости от выбранного времени года (зима, весна, лето, осень) выдает список месяцев этого времени года.

2.11. Написать программу, которая в зависимости от выбранного пересчитывает заданную в рублях сумму в эквивалент в долларах, либо в евро.

В последующих девяти задачах одинаковое условие:

Написать программу, которая в зависимости от выбранного способа (по формуле суммы или суммированием в цикле) рассчитывает сумму.

2.12. $1 + 2 + 3 + \ldots + n = n(n + 1)/2$

2.13. $p + (p + 1) + (p + 2) + \ldots + (p + n) = (n + 1)(2p + n)/2$

2.14. $1 + 3 + 5 + \ldots + (2n - 1) = n^2$

2.15. $2 + 4 + 6 + \ldots + 2n = n(n + 1)$

2.16. $1^2 + 2^2 + 3^2 + \ldots + n^2 = n(n + 1)(2n + 1)/6$

2.17. $1^3 + 2^3 + 3^3 + \ldots + n^3 = n^2(n + 1)^2/4$

2.18. $1^2 + 3^2 + 5^2 + \ldots + (2n - 1)^2 = n(4n^2 - 1)/3$

2.19. $1^3 + 3^3 + 5^3 + \ldots + (2n - 1)^3 = n^2(2n^2 - 1)$

2.20. $1^4 + 2^4 + 3^4 + \ldots + n^4 = n(n + 1)(2n + 1)(3n^2 + 3n - 1)/30$

Раздел 3. Полосы прокрутки

В приводимых ниже заданиях организовать вычисление с помощью полосы прокрутки для различных *n*. Причем, предусмотреть вычисление, как в цикле, так и по формуле, приведенной в правой части выражения

3-1. $1 + 2 + 3 + \ldots + n = n(n + 1)/2$

3-2. $p + (p + 1) + (p + 2) + \ldots + (p + n) = (n + 1)(2p + n)/2$

3-3. $1 + 3 + 5 + \ldots + (2n - 1) = n^2$

3-4. $2 + 4 + 6 + \ldots + 2n = n(n + 1)$

3-5. $1^2 + 2^2 + 3^2 + \ldots + n^2 = n(n + 1)(2n + 1)/6$

3-6. $1^3 + 2^3 + 3^3 + \ldots + n^3 = n^2(n + 1)^2/4$

3-7. $1^2 + 3^2 + 5^2 + \ldots + (2n - 1)^2 = n(4n^2 - 1)/3$

3-8. $1^3 + 3^3 + 5^3 + \ldots + (2n - 1)^3 = n^2(2n^2 - 1)$

3-9. $1^4 + 2^4 + 3^4 + \ldots + n^4 = n(n + 1)(2n + 1)(3n^2 + 3n - 1)/30$

3-10. $1 + 4 + 7 + \ldots + 3n - 2 = n(3n - 1)/2$

3-11. $-1 + 2 - 3 + \ldots + (-1)^n n = (-1)^n [(n + 1)/2]$ (здесь [ ] – означает целую часть)

3-12. $-1^2 + 2^2 - 3^2 + \ldots + (-1)^n n^2 = (-1)^n (n(n + 1)/2)$

3-13. $-1^3 + 2^3 - 3^3 + \ldots + (-1)^n n^3 = (1/8)*(1 - (-1)^n (1 - 6n^2 - 4n^3))$

3-14. $-1^4 + 2^4 - 3^4 + \ldots + (-1)^n n^4 = (-1)^n (n^4 + 2n^3 - n)/2$

3-15. $1^5 + 2^5 + 3^5 + \ldots + n^5 = (1/12)*n^2(n + 1)^2(2n^2 + 2n - 1)$

3-16. $1^5 - 2^5 + 3^5 - \ldots + (-1)^{n-1} n^5 = (1/4)(1 + (-1)^n (5n^2 - 5n^4 - 2n^5 - 1))$

3-17. $1 + 3 + 5 + \ldots + (2n + 1) = (n + 1)^2$

3-18. $1 - 3 + 5 - 7 + \ldots + (-1)^n(2n + 1) = (-1)^n (n + 1)$

3-19. $1^2 + 3^2 + 5^2 + \ldots + (2n + 1)^2 = (1/3)(n + 1)(2n + 1)(2n + 3)$

3-20. $1^2 - 3^2 + 5^2 - \ldots + (-1)^n (2n + 1)^2 = (-1)^n 2(n + 1)^2 - (1 + (-1)^n)/2$

3-21. $1^3 + 3^3 + 5^3 + \ldots + (2n + 1)^3 = (n + 1)^2(2n^2 + 4n + 1)$

3-22. $1*2 + 2*3 + \ldots + n(n + 1) = (1/3)n(n + 1)(n + 2)$

3-23. $1*2*3 + 2*3*4 + \ldots + n(n + 1)(n + 2) = (1/4)n(n + 1)(n + 2)(n + 3)$

Раздел 4. Обработка текстовых файлов

В приводимых ниже заданиях следует разработать программу с использованием кнопок на форме (см. пример ниже). Данные рекомендуется вводить с использованием объекта Memo и сохранять их в текстовом файле. Вывод на экран также производить в область Memo. В качестве примера приведем выполнение следующего задания:

Дана запись с именем STUDENT, содержащая следующие поля:

- Фамилия и инициалы;
- Номер группы;
- Успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 записей типа STUDENT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- Вывод на экран фамилий, номеров групп и оценок для всех студентов, которые являются круглыми отличниками (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по возрастанию номера группы.

Форма должна иметь примерно такой вид (рис. И-5).

