Министерство науки и высшего образования РФ
ФГБОУ ВО «Томский государственный университет
систем управления и радиоэлектроники»
Кафедра безопасности информационных систем (БИС)

С.С. Харченко

УЧЕБНАЯ ПРАКТИКА

Учебно-методическое пособие

для студентов специальностей и направлений
10.03.01 – «Информационная безопасность»,
10.05.03 – «Информационная безопасность автоматизированных систем»,
10.05.02 – «Информационная безопасность телекоммуникационных систем»,
10.05.04 – «Информационно-аналитические системы безопасности»

В-Спектр
Томск, 2021

УДК 004.42
ББК 32.973
Х 22

Х 22 **Харченко С.С.** Учебная практика: учебно-методическое пособие. – Томск: В-Спектр, 2021. – 66 с.
ISBN 978-5-91191-452-3

Учебно-методическое пособие содержит теоретические сведения и методические указания необходимые для прохождения практики по получению первичных профессиональных умений и навыков или учебно-лабораторного практикума, рекомендации по оформлению отчетной документации заданий по практике, и предназначено для студентов 1-го курса, обучающихся по специальностям: 10.03.01 – «Информационная безопасность», 10.05.03 – «Информационная безопасность автоматизированных систем», 10.05.02 – «Информационная безопасность телекоммуникационных систем», 10.05.04 – «Информационно-аналитические системы безопасности». Охарактеризованы требования к практическим заданиям, описана структура и методика выполнения работ. В пособии кратко рассматривается методика разработки алгоритмов, порядок разработки программ на языке C# в среде разработки Microsoft Visual Studio 2019. При этом, как и положено, рассматривается алфавит языка, реализация основных типов алгоритмических структур (линейная, разветвленная, циклическая), работа с массивами. Рассматриваются основные стандарты разработки технической документации на создание автоматизированных систем. В конце данного пособия, приводится глава с заданием для выполнения работ по данной дисциплине, пособие поделено на главы, соответствующие темам работ.

УДК 004.42
ББК 32.973

ISBN 978-5-91191-452-3

Содержание

Предисловие....4
Алгоритмы....6
Описание алгоритмов....7
Создание консольного приложения....14
Типы данных и операнды....17
Линейные программы....21
Программы с ветвлением....23
Программы с циклами....26
Массивы....32
Техническая документация....34
Общие требования к содержанию отчета по учебной практики....37
Задания на разработку алгоритмов....38
ЛИТЕРАТУРА....45
ПРИЛОЖЕНИЕ А....46
ПРИЛОЖЕНИЕ Б....47
ПРИЛОЖЕНИЕ В....48
ПРИЛОЖЕНИЕ Г....60

Предисловие

Цель учебной практики – научить студентов способности применять программные средства системного, прикладного и специального назначения, инструментальные средства, языки и системы программирования для решения профессиональных задач.

Задачи практики:

– научить студентов разрабатывать алгоритмы решения простых задач;

– научить студентов навыкам программирования на языке программирования высокого уровня;

– научить студентов реализовывать алгоритмы на языке программирования высокого уровня;

– научить студентов разрабатывать начальную проектную документацию по созданию программного обеспечения (автоматизированных систем).

Процесс прохождения практики направлен на поэтапное формирование и закрепление следующих компетенций:

– способностью применять программные средства системного, прикладного и специального назначения, инструментальные средства, языки и системы программирования для решения профессиональных задач (ПК-2);

– способностью к освоению новых образцов программных, технических средств и информационных технологий (ОПК-8);

– способностью разрабатывать научно-техническую документацию, готовить научно-технические отчеты, обзоры, публикации по результатам выполненных работ (ПК-7);

– способностью применять программные средства системного и прикладного назначения, языки, методы и инструментальные средства программирования для решения профессиональных задач (ОПК-5);

– способностью осуществлять анализ научно-технической информации, нормативных и методических материалов по методам обеспечения информационной безопасности телекоммуникационных систем (ПК-1);

– способностью формировать технические задания и участвовать в разработке аппаратных и программных средств защиты информационно-телекоммуникационных систем (ПСК-10.2);

– способностью применять в профессиональной деятельности современные средства вычислительной техники и программное обеспечение, достижения информационных технологий для поиска и обработки информации по профилю профессиональной деятельности (ОПК-3);

– способностью проводить комплексный анализ функционирования финансовых и экономических структур государственного или системообразующего уровня с целью выявления угроз (отрицательных тенденций) национальной безопасности Российской Федерации (ПСК-2.1);

– способностью выполнять анализ корректности и устойчивости функционирования отдельных компонентов, подсистем и в целом всей национальной системы по противодействию легализации доходов, полученных преступным путем, и финансированию терроризм (ПСК-2.2);

В результате прохождения практики обучающийся должен:

– знать основы языка программирования высокого уровня, правила оформления алгоритмов, основы проектирования автоматизированных систем;

– уметь разрабатывать линейные алгоритмы, алгоритмы с ветвлением, циклические алгоритмы, реализовывать алгоритмы в виде программного обеспечения на языке программирования высокого уровня, разрабатывать начальную проектную документацию на разработку автоматизированных систем на основе ГОСТ и/или IEEE стандартов;

– владеть основными конструкциями языка программирования высокого уровня, навыками разработки алгоритмов, навыками разработки начальной проектной документации на создание программного обеспечения.

Указанные задачи частично могут быть решены с помощью данного учебного пособия.

Алгоритмы

Алгоритм – это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, предопределённость, ввод, вывод, эффективность [1]. Существуют и другие определения алгоритма, например: алгоритм – это всякая система вычислений, выполняемых по строго определённым правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи [2]; или алгоритм – это набор предписаний, однозначно определяющий содержание и последовательность выполнения действий для систематического решения задачи [3]. Так или иначе, большинство определений понятия «алгоритм» сводится к тому, что существует набор свойств, которым должна соответствовать последовательность действий, чтобы называться алгоритмом: дискретность, результативность, определенность, понятность, конечность, универсальность. Рассмотрим подробно свойства, которыми должен обладать алгоритм.

Дискретность – заключается в том, что алгоритм должен состоять из последовательности простых шагов, кроме того, каждый шаг должен гарантированно выполняться за конечный отрезок времени.

Результативность – алгоритм должен завершаться определёнными результатами, предусмотренными условиями задачи.

Определённость – на любом шаге выполнения алгоритма, следующий шаг работы однозначно определяется состоянием системы.

Понятность – алгоритм должен включать только те команды, которые доступны исполнителю и входят в его набор команд.

Конечность – при корректно заданных входных данных, алгоритм должен завершать работу и возвращать результат за конечное число шагов.

Универсальность – алгоритм должен быть применим к различным наборам входных данных.

Существует четыре разновидности алгоритмов: линейный, алгоритм с ветвлением, циклический алгоритм и сложный алгоритм.

Алгоритм называется **линейным** если выполнение команд происходит последовательно друг за другом. **Алгоритм с ветвлением** – алгоритм, содержащий хотя бы одно условие, в результате проверки которого может осуществляться разделение на несколько альтернативных ветвей алгоритма. **Циклический алгоритм** – алгоритм, предусматривающий многократное повторение одного и того же действия или набора действий над новыми исходными данными. Сложный алгоритм – алгоритм, содержащий комбинации линейного алгоритма, алгоритма с ветвлением и циклического алгоритма. На практике чаще всего встречаются именно сложные алгоритмы.

Описание алгоритмов

Существует три основных способа записи алгоритмов: словесный способ, графический способ.

Словесный способ записи алгоритма – способ записи алгоритма на естественном языке. Данный способ очень удобен, если нужно приближенно описать суть алгоритма. Однако если придерживаться, некоторых общепринятых договоренностей при описании алгоритмов, можно это делать достаточно точно. Далее будет показано как этого добиться.

Графический способ описания алгоритмов – использование геометрических фигур для обозначения различных инструкций и команд, также называемых блок-схемами в соответствии с какой-либо нотацией, например ГОСТ 19.701-90 [4].

Так же выделяют такой способ, как **программный** – когда алгоритм записывается на каком-либо языке программирования или псевдокоде. Псевдокод – это неформальный способ записи алгоритма с использованием ключевых слов для описания операторов, близких к командам первых императивных языков программирования высокого уровня (напр. PASCAL), но опускающий несущественные подробности и специфический синтаксис. Рассмотрим, более подробно каждый из способов на примере.

Задача 1. Вычислить периметр и площадь прямоугольника, длина которого равна a, а ширина равна b.

Для начала составим алгоритм для решения задачи словесным способом, опишем его:

– начало алгоритма;

– задать численное значение стороны a.

– задать численное значение стороны b.

– вычислить площадь S прямоугольника по формуле S=a*b.

– вывести результат вычислений.

– конец алгоритма.

Теперь запишем тот же самый алгоритм, но придерживаясь более формального и краткого способа записи:

Алгоритм A:

A.0 начало;
A.1 ввод a, b;
A.2 S ← a*b;
A.3 вывод S;
A.4 остановка;

Теперь представим алгоритм A в виде блок-схемы нарисованной в соответствии с ЕСПД (рис. 1)

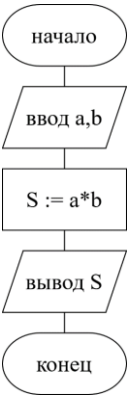


Рис. 1. Блок-схема алгоритма А

На псевдокоде алгоритм решения задачи 1 выглядит следующим образом:

```
    begin
  var a,b,S
  read a,b
  S:= a*b
  write S
    end
```

При описании алгоритмов словесным способом, рекомендуется придерживаться следующих простых правил:

1. Обозначать каждый алгоритм заглавной буквой латинского алфавита и нумеровать все шаги, например: алгоритм А и шаги А1, А2, А3 и т.д.;
2. Выполнение шагов всегда идет последовательно если явно не указано другого;
3. Последний шаг всегда – шаг «остановка», все ветви алгоритма должны приводить к нему;
4. Использовать четкие и емкие синтаксические конструкции естественного языка для обозначения ввода-вывода, разветвления, повторения действий (организация циклов): «ввод …», «вывод …», «если …, то … иначе …», «пока … повторять … после …»;
5. Для обозначения математических и логических операций не использовать обозначения, которые можно трактовать не однозначно. Например, для операции присваивания использовать обозначения «:=» или «←» вместо «=».

Рассмотрим особенности создания блок-схем в соответствии с ГОСТ 19.701-90, основные блоки, используемые для описания алгоритмов графическим способом представлены на рис. 2.

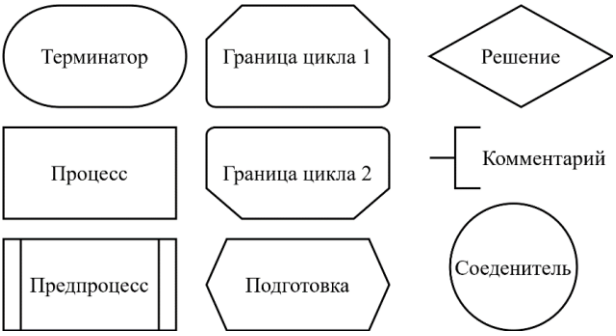


Рис. 2. Основные блоки для описания алгоритмов

Блок «терминатор» – отображает выход во внешнюю среду и вход из внешней среды. При описании алгоритмов следует использовать для обозначения начала и конца алгоритма, внешнее использование или обработки исключений(ошибок). Нередко допускают ошибку и рисуют это блок в виде овала, стоит обратить внимание, что блок терминатор изображается в виде суперэллипса.

