Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный университет имени П.М. Машерова»
Кафедра информатики и информационных технологий

Л.Е. Потапова, Т.Г. Алейникова

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ C#

Методические рекомендации к выполнению лабораторных работ

Витебск
ВГУ имени П.М. Машерова
2014

УДК 004.432(075.8)
ББК 32.973.26-018.1я73
П64

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 3 от 20.12.2013 г.

Авторы: доцент кафедры информатики и информационных технологий ВГУ имени П.М. Машерова, кандидат физико-математических наук **Л.Е. Потапова;** доцент кафедры информатики и информационных технологий ВГУ имени П.М. Машерова, кандидат физико-математических наук **Т.Г. Алейникова**

Р е ц е н з е н т :
заведующий кафедрой прикладной математики и механики
ВГУ имени П.М. Машерова, кандидат физико-математических наук,
доцент *С.А. Ермаченко*

Потапова, Л.Е.

П64 Алгоритмизация и программирование на языке C# : методические рекомендации к выполнению лабораторных работ / Л.Е. Потапова, Т.Г. Алейникова. – Витебск : ВГУ имени П.М. Машерова, 2014. – 50 с.

Методические рекомендации содержат основной материал по программированию на языке C#, необходимую справочную информацию по использованию среды разработки Visual Studio.NET, вопросы и индивидуальные задания для лабораторных занятий и самостоятельной работы.

Данное издание предназначено для обучения методам алгоритмизации с помощью языка программирования C# и адресовано студентам физико-математических специальностей.

УДК 004.432.(075.8)
ББК 32.973.26-018.1я73

СОДЕРЖАНИЕ

1. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ЭВМ 5
1.1. Кодирование числовой информации 5
1.2. Кодирование текстовой информации 7
Лабораторная работа № 1. Кодирование числовой и текстовой информации 8
2. СОЗДАНИЕ КОНСОЛЬНОГО ПРИЛОЖЕНИЯ. ВВОД И ВЫВОД ДАННЫХ 10
2.1. Консольный ввод-вывод 10
2.1.1. Методы вывода 10
2.1.2. Ввод с клавиатуры 12
2.2. Явное преобразование типа 12
2.3. Стандартные методы 13
Лабораторная работа № 2. Знакомство со средой VISUAL STUDIO.NET. Создание консольного приложения 14
3. УПРАВЛЯЮЩИЕ ОПЕРАТОРЫ. ПРОГРАММЫ С ВЕТВЛЕНИЯМИ 17
3.1. Операторы if, if ... else 17
3.2. Инструкция switch 17
Лабораторная работа № 3. Управляющие операторы ветвления. Преобразование типов 19
4. ЦИКЛИЧЕСКИЕ ПРОГРАММЫ 23
4.1. Оператор цикла с предусловием 23
4.2. Оператор цикла с постусловием 23
4.3. Оператор цикла с параметром 24
ЛАБОРАТОРНАЯ РАБОТА № 4. Операторы циклов 25
5. МАССИВЫ 27
5.1. Одномерный массив 27
5.2. Многомерные массивы 28
Лабораторная работа № 5. Обработка массивов 30
6. СТРОКИ 32
6.1. Создание строк 32
6.2. Работа со строками 33
Лабораторная работа № 6. Строки 35
7. ОБЪЯВЛЕНИЕ И ОПРЕДЕЛЕНИЕ МЕТОДОВ 36
7.1. Объявление методов 36
7.2. Параметры методов 37
7.3. Использование методов 37
Лабораторная работа № 7. Программирование с использованием методов 39
ЛИТЕРАТУРА 43

ПРИЛОЖЕНИЯ …… 44
Приложение 1. Основная таблица ASCII …… 44
Приложение 2. Встроенные типы в C# ……... 44
Приложение 3. Таблица явных числовых преобразований в C# ……. 45
Приложение 4. Основные методы пространства имен System …… 46
4.1. Основные поля и статические методы класса Math ... 46
4.2. Основные элементы класса Array ……. 47
4.3. Основные методы класса System.Char ……. 47
4.4. Основные элементы класса System.String ……... 48
4.5. Основные методы класса System.Random ……... 49

1. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ЭВМ

Цель: сформировать представление о кодировании различного вида информации и умение преобразовывать ее в машинный код.

Все данные в компьютере представляются в двоичной системе счисления. В ЭВМ используют двоичную систему потому, что она имеет ряд преимуществ перед другими.

Для хранения информации в компьютере используется совокупность разрядов, которую принято называть ячейкой памяти.

1.1. Кодирование числовой информации

Кодом целого положительного десятичного числа считается его запись в двоичной системе счисления. Множество целых чисел, представимых в памяти ЭВМ, ограничено. Диапазон значений зависит от размера области памяти, используемой для размещения чисел. В k-разрядной ячейке может храниться 2^k различных значений целых чисел.

Представление целых чисел в компьютере.

Для целых чисел существуют два представления: беззнаковое (только для неотрицательных целых чисел) и со знаком.

Представление целых чисел в беззнаковых целых типах.

При представлении неотрицательных чисел в беззнаковом формате все разряды ячейки отводятся под само число, представленное в двоичном коде (прямой код). Например, запись числа 243=11110011$_2$ в одном байте при беззнаковом представлении будет выглядеть следующим образом:

1	1	1	1	0	0	1	1

Представление целых чисел в знаковых целых типах.

Для представления чисел со знаком весь диапазон возможных значений делится пополам: половина неотрицательные, половина – отрицательные.

Для представления неотрицательных целых чисел используется прямой код. Для их представления отрицательных целых чисел используется так называемый *дополнительный код*. Дополнительный код используется для упрощения выполнения арифметических операций.

Для компьютерного представления целых чисел обычно используется один, два или четыре байта.

Дополнительный код отрицательного числа m равен 2^k-|m|, где k – количество разрядов в ячейке.

Например, если число –13 = 2^{16}-|-13|=65523,записать в ячейку из 2-х байтов, то содержимое ячейки будет следующим:

1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1

Другой алгоритм получения дополнительного кода отрицательного числа следующий:

– модуль отрицательного числа представить прямым кодом в k двоичных разрядах;

– значение всех бит инвертировать: все нули заменить на единицы, а единицы на нули (таким образом, получается k-разрядный обратный код исходного числа);

– к полученному обратному коду прибавить единицу.

Пример. Получим 16-разрядный дополнительный код числа – 13:

0000000000001101 – число |-13|=13 в прямом коде,

1111111111110010 –инвертированное значение,

1111111111110011 – число-13в дополнительном коде.

Можно заметить, что представление целого числа не очень удобно изображать в двоичной системе, поэтому используют восьмеричное или шестнадцатеричное представление: 177763_8, FFF3.

Машинное представление вещественных чисел

Практически все современные компьютеры используют представление действительных чисел в так называемой форме с *плавающей запятой.* Так как вся информация в ЭВМ хранится в двоичном представлении, то число представляется в виде:

$x = \pm M \cdot 2^p$, где мантисса $0{,}5 \le M < 1$, (1)

где *M — мантисса*, *p — порядок* числа, т.е. показатель степени, в которую нужно возвести число 2, чтобы, умножив результат на мантиссу и присвоив нужный знак, получить данное число.

Рассмотрим способы перевода вещественного числа в машинный код.

1 способ. Находим такой показатель степени числа 2, чтобы при делении числа *x* на эту степень результат (мантисса) удовлетворял условию (1)

Пример 1. $x = -2{,}75$

Чтобы это число представить в виде (1) его нужно разделить на 4, т. е. 2^2 . Получим $x=-0{,}6875 \cdot 2^2$. Переведем мантиссу в двоичный код:$x = -0{,}1011 \cdot 2^2$. Мы нашли мантиссу и порядок для машинного представления.

Пример 2. $x = 0{,}375$.

Чтобы мантисса этого числа удовлетворяла условию (1), число нужно умножить на 2^1 или разделить на 2^{-1}. Получим $x = 0{,}75 \cdot 2^{-1}$. Переведем мантиссу в двоичный код: $x = 0{,}11 \cdot 2^{-1}$.

2 способ. Переводим модуль числа *x* в двоичный код. Выполняем сдвиги двоичных цифр вправо или влево, чтобы в результате целая часть двоичного числа стала равна 0, а первая цифра дробной части стала равна 1. Количество сдвигов будет порядком, при сдвиге вправо со знаком +, при сдвиге влево со знаком–.

Пример 1. $x = -2,75$

Переводим в двоичный код: $x = -10,11$. Сдвигаем на два разряда вправо: $x = -0,1011 \cdot 2^2$. Сравните с результатом выше.

Пример 2. $x = 0,375$.

Переводим в двоичный код: $x = 0,011$.Сдвигаем на один разряд влево: $x = 0,11 \cdot 2^{-1}$.

Запись чисел в ячейке.

Рассмотрим представление в памяти компьютера величины типа single, которая занимает 4 байта.