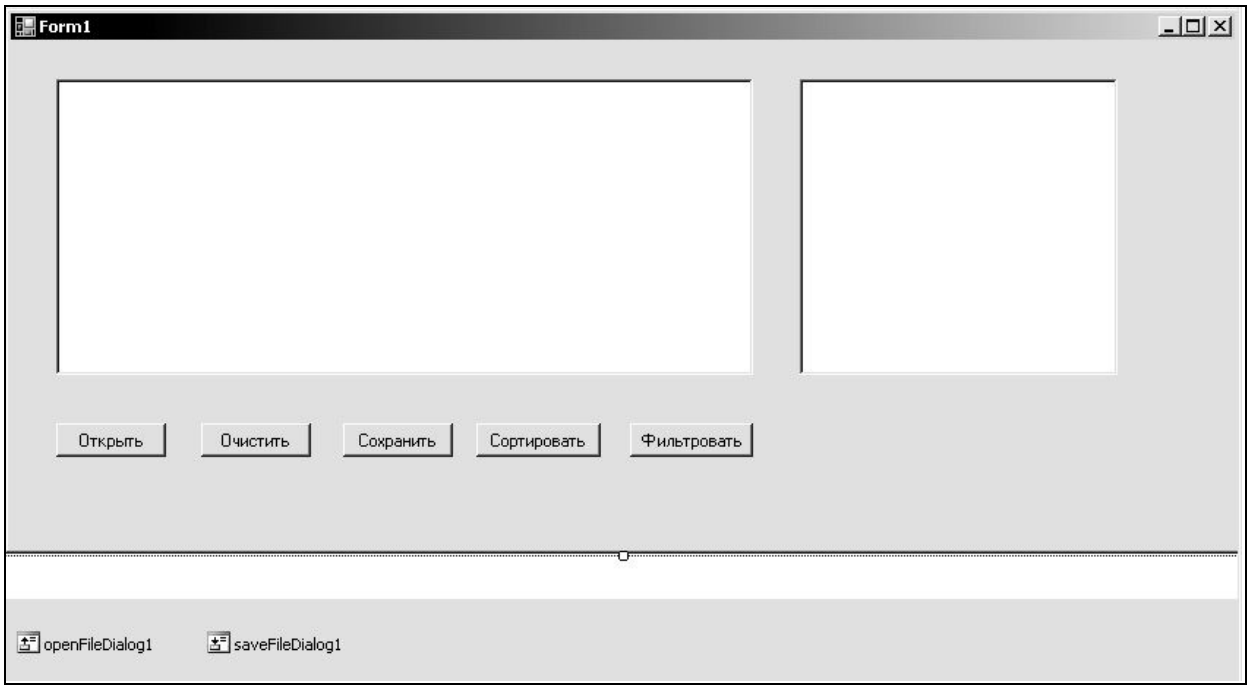


Рис. И-5

Обратите внимание на два объекта – **openFileDialog** и **saveFileDialog** . Они нам потребуются для организации диалога при вводе данных в файл (и последующего сохранения), а также при считывании из файла.

При первоначальном запуске данные на форме вносятся в поле **richTextBox** (в соответствии с той структурой, которая указана в задании), причем в пределах строки – данные одного студента: фамилия студента, номер группы, каждая из оценок вносятся через пробел. Занесем, например, данные, приведенные на рис. И-6 (окно слева).

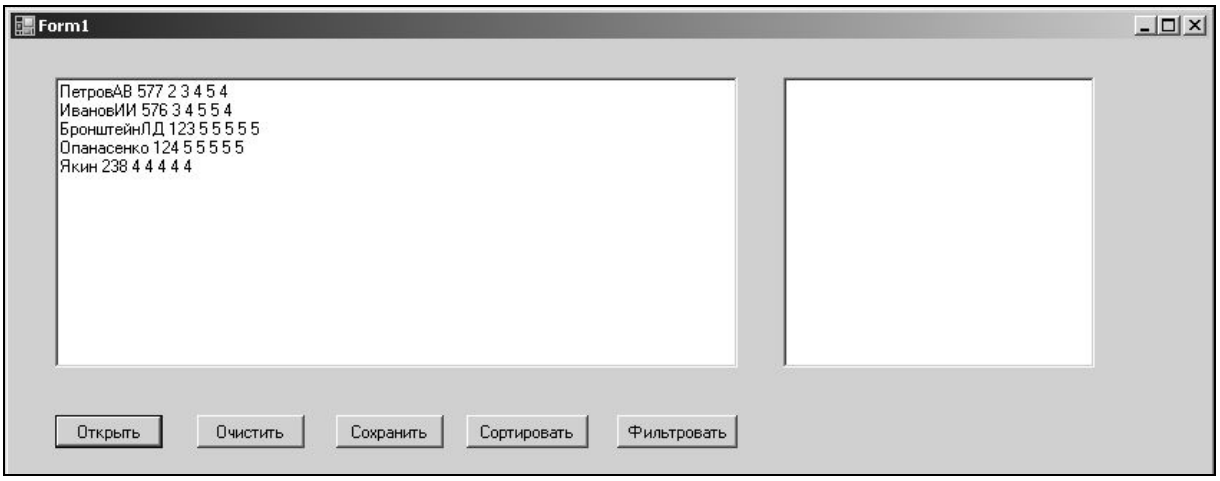


Рис. И-6

Кнопка «**Сохранить**» служит для сохранения данных из окна richTextBox. Функция для кнопки «Сохранить» имеет вид:

```
private: System::Void button3_Click(System::Object^
sender, System::EventArgs^ e) {
    saveFileDialog1->ShowDialog();
    richTextBox1->SaveFile(saveFileDialog1->FileName);
        }
```

При этом откроется окно диалога для сохранения (рис. И-7):

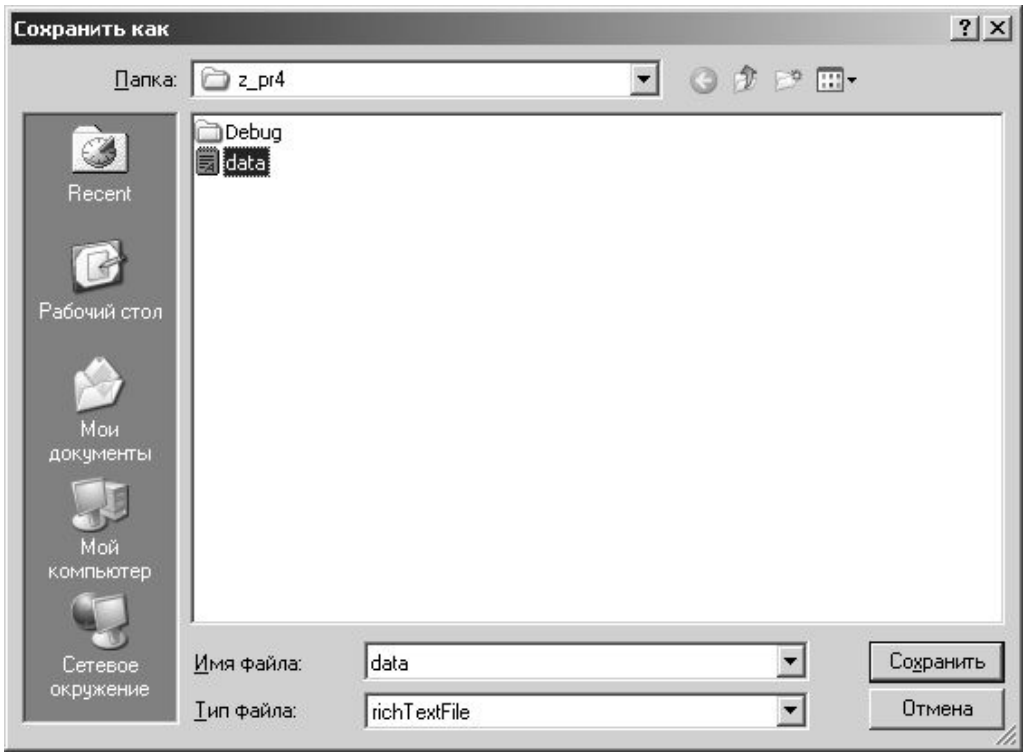


Рис. И-7

В этом окне нужно будет перейти на нужный диск и в нужную папку, а затем указать имя файла. Рекомендуется указывать у файла расширение rtf.

