МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГОУ ВО «Дагестанский государственный технический университет»

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ПРИКЛАДНОЙ ИНФОРМАТИКИ В ЭКОНОМИКЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторного практикума по дисциплине
«Алгоритмизация и программирование»
Часть 1.
«Программирование на языке C#»
для студентов направления подготовки бакалавров 09.03.03 – «Прикладная информатика».
Часть 1.

Махачкала – 2021 г.

УДК 681.3
Методические указания к выполнению лабораторного практикума по дисциплине «Алгоритмизация и программирования»: Махачкала, ДГТУ, 2021. 28 с.

Методические указания предназначены для студентов дневной и заочной форм обучения по направлению подготовки бакалавров «Прикладная информатика.

Методические указания содержат краткие теоретические сведения о системе Microsoft Studio 2019, создание консольных программ на языке программирования СиШарп, методические примеры, индивидуальные задания к выполнению лабораторных работ.

Составители: доцент кафедры ИТиПИвЭ, к.э.н. Мурадов М.М.

Рецензенты: Джабраилов Х.С., директор ООО ИВЦ Сигма, к.э.н.
Мирземагомедова М.М., доцент кафедры ПМиИ ДГТУ, к.т.н.

Печатается по решению Совета Дагестанского государственного технического университета от «____» _____2021 г., № ___.

ВВЕДЕНИЕ

В настоящее время особое значение для всех отраслей экономики принимает автоматизация производственных процессов. Для разработки приложений масштаба предприятия необходимо владеть современными методами создания сложных программных продуктов, а также инструментарием разработки программных средств.

Особое значение при подготовке специалиста в сфере информационных систем и технологий имеет практическая подготовка специалиста: производственные задачи автоматизации связаны с многопоточным программированием, использованием баз данных, применением сложных структур данных.

Сложность освоения технологий программирования, в совокупности с постоянным ростом значимости информационных технологий, указывает на необходимость подготовки специалистов в области программирования.

Учебное методическое указание (лабораторный практикум) направлено на развитие следующих профессиональных компетенций:

- ОПК-2 - способность использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности;
- ОПК-3- способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности;
- ОПК-4- способность участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью;
- ОПК-5- способность инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем;
- ОПК-7- способность разрабатывать алгоритмы и программы, пригодные для практического применения.

Основная цель изучения дисциплины «Алгоритмизация и программирование»: изучение и получение практических навыков в области современных средств и методов разработки, алгоритмов, основных синтаксических конструкция языка высокого уровня.

Задачами изучения дисциплины «Алгоритмизация и программирование» являются:

- усвоение теоретических знаний о структуре и принципах построения современных программных комплексов, а также об архитектуре современных программных платформ для автоматизации проектирования

информационных систем;

– получение практических навыков разработки приложений с использованием языка высокого уровня C#;

– изучение основных алгоритмических приемов решения задач с использованием C#;

– освоение принципов разработки программного обеспечения на основе технологий Microsoft .NET Framework;

– освоение принципов использования перспективных инструментальных средств разработки и проектирования приложений.

Знания, полученные студентами в результате выполнения заданий лабораторного практикума, позволит студентам применять полученные знания при разработке программных средств, как для образовательных целей, так и в профессиональной деятельности.

ЛАБОРАТОРНАЯ РАБОТА 1.
СТРУКТУРА КОНСОЛЬНОГО ПРИЛОЖЕНИЯ В C#

1. Цель и содержание

Цель лабораторной работы: научиться работать с переменными иконстантами простых типов в C#.

Задачи лабораторной работы:

– научиться объявлять переменные простых типов в языке C#;
– научиться объявлять константы простых типов в языке C#;
– научиться выполнять простейшие действия с переменными иконстантами.

2. Теоретическая часть

Рассмотрим структуру консольного приложения на языке C#, созданногос использованием средств MS Visual Studio (рис. 1.1).

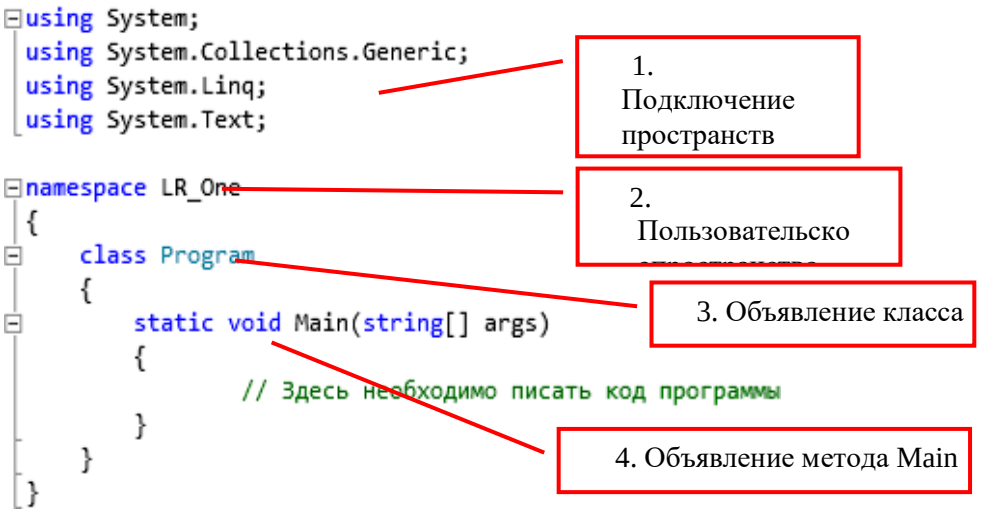

Рисунок 1.1 – Исходный код консольного приложения.