Знак числа хранится в крайнем левом разряде с номером 31 (как обычно, разряды нумеруются с 0 и располагаются справа налево). Порядок в смещенной форме занимает разряды с 23 по 30. Смещенная форма (*характеристика*) находится так: к значению порядка прибавляется число 128 («смещение») для того, чтобы значение характеристики всегда было положительным. Мантисса занимает все остальные разряды, т.е. с 0 по 22, причем первая единица мантиссы вообще не хранится (она всегда равна единице и при вычислениях это учитывается).

Пример 1. $x = -2,75$

Характеристика=128+2=130. Переводим в двоичную систему.

Знак числа	характеристика	мантисса	
1	**1 0 0 0 0 0 1 0**	**0 1 1 0 0 0 0**	**0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0**
1-ое машинное слово			2-ое машинное слово

Пример 2. $x = 0,375$.

Характеристика =127. Переводим в двоичную систему.

Знак числа	характеристика	мантисса	
0	**0 1 1 1 1 1 1 1**	**1 0 0 0 0 0 0**	**0 000000000000 0 0 0**
1-ое машинное слово			2-ое машинное слово

1.2. Кодирование текстовой информации

При хранении в компьютере любой текст рассматривается как линейная последовательность символов.

Для представления текстовой информации в компьютере чаще всего используется алфавит мощностью 256 символов. Один символ из такого алфавита несет 8 бит информации. Следовательно, двоичный код каждого символа занимает 1 байт памяти ЭВМ.

Для разных типов ЭВМ и операционных систем используются различные таблицы кодировки, отличающиеся порядком размещения символов алфавита в кодовой таблице. Международным стандартом для ПК является таблица кодировки ASCII (Приложение 1).

В настоящее время преимущественно используется кодовая таблица UNICODE, в которой для кодирования символа используется 2 байта (мощность – 65536 символов).

Контрольные вопросы:

1. Что понимается под кодированием информации?
2. В каком виде представляется информация в памяти компьютера?
3. Какие системы счисления используются в ЭВМ? Почему?
4. Как осуществляется перевод чисел (целых и вещественных) из 10-ой системы счисления в систему счисления с другим основанием? Обратный перевод?
5. Как связаны между собой 2-ая, 8-ая, 16-ая системы счисления?
6. Что такое дополнительный код? Как представить число в дополнительном коде?
7. Как записывается действительное число в нормализованной форме в ЭВМ? Что представляет собой мантисса и характеристика числа? Как их найти?
8. Как вычисляется код символа в таблице ASCII? Сколько байтов отводится при этом на представление символа в компьютере?
9. Сколько байтов для кодирования символа используется в кодовой таблице UNICODE? Сколько символов можно закодировать?

ЛАБОРАТОРНАЯ РАБОТА № 1.
Кодирование числовой и текстовой информации

Задание 1.

Заданное целое число переведите из десятичной в двоичную, восьмеричную и шестнадцатеричную системы счисления. Решение запишите подробно, демонстрируя алгоритм поэтапного деления.

Запишите двухбайтное машинное представление заданного целого положительного числа, а также равного ему по модулю отрицательного числа.

Варианты заданий:

1.	**6645**	6.	**7764**	11.	**7369**
2.	**1098**	7.	**1000**	12.	**1050**
3.	**3199**	8.	**4534**	13.	**9424**
4.	**1493**	9.	**1078**	14.	**5255**
5.	**8320**	10.	**8514**	15.	**2008**

Задание 2.

Используя нормализованную форму записи действительного числа, запишите заданные числа в машинном 4-байтном представлении. Решение приведите с подробным вычислением мантиссы, порядка и характеристики.

Варианты заданий:

1. **6645,31; 0,0012**
2. **-1098,65; 0,025**
3. **-3199,07; 0,000016**
4. **-1493,57; 0,00084**
5. **-8320,14; 0,0019**
6. **-7764,38; 0,0066**
7. **-1000,63; 0,034**
8. **-4534,21; 0,0065**
9. **-1078,39; 0,000021**
10. **-8514,05; 0,00039**
11. **-7369,65; 0,0028**
12. **-1050,11; 0,00054**
13. **-9424,37; 0,00057**
14. **-5255,24; 0,000031**
15. **-2008,96; 0,023**

Задание 3.

Используя таблицу ASCII (приложение 1) и служебную программу «Таблица символов», найдите коды символов и символы по заданным кодам для гарнитуры ***Arial*** (заполнить соответствующие столбцы таблицы 1, номера строк соответствуют вариантам). Сравните их со значениями из «Основной таблицы ASCII».

Варианты заданий:

Таблица 1

№ в-та	**Символ**	Код символа ASCII	Код символа UNICODE	Символ ASCII	Символ UNICODE	**Код символа (в 10 с/с)**
1.	**!!**					**42**
2.	**g**					**113**
3.	←					**20**
4.	**[**					**11**
5.	**)**					**67**
6.	**^**					**87**
7.	**M**					**38**
8.	**o**					**75**
9.	♫					**44**
10.	**▼**					**96**
11.	**§**					**60**
12.	**@**					**51**
13.	**#**					**84**
14.	**Z**					**17**
15.	**&**					**29**

2. СОЗДАНИЕ КОНСОЛЬНОГО ПРИЛОЖЕНИЯ. ВВОД И ВЫВОД ДАННЫХ

Цель: познакомить с назначением и возможностями среды разработки Visual Studio.NET, сформировать представление о создании консольных приложений в данной среде, об использовании методов пространства имен System, средствах ввода – вывода данных.

2.1. Консольный ввод-вывод

В языке C#, как и во многих других, нет операторов ввода и вывода. Вместо них для обмена с внешними устройствами применяются стандартные объекты. Для работы с консолью в языке C# применяется класс ***Console***, определенный в пространстве имен System.

2.1.1. Методы вывода

Методы класса Console для вывода данных – ***Write*** и ***WriteLine***.

Методы используются для вывода значений переменных и литералов различных встроенных типов. Метод ***Write*** выводит указанные данные на экран и оставляет курсор в конце выведенной строки. Метод ***WriteLine*** после вывода устанавливает курсор в начало следующей строки.

Если метод вывода вызван с одним параметром, то параметр может быть любого встроенного типа (Приложение 2), например, числом, символом или строкой.

Например,

```
double x=4.24;
Console.WriteLine(x);
```

Если требуется вывести несколько величин, их необходимо склеить в одну строку с помощью операции +. Параметры не строкового типа преобразуются в stringавтоматически.

```
Console.WriteLine("x = " + x);
```

Для управления форматированием числовых данных необходимо ввести информацию о форматировании:

```
WriteLine("строка_форматирования",arg0,arg1,...,argN);
```

В этой версии метода ***WriteLine()*** передаваемые ему аргументы разделяются запятыми. Элемент ***строка_форматирования*** содержит две составляющие: "постоянную" и "переменную". Постоянная составляющая представляет собой печатные символы, отображаемые на экране, а переменная состоит из спецификаторов формата. Общая форма записи спецификатора формата имеет следующий вид:

```
{номер_аргумента, ширина: формат}
```

Здесь элемент *номер_аргумента* определяет порядковый номер отображаемого аргумента, начиная с нулевого. Элемент *ширина* указывает ми-

нимальную ширину поля вывода, а *формат* задается элементом формата. Различные спецификации формата в применении к числовым данным представлены в таблице 1.

Таблица 1. Спецификации формата в применении к числовым данным.

Тип форматирования	Код формата		
Currency (денежные суммы)	C	C1	C7
Decimal (десятичный)	D	D1	D7
Exponential (экспоненциальный)	E	E1	E7
Fixedpoint (с фиксированной точкой)	F	F1	F7
General (общий)	G	Gl	G7
Number (числовой)	N	N1	N7
Percent (процент)	P	P1	P7
Hexadecimal (шестнадцатеричный)	X	X1	X7

Если при выполнении метода ***WriteLine***() в строке форматирования встречается спецификатор формата, вместо него подставляется и отображается аргумент, соответствующий заданному элементу *номер_аргумента*. Элементы *ширина* и *формат* указывать необязательно.

В спецификаторе *формат* часто используются *пользовательские шаблоны форматирования.* После двоеточия задается вид выводимого значения посимвольно, причем на месте каждого символа может стоять либо #, либо 0. Если указан знак #, на этом месте будет выведена цифра числа, если она не равна нулю. Если указан 0, будет выведена любая цифра, в том числе и 0.

При выводе можно использовать управляющие последовательности.

Пример 1. Фрагмент программы с форматным выводом:

```
double a=67.22;
Cosole.WriteLine("a={0,7:G}\nsin={1,10:E}",a,
Math.Sin(a));
Cosole.WriteLine("a={0,7}\nsin={1,10:#.####}",
a,Math.Sin(a);
```

На экране будет сгенерирован следующий результат:

67,22

a = 67,22

sin = -9,478925E-001

a= 67,22

sin= -,94789

2.1.2. Ввод с клавиатуры

В классе ***Console*** определены методы ввода строки и отдельного символа, но нет методов, которые позволяют непосредственно считывать с клавиатуры числа.

Для считывания строки символов используется метод ***ReadLine***.