Затем рекомендуется поле Memo очистить с помощью кнопки «**Очистить**», функция для которой имеет вид:

```
private: System::Void button2_Click(System::Object^
sender, System::EventArgs^ e) {
             richTextBox1->Clear();
         }
```

Далее следует файл открыть – кнопкой «**Открыть**», имеющей функцию:

```
private: System::Void button1_Click(System::Object^
sender, System::EventArgs^ e) {
    openFileDialog1->ShowDialog();
    richTextBox1->LoadFile(openFileDialog1->FileName);
             }
```

При выполнении этой функции появится аналогичное окно диалога (рис. *И*-8). И здесь нужно будет перейти в соответствующий каталог (папку) и выбрать нужный для открытия файл. Обратите внимание, что мы здесь заранее ограничили список файлами текстового типа (**rtf**). Такой фильтр обеспечивается выполнением (при запуске приложения, т.е. при открытии формы) функции вида:

```
   private:   System::Void   Form1_Load(System::Object^
sender, System::EventArgs^ e) {
   // фильтр списка файлов
   openFileDialog1->Filter=«RTF|*.rtf»;
   saveFileDialog1->Filter=«RTF|*.rtf»;
   }
```

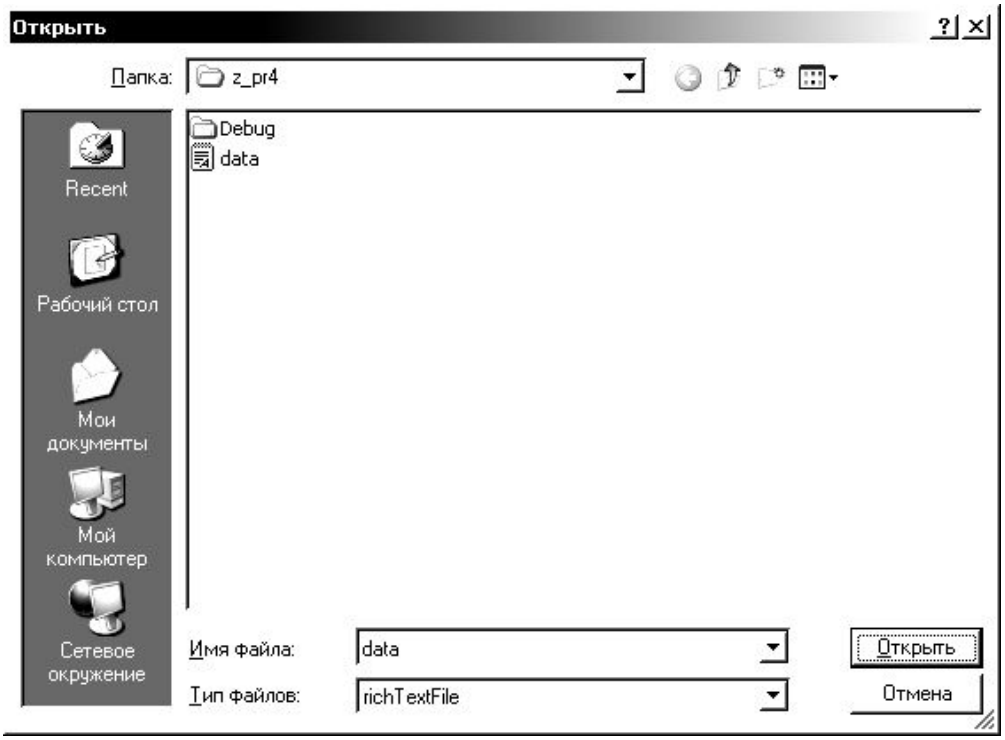


Рис. И-8

Теперь данный список (из открытого файла) вновь появится в поле richTextBox и его можно обрабатывать. Например, для сортировки (упорядочения по номеру группы) следует нажать кнопку «**Сортировать**», содержащую функцию:

```
private: System::Void button4_Click(System::Object^  sender,
System::EventArgs^  e) {
int gr[20]; int i,j; String^ s;
int n=richTextBox1->Lines->Length;  //n- число строк
 for ( i=0;i<n;i++)
{int ns=richTextBox1->Lines[i]->Length;//ns- длина i-й строки
     s =richTextBox1->Lines[i]; //вычленение i-й строки
     int k0=s->IndexOf(" "); //поиск положения 1-го пробела
     int k1=s->IndexOf(" ",k0+1);  //положение 2-го пробела
     String^ sgr=s->Substring(k0+1,k1-k0);//выделение подстроки
                  //между 1-м и 2-м пробелами - номер группы
     gr[i]=Int32::Parse(sgr); //номер группы (в числовом виде)
 }
 //собственно сортировка поиском максимума
 // и запоминанием порядка максимумов в массиве  m
 int k=0; int nmax, max, m[20];
 for ( i=0;i<n;i++)
 {  max=0;
      for (j=0;j<n;j++)
 {if (gr[j]>max) {max=gr[j];nmax=j;}}
   m[k]=nmax;k=k+1;
 gr[nmax]=0;}
  richTextBox2->Clear();
  // вывод упорядоченных строк в новое окно richTextBox2
 for ( j=0;j<n;j++)
 {richTextBox2->Text=
     richTextBox2->Text+"\n"+richTextBox1->Lines[m[j]];
   }           }
```

Внимательно изучите данную функцию. Обратите внимание, например, как вычисляется длина каждой строки.

Для преобразования номера группы (который в строке записан *символами*) в целое число используется функция `Int32::Parse`.

При нажатии на данную кнопку получим результат в отдельном окне richTextBox (см. рис. И-9, окошко справа). Как видно, список упорядочен по номеру группы в порядке убывания.