В интерактивном редакторе кода среды разработки код может быть свернут/ развернут с использованием кнопок +/-, внедренных в код. На рис. 1.1 в области 1 показано подключение пространств имен (библиотек, содержащих стандартные инструменты) с использованием зарезервированного слова using.

Программист сам создает свое пространство имен с именем LR_One (2), вкотором объявляется класс с именем Program.

В классе Program объявлен один метод – функция Main (параметры функции не рассматриваем).

Функция Main имеет особенное значение в программировании на языках

C, C++ и C#.

Функцию Main называют «точкой входа», то есть началом выполнения программы. Далее мы рассмотрим приложения, содержащие множество функций. Операционная система знает с какой именно функции начать выполнение программы – с функции Main. Очевидно, что имя этой функции менять нельзя. Это должен быть статический метод класса (или структуры), возвращающий либо значение типа int, либо void. Хотя нередко модификатор public указывается явно, поскольку по определению этот метод должен быть вызван извне программы, на самом деле неважно, какой уровень доступа вы назначите методу точки входа. Он запустится, даже если вы пометите его как private.

Когда компилируется консольное или Windows-приложение C#, по умолчанию компилятор ищет в точности один метод Main () с описанной выше сигнатурой в любом классе и делает его точкой входа программы. Если существует более одного метода Main ( ), компилятор возвращает сообщение обошибке.

Обратите внимание на вложенность конструкций на рис. 1.1: в пространство имен LR_One вложен класс Program, в класс Program вложена функция Main. В свою очередь, в функции Main содержатся инструкции на языке C#-код, который начнет выполняться при старте программы.

В C#, как и в других C-подобных языках, большинство операторов завершаются точкой с запятой (;) и могут продолжаться в нескольких строках без необходимости указания знака переноса. Операторы могут быть объединены в блоки с помощью фигурных скобок ({ }). Однострочные комментарии начинаются с двойного слеша (//), а многострочные – сослеша со звездочкой (/*) и заканчиваются противоположной комбинацией (*/). В этих аспектах язык C# идентичен C++ и Java.

Следует также помнить о том, что язык C# чувствителен к регистру символов. Это значит, что переменные с именами myVar и MyVar являются разными.

Причина присутствия оператора using в файле Program.cs связана с использованием библиотечных классов.

Весь код C# должен содержаться внутри класса. Объявление класса состоит из ключевого слова class, за которым следует имя класса и пара фигурных скобок. Весь код, ассоциированный с этим классом, размещается между этими скобками.

3. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный

(x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб ивыше, свободное дисковое пространство – не менее 1 Гб, графическое устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

4. Методика и порядок выполнения работы

1. Создайте консольное приложение, для этого выполните следующиедействия:

Выберите команду главного меню

File → *New* → *Project*...

В открывшемся диалоговом окне (рис. 1.2) выберите необходимые настройки для создаваемого проекта: язык Visual C#; фреймворк: .NET Framework 4; шаблон: Console Application.