Метод ***ReadLine()*** считывает символы до тех пор, пока не будет нажата клавиша <***Enter***>, и возвращает объект типа ***string***. При вводе вещественных чисел дробная часть отделяется от целой с помощью *запятой*. Допускается задавать числа в экспоненциальной форме.

Преобразование строки в число можно выполнить либо с помощью специального класса ***Convert***, определенного в пространстве имен ***System***, либо с помощью метода ***Parse***, имеющегося в каждом стандартном арифметическом классе.

Для считывания одного символа используется метод ***Read***. Считанный символ возвращается как значение типа **`int`**,представляющее собой код символа, или -1, если символов во входном потоке нет (например, пользователь нажал клавишу <***Enter***>). Затем это значение должно быть явно приведено к типу **`char`**.

Метод ***Read***считывает из буфера только один символ, и в отличие от***ReadLine***, не очищает буфер, поэтому следующий после него ввод будет выполняться с того места, на котором закончился предыдущий, т.е. будет читаться код клавиши <***Enter***>. Поэтому необходимо прочитать остаток строки методом ***ReadLine()***.

2.2. Явное преобразование типа

Тип результата в операции присваивания совпадает с типом левого операнда. Для правого операнда операции присваивания должно существовать неявное преобразование к типу левого операнда (Приложение 3). Когда неявного преобразования не существует, используется операция явного преобразования величины из одного типа в другой.

При преобразовании из более длинного типа в более короткий возможна потеря информации, если исходное значение выходит за пределы диапазона результирующего типа.

Формат операции:

```
(тип) выражение
```

Здесь `тип` – это имя того типа, в который осуществляется преобразование.

Пример.

```
longb = 300;
inta = (int) b;   // данные не теряются
byted = (byte) a;      // данные теряются
```

```
char h = (char)b;        // преобразование целого числа в символ
```

2.3. Стандартные методы

В C# стандартные методы можно использовать без предваряющего их описания. Математические функции реализованы в классе ***Math***, определенном в пространстве имен ***System***. (см. Приложение 4.1). Вызов функций осуществляется использованием ее в качестве операнда в выражениях.

Пример 2. Вычислить выражение:

$$y = k^2 + \frac{\ln(|a|)}{t^2 + 1};$$

Результаты вывести на экран.

Программа расчета по заданной формуле:

```
using System;
namespaceConsoleApplication1
{classClass1
 {static void Main()
  {stringbuf; //строка-буфердлявводачисел
  Console.WnteLine( "Введите k" );
  buf = Console.ReadLine(); // вводсклавиатуры
  double k = Convert.ToDouble(buf);
  //преобразование в вещественное число
  buf = Console.ReadLine();  //вводстроки
  double a = double.Parse(buf);
  Console.WriteLine( "Введите a" );
  buf = Console.ReadLine(); //вводстроки
  doublet = double.Parse( buf ); //преобразование в
                                 //вещественное число
  doubley=k*k+Math.Log(Math.Abs(a))/(t*t+1);
  Console.WriteLine("Для k={0}, a={1}и t={2}",k,a,t);
  Console.WriteLine("Результат = " +y );
  }
 }
}
```

Результат работы программы:

```
Для k=3, a=4,3и t=1,9
Результат = 9,18559804335151
```

Контрольные вопросы:

1. Основные принципы технологии .NET.
2. Что представляет собой платформа Visual Studio.NET?
3. Как создать консольное приложение?

4. Принципы объектно-ориентированного программирования.
5. Литералы. Как определяются типы литералов?
6. Какие типы относятся к встроенным?
7. Чем отличаются типы-значения и ссылочные типы?
8. Какие типы числовых переменных имеются?
9. Что такое объявление и инициализация?
10. Для чего используется упаковка и распаковка?
11. Как в C# выполняется преобразование типа?
12. Как осуществляется консольный ввод?
13. Чем отличаются методы `Read` и `ReadLine`?
14. Как обеспечить вывод данных на экран?
15. Для чего предназначен и как используется форматный вывод данных?
16. Каковы основные правила использования стандартных функций?
17. Основные приемы работы в среде разработки Visual Studio.NET:
18. Как создать консольное приложение?
19. Как сохранить проект с заданным именем?
20. Как загрузить проект?
21. Как просмотреть результаты выполнения программы?

ЛАБОРАТОРНАЯ РАБОТА № 2.
Знакомство со средой VISUAL STUDIO.NET. Создание консольного приложения.

Задание 1.

Для создания нового проекта выполните следующие действия:

1. Создайте новый проект**: File ► New ► Project**, либо **New**в окне **StartPage.**
2. В окне **New project** в левой части выбрать*Visual C# Projects,* в правой – пункт*Console Application.*
3. Загрузите файл «Среда разработки Visual Studio.NET. Консольные приложения».
4. Ознакомьтесь с основными окнами среды.
5. Измените имя проекта (*Solution*) на идентификатор, включающий фамилию, номер работы и номер варианта (например, IvanovAD_Lab2_v6).
6. Рассмотрите и определите назначение каждой строки заготовки программы.
7. Сохраните проект на диске в Вашей папке (В поле *Name* оставьтеимя проекта, в поле *Location* введите место его сохранения на диске).

Задание 2.

1. Наберите приведенный пример программы. Вставьте значения соответствующих типов в пропущенных местах операторов присваивания.

```
using System;
namespace matemat1
{
class Program
  {public static void Main(string[] args)
    //после знака = для x,y,z вставьте Ваши данные
{int x=  ;
double y=  ;
      Console.WriteLine("x: " + x);
    double b =  ;
      Console.WriteLine ("y+b: " + (y+b));
    }
  }
}
```

2. Сохраните весь проект на диске: ***File*** / ***SaveAll***
3. Для выполнения программы: ***Debug*** / ***StartWithoutDebugging***

Задание 3.

Создайте новое консольное приложение для решения задачи.

Присвойте значения вещественным числам `x, y, z` из области допустимых значений исходных данных. Вычислите `a, b.` Результаты выведите на экран.

Сохраните проект.

Варианты заданий:

1. $a=\dfrac{2tg(x)}{1-2\cos(3y)+tg(z)};\ b=(1+y)\dfrac{\sin x}{2}-\cos(4z);$

2. $a=\dfrac{3+e}{1+x^2};\ b=1+x^3+\dfrac{|y-x|^3}{3};$

3. $a=(1+y)^2\dfrac{x^2+4}{e^{-x}+x^2+4};\quad b=\dfrac{1}{x^4/2+\sin^4 z+1};$

4. $a=y+\dfrac{x^3}{y^2+\left|\dfrac{x}{y+3}\right|};\ \mathrm{b}=\left(1+tg^2 z\right);$

5. $a=\dfrac{\cos(x-\pi/6)}{x^2+1};\ b=1+\dfrac{z}{2x^3+y};$

6. $a=\dfrac{4+\sin(x+y)}{2+\left|x-1+x^2y^2\right|};\ \mathrm{b}=\cos\left(tg(z)\right);$

7. $a=\sqrt{|x|}\left(x-\dfrac{y^3}{z+x^2}\right);\ b=x-\dfrac{x^2}{3}+\dfrac{x^5}{5};$

8. $a=\dfrac{1+\sin(x+y)}{2+\left|\pi-1+\sin(x+y)\right|};\ b=x-\dfrac{2}{1+\sin(x+y)};$

9. $a=(y-\sqrt{|x|})\left(x-\sin(x+y)\right);\ b=\cos(z^2+x^2/4);$

10. $a=\dfrac{\sin(x)}{|x|+1};\ b=\dfrac{-\sqrt{|\sin x|}}{2+y^2+z^2};$

11. $a=\dfrac{10-y^3}{\sqrt{e^x+1}};\ b=\dfrac{\ln\left|x^2+1\right|}{|z+x+y|+1};$

12. $a=\dfrac{1}{1+\dfrac{x^2}{2}+\dfrac{y^4}{4}};\ b=x\left(tg(z)+e^y\right);$

13. $a=\dfrac{\sin x-y}{|x|+\cos^2 z+1};\ b=\dfrac{1-\sqrt{1+|\sin x|}}{2+y^2+z^{-2}};$

14. $a=\dfrac{10^x-y^3}{\sqrt{e^{z^2}}};\ b=\dfrac{\ln|x|+6^y\sqrt{e^z}}{|z|+10+\ln\dfrac{x^2+1}{y^4+3}};$

15. $a=(1+y)^{1/3}\dfrac{x+y\left(x^2+4\right)}{e^{-x-2}+1/\left(x^2+4\right)};\qquad b=\dfrac{1+\cos(y-2)}{x^4/2+\sin^4 z};$

Задание 4.

Измените программу:

1. Ввести данные с клавиатуры.
2. Для преобразования к числовой форме использовать класс ***Convert*** и метод ***Parse.***
3. Добавить варианты вывода результатов с использованием формата, шаблонов и управляющих символов.

3. УПРАВЛЯЮЩИЕ ОПЕРАТОРЫ. ПРОГРАММЫ С ВЕТВЛЕНИЯМИ

Цель: формирование умения использования управляющих операторов и выбора необходимых конструкций при реализации разветвляющихся алгоритмов.