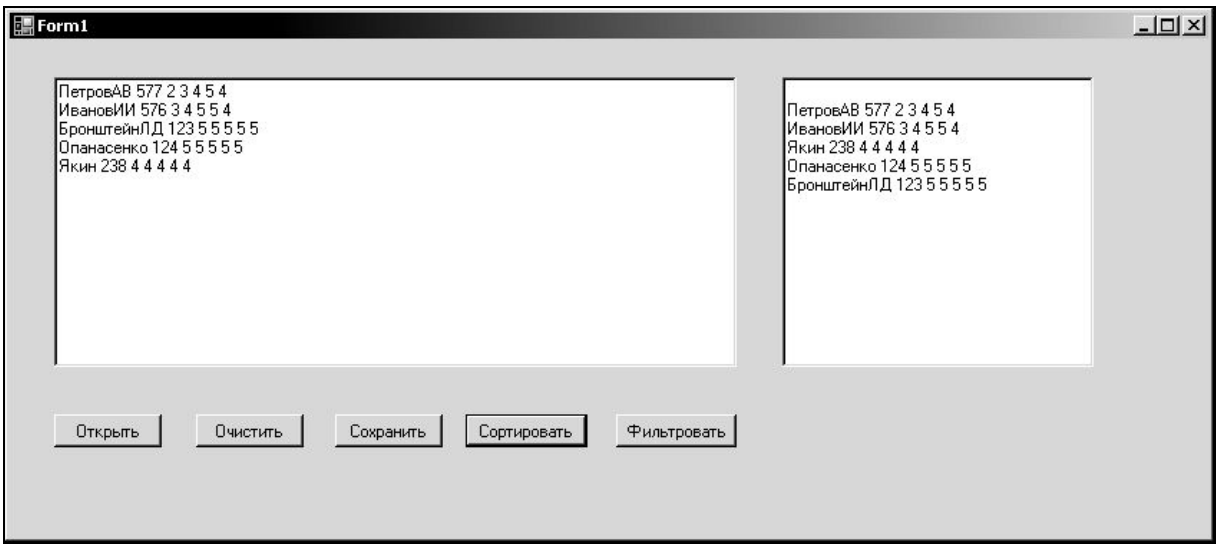


Рис. И-9

Далее потребуется отфильтровать список (в данном случае выделить отличников). Для этого служит кнопка «**Отфильтровать**» с функцией:

```
private: System::Void button5_Click(System::Object^ sender,
System::EventArgs^  e) {
String^ s;      String^ soc; int i,j; int oc[20][5];
 int n=richTextBox1->Lines->Length; //n- число строк в окне
RichTextBox
 for ( i=0;i<n;i++)
 {int ns=richTextBox1->Lines[i]->Length;//ns-длина i-й строки
                s =richTextBox1->Lines[i];
                s=s+" ";//добавили пробел в конец строки
                int k0=s->IndexOf(" ");
                int k1=s->IndexOf(" ",k0+1);  ;
                int k2=s->IndexOf(" ",k1+1);  ;
                soc=s->Substring(k1+1,k2-k1);
               oc[i][1]=Int32::Parse(soc);
                int k3=s->IndexOf(" ",k2+1);  ;
                soc=s->Substring(k2+1,k3-k2);
                oc[i][2]=Int32::Parse(soc);
                int k4=s->IndexOf(" ",k3+1);  ;
                soc=s->Substring(k3+1,k4-k3);
                oc[i][3]=Int32::Parse(soc);
                int k5=s->IndexOf(" ",k4+1);  ;
                soc=s->Substring(k4+1,k5-k4);
                oc[i][4]=Int32::Parse(soc);
                int k6=s->IndexOf(" ",k5+1);  ;
                soc=s->Substring(k5+1,k6-k5);
                oc[i][5]=Int32::Parse(soc);
                 }
 richTextBox2->Clear();
```

```
 //отбор отличников
for ( i=0;i<n;i++)
{    int sum=0;
     for (j=1;j<=5;j++)
     {    if ((oc[i][j])==5) sum++;}
     if (sum==5)
          richTextBox2->Text=
           richTextBox2->Text+"\n"+richTextBox1->Lines[i];
 }
           }
```

Здесь сначала вычленяются оценки каждого студента и подсчитывается их количество. Поскольку заранее известно, что всего оценок пять штук, то круглый отличник должен иметь пять «пятерок». В результате соответствующую строку и выводим в поле richTextBox2 . Нажмем на эту кнопку «**Отфильтровать**» и получим отдельно список отличников (см. рис. И-10, правое окошко).

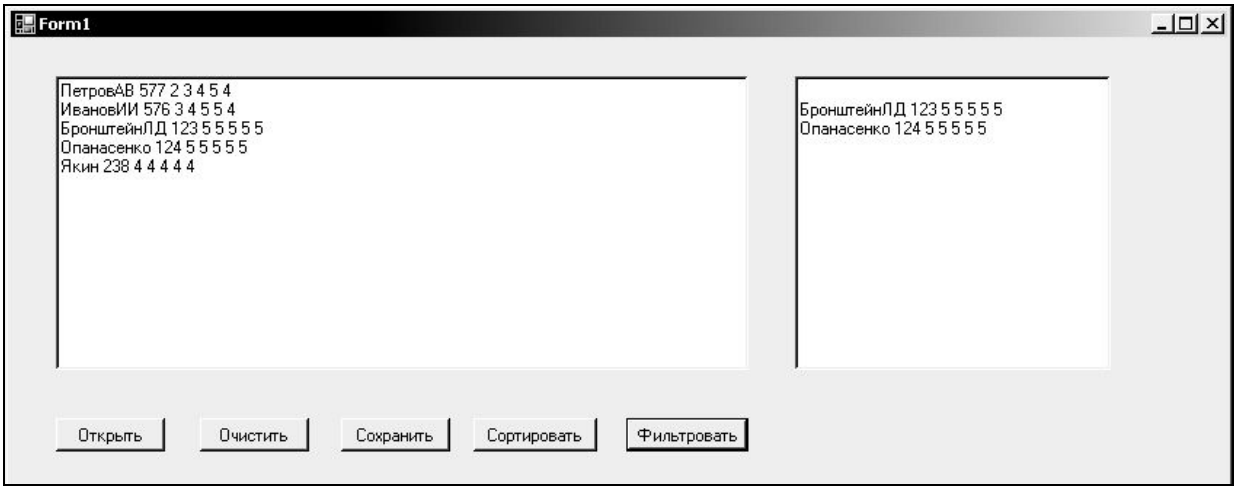


Рис. И-10

Можно, кстати, такой отфильтрованный список записать в новый файл. Для этого надо воспользоваться кнопкой «**Сохранить**» (но обратите внимание, что мы выводим отличников в новом окне richTextBox2, поэтому нужно будет поправить функцию для кнопки «Сохранить»). При нажатии на нее потребуется указать имя (другое) файла и перейти, если нужно, в другую папку. Потом уже отфильтрованный список можно открыть и провести для него сортировку (т.е. упорядочить отдельно отличников).

4-1. Дана запись с именем STUDENT, содержащая следующие поля:

- Фамилия и инициалы;
- Номер группы;
- Успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 записей типа STUDENT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- Вывод на экран фамилий и номеров групп для всех студентов, если средний балл студента больше 4 (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по возрастанию номера группы.