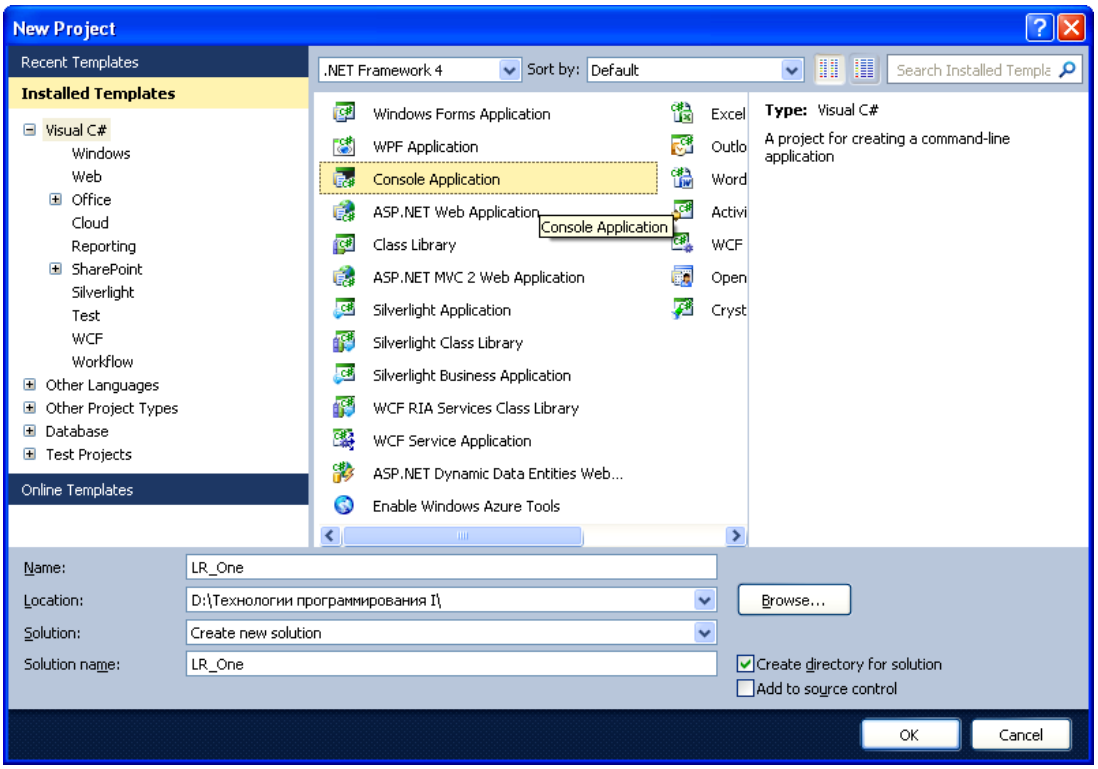

Рисунок 1.2 – Создание нового проекта консольного приложения.

В текстовом поле Name введите имя проекта (напримерLR_One).

В текстовом поле Location выберите место сохранения нового проекта.

Установите флажок-переключатель «Create directory forsolution».

Нажмите кнопку «OK».

2. После выполнения пункта 1 в среде разработки откроется новыйсозданный проект (рис. 1.3).

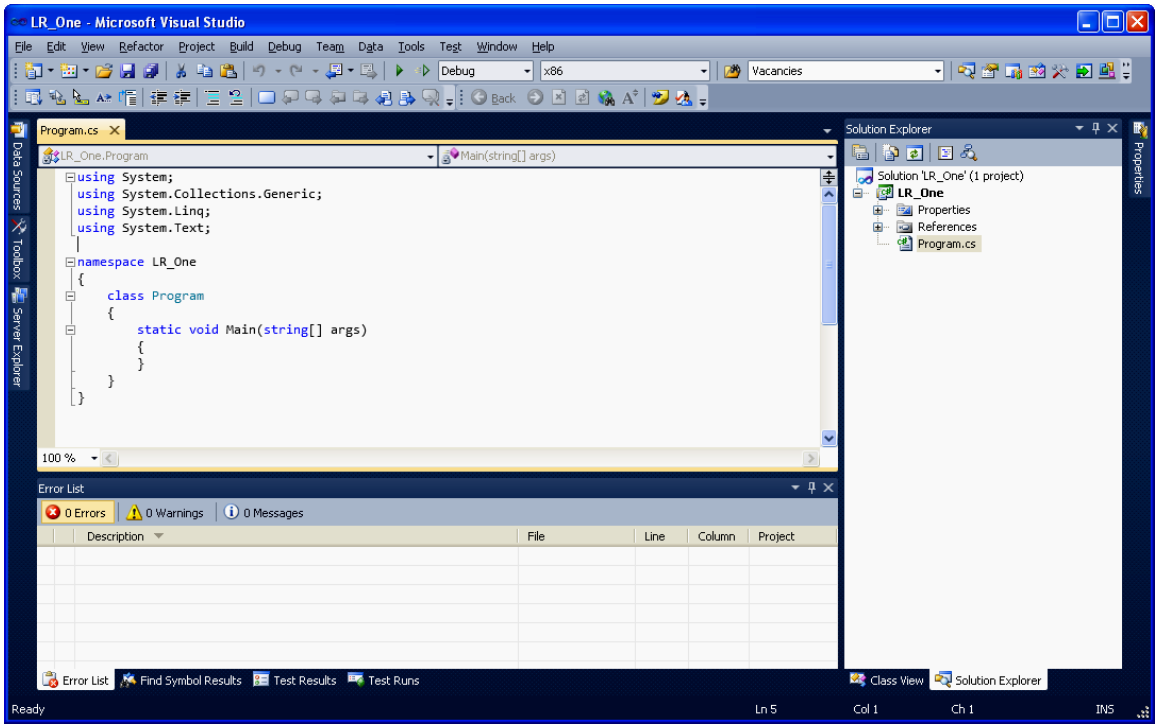

Рисунок 1.3 – Новый проект, загруженный в среду разработки: вкладка «Solution Explorer» отображает состав проекта (нас интересует только файл Program.cs); в левой части окна (сверху) открыт файл Program.cs в редакторекода.

3. На данном этапе необходимо ознакомиться со структурой исходного файла консольного приложения (рис. 1.4).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
                // Здесь необходимо писать код программы
        }
    }
}
```

Рисунок 1.4 – Исходный файл консольного приложения.

Весь код программы необходимо писать внутри функции Main. Для построения сборки (исполняемого exe-файла) выполните команду главного меню *Build* *Build Solution* (или использовать горячую клавишу *F* 6). После этого сборка создана, но приложение не будет запущено автоматически.

4. Для создания сборки и последующего запуска программы можно воспользоваться командой *Debug* *Start Debugging* главного меню среды разработки или нажать кнопку панели инструментов *Start Debugging*

(*F5*) . Можно также использовать горячую клавишу *F*5 .

5. Запустите приложение на выполнение одним из методов, указанных в пункте 6. Окно консольного приложения появится и исчезнет. Это означает, что приложение выполнило все команды, написанные программистом, и завершило свою работу.

6. Для удержания окна на экране измените исходный файл в соответствии с рисунком 1.5. В функции Main добавлен вызов только одной команды Console.DeadKey ( ) – эта функция останавливает выполнение программы и ждет, когда пользователь нажмет любую клавишу.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.ReadKey();
        }
    }
}
```

Рисунок 1.5 – Исходный файл консольного приложения дляпредотвращения закрытия окна консольного приложения.