Блок «процесс» – используется для обозначения обработки данных любого вида, выполнения определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации, в блоке следует размещать максимально простые операции, понятные исполнителю.

Блок «предпроцесс» – отображает предопределенный процесс, состоящий из одной или нескольких операций и/или шагов программы, которые определены в другом месте, в случае использования этого блока требуется представлять декомпозицию функции или метода, которая раскрывает реализацию.

Блок «граница цикла» – состоит из двух частей, используется для обозначения границ цикла, должны иметь один и тот же идентификатор, который пишется внутри блок, так же в блоке должны быть обозначены условия завершения цикла. Если речь идет о цикле с предусловием, то условие завершения цикла должно быть описано в блоке, который обозначает начало цикла, в случае с циклом с постусловием в блоке, который обозначает конец цикла.

Блок «подготовка» – отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию, другими словами, позволяет разграничивать шаги, направленные на подготовку к работе. В ГОСТ 19.701-90 нет четкого указания об использовании этого блока для обозначения цикла с параметром, однако на практике он используется именно для этого. Действительно для организации работы цикла с параметром, необходимо на каждом шаге (итерации) изменять какое-либо значение – параметр.

Блок «решение» – отображает решение или функцию переключательного типа, имеет всегда один вход и два или более выходов, из которых только один будет выполнен в зависимости от условий, обозначенных в теле этого блока.

Блок «комментарий» – используется для добавления пояснений и комментариев к блок-схеме, в целях повышения читаемости алгоритма.

Блок «соединитель» – символ отображает выходы и выходы частей блок схемы, вход и выход должны иметь одно и тоже обозначение, как правило для этого используются арабские цифры либо прописные буквы латинского алфавита. Соединитель можно использовать как для межстраничного разделения частей блок-схемы, так и для переноса направления алгоритма в рамках одной страницы, для того чтобы избежать загромождения блок схемы, пересечением линий.

Линии на блок-схемах отображают поток данных или управления. Для повышения удобочитаемости могут быть добавлены стрелки-указатели на линиях. По умолчанию считается что поток данных или управления направлен сверху-вниз и слева-право, если предполагается что имеем место быть иное направление управления или потока данных, использование стрелок-указателей **обязательно**. В блок-схемах следует избегать пересечения линий, для избегания двоякого трактования направления потока данных или управления, этого можно избежать, используя соединитель. Две или более линий могу быть объединены в одну линию, однако место объединения должно быть смещено. Линии к блокам должны подходить либо слева, либо сверху, а выходить справа и снизу, за исключением блока решения, когда используется три выхода. Линии должны выходить и входить в блоки только по центру стороны блока.

Рассмотрим несколько задач и на их примере использование словесного описания алгоритмов и использование блок-схем.

Задача 2. Составить программу для вычисления по заданному *x* значения функции: $y = \begin{cases} \sin(x), x \le 0; \\ \cos(x), x > 0. \end{cases}$

Задача 3. Составить программу для вычисления по заданному *x* значения функции: $y = \begin{cases} \sin(x), x = 1; \\ \cox(x), x = -1; \\ 0, x \ne 1, x \ne -1. \end{cases}$

Составим алгоритмы для решения задач Алгоритм B и Алгоритм C для задач 2 и 3 соответственно и составим для них блок-схемы:

Алгоритм B:

B.1 ввод x;

B.2 если $x \le 0$, то B.3, иначе B.4

B.3 y ← sin(x), перейти к B.5;

B.4 y ← cos(x);

B.5 вывод y, остановка;

Алгоритм C:

C.1 ввод x;

C.2 если x = 1, то C.3, иначе если x = -1, то C.4, иначе C.5

C.3 y ← sin(x), перейти к C.6;

C.4 y ← cos(x), перейти к C.6;

C.5 y ← 0;

C.6 вывод y, остановка;

Блок-схемы алгоритмов B и C представлены на рис. 3.

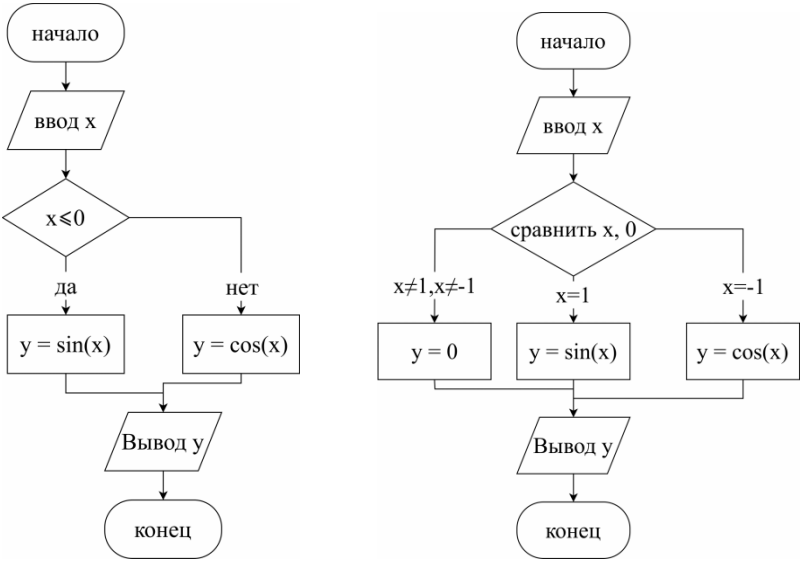


Рис. 3. Блок-схемы алгоритмов B и C

Задача 4. Рассчитать для заданного *N* величину *N!*.

Составим алгоритм для решения задачи 4 (алгоритм D).

Алгоритм D:

D.1 ввод N;

D.2 p←1; i ← 1;

D.3 если i ≤N то D.4, иначе D.5

D.4 p ←p*i; i←i+1;

D.5 вывод p, остановка.

Другой вариант алгоритма D может выглядеть следующим образом:

D.1 ввод N;

D.2 p←1;

D.3 для i от 1 до N включительно, с шагом 1 выполнять D.4;

D.4 p ←p*i; возврат;

D.5 вывод p, остановка.

Еще один вариант алгоритма решения задачи 4 выглядит следующим образом:

D.1 ввод N;

D.2 p←1; i ← 1;

D.3 пока i ≤ N выполнять D.4;

D.4 p ←p*i; i←i+1; возврат;

D.5 вывод p, остановка.

Рассмотри отличия представленных алгоритмов. Первый алгоритм предполагает использование условного оператора и оператора перехода, такой подход свойственен реализациям на языках низкого уровня, и языкам высокого уровня раннего периода и не рекомендуется к использованию так как оператор перехода (GOTO) является рудиментом языков программирования и считается не безопасным. Второй алгоритм предполагает использование цикла с параметром, важно указывать границы изменения параметра, строгость границ (включительно/не включительно) и шаг изменения параметра. Третий алгоритм реализует логику с использованием цикла с условием, важно на одном и шагов, относящихся к телу цикла не забывать изменять переменную, которая используется в проверочном условии, иначе можно получить бесконечный цикл. Поскольку словесное описание алгоритмов разное, блок-схемы этих алгоритмов так же будут выглядеть по-разному (рис. 4).

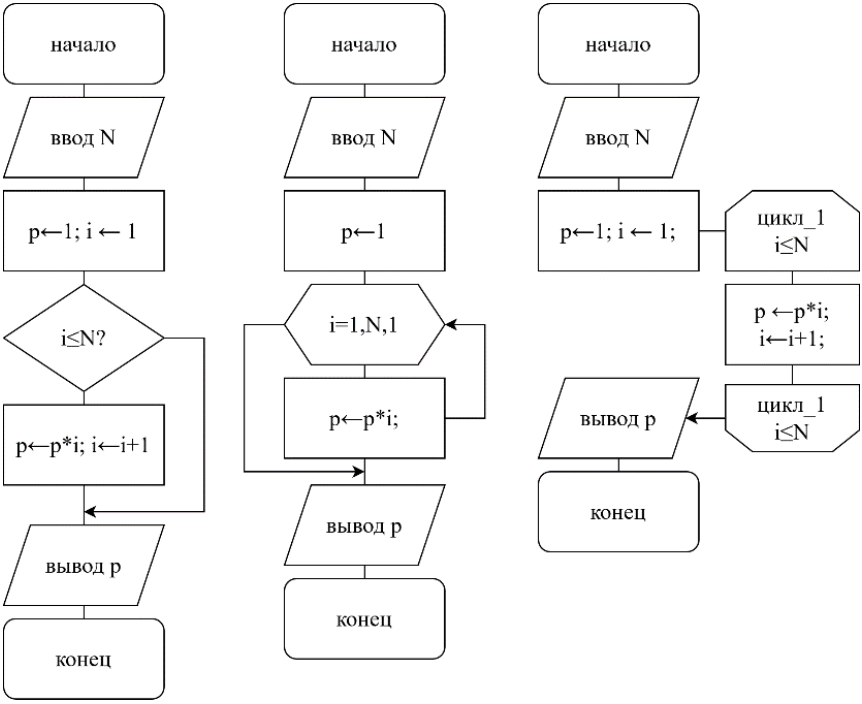


Рис. 4. Блок-схемы разных вариантов алгоритма D

Важно отметить, что на блок-схемах при использовании цикла с параметром явно указывать включительно или не включительно изменяется параметр, существует договоренность всегда считать, что указанно изменение

параметра включительно, если не оговорено иного, явно на блок-схеме при помощи комментария или сноски. Также указывается возврат управления циклу, кроме этого, можно явно указать к какому именно шагу (D.3).

Рассмотрим случай, когда требуется вложенность нескольких циклов.

Задача 5. Получить таблицу умножения в виде таблицы Пифагора.

Алгоритм E:

E.1 для i от 1 до 9 включительно, с шагом 1 выполнять E.2;

E.2 для j от 1 до 9 включительно, с шагом 1 выполнять E.3, E.4;

E.3 k ←i*j;

E.4 вывод k; возврат к E.2

E.5 переход на новую строку; возврат к E.1

E.6 остановка.

На шагах 4 и 5 важно явно указывать возврат в какому шагу передается управление. Блок-схема алгоритма E представлена на рис. 5.

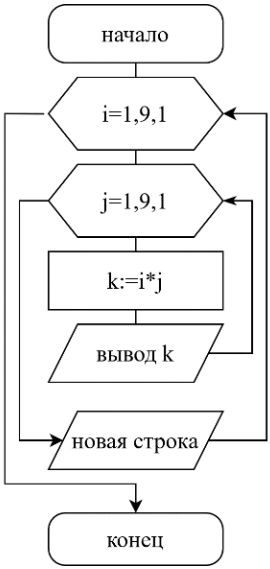


Рис. 5. Блок-схема алгоритма E

Создание консольного приложения

После запуска Visual Studio 2019 при помощи нажатия горящих клавиш «**Shift+Ctrl+N**» запускаем меню создания нового проекта, в форме поиска шаблона приложения введите «Консольное приложение» (рис. 6), далее выберите проект «Консольное приложение (.Net Framework)» либо «Консольное приложение (.Net Core)».