4-2. Дана запись с именем STUDENT, содержащая следующие поля:

- Фамилия и инициалы;
- Номер группы;
- Успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 записей типа STUDENT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- Вывод на экран фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5 (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по величине среднего балла.

4-3. Дана запись с именем STUDENT, содержащая следующие поля:

- Фамилия и инициалы;
- Номер группы;
- Успеваемость (массив из пяти элементов);

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 записей типа STUDENT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- Вывод на экран фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2 (если таких нет – вывести об этом сообщение);

Список студентов должен быть упорядочен по алфавиту фамилий.

4-4. Дана запись с именем AEROFLOT, содержащая следующие поля:

- Название пункта назначения рейса;

- Номер рейса;
- Тип самолета.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 7 элементов типа AEROFLOT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- Вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по возрастанию номера рейса.

4-5. Дана запись с именем AEROFLOT, содержащая следующие поля:

- Название пункта назначения рейса,
- Номер рейса,
- Тип самолета

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры из 7 элементов типа AEROFLOT, и занесение их в файл данных
- Чтение данных из файла и вывод их на экран
- вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры (если таких нет – вывести об этом сообщение)
- Список должен быть упорядочен по алфавиту названий пунктов назначения.

4-6. Дана запись с именем WORKER, содержащая следующие поля:

- Фамилия и инициалы работника;
- Название занимаемой должности;
- Год поступления на работу.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 элементов типа WORKER, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по алфавиту фамилий.

4-7. Дана запись с именем TRAIN, содержащая следующие поля:

- Название пункта назначения;
- Номер поезда;
- Время отправления.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа TRAIN, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по алфавиту пунктов назначения.

4-8. Дана запись с именем TRAIN, содержащая следующие поля:

- Название пункта назначения;
- Номер поезда;
- Время отправления.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 6 элементов типа TRAIN, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о поездах, отправляющихся в пункт, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по времени отправления поезда.

4-9. Дана запись с именем TRAIN, содержащая следующие поля:

- Название пункта назначения;
- Номер поезда;
- Время отправления.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа TRAIN, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о поезде, номер которого введен с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по номерам поездов.

4-10. Дана запись с именем MARSH, содержащая следующие поля:

- Название начального пункта назначения;
- Название конечного пункта назначения;
- Номер маршрута.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа MARSH, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о маршруте, номер которого введен с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по номерам маршрутов.

4-11. Дана запись с именем MARSH, содержащая следующие поля:

- Название начального пункта назначения;
- Название конечного пункта назначения;
- Номер маршрута.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа MARSH, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о маршрутах, которые начинаются или заканчиваются в пункте, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по номерам маршрутов.

4-12. Дана запись с именем NOTE, содержащая следующие поля:

- Фамилия, имя;
- Номер телефона;
- Дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа NOTE, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о человеке, номер телефона которого введен с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по датам рождения.

4-13. Дана запись с именем NOTE, содержащая следующие поля:

- Фамилия, имя;
- Номер телефона;
- Дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа NOTE, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по алфавиту.

4-14. Дана запись с именем NOTE, содержащая следующие поля:

- Фамилия, имя;
- Номер телефона;
- Дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа NOTE, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по двум первым цифрам номера телефона.

4-15. Дана запись с именем ZNAK, содержащая следующие поля:

- Фамилия, имя;
- Знак Зодиака;
- Дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа ZNAK, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если таких нет – вывести об этом сообщение).
- Список должен быть упорядочен по датам рождения.

4-16. Дана запись с именем ZNAK, содержащая следующие поля:

- Фамилия, имя;
- Знак Зодиака;

• Дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

• Ввод с клавиатуры данных из 8 элементов типа ZNAK, и занесение их в файл данных;

• Чтение данных из файла и вывод их на экран;

• вывод на экран информации о людях, родившихся под знаком, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• Список должен быть упорядочен по датам рождения.

4-17. Дана запись с именем ZNAK, содержащая следующие поля:

• Фамилия, имя;

• Знак Зодиака;

• Дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

• Ввод с клавиатуры данных из 8 элементов типа ZNAK, и занесение их в файл данных;

• Чтение данных из файла и вывод их на экран;

• вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• Список должен быть упорядочен по знакам Зодиака.

4-18. Дана запись с именем PRICE, содержащая следующие поля:

• Название товара;

• Название магазина, в котором продается товар;

• Стоимость товара в руб.

Написать программу, которая выполняет следующие действия:

• Ввод с клавиатуры данных из 8 элементов типа PRICE, и занесение их в файл данных;

• Чтение данных из файла и вывод их на экран;

• вывод на экран информации о товаре, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• Список должен быть упорядочен по алфавиту названий товара.

4-19. Дана запись с именем PRICE, содержащая следующие поля:

• Название товара;

• Название магазина, в котором продается товар;

• Стоимость товара в руб.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа PRICE, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по алфавиту названий магазинов.

4-20. Дана запись с именем ORDER, содержащая следующие поля:

- Расчетный счет плательщика;
- Расчетный счет получателя;
- Перечисляемая сумма в руб.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа ORDER, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры (если таких нет – вывести об этом сообщение).
- Список должен быть упорядочен по расчетным счетам плательщиков.

ЛИТЕРАТУРА

1. Павловская Т. C/C++. Программирование на языке высокого уровня. Учебник. – СПб.: Питер, 2001 – 460 с.

2. Павловская Т., Щупак Ю.C++. Объектно-ориентированное программирование. Практикум. – СПб.: Питер, 2006 – 264 с.

3. Лафоре Р. Объектно-ориентированное программирование в C++. – СПб.: Питер, 2007. – 928 с.

4. Хортон А. Visual C++ 2005: базовый курс. – М.: ООО «И.Д. Вильямс», 2007 – 1152 с.

ТЕСТЫ

Тесты по основам алгоритмизации[2]

1. Какой тип структуры потребуется для записи алгоритма вычисления значения функции?

$$y=|x|=\begin{cases} x, \text{ если } x\geq 0 \\ -x, \text{ если } x<0 \end{cases}$$

A) полная развилка,
B) неполная развилка,
C) следование,
D) цикл «ДО».

2. Какая фигура обязательно присутствует в блок-схеме алгоритма дважды?

A) прямоугольник, B) овал, C) параллелограмм, D) ромб.

3. Какой тип алгоритма потребуется для поиска наибольшего из двух чисел?

A) Линейный, B) циклический (типа «ПОКА»),
C) алгоритм с ветвлением, D) циклический (типа «ДО»)

4. Какой из алгоритмов является алгоритмом с ветвлением?