3.1. Операторы if, if ... else ...

Структура условного оператора:

```
if (<условное выражение>) <оператор1>; [else<оператор2>];
```

<условное выражение> – произвольное выражение логического типа. Оператор (блок операторов) выполняется в случае истинности условного выражения. В противном случае выполняется оператор (блок операторов) после ключевого слова **else**. Часть оператора в квадратных скобках может отсутствовать.

Пример 1. Фрагмент программы с вложенными условными оперпторами:

```
if (i == 10)
{
 if (j<20) a=b;
  if(k >100) c=d;
  elsea=c;  //Эта else-инструкция относится к if(k>100)
}
elsea = d;// Эта else-инструкция относится к if(i==10)
```

3.2. Инструкция switch

Инструкция **switch** (переключатель) обеспечивает многонаправленное ветвление. Она позволяет делать выбор одной из множества альтернатив.

Общий формат записи инструкции **switch**:

```
switch(<выражение>)
  {case<константа1>: <оператор1>; break;
  case<константа2>: <оператор2>;break;
  case<константа3>: <оператор3>;break;
  [default:<оператор>;break;]
  }
```

Элемент<*выражение*>инструкции **switch** должен иметь целочисленный тип (например, **char**, **byte**, **short**, **int**) или тип **string**,**case**-константы должны быть литералами, тип которых совместим с типом заданного выражения. При этом никакие две **case**-константы в одной **switch**-инструкции не могут иметь идентичных значений.

Инструкция **switch** работает следующим образом. Значение выражения последовательно сравнивается с константами из заданного списка. При обнаружении совпадения для одного из условий сравнения выполняется последовательность инструкций, связанная с этим условием, до тех пор, пока не встретится инструкция **break**. Последовательность инструкций **default**-ветви выполняется в том случае, если ни одна из заданных **case-**констант не совпадет с результатом вычисления **switch**-выражения. Ветвь **default** необязательна. Если она отсутствует, то при несовпадении результата выражения ни с одной из **case**-констант никакое действие выполнено не будет.

Пример 2. С клавиатуры вводится первый номер месяца времени года, выдать сообщение о соответствующем времени года.

```
using System;
namespaceConsoleApplication1
{classClass1
 {static void Main()
  {stringbuf;
   Console.WriteLine( "Введитеномермесяца" );
   buf = Console. ReadLine();
   int m = Convert.ToInt( buf );
   switch (m)
   {1:Console.WriteLine("зима");break;
    3: Console.WriteLine("весна");break;
    6: Console.WriteLine("лето");break;
    9: Console.WriteLine("осень");break;
    defaultConsole.WriteLine
    ("неверныйномермесяца");break;
   }
  }
 }
}
```

Пример 3. Введите символ. Преобразуйте его к знаковому типу. Если результат отрицательный выведите его значение. Если положительный и является прописной буквой латинского алфавита, выведите букву, если знак препинания – сообщение иначе комментарий.

```
using System;
namespaceсимв_выбор
class SwitchDemo1
{publicstaticvoid Main()
   {Console.WriteLine("Введитесимвол");
    char c = (char)Console.Read();
    Console.ReadLine();
    Console.WriteLine("значениесимвола"+c);
```

```
    sbyte z =(sbyte)c;
    int c1=Math.Sign(z); // Знакчисла
    Console.WriteLine("значение числа "+z+"знак
    "+c1);
    switch (c1)
    {case -1:Console.WriteLine("значение отрицатель-
              но"+z);
     break;
     case 1 :
        if (c>='A'& c<= 'Z')
        Cosole.WriteLine("прописная латинская буква
        "+c);
        elseif (Char.IsPunctuation(c))
             Console.WriteLine("знакпрепинания" +c);
             elseConsole.WriteLine("другой " +c);
     break;
   }
  }
}
```

Контрольные вопросы:

1. Когда используют явное приведение типа?
2. Выражением какого типа является условие в операторе **if**? Какие значения оно может принимать?
3. Как работает оператор **if**, если отсутствует часть **else**<оператор 2>?
4. В каких случаях используется оператор **switch**?
5. Какого типа может быть<выражение>в операторе **switch**?
6. В каком случае выполняется последовательность инструкций **default**-ветви?

ЛАБОРАТОРНАЯ РАБОТА № 3
Управляющие операторы ветвления. Преобразование типов

Задание 1.

Создайте проект для решения задачи, используя условный оператор. На экран выводить исходные данные и результаты. В работе использовать только стандартные типы: числовые, символьный и булевский.

Варианты заданий:

1. По введенному целочисленному значению определите, является ли соответствующий ему символ строчной или прописной буквой русского алфавита. Вывести число, букву и комментарий.

2. Введите целое число. Если оно больше 255, вычислите корень квадратный из него, если меньше и соответствует отображаемым символам кодовой таблицы, выведите символ и его код, и комментарий для любого случая.

3. Определить какому алфавиту (латинскому или русскому) принадлежит введенный с клавиатуры символ. На экран вывести символ, его код и комментарий для любого случая.

4. Введите символ. Если он не является буквой латинского алфавита, замените его на знак ‘?’, иначе если прописная буква, то замените ее на строчную букву. Вывести введенный и преобразованный-символы, комментарий.

5. В компьютер поступают результаты двух спортсменов (время в секундах). По выбору пользователя (символ) выведите:

а) лучший результат

б) средний результат.

6. Введите целое число. Если оно является кодом отображаемого символа основной части таблицы ASCII, выведите символ и код, иначе выведите число и комментарий.

7. Введите 2 символа. Если символы – цифры, найдите и выведите их сумму и цифры, иначе выведите коды символов.

8. Введите символ. Если символ – цифра, выведите ее значение, увеличенное на единицу, если буква латинского алфавита, выведите следующий за ней символ, иначе код введенного символа. Выведите соответствующий комментарий.

9. Введите два вещественных числа. Меньшее из них соответствует нормативу площади на 1 человека, большее – общей площади здания. Вычислите максимально возможное количество человек, которых можно разместить в этом здании.

10. Определите номер строки и столбца основной кодовой таблицы для введенного символа. Если символ не принадлежит указанной части таблицы, то вывести соответствующий комментарий.

11. Введите символьное название времени суток (`a` – до полудня, `p` – после полудня) и время в часах и минутах. Выведите время в 24-часовом формате: ЧЧ.ММ.

12. Для введенного двузначного целого четного числа выведите соответствующий ему символ, для нечетного – символ, соответствующий сумме его цифр. Для неотображаемых символов выдать соответствующий комментарий.

13. Ввести два символа, сравнить их коды. Для большего вывести следующий символ, для меньшего – предыдущий. Для неотображаемых символов выдать соответствующий комментарий.

14. Для введенного символа вычислить его порядковый номер в английском алфавите. Вывести символ и его номер. Если символ не принадлежит этому алфавиту, выдать комментарий.

15. Для двух введенных символов найти разность кодов и соответствующий этой разности символ. Если полученный символ отображаемый, вывести его и значение кода на экран, иначе соответствующий комментарий.

Задание 2.

Составьте программу, используя инструкцию **`switch`**. Ввод данных осуществлять в строковую переменную. Для преобразования к числовым типам использовать класс **`Convert`** и метод **`Parse`**. Вывод результатов выполнить с использованием форматов и шаблонов.

Варианты заданий:

1. Вводится два целых числа: одно типа **`ushort`** и другое типа **`int`**. Найдите сумму этих чисел, предварительно преобразовав тип **`ushort`** в **`int`**. Определите, является ли результат цифрой и какой: нулем, четной или нечетной

2. Дано натуральное число N (< 20), определяющее сумму денег в рублях и целое k, определяющее начисленный процент (например, 5%). Измените N с учетом начисленных процентов. Вывести N и для этого числа наименование: "рубль", "рубля", "рублей".

3. Дано натуральное число G, определяющее год рождения человека, и целое число – текущий год. Определите возраст человека в годах, описав его типом**`uint`**. Вывести для этого числа наименование: "год", "года", "лет".

4. Вводится два числа: a типа **`ushort`** и b типа **`uint`** (b<=9). Найти остаток от деления a на b. Вывести название этого числа (0 – ноль, 1 – один, ..., 8 – восемь).

5. Вводится числа a типа **`ushort`** и b типа **`int`**. Уменьшить a на b. Если a лежит в диапазоне от 20 до 30, вывести название этого числа (21 – двадцать один, 22 – двадцать два, ...), иначе выдать сообщение, что число вне диапазона.

6. Вводится натуральное число в диапазоне от 2 до 101 и целое в диапазоне от -100 до -1. Найти сумму этих чисел и написать эффективный алгоритм вывода абсолютного значения полученного числа и его название (1 – один, 2 – два, ..., 100 – сто).

7. Вводится сумма в рублях (вещественное число < 1). Перевести его в копейки – целочисленный тип. Вывести полученное значение числа (1 – 99), дописав слово "копейка" в правильной форме (например, 12 копеек, 1 копейка).

8. Вводятся с клавиатуры вещественное число X типа **long**, и Y типа **float** и символ K$. В зависимости от значения K$ (+, - , * , / или % - остаток от целочисленного деления) вычислить и вывести соответствующий результат и комментарий.