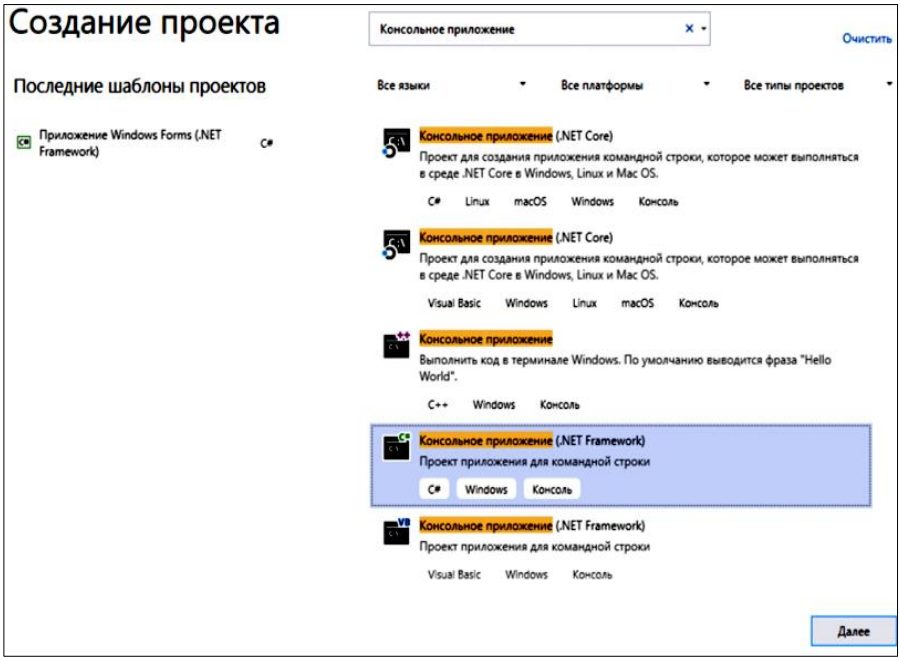


Рис. 6. Меню создания нового проекта

После выбора типа проекта откроется меню настройки проекта, где необходимо выбрать название проекта, его расположение на диске, версию .Net Framework [10] и другие характеристики (рис. 7). После создания проекта откроется главное окно Visual Studio для работы с проектом (рис. 8). В правой части окна располагается обозревателей решений, в котором можно увидеть всю иерархию файлом проекта. Большую часть по центру окна занимает редактор кода, в котором можно открывать сразу несколько вкладок для редактирования разных файлов проекта. В нижней части окна располагаются дополнительные элементы, такие как «Вывод», «Список ошибок» и др. в зависимости от пользовательских настроек.

Попробуйте ввести простой фрагмент кода, в тело основной функции «Main» внутри фигурных скобок, представленной ниже:

```
Console.WriteLine("Основы программирования");
```

Это строка кода выводит сообщение, которое размещено в кавычках. После чего исходный код должен принять следующий вид:

```
using System;
namespace BasicsOfProgrammingConsoleApplication
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Основы программирования");
      }
   }
}
```

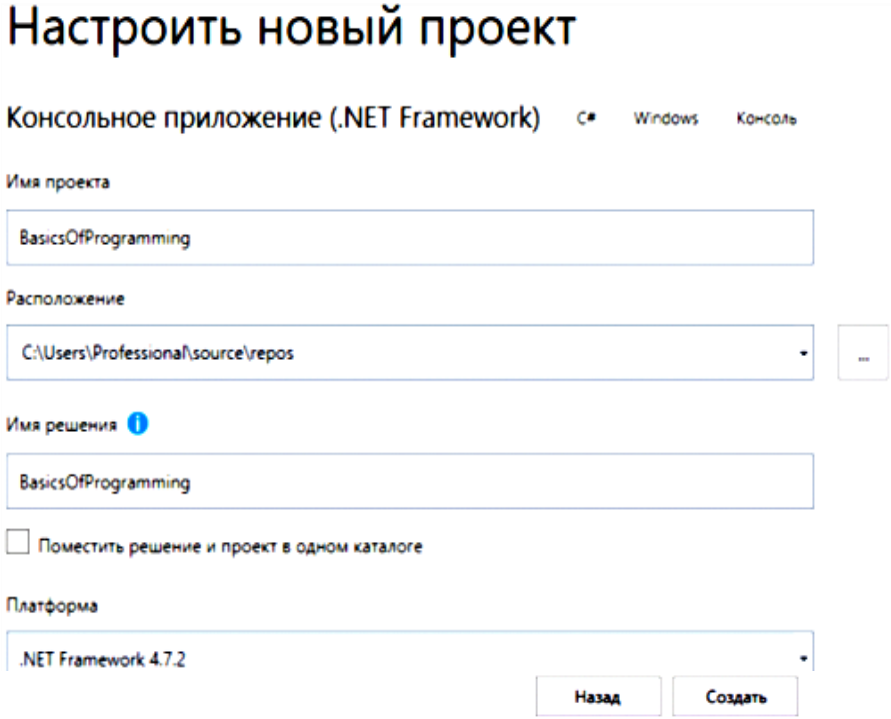


Рис. 7. Меню настройки создаваемого проекта

Название «BasicsOfProgrammingConsoleApplication» может отличается в вашем коде, оно зависит от того, как вы назвали свое приложение на этапе создания проекта.

Для запуска программы ее необходимо сначала скомпилировать и собрать проект, для этого на панели меню выберите **«Сборка → Собрать решение»** или воспользуйтесь горячими клавишами **«Ctrl+Shift+B».** Если все прошло успешно автоматически откроется оснастка «Вывод» (рис. 9).

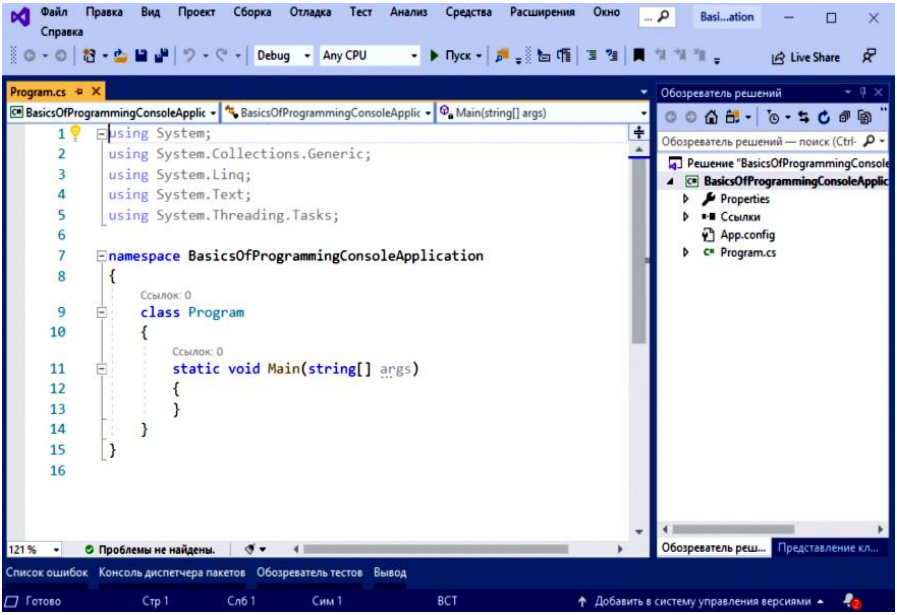


Рис. 8. Основное окно для работы с проектом

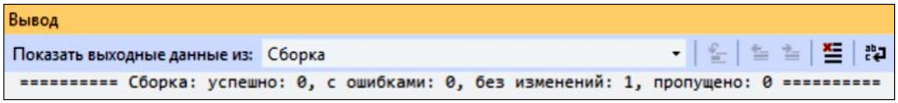


Рис. 9. Сообщение об успешной сборки проекта

Теперь для того, чтобы запустить программу перейдите в меню **«Отладка → Запуск без отладки»**. Если на этом шаге выбрать просто **«Отладка»** программа сразу же закроется после выполнения. После запуска приложения откроется стандартная консоль и в ней будет выведено сообщения «Основы программирования» (рис. 10).

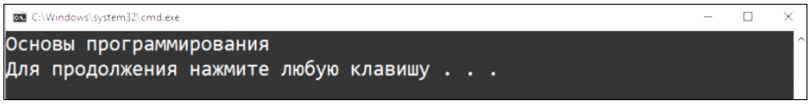


Рис. 10. Результат выполнения программы

Типы данных и операнды

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 на рис. 11.

Ключевое слово или тип C#	Диапазон	Размер/точность
sbyte	От -128 до 127	8-разрядное целое число со знаком
byte	От 0 до 255	8-разрядное целое число без знака
short	От -32 768 до 32 767	16-разрядное целое число со знаком
ushort	От 0 до 65 535	16-разрядное целое число без знака
int	От -2 147 483 648 до 2 147 483 647	32-разрядное целое число со знаком
uint	От 0 до 4 294 967 295	32-разрядное целое число без знака
long	От -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807	64-разрядное целое число со знаком
ulong	От 0 до 18 446 744 073 709 551 615	64-разрядное целое число без знака
float	От ±1,5 x 10−45 до ±3,4 x 1038	4 байта/6–9 знаков
double	от ±5,0 × 10−324 до ±1,7 × 10308	8 байт/15–17 знаков
decimal	от ±1,0 x 10-28 до ±7,9228 x 1028	16 байт/28-29 знаков
bool	true; false	16 разрядов
char	От U+0000 до U+FFFF	16 разрядов

Рис. 11. Основные типы данных в C#

Все представленные на рисунке типы данных – типы значений, кроме этого, в языке C# есть ссылочные типы – object, string, delegate, dynamic. О ссылочных типах данных поговорим позже, в первых лабораторных работах могут понадобиться только простые типы данных, такие как – **«string», «int», «float»**. Ранее мы уже использовали тип данных **«string»**, когда создавали первое приложение, этот тип используется для хранения текстовой информации – строк.

Объявление типов переменных делается соответствующим служебным словом с последующим перечислением имен переменных:

int i, j, k, l, m; – перечисленные переменные будут целого типа и смогут принимать значения от –2147483648 до –2147483647, что соответствует 2^{32} различным значениям так как исходя из таблицы, представленной на рис. 11,

переменные типа **«int»** занимают в памяти 32 бита. Объявление переменных любых типов можно совместить с присваиванием. Присваивание значения переменным делается с помощью команды присваивания **«=»**. Значения для символьных переменных заключаются в одинарные кавычки, для строковых переменных в двойные кавычки: int i = 33; float f = 0.156F; double d = 0.5; char c = '5'; string s = "Привет!";

Обратите внимание, что для вещественных чисел в качестве разделителя используется точка, а не запятая – **«0.5»**, так же для использования числа типа «float» необходимо в конце указывать литер «F», иначе компилятором это число будет восприниматься как тип **double**. Большинство типов данных – это типы данных, реализующие работу с числами, поэтому должны быть реализованы и операции работы с ними. Основные арифметические и логические операции, используемые в **C#**, приведены на рис. 12.

Арифметические операции	Обозначение	Логические операции	Обозначение
Сложение	+	Равно	= =
Вычитание	–	Не равно	!=
Умножение	*	Больше	>
Деление	/	Меньше	<
Вычисление остатка	%	Больше или равно	>=
Присваивание	=	Меньше или равно	<=
Отрицание	!	Логическое умножение	&&
		Логическое сложение	||

Рис. 12. Основные арифметические и логические операции в C#

Присваивание значений переменных, как уже было показано в примерах выше, делается командой присваивания «=»: int i = 5. В C# помимо привычных имеются и сокращенные формы записи арифметических операторов:

– «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».