A) алгоритм вычисления площади треугольника по его сторонам,
B) алгоритм удаления всех шаров из урны по одному,
C) алгоритм определения кислотности раствора,
D) алгоритм поиска наибольшего из *N* чисел,

5. Алгоритм должен быть применим для некоторого класса задач, различающихся лишь исходными данными. Это свойство называется:

A) Результативностью, B) массовостью,
C) дискретностью, D) определенностью.

6. Какой тип алгоритма потребуется для поиска набольшего из N чисел?

A) Линейный, B) циклический без условия внутри,
C) циклический с условием внутри, D)алгоритм с ветвлением.

[2] Тесты с 11 по 35 данного раздела взяты с сайта www.fepo.ru

7. Какой тип алгоритма потребуется для вычисления суммы:
1–1/2+1/3–1/4+…с заданной точностью
A) линейный, B) циклический (типа «ПОКА»),
C) алгоритм с ветвлением, D) циклический (типа «ДО»)

8. Какой из алгоритмов не является циклическим?
A) поиск наибольшего из N чисел,
B) удаление всех шаров из урны по одному,
C) упорядочивание N элементов по возрастанию,
D) решение квадратного уравнения.

9. Алгоритм должен приводить к решению задачи за конечное число шагов. Это свойство называется:
A) результативностью, B) массовостью,
C) дискретностью, D) определенностью,

10. Часть программы, служащую для решения некоторой вспомогательной задачи, называют…
A) подпрограммой, B) библиотекой,
C) надстройкой, D) объектом,

11. Процедура KLMN реализует следующее алгебраическое выражение…

```
ПРОЦЕДУРА KLMN;
НАЧАТЬ ПРОЦЕДУРУ
||ПИСАТЬ('ВВЕДИТЕ ЗНАЧЕНИЕ K, L, M, N');
||ЧИТАТЬ (K, L, M, N);
||ЕСЛИ K=L ТО
||||ЕСЛИ M<N ТО
||||||X:=1
||||ИНАЧЕ
||||||X:=2
||||КОНЕЦ ЕСЛИ
||ИНАЧЕ X:=3
||КОНЕЦ ЕСЛИ
КОНЕЦ ПРОЦЕДУРЫ;
```

A) $$X = \begin{cases} 1, \text{если} K = L \text{ и } M < N; \\ 2, \text{если} K = L; \\ 3, \text{если} K \neq L \text{ и } M \geq N. \end{cases}$$

B)

$$X = \begin{cases} 1, \text{если} K = L \text{ и } M < N; \\ 2, \text{если} K = L \text{ и } M \geq N; \\ 3, \text{если} K \neq L. \end{cases}$$

C)

$$X = \begin{cases} 1, \text{если} K = L; \\ 2, \text{если} K \leq L \text{ и } M < N; \\ 3, \text{если} K \neq L \text{ и } M \geq N. \end{cases}$$

D)

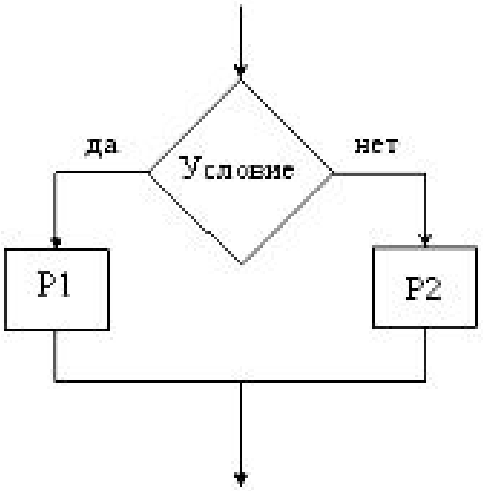
$$X = \begin{cases} 1, \text{если} K = L; \\ 2, \text{если} K \leq L \text{ и } M \leq N; \\ 3, \text{если} K \neq L \text{ и } M \geq N. \end{cases}$$

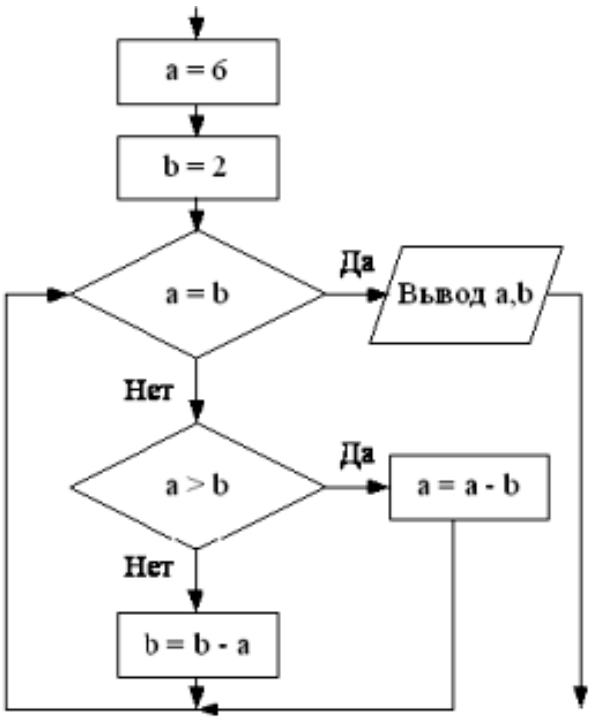
12. Обнаруженное при тестировании нарушение формы записи программы приводит к сообщению об ошибке…

A) грамматической, B) тематической,
C) синтаксической, D) орфографической.

13. На рисунке представлен фрагмент алгоритма, имеющий _____ структуру.

A) циклическую с предусловием,
B) разветвляющуюся,
C) циклическую с постусловием,
D) линейную.





14. Дан фрагмент блок-схемы алгоритма. На выходе **a, b** будут принимать значения…

A) a = 4, b = 2,
B) a = 4, b = 4,
C) a = 2, b = 2,
D) a = 2, b = 4,
E) a = 0, b = 0.