7. Запустите измененное приложение, убедитесь, что окно удерживается на экране.

8. Добавьте несколько строк кода в исходный файл (рис. 1.6).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_One
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Лабораторная работа №1");
            Console.WriteLine("");
            Console.WriteLine("Выполнил: Иванов Иван иванович");
            Console.WriteLine("Группа: ИСТБ-121");
            Console.WriteLine("Наименование ЛР: Структура консольного приложения");
            Console.WriteLine("");
            Console.WriteLine("для завершения работы программы нажмите любую клавишу...");

            Console.ReadKey();
        }
    }
}
```

Рисунок 1.6 – Исходный файл консольного приложения для вывода информации на экран.

9. Внимательно изучите исходный код примера на рис. 1.5. Запустите приложение и убедитесь, что отсутствуют ошибки и информация

выводится.

10. Выполните индивидуальное задание.

Индивидуальное задание.

Измените приложение, созданное в ходе выполнения данной лабораторной работы таким образом, чтобы программа выводила на экран следующую информацию (каждый студент должен использовать персональнуюинформацию о себе):

- Название и номер лабораторной работы;
- ФИО студента;
- Группа студента и шифр специальности;
- Дата рождения студента;
- Населенный пункт постоянного места жительства студента;
- Любимый предмет в школе;
- Краткое описание увлечений, хобби, интересов.

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Какая функция имеет особенное значение при выполнении программына языках C, C++, C#?
2. Что такое «точка входа» в программе?
3. Как вы понимаете термины «пространства имен», «класс», «метод»,

«функция»?

4. Чем отличается структура консольного приложения от приложения, построенного с использованием технологий Windows Forms, WPF?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-3].

ЛАБОРАТОРНАЯ РАБОТА 2. ПРЕДОПРЕДЕЛЕННЫЕ ТИПЫДАННЫХ, ПЕРЕМЕННЫЕ, КОНСТАНТЫ

1. Цель и содержание

Цель лабораторной работы: научиться работать с переменными иконстантами простых типов в C#.

Задачи лабораторной работы:

– научиться объявлять переменные простых типов в языке C#;
– научиться объявлять константы простых типов в языке C#;
– научиться выполнять простейшие действия с переменными и константами.

2. Теоретическая часть

Перед выполнением лабораторной работы необходимо изучить материалы лекций. Следует понимать принцип деления типов .NET на типы значений и ссылочные типы.

В данной лабораторной работе необходимо освоить приемы работы с предопределенными типами значений

Типы значений C#

Язык C# поддерживает 8 предопределенных целочисленных типов(таблица 2.1).

Таблица 2.1 – Целочисленные типы C#.

Имя типа	Тип CTS	Описание	Диапазон (минимум : максимум)
sbyte	System.SByte	8-битное целое со знаком	-128 : 127
short	System.Int16	16-битное целое со знаком	-32 768 : 32 767
int	System.Int32	32-битное целое со знаком	-2 147 483 648 2 147 483 647
long	System.Int64	64-битное целое со знаком	$-2^{63} : 2^{63}-1$
byte	System.B	8-битное целое без	0 : 255

	yte	знака	
ushort	System.UInt16	16-битное целое без знака	0 : 65 535
uint	System.UInt32	32-битное целое без знака	0 : 2^{32}-1
ulong	System.UInt64	64-битное целое без знака	0 : 2^{64}-1

Язык C# также поддерживает и типы с плавающей точкой (таблица 2.2).

Таблица 2.2 – Типы с плавающей точкой C#.

Имя типа	Тип CTS	Описание	Кол-во знаков	Диапазон (минимум : максимум)
float	System.Single	32-битное с плавающей точкой одинарной точности	7	от $\pm 1.5\times10^{-45}$ до $\pm 3.4\times10^{38}$
double	System.Double	64-битное с плавающей точкой двойной точности	15/16	от $\pm 5.0\times10^{-324}$ до $\pm 1.7\times10^{308}$

В таблице 2.3 представлен десятичный тип C#. Данный тип реализовандля финансовых операций.

Таблица 2.3 – Десятичный тип C#.

Имя типа	Тип CTS	Описание	Кол-во знаков	Диапазон (минимум : максимум)
decimal	System.Decimal	128-битное с плавающей точкой в десятичной нотации с высокой точностью	28	от $\pm 1.0\times10^{-28}$ до $\pm 7.9\times10^{28}$

Как и во многих языках программирования существует булевский тип(таблица 2.4).

Таблица 2.4 – Булевский тип.

Имя типа	Тип CTS	Значения
bool	System.Boolean	true или false

Для хранения одиночных символов в языке C# используется тип char(таблица 2.5)

Таблица 2.5 – Булевский тип.

Имя типа	Тип CTS	Значения
char	System.Char	Представляет отдельный 16-битный (Unicode) символ

Литералы типа char записываются как одиночные, заключенные в одинарные кавычки символы: ‘F’, ‘w’, ‘ц’, ‘Я’ и т.д.

В переменных типа char можно хранить и специальные символы в видеуправляющих последовательностей (таблица 2.6).

Таблица 2.6 – Представление символов в виде управляющихпоследовательностей.

Управляющая последовательность	Символ
\’	Одиночная кавычка
\”	Двойная кавычка
\\	Обратный слэш
\0	Пусто
\a	Предупреждение (звуковой сигнал)
\b	Забой
\f	Подача формы
\n	Новая строка
\r	Возврат каретки
\t	Символ табуляции
\v	Вертикальная табуляция

Если отдельные символы объединены в строку, то необходимо использовать тип string, который отображается на тип CTS – System.String.

Объявление и инициализация переменных в C#

Синтаксис объявления переменных в C# выглядит следующим образом:

Тип Данных Например: *Идентификатор Переменной*;