Кроме того, существуют операции инкремента и декремента для более удобной записи увеличения или уменьшения переменных на 1, поскольку это часто встречаемая операция:

– «i++» эквивалентно «i=i+1»;

– «++i» эквивалентно «i=i+1»;

– «i--» эквивалентно «i=i-1»;

– «--i» эквивалентно «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.

Вызов функции	Описание
Math.Abs(x)	Возвращает абсолютное значение числа x.
Math.Acos(x)	Возвращает угол, косинус которого равен указанному числу.
Math.Asin(x)	Возвращает угол, синус которого равен указанному числу.
Math.Atan(x)	Возвращает угол, тангенс которого равен указанному числу.
Math.Ceiling(x)	Возвращает наименьшее целое число, которое больше или равно заданному числу.
Math.Cos(x)	Возвращает косинус указанного угла.
Math.Exp(x)	Возвращает e, возведенное в указанную степень.
Math.Floor(x)	Возвращает наибольшее целое число, которое меньше или равно заданному числу.
Math.Log(x)	Возвращает натуральный логарифм (с основанием e) указанного числа.
Math.Log10(x)	Возвращает логарифм с основанием 10 указанного числа.
Math.Max(x, y)	Возвращает большее из двух чисел.
Math.Min(x, y)	Возвращает меньшее из двух чисел.
Math.Pow(x, y)	Возвращает указанное число, возведенное в указанную степень.
Math.Round(x)	Округляет значение до ближайшего целого значения.
Math.Sin(x)	Возвращает синус указанного угла.
Math.Sqrt(x)	Возвращает квадратный корень из указанного числа.
Math.Tan(x)	Возвращает тангенс указанного угла.

Рис. 13. Набор основных математических функций реализованных в C#

Поскольку языки предшественники языка программирования C# разрабатывались, прежде всего, для решения научно-технических задач, обладая преемственностью здесь имеется целый набор математических функций. Основные функции необходимые для выполнения лабораторных работ приводятся на рис. 13, более широкий набор всегда можно найти в официальной документации [1].

Для примера решение функция $y = \sqrt{a^2 + b^3}$, на языке программирования C# можно представить следующим образом:

```
double y;
int a = 5, b = 6;
y = Math.Sqrt(Math.Pow(a, 2) + Math.Pow(b, 3));
```

Линейные программы

Программа на C# состоит из одной или более функций. Причем, начинается выполнение программы с функции Main(), которая, по сути, есть главный элемент программы, также ее называют – точкой входа. Реализация алгоритма – исполняемая часть программы заключается в составной оператор, который обозначается фигурными скобками.

Составной оператор «{...}» – действие состоит в последовательном выполнении содержащихся в нем операторов. Каждая фраза алгоритма заканчивается пустым оператором, который обозначается точкой с запятой «**;**». Выполнение пустого оператора не меняет состояния программы.

Обычно перед функцией Main задают так называемые директивы препроцессору. Такие директивы начинаются с ключевого слова using, например, директива «using System;» подключает пространство имен, которое содержит фундаментальные и базовые классы, определяющие часто используемые типы значений и ссылочных данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки.

Для вывода на экран служит команда – Console.Write ("Сообщение для вывода"); В ней для вывода текстовых сообщений – помещают их в кавычки. Также там можно разместить управляющие последовательности, встраиваемые в текст, для управления форматом, цветом и другими опциями вывода в текстовом терминале, например:

- «\n» – перевод на новую строку;
- «\t» – табуляция;
- «\a» – подача звукового сигнала;
- «\b» – возврат на один символ назад;

Для вывода сообщения и перевода курсора на новую строку можно использовать команду Console.WriteLine("Сообщение для вывода");

Для ввода значений переменных в программу в процессе ее исполнения служат команды: Console.Read() и Console.ReadLine(), первая возвращает код ASCII введенного символа, вторая возвращает введенную строку.

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

Изученных сведений достаточно для составления простой программы на C#. Программу, реализующую линейный алгоритм, исполняющуюся прямолинейно от начала и до конца – линейной программой. Рассмотрим простой пример реализующий алгоритм А и решение задачи 1.

Задача 1. Вычислить периметр и площадь прямоугольника, длина которого равна a, а ширина равна b.

Реализация алгоритма А на языке C# будет выглядеть следующим образом:

```
using System;
namespace Sample1
{
   class Program
   {
      static void Main(string[] args)
      {
         int a, b, S; // Объявление переменных a и b
         // Вывод сообщения о необходимости ввода переменной a
         Console.Write("Введите значение a = ");
         a = int.Parse(Console.ReadLine()); // Ввод переменной a
         Console.Write("Введите значение b = ");
         b = int.Parse(Console.ReadLine()); // Ввод переменной b
         S = a * b; // Вычисление S
         Console.WriteLine("S = {0}", S); // Вывод S
      }
   }
}
```

Разберем некоторые нюансы приведенного кода.

– Все используемые в программе переменные, необходимо обязательно объявить перед использованием;

– int.Parse(Console.ReadLine()) – преобразует введённые с клавиатуры значение в целочисленный тип;

– Console.WriteLine("S = {0}", S); – при выводе вместо {0} подставляется переменная S, этот приме называется форматирование строк, вместо него можно использовать интерполяцию строк, тогда эту строчку можно заменить на «Console.WriteLine($"S = {S}");».

Проведем «построение» решения (напомним, что для этого достаточно нажать горячие клавиши **Ctrl+Shift+B**) и, если нет ошибок, запускаем на выполнение (**Ctrl+F5**).

После запуска на исполнение получим результат как на рис. 14.

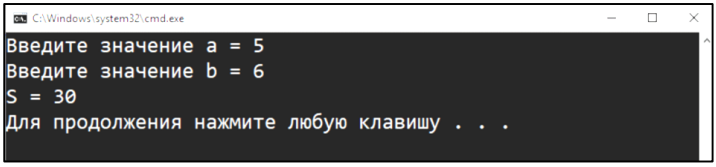


Рис. 14. Результат работы программы – задача 1

Программы с ветвлением

Разветвляющийся алгоритм – алгоритм, содержащий хотя бы одно условие, в результате проверки которого может осуществляться разделение на несколько альтернативных ветвей алгоритма.

Условный оператор на языке C# имеет вид:

```
if (условие)
      оператор_1;
    else
      оператор_2;
```

Обратите внимание, что 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. Кроме того, для реализации разветвляющихся алгоритмов в языке C# присутствует оператор выбора:

```
switch (a)
    {
          case 1:
             Console.WriteLine("Выбор 1");
             break;
          case 2:
             Console.WriteLine("Выбор 2");
             break;
          default:
             Console.WriteLine("Выбор по умолчанию");
             break;
 }
```

Рассмотрим два примера, где может понадобиться разветвление алгоритма. Составим программы, реализующие разработанные ранее алгоритмы, код программы задачи 2 с использованием уловного оператора представлен ниже:

```
using System;
namespace Sample2
{
   class Program
   {
      static void Main(string[] args)
      {
         double x, y;
         Console.Write("Введите значение x = ");
         x = double.Parse(Console.ReadLine());
         if (x <= 0)
            y = Math.Sin(x);
         else
            y = Math.Cos(x);
         Console.WriteLine($"y = {y}");
      }
   }
}
```

Результат работы программы представлен на рис. 15.

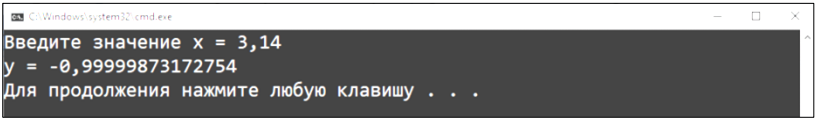


Рис. 15. Результат работы программы – задача 2

Исходный код программы реализующий алгоритм С:

```
using System;
namespace Sample3
{
            class Program
            {
      static void Main(string[] args)
      {
         double x, y;
         Console.Write("Введите значение x = ");
         x = double.Parse(Console.ReadLine());
         switch (x)
         {
            case 1:
               y = Math.Sin(x);
               break;
            case -1:
               y = Math.Cos(x);
               break;
            default:
               y = 0;
               break;
         }
         Console.WriteLine($"y = {y}");
       }
            }
}
```

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

```
if (условие)
{ оператор_1_1; оператор_1_2; }
    else
                        { оператор_2_1; оператор_2_2;}
```

Задача 6. Составить программу, которая перераспределит заданные значения x, y так, что в x окажется большее значение, а в y меньшее.

```
using System;
namespace Sample6
{
   class Program
   {
      static void Main(string[] args)
      {
         double x, y, z;
         Console.Write("Введите значение x = ");
         x = double.Parse(Console.ReadLine());
         Console.Write("Введите значение y = ");
         y = double.Parse(Console.ReadLine());
         if (x < y) { z = x; x = y; y = z; }
         Console.WriteLine($"x = {x}");
         Console.WriteLine($"y = {y}");
      }
   }
}
```

Обратите внимание, что здесь использована сокращенная форма записи условного оператора. 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 **«!»**). Результат работы программы, решающей пример 4 представлен на рис. 16.

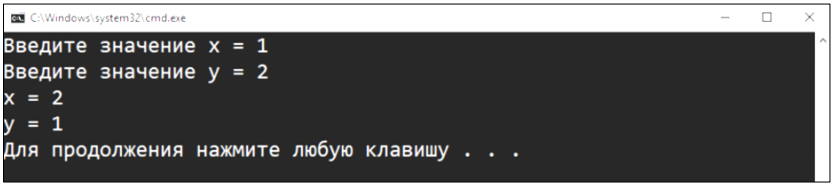


Рис. 16. Результат работы программы – задача 6

Программы с циклами

Цикл – управляющая конструкция, предназначенная для организации многократного выполнения какого-либо набора инструкций или команд. Циклы могут быть организованны как при помощи специальных операторов языка программирования, так и при помощи операторов сравнения и условного перехода.

Бесконечный цикл – цикл, в котором отсутствует условие выхода из цикла, либо условие выхода из цикла не достижимо в реальности.

Циклический алгоритм – алгоритм, предусматривающий многократное повторение одного и того же действия над новыми исходными данными.

В результате выполнения алгоритма D в переменной *p* будет находится значения факториала числа N, каждый раз после выполнения шага 4, мы увеличиваем значение переменной i и проверяем достигнул ли переменная i значения N. Такие переменные принято называть счетчиком цикла.

Счетчик цикла – переменная, контролирующая повторы выполнения циклов. Счетчики циклов изменяют своё значение при каждой итерации цикла, принимая уникальное значение для каждой отдельной итерации. Счетчик цикла используется для определения момента, когда цикл должен завершить свою работу.

Бывают ситуации, когда условиями окончания цикла нельзя представить в виде значения счетчика цикла. Тогда принято говорить о циклах с условием. Существует два вида таких циклов: с предусловием и постусловием, отличаются они тем в какой момент проверяется условие выполнения цикла. Для обозначения циклов с условием на языке блок-схем используется два вида обозначений для разных ситуаций, символы начала и конца цикла содержат имя и условие. Условие может отсутствовать в одном из символов пары. Расположение условия, определяет тип оператора, соответствующего символам на языке высокого уровня — оператор с предусловием или постусловием.