15. Параметры, указываемые в заголовке подпрограммы, называются…

A) формальными, B) глобальными,
C) фактическими, D) абсолютными,

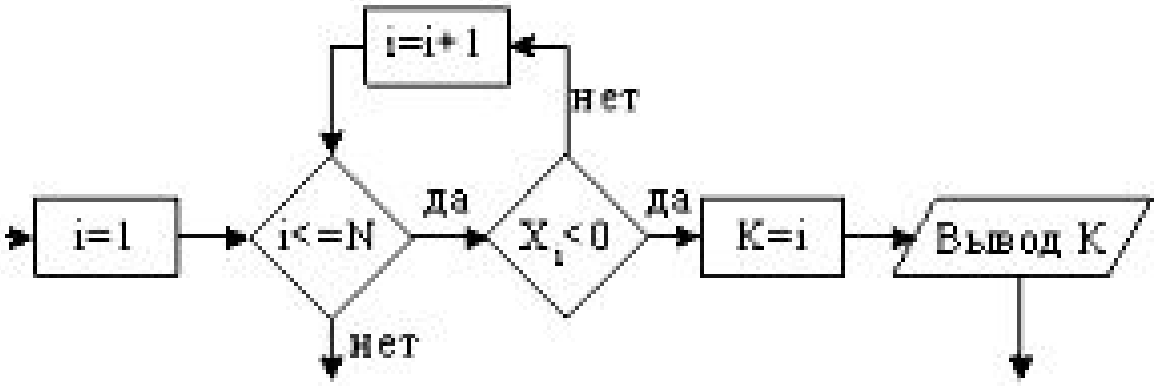
16. Что означает положительный результат при тестировании программных систем:

A) есть замечания, B) ошибки исправлены,
C) ошибки найдены, D) ошибки не найдены,

17. Какие типы конструкций не должны применяться для реализации логики алгоритма и программы, с точки зрения структурного программирования:

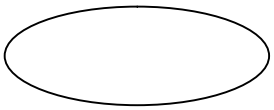
A) безусловные переходы, B) повторение (циклы),
C) последовательное выполнение, D) ветвление,

18. Задан одномерный массив X1, X2,… XN. Фрагмент алгоритма (приведен ниже) определяет…



A) количество отрицательных элементов,
B) индекс первого отрицательного элемента,
C) индекс последнего отрицательного элемента,
D) минимальный элемент массива,
E) индекс минимального элмемента массива.

19. Приведенный символ обозначает:



A) начало программы, B) ввод/вывод данных,
C) условный оператор, D) конец программы,

20. Методика анализа, проектирования и написания приложений с помощью структуры классов, каждый из которых является целостным фрагментом кода и обладает свойствами и методами, называется _____ программированием

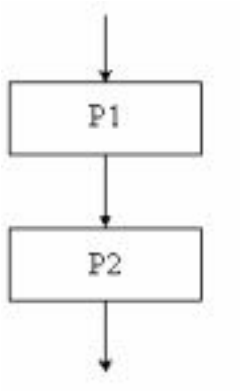
A) объектно-ориентированным, B) модульным,
C) структурным, D) формальным.

21. Этап разработки программ, состоящий в формировании исходного текста программы на одном из языков программирования в соотвествии с заданным алгоритмом, получил название…
A) исходный код, B) этап системного анализа,
C) этап реализации, D) этап моделирования,
E) этап кодирования.

22. Ни в одном языке программирования нет _____ выражений
A) логических, B) арифметических, C) этап реализации,
D) физических, E) текстовых

23. как называется деятельность, направленная на обнаружение и исправление ошибок в программной системе:
A) рефакторинг, B) тестирование,
C) отладка, D) демонстрация.

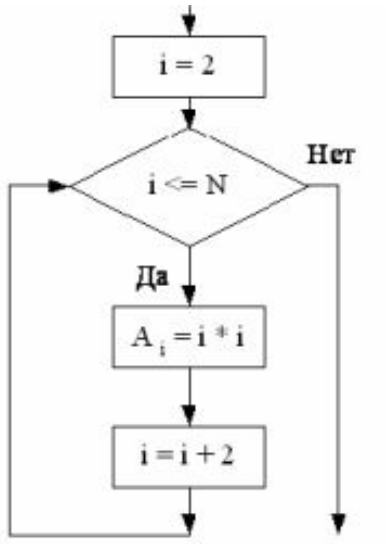
24. Операторы присваивания…
A) вычисляют значения математических выражений,
B) меняют значения констант,
C) организуют выполнение повторяемых действий,
D) задают значения переменных.



25. На рисунке представлен фрагмент алгоритма, имеющий _____ структуру

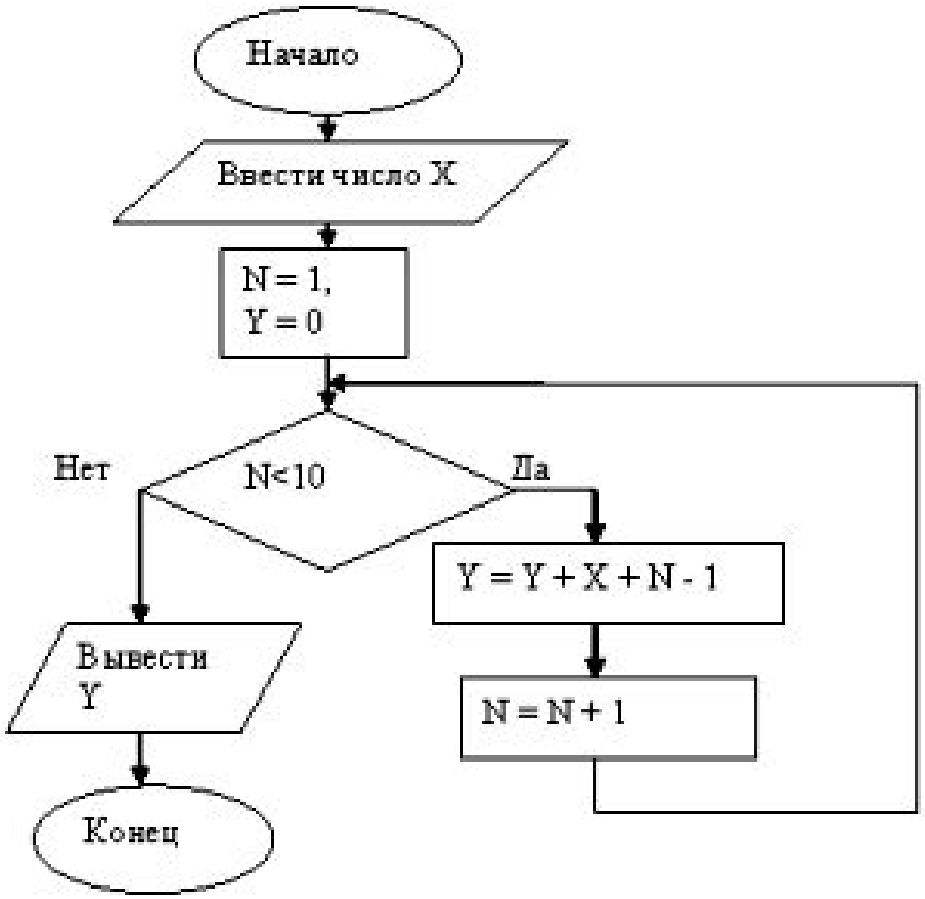
A) циклическую с предусловием,
B) разветвляющуюся,
C) циклическую с постусловием,
D) линейную.