9. Вводятся два числа: одно типа **int**, второе типа **double**, которые соответствуют сторонам прямоугольника, либо высоте и основанию треугольника, либо диагоналям ромба. Найти остаток от деления второго числа на 3, и в зависимости от полученного значения произвести расчет площади одной из названных фигур.

10. Вводится числа a типа **long** и b типа **int** (|b|<10). Вывести название этого числа |b| (0 – ноль, 1 – один, 2 – два, ..., 9 – девять) и частное от деления a на b с тремя знаками после запятой.

11. Вводится числа a и c типа short и b типа int, являющиеся коэффициентами квадратного уравнения. В зависимости от знака дискриминанта вывести сообщение о наличии и количестве корней, и значения корней, если они есть.

12. Определить для прописной буквы английского алфавита, является ли она гласной. Если буква гласная, определить букву алфавита, код которой на 1 больше кода для A, на 2 – для E и т.д. Выведите введенный символ и его код, результат и комментарий.

13. Вводятся символ и два целых числа. Если символ является знаком отношения, то выполнить соответствующее сравнение двух введенных чисел. Вывести сообщение, содержащее первое число, знак отношения, второе число и результат выполненного сравнения.

14. Выполните перевод из сантиметров, заданных натуральным число в футы и дюймы(1 фут = 12 дюймов, 1 дюйм = 2.54 см)и наоборот. При вводе величин укажите единицу измерения – ‘i’ для дюймов, ‘c’ для см, ‘f’ – для футов.

15. Водится с клавиатуры число S типа **ushort**и число X типа **int**, имеющие смысл номера сезона (зима – 1 и.т.д.) и количества дней, прошедших с начала сезона. Вывести соответствующие значения дня и месяца.

4. ЦИКЛИЧЕСКИЕ ПРОГРАММЫ

Цель: формирование умений выбора необходимых конструкций при реализации циклических алгоритмов, навыков применения операторов цикла языка C#.

4.1. Оператор цикла с предусловием

Общая форма цикла **while** имеет вид:

```
while (<условие>) <оператор>;
```

Работой цикла управляет элемент условие, который представляет собой любое допустимое выражение типа **bool**. Под элементом оператор понимается либо одиночная инструкция, либо блок инструкций. Инструкция выполняется до тех пор, пока условное выражение возвращает значение ИСТИНА. Как только это условие становится ложным, управление передается инструкции, которая следует за этим оператором.

Пример 1. Найти сумму ряда с точностью ε, общий член которого: $a_n = \frac{1}{n}$

```
using System;
 classWhileDemo
 {public static void Main()
  {doublee=0.0001;// точность ε
   doubles=0;//s – сумма
   inti=1;
   doublea=1/i;// a – слагаемое
   //подсчётсуммывциклеwhile
   while (a>=e){s += a; i++; a=(double)1/i;}
   Console.WriteLine("Ответ
   while{0,10:#0.00000}",s);
  }
 }
```

Результат работы программы:

Ответ while 9,78761

Если точность вычислений ε взять больше 1 (например, 1.2), то цикл не выполнится ни разу, и значение sостанется равным 0.

4.2. Оператор цикла с постусловием

Общий формат цикла с постусловием имеет вид:

```
do {<оператор>;} while (<условие>);
```

Цикл **do-while**выполняется до тех пор, пока остается истинным элемент условие. В отличие от цикла **while**, в котором условие проверяется при входе, цикл **do-while** проверяет условие при выходе из цикла. Это значит, что цикл **do-while** всегда выполняется хотя бы один раз.

Фигурные скобки необязательны, если элемент оператор состоит только из одной инструкции, но их желательно использовать для улучшения читабельности конструкции **do-while**.

Напишем программу для примера 1 с циклом **do-while**

```
using System;
 classDoWhile
 {public static void Main()
  {doublee=0.0001;
   doubles=0;//s – сумма
   inti=0;
   doublea=0;// a – слагаемое
   //подсчёт суммы в циклedo-while
   do{s += a; i++; a=(double)1/i;}
   while (a>=e);
   Console.WriteLine("Ответdo-
   while{0,10:0.00000}",s);
   }
  }
 }
```

Результат работы программы:

Ответ do-while9,78761

Обратите внимание на начальные значения переменных i и a. Если бы мы их не изменили, то для $\varepsilon > 1$ значение s было бы равно1, что неверно. Поэтому мы делаем сдвиг влево на одно слагаемое, начинаем i не с 1, а с 0.

4.3. Оператор цикла с параметром

Общий формат записи цикла с параметром имеет следующий вид:

for(<инициализация>;<условие>;<итерация>)<оператор>;

Элемент инициализация представляет собой инструкцию присваивания управляющей переменной цикла начального значения. Эта переменная действует в качестве счетчика, который управляет работой цикла.

Элемент условие представляет собой выражение типа **bool**, в котором тестируется значение управляющей переменной цикла.

Элемент итерация – это выражение, которое определяет, как изменяется значение управляющей переменной цикла после каждой итерации. Цикл f**or** будет выполняться до тех пор, пока условие истинно. Как только условие станет ложным, выполнение программы продолжится с инструкции, следующей за циклом **for**. Тело цикла может быть пустым.

Пример 2. Найти сумму чисел от 1 до 10.

```
usingSystem;
classClass3
 {public static void Main() {
    {ints = 0;
    // Суммируем числа от 1 до 10
    for (inti = 1; i<= 10; i++) s += i;
    Console.WriteLine("Суммаравна " + s);
    }
 }
```

Результат работы программы:
Сумма равна 55

Контрольные вопросы:

1. В чем отличие операторов **while и do ... while**
2. Что представляет собой элемент<инициализация>в операторе **for**?
3. Какого типа может быть элемент<условие>в цикле **for**?
4. Назначение управляющей переменной цикла **for**?
5. Назначение управляющих операторов **goto, break, continue, return**?
6. Как программируются циклические алгоритмы с явно заданным числом повторений цикла?

ЛАБОРАТОРНАЯ РАБОТА № 4.
Операторы циклов

Задание.

Вычислить сумму ряда, используя циклы **while**и **do-while.** Слагаемые, по модулю меньше заданного вещественного числа ε, не учитывать. Вычисление слагаемых выполнять, используя рекуррентные отношения. Сравнить результаты, полученные с использованием разных операторов цикла.

Варианты заданий:

1. $1+\frac{1+\cos x}{2}+\frac{1+\cos x+\cos 2x}{3}+\ldots$
2. $t+\frac{t-t^2}{2}+\frac{t-t^2+t^3}{4}+\ldots$, где $|t|<1$
3. $\sin x+\frac{\sin x^2}{1+2}+\frac{\sin x^3}{1+2+3}+\ldots$
4. $\frac{x}{\ln 2}-\frac{x^3}{\ln 2+\ln 3}+\frac{x^5}{\ln 2+\ln 3+\ln 4}+\ldots$, где $|x|<1$
5. $\sin x-\sin\sin x+\sin\sin\sin x-\ldots$
6. $a^x=1+\ln a\cdot x+\frac{(\ln a)^2}{2!}\cdot x^2+\frac{(\ln a)^3}{3!}\cdot x^3+\ldots$
7. $\frac{x-1}{x^2}+\frac{(x-1)^2}{2x^3}+\frac{(x-1)^3}{3x^4}+\ldots$, где $x>1$
8. $\frac{\cos x}{x}+\frac{\cos^2 x}{3x}+\frac{\cos^3 x}{5x}+\ldots$
9. $\sqrt{1+x}=1+\frac{1}{2}x-\frac{1}{2\cdot 4}x^2+\frac{1\cdot 3}{2\cdot 4\cdot 6}x^3-\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 8}x^4+\ldots$, где $|x|<1$
10. $\frac{x}{2}-\frac{x-2x^2}{4}+\frac{x-2x^2+3x^3}{8}-\ldots$, где $|x|<1$
11. $1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-\ldots$, где $|x|<1$
12. $\frac{1}{x}-\frac{1}{2\cdot 3x^3}+\frac{1\cdot 3}{2\cdot 4\cdot 5x^5}-\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7x^7}+\ldots$, где $|x|>1$
13. $x^2\left(\frac{1}{1!}+\frac{1}{2!}\right)-x^4\left(\frac{1}{2!}+\frac{1}{4!}\right)+x^6\left(\frac{1}{3!}+\frac{1}{6!}\right)-\ldots$
14. $\frac{x}{2}-\frac{x^3}{6}+\frac{x^5}{10}-\ldots$, где $|x|<1$
15. $-\cos x+\frac{\cos 2x}{2^2}-\frac{\cos 3x}{3^2}+\ldots$

5. МАССИВЫ

Цель: усвоение основных приемов работы со структурированными переменными типа массив, формирование представлений о различных алгоритмах обработки данных этого типа.

Массивом называется конечная именованная последовательность однотипных величин. В C# массивы могут быть одномерными или многомерными. Положение каждого элемента в массиве определяется индексом.