```
int a;
```

Этот код объявляет переменную типа int с именем a. Компилятор не позволит использовать эту переменную до тех пор, пока она не будет инициализирована (т.е. пока ей не будет присвоено значение).

Для инициализации переменной a необходимо написать следующий код:

```
a = 123;
```

Переменную можно инициализировать во время объявления:

```
int b = 7;
```

или

```
string str = "Hello, World!!!";
```

Синтаксис C# позволяет объявить несколько переменных (и инициализировать их) одного типа в одной синтаксической конструкции.

Например:

```
float b, i, myPerem, U_t = 0.3F, z_11 = 23.56F;
```

В данном примере объявляется 5 переменных типа float, некоторые из них инициализируются в процессе объявления.

Объявление и инициализация констант в C#

Константа – это переменная, значение которой не меняется за время выполнения программы. Для объявления константы необходимо воспользоваться ключевым словом const. Например:

```
const char simv = 'A';
const double pi = 3.14;
```

Очевидно, что при таком объявлении, поменять значения simv и pi в дальнейшем будет нельзя.

3. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмомописанном в лабораторной работе №1.
2. Изучите материал в разделе «Теоретическое обоснование» даннойлабораторной работы.

3. Модифицируйте исходный файл как показано на рис. 2.1.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5
6  namespace LR_One
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int a;
13             int b = 7;
14
15             string str = "Hello, World!!!";
16
17             a = 123;
18
19             Console.ReadKey();
20         }
21     }
22 }
```

Рисунок 2.1 – Объявление переменных в C#.

4. Рассмотрим приведенный пример подробнее:

В строке 12 объявляется переменная типа int с именем a.

В строке 13 объявляется переменная типа int с именем b, причем при объявлении для нее устанавливается начальное значение, равное 7.

В строке 15 объявляется переменная типа string с именем str иинициализируется строковым значением «Hello, World!!!».

В строке 17 переменной a присваивается целочисленное значение

123.

5. Запустите приложение на выполнение. У вас должно появиться пустое

окно консольного приложения. Очевидно, что исходный код, представленный на рис. 2.1 не предполагает вывода какой-либо информации на экран.

6. Для вывода информации на экран воспользуемся функцией Console.WriteLine, изученной в лабораторной работе 1. Добавим в исходный файл следующие строки (строки 19 – 21 на рис. 2.2).

```
6  namespace LR_One
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int a;
13             int b = 7;
14
15             string str = "Hello, World!!!";
16
17             a = 123;
18
19             Console.WriteLine(a);
20             Console.WriteLine(b);
21             Console.WriteLine(str);
22
23             Console.ReadKey();
24         }
25     }
26 }
27
```

Рисунок 2.2 – Добавление строк кода для вывода значений объявленных переменных на экран.

7. В примере на рис. 2.2 используется простой вывод, то есть имя переменной просто передается в качестве параметра функции Console.WriteLine.

8. Для выполнения форматного вывода (изменения формата представления выводимой информации) необходимо реализовать вывод в следующем виде (рис. 2.3 изменены строки 19-21):