26. В результате работы фрагмента алгоритма элементы массива A2, A4, A6, A8 при N=8 получат соответственно значения…



A) 4, 16, 36, 64,
B) 2, 4, 16, 32,
C) 4, 16, 32, 48,
D) 4, 12, 24, 36,
E) 1, 9, 25, 49.

27. Данная блок-схема программы

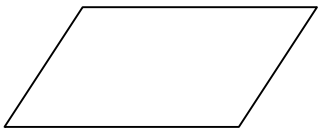


A) производит сложение 9 подряд идущих натуральных чисел, начиная с введенного и выводит результат,

B) возводит введенное число в 9 степень и выводит результат,

C) возводит введенное число в 10 степень и выводит результат,

D) производит сложение 10 подряд идущих натуральных чисел, начиная с введенного и выводит результат.

28. Символ  обозначает:

A) условный оператор, B) начало программы,

C) ввод/вывод данных, D) конец программы.

29. Процесс поиска ошибок в программе принято называть…

A) профилактикой, B) испытанием,

C) отладкой, D) ремонтом, E) диагностикой.

30. Многократное исполнение одного и того же участка программы называется…

A) обращением к подпрограмме, B) рекурсией,

C) итерацией, D) циклическим процессом.

31. Какие из перечисленных языков наиболее подходят для системного программирования:

A) Java, B) PHP, C) C,C++, D) Accembler.

32. Использование подпрограмм в программе позволяет организовать выполнение…

A) ввода данных в программу,

B) циклических вычислений в соотвествии с алгоритмом,

C) разветвление вычислений в соотвествии с алгоритмом,

D) повторяющихся действий над различными исходными данными.

33. Какие виды подпрограмм **не присущи** языку программирования C:

A) процедуры, B) функции, C) методы, D) модули,

34. Параметры, указываемые в момент вызова подпрограммы из основной программы, называются…

A) постоянными, B) фактическими,
C) глобальными, D) абсолютными,

35. Дан массив целых чисел {Ai}, где i = 1, 2, 3 …, M. Пусть M равно 15. Программа вычисляет произведение сумм некоторых элементов этого массива. В программе введены следующие константы: G = 1; W = 12; T = 8; L = 15.

```
ПРОГРАММА 15;
    ФУНКЦИЯ SUMMA(I1,I2);
    НАЧАТЬ ФУНКЦИЮ
    ||S:=0;
    ||НЦ ДЛЯ I:=I1 ДО I2
    ||||S:=S + A[I]
    ||КЦ;
    ||SUMMA:=S
    КОНЕЦ ФУНКЦИИ;
НАЧАТЬ ПРОГРАММУ
||ПИСАТЬ ('ВВЕДИТЕ ЗНАЧЕНИЯ МАССИВА A:' );
||НЦ ДЛЯ J:=1 ДО M
||||ЧИТАТЬ (A[J]);
||КЦ;
||P:=SUMMA (G, W)*SUMMA(T, L);
||ПИСАТЬ ('ПРОИЗВЕДЕНИЕ РАВНО:', P:6)
КОНЕЦ ПРОГРАММЫ.
```

A)

$$P = \sum_{i=1}^{15} A_i \cdot \sum_{j=1}^{6} A_j$$

B)

$$P = \sum_{i=1}^{8} A_i \cdot \sum_{j=1}^{12} A_j$$

C)

$$P = \sum_{i=1}^{12} A_i \cdot \sum_{j=8}^{12} A_j$$

D)

$$P = \sum_{i=1}^{12} A_i \cdot \sum_{j=8}^{15} A_j$$

Тесты по программированию на C++

1. Какой из указанных типов описывает вещественный тип (с плавающей точкой):

A) int B) float C) double D) real E) long double

2. Как обозначается логическое умножение (конъюнкция) на Borland C++

A) and B) & C) && D) | | E) **

3. Логическая операция равенства в C++ обозначается

A) := B) =: C) = = D) = E) | |

4. Какая запись соответствует операции декремента в префиксной форме

A) i++ B) ++I C) i– – D) – –I E) такого нет

5. Дано значение угла X (в градусах). Требуется найти тангенс данного угла. Укажите правильную запись (здесь PI=3,141592…):

A) tan(x)
B) sin(x)/cos(x)
C) sin(x*PI/180)/cos(x*PI/180)
D) tg (x)
E) tan (x*PI/180)

6. Для заданных целых **a** и **b** вывести сумму a+b, если они одного знака и разность a – b, если они имеют разные знаки

A) if ((a>0)&&(b>0)) | | ((a<0)&&(b<0))) cout>>a+b; else cout a–b;
B) if (a*b>0) cout<<(a+b); else cout<<(a-b)
C) if (a*b<0) cout<<(a+b); else cout<<(a-b)
D) if a*b>0 cout<<(a+b); else cout<<(a-b)
E) if a*b>0 cout<<(a+b); else cout<<(a-b)

7. Какие значения напечатает программа:

```
x=1;
for (i=1; i<= 6; i++) {cout<<(x*x); x=x+2 ;}
```

A) 1; 3; 5
B) 1; 3; 5; 7; 9; 11
C) 1; 9; 25; 49; 81; 121
D) 1; 9; 25

8. Укажите, где правильно описан двумерный целочисленный массив, состоящий из 20 элементов

A) int a [10] [10]
B) int a [20]
C) int a [2][10]
D) int a [2, 10]

9. При заданных исходных данных a=3, b=2 определите результат выполнения фрагмента программы

```
cout<<«введи a,b»; cin>>a>>b;
while (a>b) a=a–1;
cout<< a <<b;
```

A) a=2, b=2 B) a=1, b=1 C) a=1, b=2 D) a=0, b=2

10. Где собственно описывается функция при создании функции пользователя?

A) до открытия главной функции main()
B) за закрывающей скобкой функции main()
C) в любом месте
D) после описания переменных внутри функции main()
E) до описания переменных внутри функции main()

11. Какой символ ставится перед именем указателя?

A) & B)= C)* D)P E) никакой не ставится

Учебное издание

Александр Александрович Шелупанов,
Василий Николаевич Кирнос

ИНФОРМАТИКА. БАЗОВЫЙ КУРС

Учебник
(в четырех частях)

Часть 3. Основы алгоритмизации и программирования в среде Visual C++2005.

Издательство «В-Спектр»
Подписано к печати 10.01.2009.
Формат $60\times84^{1}/_{16}$. Печать трафаретная.
Печ. л. 13,5. Усл. печ. л. 12,7.
Тираж 500 экз. Заказ 1.

Тираж отпечатан в издательстве «В-Спектр»
ИНН/КПП 7017129340/701701001, ОГРН 1057002637768
634055, г. Томск, пр. Академический, 13-24, Тел. 49-09-91.
E-mail: bmwm@list.ru