В языке программирования C# оператор for выполняет оператор или блок операторов, пока определенное логическое выражение равно значению true. Оператор for определяет разделы инициализатора, условия и итератора:

```
for (инициализатор; условие; итератор)
{
}
```

Например:

```
for (int i = 1; i < N; i++)
{
 p = p * i;
}
```

Таким образом, оператор for имплементирует цикл с параметром. В любой момент в блоке операторов for вы можете прервать цикл с помощью оператора break или перейти к следующей итерации в цикле с помощью оператора

continue. Также можно выйти из цикла for с помощью операторов goto, return или throw.

Листинг программы, реализующей алгоритм D с использованием цикла с параметром, представлен ниже:

```
using System;
namespace SampleD
{
   class Program
   {
      static void Main(string[] args)
      {
         int N, p = 1;
         Console.Write("Введите значение N = ");
         N = int.Parse(Console.ReadLine());
         for (int i = 1; i < N; i++)
         {
            p = p * i;
         }
         Console.WriteLine($"Факториал N = {p}");
      }
   }
}
```

Результат работы программы, реализующей алгоритм D представлен на рис. 17.

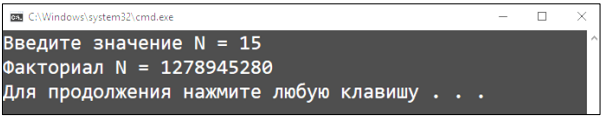


Рис. 17. Результат работы программы, реализующей алгоритм D

Оператор while выполняет оператор или блок операторов, пока определенное логическое выражение равно значению true. Так как это выражение оценивается перед каждым выполнением цикла, цикл while выполняется ноль или несколько раз. В любой точке блока операторов while можно разорвать цикл с помощью оператора break. Также можно выйти из цикла while с помощью операторов goto, return или throw. Оператор while имплементирует цикл с предусловием в языке программирования C#.

Пример использования оператора while:

```
while (n > 5)
    {
      Console.WriteLine(n);
      n--;
    }
```

Оператор do-while выполняет оператор или блок операторов, пока определенное логическое выражение равно значению true. Так как это выражение оценивается после каждого выполнения цикла, цикл do-while выполняется один

или несколько раз. Это отличает его от цикла while, который выполняется от нуля до нескольких раз. В любой точке блока операторов do можно разорвать цикл с помощью оператора break или с помощью операторов goto, return или throw.

Пример использования оператора do-while:

```
do
{
Console.WriteLine(n);
 n--;
} while (n > 5);
```

Внутри циклов можно организовывать и вложенные циклы, причем число вложений ничем не ограничено. Рассмотрим на примере алгоритм решения, которого приведен выше.

Задача 5. Получить таблицу умножения в виде таблицы Пифагора.

Листинг программы, реализующей алгоритм решения представлен ниже.

```
using System;
namespace Sample5
{
  class Program
  {
    static void Main(string[] args)
    {
      for (int i = 1; i <= 9; i++)
      {
        for (int j = 1; j <= 9; j++)
        {
          int k = i * j;
          if (k >= 10)
            Console.Write($" {k}");
          else
            Console.Write($"  {k}");
        }
        Console.WriteLine();
      } } }}
```

Обратите внимание при выводе чисел, проводится проверка двузначное число или нет и в зависимости от этого k выводиться либо с одним, либо с двумя предшествующими пробелами, но на блок-схеме это не отражено, поскольку это имеет незначительное значение для самого алгоритма, а относится непосредственно к стилизации вывода, нюансы, связанные с форматом вывода, можно опускать при описании алгоритмов.

Рассмотрим пример, когда актуально использовать циклы с условием.

Задача 7. Рассчитать сумму вида: $1-1/2+1/3-1/4+...$ с заданной точностью *E*. Иначе говоря, проводить суммирование, пока очередное слагаемое является больше, чем *E*.

Блок-схема алгоритма решения задачи 7 с использованием цикла с предусловием представлена на рис. 18.

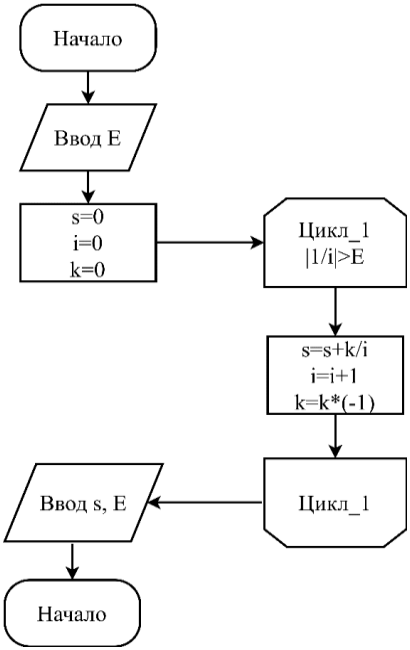


Рис. 18. Блок-схема алгоритма решения задачи 7

Листинг программы решения задачи 7 представлен ниже:

```
using System;
namespace Sample7
{
   class Program
   {
      static void Main(string[] args)
      {
         int i = 1, k = 1;
         double s = 0, E;
         Console.Write("Введите точность E = ");
         E = double.Parse(Console.ReadLine());
         while (1.0 / i > E)
```

```
        {
          s = s + (double)k / i;
          i++;
          k = k * (-1);
        }
        Console.WriteLine($"Сумма последовательности равна {s} с точностью {E}");
      }
    }
}
```

Результат работы программы задачи 7 представлен на рис. 19.

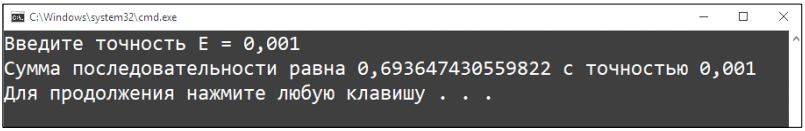


Рис. 19. Результат работы программы – задача 7

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. Рассмотрим пример, где это может быть необходимо.

Задача 8. Среди K первых членов последовательности вида: 1, 1+1/2, 1+1/2+1/3... найти первый, больший заданного числа A. Если же его нет (среди первых K членов), то вывести об этом сообщение. Результаты работы программы задача 8 представлен на рис. 20. Листинг программы решения задачи 8 представлен ниже:

```
using System;
namespace Sample8
{
  class Program
  {
    static void Main(string[] args)
    {
      int i = 1, k; double s = 0, A;
      Console.Write("Введите количество k = ");
      k = int.Parse(Console.ReadLine());
      Console.Write("Введите A = ");
      A = double.Parse(Console.ReadLine());
      while (i <= k)
      {
        s = s + 1.0 / i;
        if (s > A) break;
```

```
            i++;
        }
        if (i > k)
            Console.WriteLine("Такого числа нет");
        else
            Console.WriteLine($"При i = {i} первое число которое больше A это {s} ");
    }
  }
}
```

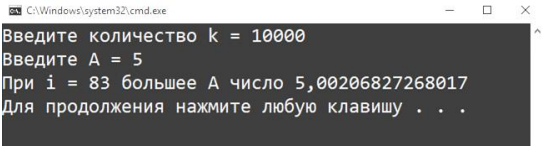


Рис. 20. Результат работы программы – пример 8

Контрольные вопросы

1. Что такое цикл?
2. Какие разновидности циклов бывают?
3. В каких ситуациях используются те или иные разновидности циклов?
4. Как циклы обозначаются на блок-схемах?
5. Какие операторы используются для пропуска итераций и выхода из цикла?

Массивы

Массив – это упорядоченный набор величин, принадлежащих одному типу данных, обозначаемых одним именем. Данные, являющиеся элементами массива, располагаются в памяти компьютера в определенном порядке, который задается индексами (порядковыми номерами элементов массива).

Размерность массива — это количество индексов, необходимое для однозначной адресации элемента в рамках массива. Форма или структура массива — сведения о количестве размерностей и размере массива для каждой из размерностей. Нуль-мерный массив называется скаляром, одномерный – вектором, двумерный – матрицей.

В C# массив, как и любая переменная, должен быть объявлен. Делается это с помощью служебного слова, указывающего тип с квадратными скобками, затем указывается имя массива, ставится знак равенства и ключевые слова new, далее снова указывается тип и в квадратных скобках размерность массива.

```
int[] array1 = new int[5];
```

Кроме того, можно сразу задать и значение элементов массива, сделать это можно одним из следующих способов:

```
int[] array2 = { 1, 2, 3, 4, 5};
int[] array3 = new int[5] { 1, 2, 3, 4, 5 };
```

Заметим, что индексы массива ведут счет с нуля, поэтому запись вида: int[] array1 = new int[5]; означает, что резервируется память для 5 чисел типа int с именем `array1` и порядковыми номерами от 0 до 4. Рассмотрим небольшой пример.

Задача 9. Ввести одномерный массив из 5 целых чисел. Вывести четные по порядковому номеру элементы этого массива.

Результат работ программы задача 9 представлен на рис. 21. Листинг программы для решения данного примера приведен ниже.

```
using System;
namespace Sample9
{
  class Program
  {
    static void Main(string[] args)
    {
      int[] numbers = new int[5];
      for (int i = 0; i < 5; i++)
      {
        Console.Write($"Введите {i+1} элемент массива: ");
        numbers[i] = int.Parse(Console.ReadLine());
      }
```

```
        for (int i = 1; i < 5; i+=2)
        {
            Console.WriteLine($"{i+1} элемент массива равен: {numbers[i]}");
        }
    }
  }
}
```

Обратите внимание при выводе индексов массива мы указывали i+1, т.к. в памяти нумерация элементов массива начинается с 0!

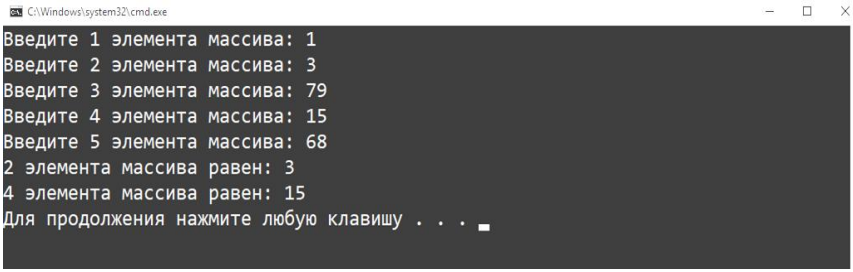


Рис. 21. Результат работы программы – задача 9

При работе с массивами необходимо внимательно следить за тем, чтобы не выходить за их объявленные границы. Компилятор не предупреждает об этой ошибке! Попытка ввести больше элементов, чем описано, приведет к неверным результатам и необработанному исключению – «Необработанное исключение: System.IndexOutOfRangeException: Индекс находился вне границ массива.», а попытка вывести – выведет случайный результат, находящийся в памяти.

Попробуйте в предыдущем примере описать массив numbers[5] (напомним, что при этом максимальный элемент `numbers[4]`, т.к. счет элементам идет с нуля), и вы не получите последнего числа (точнее получите какое-то случайное число при выводе).

Еще раз обратим внимание, что работа с целыми переменными требует осторожности. Деление целого на целое дает в результате целое! Если требуется получать «правильный», вещественный, результат следует использовать преобразование. Если же требуется получать остаток от деления целого на целое, то применяют операцию %. Проиллюстрируем использование операции деления и вычисления остатка на простом примере.

Массивы могут быть и многомерными, например, двумерными. В этом случае размерности записываются через запятую:

int[,] numbers = new int[5, 4]; – целочисленный массив из 5 строк и 4 столбцов.