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5
6  namespace LR_One
7  {
8      class Program
9      {
10         static void Main(string[] args)
11         {
12             int a;
13             int b = 7;
14
15             string str = "Hello, World!!!";
16
17             a = 123;
18
19             Console.WriteLine("Значение переменной a равно {0}", a);
20             Console.WriteLine("a значение переменной b равно {0}", b);
21             Console.WriteLine("значение a+b = {0}+{1} = {2}", a, b, a+b);
22             Console.WriteLine(str);
23
24             Console.ReadKey();
25         }
26     }
27 }
```

Рисунок 2.3 – Вывод информации с использованием форматных строк.

9. Внимательно изучите код полученной программы, Затем выполнитеиндивидуальное задание.

Индивидуальное задание.

Объявите требуемые переменные, присвойте им начальные значения (определите самостоятельно, значения какого типа могут принимать переменные), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения в соответствие с вариантом:

Вариант	Выражение для вычисления
1	$(0,2x)^3 + \cos x$
2	$x-10\sin x+ 76$
3	$2^{-x+} \sin x$
4	$2^x-2\cos x +10$
5	$\lg(x+5) + \cos x$
6	$\sqrt{4x+1} + 3\cos x$
7	$x \sin x- 1 + 4$
8	$8\cos x-x + 6$
9	$\sin x-0,2x + 3$
10	$10\cos x-0,1x^2 +2$
11	$21g(x+7)-5\sin x +50$
12	$4 \cos x+0,3x -12$
13	$5\sin 2x + \sqrt{1-x}$
14	$l,2x^4+2x^3-24.1+ 13x^2+l4.2x$
15	$\ln(x+6,l) / 2\sin (x-1,4)$

7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе

её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

8. Контрольные вопросы

1. Что такое переменная? Как объявляется переменная?
2. Как объявляется константа? Чем константа отличается от переменной?
3. Какие типы значений применяются C#?
4. Чем тип char отличается от типа string?
5. Как производится инициализация переменных? Как производитсяинициализация констант?
6. Что такое управляющие последовательности?

9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [1], [7].

ЛАБОРАТОРНАЯ РАБОТА 3. ИСПОЛЬЗОВАНИЕ ВОЗМОЖНОСТЕЙКОНСОЛЬНОГО ВВОДА-ВЫВОДА.

1. Цель и содержание

Цель лабораторной работы: изучить команды ввода и вывода данных в консольном приложении.

Задачи лабораторной работы:

– научиться применять функции Console.Write и Console.WriteLine;
– научиться применять функции Console.Read и Console.ReadLine;
– научиться использовать форматы вывода для различных типов данных.

2. Теоретическая часть

В предыдущих лабораторных работах уже использовалась команда Console.WriteLine. Рассмотрим ее синтаксис подробнее:

```
Console.WriteLine("Строка текста");
```

Эта запись означает вызов статического метода класса Console. В качестве параметра функции передается строка, которая выводится в

консоль.

Существует также функция:

```
Console.Write("Строка текста");
```

Этот метод также осуществляет вывод, только не переводит каретку на следующую строку.

Рассмотренные методы также позволяют осуществлять форматный вывод, аналогичный функции printf языка Си, например:

В данном примере в качестве первого параметра передается строка,

```
int a = 100, b = 13, c = 23, d = 0, e = 0;
Console.WriteLine("Переменная a = {0}, переменная b = {1}," +
"переменная c = {2}, " +
"переменная d = {3}, " +
"переменная e = {4}", a, b, c, d, e);
```

содержащая маркеры в фигурных скобках. При выводе на места маркеров будутподставлены параметры, которые следуют за стрококй.

При форматном выводе можно задавать ширину поля вывода, для этого необходимо воспользоваться следующим приемом:

В данном примере видно, чтобы задать ширину вывода необходимо

```
int a = 100, b = 13, c = 23, d = 0, e = 0;
Console.WriteLine("Переменная a = {0, 5}, переменная b = {1, 5}," +
"переменная c = {2, 5}, " +
"переменная d = {3, 5}, " +
"переменная e = {4, 5}", a, b, c, d, e);
```

использовать в формате {n, w}, где n – порядковый номер параметра, а w – ширина области вывода (знак w позволяет осуществлять выравнивание выводимого значения по левому или правому краю).

При выводе также можно добавлять строку формата вместе с необязательным значением точности (таблица 2.1).

Таблица 3.1 – Основные строки формата.

Стока	Описание
C	Локальный формат валюты
D	Десятичный формат. Преобразует целое к основанию 10 и снабжает ведущими нулями, если естьспецификатор точности.
E	Научный (экспоненциальный) формат. Спецификатор точности устанавливает количество десятичных разрядов (по умолчанию 6).
F	Форматс фиксированной запятой. Спецификатор точности задает количество десятичных разрядов.

G	Общий формат. Форматирование E или F.
N	Числовой формат. Форматирует число с разделителями тысяч – запятыми.
P	Процентный формат
X	Шестнадцатиричный формат. Спецификатор точности используется для указания ведущих нулей.

Пример использования формата:

```
float a, b, c, res;

Console.Write("Введите a > ");
a = Convert.ToSingle(Console.ReadLine());

Console.Write("Введите b > ");
b = Convert.ToSingle(Console.ReadLine());

Console.Write("Введите c > ");
c = Convert.ToSingle(Console.ReadLine());

res = (a + b) * c;

Console.WriteLine("\n\n");

Console.WriteLine("Денежный формат {0:C} \n", res);
// Console.WriteLine("Десятичный формат {0:D10} \n", res);
Console.WriteLine("Экспоненциальный формат {0:E} \n", res);
Console.WriteLine("Формат с фиксированной запятой {0:F3, 7} \n", res);
Console.WriteLine("Общий формат {0:G} \n", res);
Console.WriteLine("Числовой формат {0:N} \n", res);
Console.WriteLine("Процентный формат {0:P} \n", res);
// Console.WriteLine("Шестнадцатиричный формат {0:X} \n", res);
```

Изучите представленный пример. Выполните данный код в средеи

В классе Console также существуют методы, позволяющие считывать информацию, вводимую пользователем:

В данном примере строка, введенная пользователем, считывается в

```
string str;

str = Console.ReadLine();