5.1. Одномерный массив

Одномерный массив является линейной таблицей, в которой для точного указания на элемент данных достаточно знания только одного индекса.

Для объявления одномерного массива используется следующая форма записи:

```
<тип>[] <имя_массива>( = new<тип> [<размер>]);
```

Здесь с помощью элемента `тип` объявляется базовый тип массива. Выражение в круглых скобках может отсутствовать.

Количество элементов, которые будут храниться в массиве, определяется элементом записи `размер` (может быть выражение). Поскольку массивы реализуются как объекты, то сначала объявляется ссылочная переменная на массив, а затем для него выделяется память, и переменной массива присваивается ссылка на эту область памяти. Таким образом, в C# массивы динамически размещаются в памяти с помощью оператора **new**.

Массивы можно инициализировать в момент их создания. Формат инициализации одномерного массива имеет следующий вид:

```
<тип>[] <имя_массива> = {val1, val2,...,valN};
```

Здесь начальные значения, присваиваемые элементам массива, задаются с помощью последовательности `val1–valN`. Область памяти для массива выделяется автоматически в соответствии с заданными значениями инициализации (инициализаторами). В этом случае нет необходимости использовать в явном виде оператор **new**, однако он может использоваться.

Примеры возможных вариантов инициализации массива:

```
int[] n = {99,10,100,18,78,23,63,9,8,9};

int[] n = newint[]{99,10,100,18,78,23,63,9,8,9};
int [] n;
n = newint[]{99,10,100,18,78,23,63,9,8,9};
```

```
int[] n = newint[10]{99,10,100,18,78,23,63,9,8,9};
```

В последнем варианте размер должен соответствовать количеству инициализаторов.

Доступ к элементу массива осуществляется указанием индекса в квадратных скобках после имени массива, например: `n[6]`. Нумерация индексов начинается с 0.

5.2. Многомерные массивы

Многомерным называется такой массив, который характеризуется двумя или более измерениями, а доступ к отдельному элементу осуществляется посредством двух или более индексов.

Простейший многомерный массив – двумерный.

Для объявления двумерного массива используется следующая форма записи.

```
<тип>[,] <имя_массива>( = new<тип> [разм1,разм2]);
```

Элемент `разм1` соответствует количеству строк, `разм2` – количеству столбцов.

Чтобы получить доступ к элементу двумерного массива, необходимо указать оба индекса, разделив их запятой.

Например:

```
int [,] table = newint[10, 20];
table [3,5] = 10;
```

Пример 1. Программа заполняет двумерный массив числами от 1 до 12, а затем выводит содержимое этого массива.

```
using System;
 classTwoD
 {publicstaticvoid Main()
  {int t, i;
    int[,] table = newint[3, 4];
    for(t=0; t < 3; ++t) {
     for(i=0; i< 4; ++i) {
      table[t,i] = (t*4)+i+1;
      Console.Write(table[t,i] + " ") ;
     }
    Console.WriteLine();
    }
  }
 }
```

В этом примере элемент массива table [0,0] получит число 1, элемент table [0,1] –число 2, и т.д.

В общем случае многомерный массив объявляется так:

```
<тип> [,..,] <имя>= new<тип>[размер1,...,размерN];
```

Многомерный массив можно инициализировать, заключив список инициализаторов каждой размерности в собственный набор фигурных скобок.

Формат инициализации двумерного массива:

```
<тип>[,] <имя_массива> = {
{val, val, val, ..., val},
{val, valval, ..., val},
{val, val, val, ..., val}
};
```

Здесь элемент `val`–значение инициализации. Каждый внутренний блок означает строку.

Пример 2. Программа инициализирует массив `s` числами от 1 до 5 и квадратами этих чисел.

```
using System;
 class Squares {
  public static void Main() {
   int[,] s= //объявляем двумерный массив с инициализацией
   {
      { 1, 1 },
      { 2, 4 },
      { 3, 9 },
      { 4, 16 },
      { 5, 25 }
   }
   for(int i=0; i< 5; i++) {
    for(int j=0; j < 2; j++)
     Console.Write(s[i,j] + " " ) ;
    Console.WriteLine();
   }
  }
 }
```

Результаты выполнения этой программы:

```
11
2 4
3 9
4 16
525
```

Контрольные вопросы:

1. Что понимается под массивом?
2. Каковы возможные способы описания массивов (одномерных и многомерных)?
3. Как выделяется память для хранения массива?

4. Какие формы инициализации массива?
5. Какие типы допустимы для описания индексов массивов?
6. Какие типы могут использоваться в качестве базовых для описания массивов?
7. Как осуществляется ввод и вывод массивов?

ЛАБОРАТОРНАЯ РАБОТА № 5
Обработка массивов

Задание 1. Составьте программу для решения следующей задачи обработки одномерных массивов произвольной длины. Выведите на монитор исходные данные и результат.

Вариантызаданий:

1. Заданы два одномерных массива различных размеров. Объединить их в один массив, включив второй массив между K-м и (K+1)-м элементами первого (K задано),
2. Найти среднее арифметическое заданного массива размером M. Преобразовать исходный массив, вычитая из каждого элемента среднее значение.
3. Найти количество перемен знака в массиве из N чисел. Нулевые элементы заменить абсолютным значением предыдущего. Если первый элемент нулевой, то заменить его числом F.
4. Задан массив из K чисел, составить программу замены нулей полусуммой следующего и предыдущего чисел. На место последнего или первого нуля ставить соответственно предыдущее или последующее число.
5. Найти минимальный элемент среди положительных чисел.
6. Удалить из массива целых чисел размером P все четные числа, стоящие на нечётных местах, сдвинув оставшиеся в начало массива.
7. Удалить из массива вещественных чисел все максимальные элементы, сдвинув оставшиеся влево на освободившиеся места, справа записать нули.
8. Найти среднее арифметическое элементов, расположенных между первым чётным и последним нечётным числом, встречающимся в целочисленной линейной таблице.
9. В заданный массив целых чисел из N элементов вставить элемент, равный M после последнего минимума.
10. Задан массив целых чисел из T элементов. Найти первую пару соседних противоположных чисел (их сумма равна 0).
11. В заданном массиве вещественных чисел найти наибольшую длину цепочки стоящих рядом знакочередующихся элементов.

12. Исключить из заданной целочисленной таблицы из M элементов числа k-го десятка, на их место сдвинуть элементы влево.

13. Найти количество различных элементов в заданном массиве вещественных чисел.

14. В целочисленном одномерном массиве длины K найти индекс элемента, равного заданному числу. Если таковых несколько, то индексы сохранить в массиве.

15. Даны два одномерных массива различных размеров. Упорядочить их по убыванию, получить из них один упорядоченный массив.

Задание 2. Составьте программу для решения следующей задачи обработки двумерных массивов произвольного размера. Выведите на монитор исходные данные и результат.

Варианты заданий:

1. Элементы столбца матрицы с максимальным по модулю элементом в *k*-ой строке заменить на число *X*.
2. Переставить *i*-ую и *j*-ую строки матрицы.
3. Упорядочить *m*-ую строку по невозрастанию элементов и записать ее в одномерный массив. Вывести исходный массив и полученный вектор.
4. Из заданной матрицы удалить *k*-ую строку и*l*-ый столбец.
5. Из матрицы *B(m,n)* сформировать матрицу *C(m,n)*, каждый элемент которой получается путем вычитания из соответствующего элемента матрицы *B* первого элемента *k*-ой строки.
6. Задана матрица размером $m \times n$. Просуммировать элементы, расположенные на главной и побочных (соседних с главной) диагоналях. Результат получить в виде вектора.
7. В нечетных строках матрицы найти минимальный элемент и количество повторений в этих строках.
8. В данной действительной матрице найти суммы элементов строк, в которых расположен элемент с наименьшим значением.
9. Найти строки матрицы с наибольшей и наименьшей суммой элементов. Сформировать из этих строк вектор.
10. Задан двумерный массив размерности $m \times n$. Дополнить его строкой и столбцом, в которых записать суммы элементов соответствующих строк и столбцов исходного массива. В элементе (*m+1,n+1*) должна храниться сумма всех элементов первоначального массива.
11. Целочисленная двумерная матрица разделяется главной диагональю на два треугольника. Из элементов верхнего и нижнего треугольников сформировать вектора.
12. В заданной двумерной матрице замените строки, содержащие максимальный элемент, на соответствующие строки единичной матрицы.

13. Заданы натуральное четырёхзначное число и двумерный массив, элементами которого являются натуральные четырехзначные числа. Определить, имеется ли в таблице число с обратным порядком расположения цифр по отношению к данному. Найти его индексы.

14. Дана двумерная таблица. Среди её элементов найти первый, удовлетворяющий свойству: абсолютная величина этого элемента превосходит сумму всех предшествующих ему в строке. Вывести элемент и его индексы.

15. Задана двумерная таблица из вещественных чисел. Сформировать вектор из элементов таблицы, принадлежащих данному диапазону.

6. СТРОКИ

Цель: освоить приемы использования строковых переменных для обработки текстовой информации.