Техническая документация

Техническое задание (ТЗ) – это исходный документ на проектирование какого-либо технического изделия, в контексте данного пособия – программного обеспечения (ПО). ТЗ устанавливает основное назначение разрабатываемого ПО, его технические характеристики, показатели качества и технико-экономические требования, предписание по выполнению необходимых стадий создания документации и её состав, порядок контроля и приемки, а также специальные требования. Существуют различные стандарты для написания ТЗ, ниже приведен список наиболее популярных стандартов:

1. ГОСТ 34 [7].
2. ГОСТ 19 [8].
3. IEEE STD 830-1998 [11].
4. ISO/IEC/ IEEE 29148-2018 [12].

ГОСТ 34 регламентирует структуру ТЗ на создание системы, в которую входят ПО, аппаратное обеспечение, персонал, которые работают с ПО, и автоматизируемые процессы. Согласно ГОСТ 34 техническое задание должно включать следующие разделы, которые могут быть разделены подразделы:

1. Общие сведения.
2. Назначение и цели создания.
3. Характеристики объектов автоматизации.
4. Требования к системе.
5. Состав и содержание работ по созданию системы.
6. Порядок контроля и приемки системы.
7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие.
8. Требования к документированию.
9. Источники разработки.

В ТЗ могут включаться приложения. Кроме того, в зависимости от вида, назначения, специфических особенностей объекта автоматизации и условий функционирования системы допускается оформлять разделы ТЗ в виде приложений, вводить дополнительные, исключать или объединять подразделы ТЗ. В ТЗ на части системы не включают разделы, дублирующие содержание разделов ТЗ на автоматизированную систему в целом. С полным списком подразделов можно ознакомится в документе [7]. При разработке ТЗ для государственных проектов Заказчики, как правило, требуют соблюдение именно этого стандарта. Пример технического задания на основе ГОСТ 34 представлен в приложении В.

ГОСТ 19.ххх. Единая система программной документации (ЕСПД) – это комплекс государственных стандартов, устанавливающих взаимоувязанные правила разработки, оформления и обращения программного обеспечения и

программной документации. Этот стандарт относится к разработке именно программного обеспечения.

Согласно ГОСТ 19.201-78. Техническое задание, требования к содержанию и оформлению, техническое задание должно включать следующие разделы:

1. Введение.
2. Основания для разработки.
3. Назначение разработки.
4. Требования к программе или программному изделию.
5. Требования к программной документации.
6. Технико-экономические показатели.
7. Стадии и этапы разработки.
8. Порядок контроля и приемки.
9. Приложения.

IEEE STD 830-1998 – это методика, которая имеет своей целью установление требований к разрабатываемому программному обеспечению, но также может применяться, чтобы помочь в выборе собственных и коммерческих программных изделий. Согласно стандарту, техническое задание должно включать следующие основные разделы:

1. Введение.
2. Общее описание.
3. Детальные требования.
4. Функциональные требования;
5. Требования к производительности.
6. Проектные ограничения (и ссылки на стандарты).
7. Нефункциональные требования (надежность, доступность, безопасность и др.).
8. Другие требования.
9. Приложения.
10. Алфавитный указатель.

Стандарт IEEE 29148-2018 обеспечивает единую трактовку процессов и продуктов, используемых при разработке требований на протяжении всего жизненного цикла систем и программного обеспечения.

В ходе выполнения учебной практики необходимо согласовать с преподавателем один из вариантов программного обеспечения, а именно формальные требования к ПО, и используя их как требования заказчика разработка ТЗ в соответствии с ГОСТ 34 или ГОСТ 19, по согласованию с преподавателем ТЗ может быть оформлено в соответствии с одним из IEEE стандартов.

Перед разработкой проекта технического задания необходимо выполнить предпроектное исследование, которое заключается в изучении требова-

ний заказчика, ознакомление с предметной областью по тематике разрабатываемого ПО, обзор и обоснование выбранных технологий для реализации. В качестве требований заказчика следует рассматривать задание на курсовую работу согласно варианту согласованным с преподавателем из учебно-методического пособия по курсовой работе [9], с которым необходимо полностью ознакомится перед выполнением этого этапа практики.

Общие требования к содержанию отчета по учебной практике

Выполнение работ по учебной практике требует разработки 6 алгоритмов к задачам, приведенных в пособии и, как результат, создание программ на языке программирования C# в среде разработки Microsoft Visual Studio 2019, а также создания проектной документации на разработку программного обеспечения в рамках курсовой работы по дисциплине «Основы программирования» на основе одного из стандартов. Порядок и объем работ, а также требования к содержанию представлены ниже.

На первом занятии необходимо согласовать вариант работ с преподавателем. Пример оформления титульного листа отчета представлен в приложении А. Пример заявления на прохождение учебной практики представлен в приложении Б. Пример оформления дневника студента представлен в приложении Г. Отчет по учебной практике должен содержать следующие разделы:

– титульный лист;
– реферат на русском и английском языках;
– задание на работу;
– содержание;
– разработка алгоритмов;
– реализация алгоритмов на языке программирования C#;
– предпроектное исследование для создания проект технического задания;
– заключение;
– список использованных источников;
– приложение А – обязательное (исходные коды программы);
– приложение Б – обязательное (проект технического задания);

Содержательная часть разделов пояснительной записки рассматривается в разделах данного пособия. Основные требования, предъявляемые к работе в рамках учебной практики:

– словесное описание алгоритмов в соответствии, с правилами, описанными в первом разделе данного пособия;
– разработка блок-схем алгоритмов в соответствии с ГОСТ 19.701-90.
– реализация алгоритмов на языке программирования C# и их дальнейшая доработка при необходимости;
– разработка технического задания на создание программного обеспечения на основе ГОСТ 34 или ГОСТ 19;

Особых требований к сложности разрабатываемых алгоритмов не предъявляется. Перед тем как приступить к выполнению учебной практики необходимо в полном объеме ознакомиться с данным пособием. В ходе выполнения учебной практики рекомендуется ознакомится со списком литературы, который прилагается к данному пособию. Ознакомление с ГОСТ 34, ГОСТ 19 и стандартами IEEE является обязательной частью прохождения учебной практики. Пояснительная записка должна быть оформлена в соответствии ОС ТУСУР 01–2013 или ГОСТ 7.32–2017.

Задания на разработку алгоритмов

Линейные программы

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.

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. 1-3. Дана длина ребра куба a. Найти объем куба V и площадь его боковой поверхности S. Пример: a=5 Ответ: V=125, S=100.

4. Каков объем кислорода, содержащегося в комнате размером a*b*c, если кислород составляет 21% объема воздуха? Пример: a=3, b=4, c=5. Ответ: 12,6.

5. Найти площадь равнобочной трапеции с основаниями a и b и углом при большем основании, равным x. Пример: a=6, b=5, x=45o. Ответ: 2,75.

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°.

7. Определить время падения камня на поверхность земли с высоты h. Пример: h=10 м. Ответ: 1,4278 с.

8. Три сопротивления R1, R2, R3 соединены параллельно. Найти сопротивление соединения. Пример: R1=10, R2=15, R3=20. Ответ: 4,62.

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.

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.

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.

12. Написать программу вычисления объема цилиндра. Извне вводятся радиус основания R и высота цилиндра h. Пример: R=10, h=15. Ответ: 4712,39.

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.

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.

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.

16. Написать программу вычисления сопротивления электрической цепи, состоящей из двух параллельно соединенных сопротивлений. Извне вводятся величина первого и второго сопротивления. Пример: R1=10, R2=15. Ответ: 6.

17. Написать программу вычисления сопротивления электрической цепи, состоящей из двух последовательно соединенных сопротивлений. Извне вводятся величина первого и второго сопротивления. Пример: R1=10, R2=15. Ответ: 25.

18. Написать программу вычисления силы тока в электрической цепи. Извне вводятся напряжение U и сопротивление R. Пример: U=10, R=15. Ответ: 0,6667.

19. Составить программу, которая поменяет местами значения введенных переменных x и y не используя дополнительной переменной.

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 не используя дополнительной переменной.

Программа с ветвлением

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

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

3. Даны числа x, y (x не равно y). Меньшее из них заменить полусуммой, а большее – их удвоенным произведением.

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

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

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.

7. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, а.

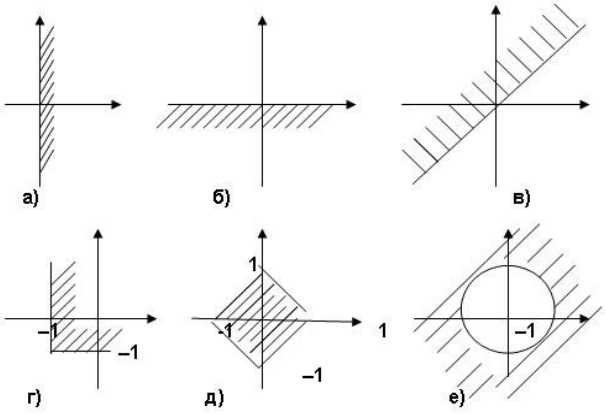
8. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, б.

9. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, в.

10. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, г.

11. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, д.

12. Даны координаты точки (x, y). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, е.



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.

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?

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

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).

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)

18. Написать программу вычисления сопротивления электрической цепи, состоящей из двух сопротивлений. Извне вводятся величина первого, второго сопротивления и указывается тип соединения (например, 1 – последовательное, 2 – параллельное соединение).

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

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).

Работа с одномерными массивами

1. Определить, содержит ли массив данное число x
2. Найти количество четных чисел в массиве.
3. Найти количество чисел в массиве, которые делятся на 3, но не делятся на 7.
4. Определите, каких чисел в массиве больше: которые делятся на первый элемент массива или которые делятся на последний элемент массива.
5. Найдите сумму и произведение элементов массива.
6. Найдите сумму четных чисел массива.
7. Найдите сумму нечетных чисел массива, которые не превосходят 11.
8. Найдите сумму чисел массива, которые расположены до первого четного числа массива. Если четных чисел в массиве нет, то найти сумму всех чисел за исключением крайних.
9. Найдите сумму чисел массива, которые стоят на четных местах.
10. Найдите сумму чисел массива, которые стоят на нечетных местах и при этом превосходят сумму крайних элементов массива.
11. Найти наибольший элемент массива.
12. Найдите сумму наибольшего и наименьшего элементов массива.
13. Найдите количество элементов массива, которые отличны от наибольшего элемента не более чем на 10%.
14. Найдите наименьший четный элемент массива.
15. Среди элементов с нечетными номерами найдите наибольший элемент массива, который делится на 3.
16. Дан массив и число p. Найдите два различных числа в массиве, сумма которых наиболее близка к p.
17. Дан массив. Найдите два соседних элемента, сумма которых минимальна.
18. Дан массив. Найдите три последовательных элемента в массиве, сумма которых максимальна.
19. В данном массиве найдите количество чисел, соседи у которых отличаются более чем в 2 раза.
20. Найдите количество чисел, каждое из которых равно сумме квадратов своих соседей и при этом не является наибольшим в массиве.