Console.WriteLine("вы ввели строку: " + str);
```

переменную str, затее осуществляется ее вывод. Наберите данный пример в среде VS и изучите код.

3. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом,представленным в лабораторной работе №1.
2. Выполните индивидуальное задание, выполнив вывод результата с использованием всех возможных форматов, представленных в

таблице 2.1.

Индивидуальное задание.

Объявите требуемые переменные, значения переменным пользователь должен присваивать в процессе выполнения программы (считываются с консоли), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения во всех возможных форматах:

Уравнения математических формул выдаются для студента преподавателем индивидуально.

5. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

6. Контрольные вопросы

1. Как оформляется комментарий?
2. Какие функции используются для вывода информации в консоль?
3. Какие функции используются для считывания информации с консоли?
4. Какие форматы вывода вы знаете? Опишите их.
5. Какой класс содержит статические методы для конвертированиязначений и приведения их к требуемому типу?
6. Какой класс содержит статические методы – математические функции? Что такое управляющие последовательности?

7. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [4], [7], [8].

ЛАБОРАТОРНАЯ РАБОТА 4.
УПРАВЛЕНИЕ ПОТОКОМ ВЫПОЛНЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРОВ IF, SWITCH

1. Цель и содержание

Цель лабораторной работы: изучить операторы, позволяющие организовывать непоследовательное выполнение программного кода.

Задачи лабораторной работы:

– научиться применять условный оператор if;
– научиться применять оператор цикла for;
– научиться применять операторы выбора switch.

2. Теоретическая часть

Условный оператор.

В предыдущих лабораторных работах были рассмотрены вопросы программирования на языке C# с использованием только последовательного выполнения операторов программы.

В языке C# (как и в большинстве языков) можно вводить условия, циклы, ветвления.

Условный оператор в C# позволяет организовать условие типа «если …,то …, иначе …».

Синтаксис условного оператора:

if (*условие*)
оператор
[else if (*условие*)

...
else
оператор

оператор]

В данном синтаксисе *оператор* может быть составным оператором (группа операторов языка, заключенные в фигурные скобки). Условный оператор содержит только одно обязательное зарезервированное слово – if. Все остальные конструкции не являются обязательными (в синтаксическом описании на это указывают скобки [ ]).

Пример использования условного оператора:

```
int a = 100, b = 13, c = 23, d = 0, e = 0;


if (a == 100) // Если a равно 100
{
    // Увеличиваем b на 1000
    b += 1000;
}
else if (a < 100) // иначе если a меньше 100
{
    // изменяем значение d
    d = b + c;
    // и изменяем значение e
    e = a + b;
}
else // иначе (остальные варианты не предусмотренные первыми двумя условиями)
    // edtkbxbdftv a на 1
    a++;

// продолжается последовательное выполнение программы
Console.WriteLine("{0}", a);
```

В представленном примере сначала проверяется является ли значение переменной a равным 100.

Если «Да», то выполняется единственный оператор (увеличение b на 1000). На этом выполнение всего условного оператора завершено и программа переходит к выполнению операторов, следующих за ним, то есть к выводу значения a.

1. Если первое условие не выполнилось, проверяется второе – является ли a меньше 100 (конструкция else if). Если это утверждение истинно, то выполняется составной оператор, состоящий из двух – изменение значений переменных d и e. Затем осуществляется переход к выводу переменной a.
2. Если не выполнилось ни одно условие с оператором if, то выполняется оператор, указанный после зарезервированного слова else. В данном случае это несоставной оператор (a увеличивается на 1), поэтому фигурные скобки можно не писать.

Во всей этой конструкции только оператор if является обязательным. Конструкций else if может быть любое количество, а if и else – только по одному.

Еще один пример условного оператора:

```
int a = 100;

if (a >= 100)
{
    Console.WriteLine("Значение переменной a больше или равно 100");
}
```

В данном случае вывод в консоль выполнится только если a больше либо равно 100.Иначе условный оператор ничего не выведет. Фигурные

скобки в данном примере можно опустить.

Обратите внимание, что для проверки на равенство используется оператор ==, а не =. Знак «равно» используется в C# для присваивания значений. Следует понимать, что условное выражение, стоящее в конструкцииif должно возвращать булево значение.

Например, следующий условный оператор всегда будет выполняться:

```
if (true)
    Console.WriteLine("Всегда выводится");
else
    Console.WriteLine("Никогда не выводится");
```

Оператор выбора.Синтаксис оператора:

switch (*перем*)
case *константа*1:
{
оператор _1;
break;
}
case *константа*2 :
{*оператор* _ 2;
break;
}
default :
оператор _ *n*;
break;
}

В данном операторе осуществляется выбор оператора (который может быть составным) в зависимости от значения переменной *перем* . Если *перем* равна значению *константа*1, то выполнится *оператор* _1, если *константа*2 – *оператор* _ 2, и т.д. Если ни одно значение, указанное в каком-либо операторе case, не совпало со значением *перем*, то выполнится оператор, указанный в секции default . Внутри каждой секции case следует указать оператор break, который предотвращает проверку других условий после выполнения данного оператора. Следует обратить внимание, что в секциях case следует использовать только константы (переменные не допускаются).

Пример использование оператора switch … case:

```
// Организация ответа на действие пользователя:
string text = "1 - Вывод группы \n" +
    "2 - Вывод фамилии преподавателя \n" +
    "3 - Вывод названия предмета \n" +
    "4 - Вывод приветствия \n";