Строки в C# предназначены для работы со строками символов в кодировке ***Unicode***, являются встроенным типом ***C#***. Ключевое слово типа – **string.** Ему соответствует базовый класс ***System.String*** библиотеки **. NET.**Таким образом, **string** – это ссылочный тип. Строки типа **string** относятся к так называемым неизменяемым типам данных

6.1. Создание строк

Создать строку можно несколькими способами:

```
string<имя_строки>;    // инициализация отложена
```

Самый простой способ создать объект типа **string** – использовать строковый литерал. После выполнения инструкции

```
string<имя_строки>=<строковый литерал>;
```

будет объявлена ссылочная переменная типа **string**, которой присваивается ссылка на строковый литерал.

Например,

```
stringstr = "test";
```

Можно также создать **string**-объект из массива типа **char**.

Пример 1.

```
char[]  ca = {'t','e','s','t'};// массив для инициализации строки
stringstr = newstring(ca);
```

После создания **string**-объект можно использовать везде, где разрешается использование строки символов, заключенной в кавычки.

6.2. Работа со строками

Для строк определены следующие операции:

- присваивание (=);
- проверка на равенство (==);
- проверка на неравенство (!=);
- обращение по индексу ([]);
- сцепление (конкатенация) строк (+).

Несмотря на то, что строки являются ссылочным типом данных, на равенство и неравенство проверяются не ссылки, а значения строк. Строки равны, если имеют одинаковое количество символов и совпадают посимвольно.

Как и у массивов, индексация символов в строке начинается с нуля. Обращаться к отдельному элементу строки по индексу можно только для получения значения, но не для его изменения, так как строки типа **string** являются неизменяемым типом данных.

В классе ***System.String*** предусмотрено множество методов, полей и свойств (Приложение 4.4).

Методы, изменяющие содержимое строки, на самом деле создают новую копию строки. Неиспользуемые «старые» копии автоматически удаляются сборщиком мусора.

Пример 2. Написать программу, которая заменяет заданную с клавиатуры последовательность символов в тексте на многоточие и подсчитать количество замен.

```
using System;
namespace ctroki_primer
{
  class Program
  {
   static void Main(string[] args)
   {string s, str;  // str – текст, s – последовательность символов
    int dl, j, n=0;
    Console.WriteLine("Введитетекст");
    str = Console.ReadLine();
    Console.WriteLine("Какую последовательность за-
      менять?");
    s = Console.ReadLine();
    dl = s.Length;
    j = 0;
    //подсчет количества вхождений подстроки в строку
    while(j <str.Length - dl+1){
     if(str.Substring(j,dl) == s) { n++; j += dl;}
     elsej++;
```

```
    }
    str = str.Replace(s, "...");//замена подстроки на...
    Console.WriteLine(str);
    Console.WriteLine("числозамен"+n);
   }
  }
}
```

Пример 3. Задан массивстрок. Изменить отдельные элементы массива. Преобразоватьмассиввстроку.

```
using System;
  classStringArrays
   {publicstaticvoid Main() {
      string[] str =
      {"Это","очень","простой","тест"};
      Console.WriteLine("Исходныймассив: " );
      for(inti=0; i<str.Length; i++)
        Console.Write (str[i] + " " );
      Console.WriteLine("\n");
      str[1] = "тоже";      //изменяем строку
      str[3] = "тест, не так ли?";
      Console.WriteLine("Измененныймассив: ");
      for(inti=0; i<str.Length;i++)
        Console.Write (str[i] + " ");//выводэлементов
                                     // массива через пробел
        Console.WriteLine();
        //склеиваниеэлементовмассивавстроку
        strings=string.Join(" ",str,0,str.Length);
        Console.WriteLine(s);
    }
}
```

После выполнения этой программы получаем такие результаты:

Исходный массив:

Это очень простой тест

Модифицированный массив:

Это тоже простой тест, не так ли?

Это тоже простой тест, не так ли?

Ссылочные переменные типа **string** могут менять объекты, на которые они ссылаются. А содержимое созданного **string** – объекта изменить уже невозможно.

Контрольные вопросы:

1. Чем являются строки в C#?

2. Какая информация может быть представлена с помощью строк? Каковы основные правила их описания?

3. Как осуществляется ввод и присваивание значений строковым переменным?

4. Какие операции определены для строк?

5. Как создаются строки?

6. Можно ли изменять значение строки?

ЛАБОРАТОРНАЯ РАБОТА № 6
Строки

Задание.

Дана символьная строка. Слово – последовательность символов между пробелами, не содержащая пробелы внутри себя. Составьте программу для решения следующей задачи:

Варианты заданий:

1. Разбить текст на строки длиной в два слова. Перенос на новую строку осуществить на месте пробела.
2. Расположите слова данного предложения в порядке возрастания числа букв в словах.
3. Определить количество и вывести все самые длинные слова.
4. Определить длину слова, стоящего на N-ом месте и вывести все слова, состоящие из такого же количества символов, что и найденное слово. Если N больше количества слов в предложении, то вывести соответствующее сообщение.
5. Определить количество слов, которые начинаются и оканчиваются одним и тем же символом.
6. Вывести все слова, в которых есть заданный символ.
7. Для каждого из слов указать, сколько раз оно встречается в данной строке.
8. Даны две символьные строки. Вывести слова, которые встречаются в обеих строках.
9. Отредактировать заданное предложение, удаляя из него слова, которые уже встречались.
10. Найти слова, в которых нет повторяющихся букв.
11. Подсчитать количество слов, имеющих удвоенные буквы.
12. Расположить слова данного предложения в алфавитном порядке по первой букве.
13. Отредактировать заданное предложение, удаляя из него слова с нечетными номерами и переворачивая слова с четными (например, howdoyoudo -->odod).

14. Определить, есть ли в данной последовательности два любых одинаковых слова, и вывести эти слова и соответствующее сообщение.

15. Удалить среднюю букву в словах нечетной длины, и вставить посередине пробел в ином случае.

7. ОБЪЯВЛЕНИЕ И ОПРЕДЕЛЕНИЕ МЕТОДОВ

Цель: формированиеуменийприменять методы при реализации алгоритмов сложной структуры, усвоение понятий формальных и фактических параметров и способов их передачи между вызывающей и вызываемой функциями.

7.1. Объявление методов

Метод – это функциональный элемент класса, который реализует вычисления или другие действия, выполняемые классом или экземпляром. Методы определяют поведение класса.

Метод представляет собой законченный фрагмент кода, к которому можно обратиться по имени. Он описывается один раз, а вызываться может столько раз, сколько необходимо. Один и тот же метод может обрабатывать различные данные, переданные ему в качестве аргументов.

Синтаксис метода:

```
<тип><имя_метода> ([<параметры>])
                  {тело_метода}
```

Первая строка представляет собой заголовок метода. `Тело метода`, задающее действия, выполняемые методом, чаще всего представляет собой блок.

`Тип` определяет, значение какого типа вычисляется с помощью метода. Это значение, как правило, возвращается с помощью оператор `ereturn` в теле метода в то место вызывающей функции, откуда был вызван метод. Если метод не возвращает никакого значения, в его заголовке задается тип **`void`**, а оператор `return` отсутствует.

`Параметры`, описываемые в заголовке метода, определяют множество значений *аргументов,* которые можно передавать в метод. Параметр представляет собой локальную переменную, которая при вызове метода принимает значение соответствующего аргумента. Область действия параметра – весь метод. Для каждого параметра должны задаваться его *тип* и *имя.*

При вызове метода аргументы должны записываться в том же порядке, что и в заголовке метода, и должно существовать неявное

преобразование типа каждого аргумента к типу соответствующего параметра. При несоответствии типов выдается диагностическое сообщение.

Метод, не возвращающий значение, вызывается отдельным оператором, а метод, возвращающий значение, – в составе выражения.

7.2. Параметры методов

Существуют два способа передачи параметров: по значению и по ссылке.

При передаче по значению метод получает копии значений аргументов, и операторы метода работают с этими копиями. Доступа к исходным значениям аргументов у метода нет, а следовательно, нет и возможности их изменить. ***При передаче по ссылке (по адресу)*** метод получает копии адресов аргументов, он осуществляет доступ к ячейкам памяти по этим адресам и может изменять исходные значения аргументов, модифицируя параметры.

В C# для обмена данными между вызывающей и вызываемой функциями предусмотрено четыре типа параметров:

– параметры-значения. Описывается в заголовке метода следующим образом:<тип><имя>

– параметры-ссылки – описываются с помощью ключевого слова**ref**: **ref**<тип><имя>.Параметры-ссылки используются, если в методе требуется изменить значение каких-либо передаваемых в него величин. При вызове метода на месте параметра-ссылки может находиться только ссылка на инициализированную переменную точно того же типа.

– выходные параметры - описываются с помощью ключевого слова **out**. Параметру, имеющему этот спецификатор, должно быть обязательно присвоено значение внутри метода. В вызывающем коде можно использовать переменную без инициализации.

– параметры-массивы – описываются с помощью ключевого слова **params**. Параметр-массив может быть только один и должен располагаться последним в списке.