Циклы с параметром

1. Найти сумму четных цифр числа, введенного с клавиатуры.
2. Составьте таблицу значений функции $y = x^2+3x-17$, диапазон значений x вводится с клавиатуры.
3. Посчитать четные и нечетные цифры числа, введенного с клавиатуры.
4. Вывести число-перевертыш числа, введенного с клавиатуры (без использования строкового типа).
5. Вывести кубы чисел от A до B, введенных с клавиатуры.
6. Вывести факториал числа, введенного с клавиатуры.
7. Определить количество простых чисел в диапазоне от 1 до N, N вводится с клавиатуры.
8. Вывести столбиком цифры числа, введенного с клавиатуры.
9. Вывести сумму цифр числа, введенного с клавиатуры.
10. Вывести произведение цифр числа, введенного с клавиатуры.
11. Вывести квадраты натуральных чисел от 1 до N, N вводится с клавиатуры.
12. Вывести последовательность первых N чисел Фибоначчи, N вводится с клавиатуры.
13. Возвести в степень N число X, N и X вводятся с клавиатуры.
14. Вывести таблицу умножения на экран.
15. Удалить цифру X из числа N, N и X вводятся с клавиатуры.
16. Вывести сумму первой и последней цифр числа, введенного с клавиатуры.
17. Определить, наибольшую цифру числа, введенного с клавиатуры.
18. Найти число под номером N в ряду чисел, введенных с клавиатуры.
19. Определить количество разрядов числа и вывести на экран в формате $\sum_{i=0}^{n-1} a_i b_i$, например 325=3*10^2+2*10^1+1*10^0.
20. Найти одинаковые цифры двух чисел, введенных с клавиатуры.

Циклы с условием

1. Дана непустая последовательность целых чисел, оканчивающаяся нулем. Найти сумму всех чисел последовательности; Последовательность вводится с клавиатуры.
2. Дана непустая последовательность целых чисел, оканчивающаяся нулем. Найти количество всех чисел последовательности. Последовательность вводится с клавиатуры.
3. Дана непустая последовательность неотрицательных целых чисел, оканчивающаяся отрицательным числом. Найти среднее арифметическое всех

чисел последовательности (без учета отрицательного числа). Последовательность вводится с клавиатуры.

4. Дана последовательность из n вещественных чисел. Первое число в последовательности нечетное. Найти сумму всех идущих подряд в начале последовательности нечетных чисел. Последовательность вводится с клавиатуры.

5. Дана последовательность из n вещественных чисел, начинающаяся с отрицательного числа. Определить, какое количество отрицательных чисел записано в начале последовательности. Последовательность вводится с клавиатуры.

6. Дана последовательность целых чисел a1, a2, ..., a18, в начале которой записано несколько равных между собой элементов. Определить количество таких элементов последовательности. Последовательность вводится с клавиатуры.

7. Дано число n. Из чисел 1, 4, 9, 16, 25, ... напечатать те, которые не превышают n. n – вводится с клавиатуры.

8. Среди чисел 1, 4, 9, 16, 25, ... найти первое число, большее n. n – вводится с клавиатуры.

9. Дано число n. Напечатать те натуральные числа, квадрат которых не превышает n. n – вводится с клавиатуры.

10. Дано число n. Найти первое натуральное число, квадрат которого больше n. n – вводится с клавиатуры.

11. Дано натуральное число n. Определить: количество цифр 3 в нем. n – вводится с клавиатуры.

12. Дано натуральное число n. Определить: сколько раз в нем встречается последняя цифра; n – вводится с клавиатуры.

13. Дано натуральное число n. Определить: количество четных цифр в нем. n – вводится с клавиатуры.

14. Дано натуральное число n. Определить: сумму его цифр, больших пяти; n – вводится с клавиатуры.

15. Дано натуральное число n. Определить: произведение его цифр, больших семи. n – вводится с клавиатуры.

16. Дано натуральное число n. Определить: сколько раз в нем встречаются цифры 0 и 5 (всего). n – вводится с клавиатуры.

17. Дано натуральное число n. Определить: сколько раз в нем встречается цифра a; n – вводится с клавиатуры.

18. Дано натуральное число n. Определить: сколько раз в нем встречаются цифры x и y. n – вводится с клавиатуры.

19. Дана непустая последовательность целых чисел, оканчивающаяся нулем. Найти: сумму всех чисел последовательности, больших числа x. Последовательность вводится с клавиатуры.

20. Дано натуральное число. Определить его максимальную и минимальную цифры. Число вводится с клавиатуры.

Циклы с параметром и условием

1. Вывести на экран все целые числа от 100 до 200, кратные трем.
2. Вывести на экран все целые числа от a до b, кратные некоторому числу c.
3. Найти сумму положительных нечетных чисел, меньших 50.
4. Найти сумму целых положительных чисел из промежутка от a до b, кратных четырем.
5. Составить программу поиска трехзначных чисел, которые при делении на 47 дают в остатке 43, а при делении на 43 дают в остатке 47.
6. Составить программу поиска четырехзначных чисел, которые при делении на 133 дают в остатке 125, а при делении на 134 дают в остатке 111.
7. Определить количество трехзначных натуральных чисел, сумма цифр которых равна целому числу n (0 <n <27). Найти: все двузначные числа, сумма квадратов цифр которых делится на 13;
8. Определить количество трехзначных натуральных чисел, сумма цифр которых равна целому числу n (0 <n <27). Найти: все двузначные числа, обладающие следующим свойством: если к сумме цифр числа прибавить квадрат этой суммы, то получится снова искомое число.
9. Найти все двузначные числа, которые делятся на n или содержат цифру n.
10. Найти: все трехзначные числа, чьи квадраты оканчиваются тремя цифрами, которые и составляют искомые числа;
11. Найти: все трехзначные числа, кратные семи и у которых сумма цифр также кратна семи.
12. Найти сумму целых положительных чисел, больших 30 и меньших 100, кратных трем и оканчивающихся на 2, 4 и 8.
13. Дано натуральное число. Получить все его делители.
14. Дано натуральное число. Получить все его делители.
15. Дано натуральное число. Найти сумму его делителей.
16. Дано натуральное число. Найти сумму его четных делителей.
17. Дано натуральное число. Определить количество его делителей
18. Дано натуральное число. Определить количество его нечетных делителей.
19. Дано натуральное число. Найти количество его делителей, больших d.
20. Дано натуральное число. Определить номер цифры 3 в нем, считая от конца числа. Если такой цифры нет, ответом должно быть число 0, если таких цифр в числе несколько — должен быть определен номер самой правой из них.

ЛИТЕРАТУРА

1. Кнут Д.Э. Искусство программирования. – Т. 1: Основные алгоритмы. – 3-е изд.; пер. с англ. – М.: ИД «Вильямс», 2016. – 720 с.
2. Колмогоров А.Н. Теория информации и теория алгоритмов. – М.: Наука, 1987. – 380 с.
3. Мелехин В.Ф., Павловский Е.Г. Вычислительные машины. – М.: ИЦ «Академия», 2013. – 384 с.
4. ГОСТ 19.701-90 (ИСО 5807-85). Единая система программной документации (ЕСПД). Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.
5. Технологии разработки программного обеспечения. Разработка сложных программных систем: учеб. пособие для вузов. – СПб.: Питер, 2002. – 464 с.
6. Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. Проектирование информационных систем. Курс лекций: учеб. пособие для вузов. – М.: Интернет-университет информационных технологий, 2005. – 298 с.
7. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
8. ГОСТ 19.201-78. Техническое задание, требования к содержанию и оформлению.
9. Харченко С.С. Основы программирования: учебно-методическое пособие по курсовой работе. – Томск: В-Спектр, 2019. – 48 с.
10. Документация по .Net | Microsoft Docs. – URL: https://docs.microsoft.com/ru-ru/dotnet/
11. IEEE Recommended Practice for Software Requirements Specifications // IEEE Std 830-1998. – 20 Oct. 1998. – PP. 1–40. doi: 10.1109/IEEESTD.1998.88286.
12. ISO/IEC/IEEE International Standard – Systems and software engineering – Life cycle processes – Requirements engineering // ISO/IEC/IEEE 29148:2018(E), 30 Nov. 2018. – PP. 1–04. doi: 10.1109/IEEESTD.2018.8559686.

ПРИЛОЖЕНИЕ А

Форма титульного листа отчета по учебной практике

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра безопасности информационных систем (БИС)

Разработка алгоритмов и программ на языке программирования C#

ОТЧЕТ

ПО РЕЗУЛЬТАТАМ
Учебной практики: Учебно-лабораторный практикум

Обучающийся гр. _____

_____ _____
(подпись) (И.О. Фамилия)

8 февраля 2021 г
(дата)

Руководитель практики от Университета:
Доцент каф.БИС, канд. техн. н.

_____ _____ С.С. Харченко
оценка (подпись)

_____
(дата)

Томск 2021

ПРИЛОЖЕНИЕ Б

Форма заявления на учебную практику

Заведующему кафедрой КИБЭВС
А.А.Шелупанову
(ФИО зав. кафедрой)
от студента гр. 710-1

_____
(ФИО студента)

_____
(эл. почта)

_____
(телефон)

Заявление

Прошу направить меня для прохождения учебной практики (практика по получению первичных профессиональных умений и навыков)
в профильную организацию кафедра КИБЭВС факультета безопасности (адрес:г.Томск, ул.Красноармейская, 146) форма проведения - дискретно по периодам проведения практик:
с 08 февраля 2021 г. по 13 марта 2021 г.; с 22 марта 2021 г. по 24 апреля 2021 г.; с 03 мая 2021 г. по 05 июня 2021 г.; с 14 июня 2021 г. по 26 июня 2021 г.

Дата _____ Подпись _____

Согласовано:

Зав. кафедрой КИБЭВС _____ А.А.Шелупанов
(Ф.И.О.)

Руководитель практики от университета _____ С.С.Харченко
(Ф.И.О.)

ПРИЛОЖЕНИЕ В

Пример технического задания

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра безопасности информационных систем (БИС).

Решето Эратосфена
ТЕХНИЧЕСКОЕ ЗАДАНИЕ

На 12 листах

СОГЛАСОВАНО
доцент кафедры БИС,
канд. техн. наук
_____ Харченко С.С.
__.__.2021

РАЗРАБОТЧИК
Студент гр. 720-1
_____Иванов А.С.
__.__.2021

Томск 2021

1 Общие сведения

1.1 Полное наименования системы и ее условное обозначение

Полное наименование системы: «Десктопное приложение «Решето Эратосфена».

1.2 Заказчик

Заказчиком является Томский государственный университет систем управления и радиоэлектроники, кафедра безопасности информационных систем (БИС).

1.3 Исполнитель

Исполнителем является студент группы 720-1 Иванов Андрей Сергеевич.

1.4 Основания разработки

Основанием для разработки является задание на выполнение курсовой работы по дисциплине “Основы программирования” для студентов направления подготовки 10.03.05 - Информационная безопасность автоматизированных систем, профиля «Информационная безопасность автоматизированных банковских систем».

2 Назначение и цель создания системы

2.1 Назначение системы

Система предназначена для генерации списка простых чисел в диапазоне от 1 до введенного N на основе алгоритма – решето Эратосфена.

2.2 Цели создания системы

Целью разработки является автоматизация процесса проверка списка чисел на простоту.

3 Характеристика объектов автоматизации

3.1 Объект автоматизации

Объектом автоматизации является проверка списка чисел на простоту.

4 Требования к системе

4.1 Требования к структуре и функционированию

Приложение должно выполнять следующие функции:

– Возможность для авторизации и регистрации.

– Генерация списка простых чисел.

– Генерация «решета Эратосфена».

– Сохранение сгенерированного «решета» в виде изображения в файл.

4.2 Перечень подсистемы, их назначение и основные характеристики

В системе предлагается выделить следующие функциональные подсистемы:

– Подсистема графического интерфейса, для более удобного взаимодействия с приложением;

– Подсистема авторизации;

– Подсистема регистрации;

– Подсистема взаимодействия с базой данных.