Console.Write(text + "Введите команду > ");
int result = Convert.ToInt32(Console.ReadLine());

switch (result)
{
    case 1: { Console.WriteLine("ИСТБ - 101"); break; }
    case 2: Console.WriteLine("Николаев Евгений Иванович"); break;
    case 3: { Console.WriteLine("Технологии программирования"); break; }
    case 4: Console.WriteLine("Привет !"); break;
    default: Console.WriteLine("Недопустимый вариант !!!"); break;
}
```

Изучите представленный пример самостоятельно.

3. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.
2. Выполните индивидуальное задание. Во всех заданиях переменные X, Y являются вещественными и вводятся пользователем. Количество слагаемых также вводится пользователем. Программа должна вывести сумму заданного числа членов последовательности.

Индивидуальное задание.

Перед выполнением задания требуется самостоятельно определить закономерность изменения членов последовательности, чтобы применить цикл,условный оператор или, если потребуется, оператор выбора.

Задача №1

Напишите программу, которая считывает три строки. Если эти три строки – «раз», «два» и «три», то программа выводит «ГОРИ», если нет, то «НЕ ГОРИ».

Задача №2

Напишите программу, которая считывает две строки и выводит «ВЕРНО», если в каждой из них записано или слово только да, или только слово нет (в любой комбинации). Если это не так, выведите «НЕВЕРНО».

Задача №3

При регистрации нового ящика электронной почты пользователя обычно просят ввести, помимо прочего, желаемый логин, а также резервный адрес электронной почты (на случай, если понадобится восстановить забытый пароль). Напишите программу, которая проверяет, что пользователь ничего не перепутал и ввёл корректный логин (не содержащий символ «@») и корректный резервный адрес (содержащий символ «@»). Иных проверок, кроме указанных здесь, выполнять не надо.

Формат ввода

Вводятся две строки: предлагаемые пользователем логин и резервный адрес.

Формат вывода

Выводится одна строка: если все условия выполнены, то выводится «OK» (латиницей); если в логине присутствует «@», то выводится «Некорректный логин»; если

логин корректный, но в адресе отсутствует «@», то выводится «Некорректный адрес».

Задача №4

Напишите программу-тест, которая по некоторым простым вопросам выдаёт «строго индивидуальный» анализ личностных качеств. Задайте пользователю два-четыре вопроса с тремя-пятью вариантами ответа (например, «Какое ваше любимое время года?») и считайте его ответы.

Если пользователь при ответе на любой вопрос (в том числе и первый) дал не предусмотренный вами вариант ответа, то надо сообщить ему об ошибке и сразу же завершить работу.

Если же он дал предусмотренный ответ на каждый из вопросов, выдаём пользователю результат (например, «Вы обладаете незаурядным умом.»), причём должно быть не менее пяти разных вариантов результата.

Обратите внимание, что программа должна задавать следующий вопрос только при получении корректного ответа на предыдущий.

5. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

6. Контрольные вопросы

1. Как оформляется комментарий?
2. Опишите синтаксис условного оператора. Какие части данногооператора являются обязательными?
3. Опишите синтаксис оператора выбора. Поясните почему следуетиспользовать оператор break внутри каждого выбора.
4. Можно ли с помощью for реализовать бесконечный цикл? Пояснитеответ на примерах.

7. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [2-4].

Перечень рекомендуемой литературы

1. Язык Си# Базовый курс [Электронный ресурс]: учебное пособие/ Под-бельский В.В.— Электрон. текстовые данные.— М.: Финансы и статистика, 2011.— 384 с.— Режим доступа: http://www.iprbookshop.ru/18866.— ЭБС «IPRbooks».

2. Кудинов Ю.И. Основы алгоритмизации и программирования. Часть 1 [Электронный ресурс]: учебное пособие/ Кудинов Ю.И., Келина А.Ю.— Электрон. текстовые данные.— Липецк: Липецкий государственный техни-ческий университет, ЭБС АСВ, 2013.— 71 с.— Режим доступа: http://www.iprbookshop.ru/55121.— ЭБС «IPRbooks».

3. Липаев В.В. Документирование сложных программных комплексов [Элек-тронный ресурс]: электронное дополнение к учебному пособию «Про-граммная инженерия сложных заказных программных продуктов» (для бакалавров)/ Липаев В.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2015.— 115 с.— Режим доступа: http://www.iprbookshop.ru/27294.— ЭБС «IPRbooks».

4. Влацкая И.В. Проектирование и реализация прикладного программного обеспечения [Электронный ресурс]: учебное пособие/ Влацкая И.В., Заель-ская Н.А., Надточий Н.С.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2015.— 119 с.— Режим доступа: http://www.iprbookshop.ru/54145.— ЭБС «IPRbooks».

5. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual C# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский госу-дарственный строительный университет, ЭБС АСВ, 2011.— 180 с.— Ре-жим доступа: http://www.iprbookshop.ru/19258.— ЭБС

СОДЕРЖАНИЕ

Мурадов М.М.

Методические указания предназначены для студентов дневной и заочной форм обучения по направлению подготовки бакалавров «Прикладная информатика».

.