4.3 Требования к надежности

При возникновении сбоев в аппаратном обеспечении, включая разряд аккумулятора устройства, информационная система восстанавливает свою работоспособность после устранения сбоев и корректного перезапуска аппаратного обеспечения (за исключением случаев повреждения рабочих носителей информации с исполняемым программным кодом).

4.4 Требования по безопасности

Все технические решения, использованные при создании системы, а также при определении требований к аппаратному обеспечению, соответствуют действующим нормам и правилам техники безопасности,

пожарной безопасности, а также охраны окружающей среды при эксплуатации.

4.5 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению

Для эксплуатации разрабатываемой информационной системы необходимы следующие условия:

- Компьютер под управлением операционной системы Windows 10;
- Предустановленный .Net Framework v 4.5;
- Питание компьютера от сети или батареи;
- Предустановленная СУБД PostgreSql v 10.0;
- Наличие таких периферийных устройств, как мышь и клавиатура, для взаимодействия.

4.6 Требования к защите информации от несанкционированного доступа

Доступ к работе с интерфейсом системы имеют только авторизованные пользователи.

4.7 Требования к функциям разработчика

Роль разработчика заключается в обновлении и пополнении системы новыми функциями, а также исправление возможных ошибок в функционировании системы.

4.8 Требования к функциям пользователя

Пользователь может использовать все функции, которыми обладает система.

4.9 Описание процессов и функций работы с системой

Процессы и функции, выполняемые при эксплуатации системы, приведены в разбивке по подсистемам: подсистема графического интерфейса,

для более удобного взаимодействия с приложением, подсистема авторизации, подсистема регистрации, подсистема взаимодействия с базой данных. Процессы, реализованные под управлением различных подсистем, реализуются на основе системных процедур, которые являются составной частью функций системы. Системные процедуры группируются в соответствии с их назначением:

- Графический интерфейс пользователя;
- Авторизация/Регистрация пользователей.

4.10 Требования к информационному обеспечению системы

Компоненты системы должны активно взаимодействовать с системой управления базой данных (СУБД). Обмен информацией с СУБД должен происходить автоматически. Уровень хранения данных в системе должен быть построен на основе современных реляционных или объектно-реляционных СУБД. Доступ к данным должен быть предоставлен только авторизованным пользователям.

4.11 Требования к программному обеспечению

- ОС Windows 10;
- СУБД PostgreSql 10;
- Язык программирования C;
- .Net Framewrok 4.5;
- Установлено ПО;

5 Состав и содержание работ по созданию системы

Состав и содержание работ по созданию системы приведены в таблице в таблице 5.1.

Таблица 5.1 – Этапы разработки

№	Этап	Результат	Срок выполнения
1	Проектирование системы	UML диаграмма классов	09.11.2021
2	Разработка ПО	Приложение, репозиторий на github	16.11.2021
3	Создание БД и подключение к СУБД	SQL запросы инициализации и миграции БД	23.11.2021
4	Тестирование и отладка приложения	Набор автотестов, отчет о тестировании	07.12.2021
5	Защита курсовой работы	Пояснительная записка	21.12.2021

6 Порядок контроля и приемки системы

6.1 Перечень этапов испытаний и проверок

Этапы испытаний подразделяются на предварительные и приемочные. Предварительные испытания проводятся на стадии тестирования разработчиком. Приемочные испытания проводятся во время сдачи проекта разработчиком совместно с заказчиком. Все подсистемы испытываются одновременно на корректность взаимодействия подсистем, влияние подсистем друг на друга, то есть испытания проводятся комплексно.

Во время приемочных испытаний оценивается:

– Полнота и качество реализации функций, указанных в настоящем техническом задании;

– Демонстрация объектно-ориентированного подхода при реализации функций, указанных в настоящем техническом задании;

– Выполнение каждого требования, относящегося к интерфейсу системы;

– Полнота действий, доступных пользователю:

1. Регистрация и авторизация пользователей;
2. Генерация простых чисел;
3. Создание и сохранение на внешний носитель изображения сгенерированного «Решета»;
4. Выход авторизированного пользователя из приложения.

При проверке регистрации производится несколько попыток регистрации пользователя.

При авторизации пробуется авторизация зарегистрированного пользователя и не зарегистрированного пользователя.

При проверке корректного выхода авторизированного пользователя из приложения при нажатии на выход из приложения должна открыться главная страница приложения, и, чтобы продолжить работу с приложением, нужно

заново авторизоваться. Также просматриваются коды графического интерфейса, коды выполнения запросов и просматривается база данных.

Приемка результатов должна осуществляться в сроки, установленные таблицей 5.1. Результаты проектирования системы предоставляются в электронном виде с помощью ЭИОС sdo.tusur.ru. Результаты разработки программного обеспечения будут располагаться на репозитории github. Результаты тестирования будут представлены в электронном виде с помощью ЭИОС sdo.tusur.ru.

6.2 Общие требование к приемке работы

Приемка осуществляется представителями Заказчика и Исполнителя. Все создаваемые в рамках настоящей работы программные изделия передаются Заказчику, как в виде готовых модулей, так и в виде исходных кодов, представляемых в электронной форме на репозитории github.

7 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

Для обеспечения готовности объекта к вводу системы в действие провести комплекс мероприятий:

- Загрузка файлов приложения;
- Проведение предварительных испытаний;
- Проверка приемочных испытаний.

8 Требования к документированию

Состав программной документации:

- Задание на курсовую работу;
- Техническое задание (ТЗ);
- Пояснительная записка (ПЗ);
- Документация к системе в электронном виде.

Документация должна быть оформлена с использованием:

- ГОСТ 34.602-89;
- ОС ТУСУР 01-2013 для технического задания;
- ОС ТУСУР 01-2013 для пояснительной записки.

ПРИЛОЖЕНИЕ Г

Пример заполнения дневника студента

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

(ТУСУР)

ДНЕВНИК СТУДЕНТА

по учебной практике

(Практика по получению первичных профессиональных умений и навыков)

Краткая инструкция

1. Перед отъездом на практику каждый обучающийся получает на кафедре дневник по практике.

2. В процессе практики обучающийся регулярно ведет запись о проделанной работе.

3. Раздел 4 заполняется работником профильной организации, ответственным за проведение инструктажа по ОТ и ТБ, заверяется его подписью и печатью организации. Раздел 5 заполняются руководителем практики от профильной организации.

4. Заполнение всех разделов является обязательным.

5. В течение двух недель с начала занятий в последующем семестре обучающийся сдает отчет и дневник на кафедру.

С инструкцией ознакомлен

Подпись обучающегося_____

«8» февраля 2021г.

ДЛЯ ЗАМЕТОК

1. Общие сведения

1. Фамилия, имя, отчество студента.....
Иванов Иван Иванович..... .....
2. Факультет.....:
безопасности (ФБ).
3. Курс 1..... 4. Группа №710-1 ....
5. Место практики ТУСУР, ФБ, КИБЭВС.....
6. Срок практики 8.02.2021-13.03.2021, 22.03.21-24.04.2021, 03.05.21-05.06.21,14.06.21-26.06.21

График прохождения практики:

Цех и рабочее место, лекции, экскурсии, экзамены, отчет	Недели														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ТУСУР, ФБ	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
408 УЛК															

Составил:
Руководитель практики от Университета _____ (Подпись) Харченко С.С. (Ф.И.О.)

2. Индивидуальное задание

а) тема практики Разработка алгоритмов и программ на языке программирования C#.

б) цель практики Получение навыков разработки алгоритмов решения практических задач, их реализация на языке программирования C#, с использованием интегрированной среды разработки Visual Studio

в) задачи практики

1) Пройти инструктаж;
2) Изучить методические указания к практике;
3) Повторить пройденный материал по дисциплине Информатика;
4) Составить алгоритмы решения задач 1-6 и разработать блок-схемы алгоритмов;
5) Реализовать разработанные алгоритмы на языке Программирования C#;
6) Разработать техническое задание на разработку программного обеспечения в соответствии со стандартом;
7) Написать отчет согласно действующего ОС ТУСУР.

6. Заключение руководителя практики от Университета

Оценка за учебную практику
(Учебно-лабораторный практикум): .....

Руководитель
практики от Университета _____Харченко С.С.
(Подпись) (Ф.И.О.)

«____» _____ 2021 г.

5. Оценка работы обучающегося

(заполняется руководителем практики от профильной организации)

а) Заключение о работе обучающегося в период практики (технические навыки, активность, дисциплина, участие в производственных мероприятиях, помощь производству).

б) поощрения и взыскания (по приказам)

Оценка за практику: .....

Руководитель практики от профильной организации _____ (Подпись) Харченко С.С. (Ф.И.О.)

3. Содержание работ практики

Дата	Подразделение, рабочее место	Краткое содержание проделанной работы	Подпись руководителя практики от профильной
08.02	ФБ, 408 УЛК	Ознакомление с заданием	
		по учебной практике.	
		Прохождение инструктажа	
22.02	ФБ, 402 УЛК	Разработка алгоритмов к задачам	
		1-2	
08.03	ФБ, 402 УЛК	Разработка алгоритмов к задачам	
		3-4	
22.03	ФБ, 402 УЛК	Разработка алгоритмов к задачам	
		5-6	
05.04	ФБ, 402 УЛК	Реализация программы по	
		алгоритмам 1-3	
19.04	ФБ, 402 УЛК	Реализация программы по	
		алгоритмам 4-6	
03.05	ФБ, 402 УЛК	Тестирование и отладка	
		разработанных программ	
17.05	ФБ, 402 УЛК	Тестирование и отладка	
		разработанных программ	
14.06	ФБ, 402 УЛК	Разработка технического задания	
		на курсовую работу	
14.06	ФБ, 402 УЛК	Разработка технического задания	
		на курсовую работу	
14.06	ФБ, 402 УЛК	Разработка технического задания	
		на курсовую работу	

Дата	Подразделение, рабочее место	Краткое содержание проделанной работы	Подпись руководителя практики от профильной
28.06	ФБ, 402 УЛК	Написание отчета	
		Написание отчета	
28.06	ФБ, 402 УЛК	Написание отчета	
		Написание отчета	
		Написание отчета	
28.06	ФБ, 402 УЛК	Написание отчета	

4. Отметка о прохождении инструктажа по охране труда и пожарной безопасности

Инструктаж проведен.

Ответственный за ОТ и ТБ_____ Харченко С.С.

(Подпись) (Ф.И.О.)

Учебное издание

Сергей Сергеевич Харченко

УЧЕБНАЯ ПРАКТИКА

Учебно-методическое пособие

для студентов специальностей и направлений
10.03.01 – «Информационная безопасность»,
10.05.03 – «Информационная безопасность автоматизированных систем»,
10.05.02 – «Информационная безопасность телекоммуникационных систем»,
10.05.04 – «Информационно-аналитические системы безопасности»

Верстка – В.М. Бочкаревой
Текст дан в авторской редакции, без корректуры

Издательство «В-Спектр»
Подписано к печати 10.04.2021.
Формат $60\times84^{1/16}$. Печать трафаретная.
Печ. л. 4. Тираж 100 экз. Заказ 12.

Тираж отпечатан ИП В.М. Бочкаревой
ИНН 701701817754
634055, г. Томск, пр. Академический 13-24
т. 8-905-089-92-40. Эл. почта: bvm@sibmail.com