Ключевое слово предшествует описанию типа параметра. Если оно опущено, параметр считается параметром-значением.

7.3. Использование методов

Пример 1. Параметры-значения, параметры-ссылки, выходные параметры

```
using System;
namespaceConsoleApplicationl
{classClassl
```

```
 {static void P(int a,
 ref int b, out intc)
  {a = 44; b = 33; c=55;
    Console.WriteLine("внутриметода {0} {1} {2}",
    a,b,c);
   }
  static void Main()
   {int a = 2, b = 4,
   c;
     Console.WriteLine("довызова {0} {1}",a, b);
     P(a, ref b, out c);
     Console.WriteLine("послевызова {0} {1} {2}",
     a, b, c);
   }
  }
}
```

Пример 2. Введите символ. Преобразуйте его к знаковому типу. Если результат отрицательный, выведите его значение. Если положительный и является прописной буквой латинского алфавита, выведите букву, если знак препинания – сообщение, иначе комментарий.

Пример 3. Для каждого числа в заданном диапазоне найти наименьший множитель: 2,3,5 или 7

```
using System;
namespace primer_7
{classClass1
 {staticintdelit(intn) //методнахождениямножителя
   {if((n % 2) == 0) return 2;
   elseif((n % 3) == 0) return 3;
   elseif((n % 5) == 0) return 5;
   elseif((n % 7) == 0) return 7;
   else return -1;
   }
  publicstatic void Main()
   {stringbuf;
   buf=Console.ReadLine();
   int m=int.Parse(buf);          // нижняяграницадиапазона
   buf=Console.ReadLine();
   int k=Convert.ToInt32(buf); //верхняяграницадиапазона
   Console.WriteLine("Нижняяграницадиапазона" +m);
   Console.WriteLine("Верхняяграницадиапазона"+k);
   for(intnum = m; num< k; num++)
     {int d=delit(num);
     if (d!=-1)
```

```
        Console.WriteLine("Наименьший множитель числа
        " +num + " равен "+ d);
        else Console.WriteLine(num + "не делится на
        2,3,5 или 7");
        }
     }
   }
}
```

Результат работы программы:
Нижняя граница диапазона 23
Верхняя граница диапазона 32
23 не делится на 2, 3, 5 или 7.
Наименьший множитель числа 24 равен2
Наименьший множитель числа 25 равен5
Наименьший множитель числа 26 равен2
Наименьший множитель числа 27 равен3
Наименьший множитель числа 28 равен2
29 не делится на 2, 3, 5 или 7.
Наименьший множитель числа 30 равен2
31 не делится на 2, 3, 5 или 7

Контрольные вопросы:

1. Что представляют собой методы?
2. Как объявляется метод?
3. Какие типы можно использовать для описания значения функции?
4. Назначение управляющего оператора `return`.
5. Какова область действия параметров метода?
6. Как вызываются методы?
7. Какая существует связь между формальными и фактическими параметрами?
8. В чем особенности механизма передачи параметров по значению и по ссылке?

ЛАБОРАТОРНАЯ РАБОТА № 7
Программирование с использованием методов

Задание 1. Составьте программу с использованием типизированного метода с параметрами. Выполните ее тестирование и отладку.

Варианты заданий:

1. Для заданного вектора $(x_1, x_2, ..., x_n)$ вычислить величину:

$$N = \begin{cases} 0, & если \quad x_1 \neq 0 \\ n, & если \quad x_1 = x_2 = ... = x_n \\ m, & если \quad x_1 = x_2 = ... = x_m = 0, \quad x_{m+1} \neq 0 \quad (m < n) \end{cases}$$

2. Вычислить z – сумму значений функции:

$$z = f(|x|,y)+f(a,b)+f(|x|+1,-y)+f(|x|-|y|,x)+f(x+y,a+b),$$

$$\text{где } f(u,t) = \begin{cases} u + 2t & u \geq 0; \\ u + t & u \leq -1; \\ u^2 - 2t + 1, & -1 < u < 0. \end{cases}$$

3. Найти наименьшую сторону треугольника, заданного координатами своих вершин.

4. По заданному значению величины x определить, имеет ли функция

$$y(x) = \frac{1}{\sqrt{x-1}} + \frac{1}{x^2 - 8x + 12}$$

значение, и найти его.

5. Вычислить значение функции $y=ax^2+bx+d$, где

$$a = \sum_{i=1}^{n} t_i; \; b = \sum_{i=n+1}^{100} t_i; \; d = \sum_{i=1}^{20} q_i; \; 1 < n < 100;$$

используя функцию $sum = \sum_{i=k}^{l} mas_i$.

6. Вычислить корни x и y системы уравнений:

$$\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}.$$

7. Вычислить величину:

$$z = \frac{sh^2 a + sh(a-b)}{sh\, a + \sqrt{sh(a^2 - b^2)}},$$

используя функцию $sh\, x = \frac{e^x - e^{-x}}{2}$.

8. Даны действительные числа s и t, получить значение выражения:

$$h(s,t) + \max(h^2(s-t, st), h^4(s-t, s+t)) + h(1,1),$$

$$\text{где } h(a,b) = \frac{a}{1+b^2} + \frac{b}{1+a^2} - (a-b)^3.$$

9. Найти сумму ряда с заданной точностью ε, общий член которого имеет вид:

$$a_n = \frac{2n-1}{2^n}$$

10. Вычислить приближённое значение кубического корня из*a*, используя соотношение:

$$x_i = 2x_{i-1}/3 + a/3x_{i-1}^2,\ x_1 = 1.$$

11. Дано натуральное число *n*. Выяснить, является ли оно полным квадратом. Определить функцию, позволяющую распознавать полные квадраты.

12. Дано натуральное число. Найти количество цифр в числе.

13. Найти в диапазоне *[a, b]* первое число, кратное *c* и *d*.

14. Найти количество отрицательных значений функции $y = x^2 - a*b+c*x$ на отрезке от x_1до x_2 с шагом *h*. Параметры *a, b, c* заданы.

15. Дано*n*-значное целое число, записать его цифры в обратном порядке.

Задание 2. Создайте проект, в котором опишите метод обработки одномерных массивов. Результаты работы метода передавать параметрами. Напишите тестирующую программу с выдачей результатов на экран.

Варианты заданий:

1. Найти наименьший и наибольший элементы целочисленного массива.

2. "Сжать" числовой одномерный массив, удалив из него заданный элемент *b*. Элементу *b* присвойте значение длины массива.

3. Найти сумму и количество положительных элементов в целочисленном массиве, расположенных между заданными числами *a* и *b*.

4. Объединить два упорядоченных массива в один упорядоченный и возвратить размер объединенного массива.

5. Найти в заданном массиве натуральных чисел первый палиндром (число, которое читается одинаково как с начала, так и с конца, например, 313, 66), и определите количество палиндромов.

6. Найти минимальный элемент среди положительных чисел массива и максимальный среди отрицательных.

7. В целочисленном массиве нулевые элементы заменить заданным числом, найти количество перемен знака в полученном массиве и присвоить его этому числу.

8. В целочисленном одномерном массиве найти число, повторяющееся наибольшее количество раз. Если таковых несколько, то сохранить их в массиве.

9. В заданном массиве вещественных чисел найти наибольшую длину цепочки стоящих рядом знакочередующихся элементов и количество отрицательных.

10. Упорядочить массив по убыванию, передать параметрами первый и последний элемент.

11. Уменьшить размерность массива на заданное число *n*, присвойте переменной *n* значение нового размера массива и затем замените все элементы, следующие за первым отрицательным элементом значением *n*.

12. Найти сумму нечетных положительных элементов целочисленного массива и количество четных.

13. Найдите наименьший положительный элемент среди элементов с четными номерами массива и его номер.

14. Определить количество элементов, меньших заданного числа m и первый среди них в массиве.

15. Вычислить $z = \frac{x_{\max} - y_{\min}}{2}$, где x_{max}– максимальный элемент массива $X(n)$; y_{min}– минимальный элемент массива $Y(m)$. Элементы x_{max} и y_{min} вычислять одним методом.

ЛИТЕРАТУРА

1. Бен Ватсон C# 4.0 на примерах – СПб.: БХВ-Петербург, 2011, 608 с.
2. Биллиг, В.А. Основы программирования на C#. – М.: Изд-во «Интернет-университет информационных технологий – ИНТУИТ.ру», 2006. – 488 с.
3. Ватсон, К.C#. – М.: Лори, 2004. – 880 с.
4. Вирт, Н. Алгоритмы и структуры данных. – СПб: Невский диалект, 2001. – 352 с.
5. Гуннерсон, Э. Введение в C#. Библиотека программиста. – СПб.: Питер, 2001. – 304 с.
6. КристианНейгел, Билл Ивьен, ДжейГлинн, Карли Уотсон, Морган Скиннер. C# 4.0 и платформа .NET 4 для профессионалов – М.: Диалектика, Вильямс, 2011 – 1440 с.
7. Либерти, Д. Программирование на C#. – СПб.: Символ-Плюс, 2003. – 688 с.
8. Майо, Д.C#. Искусство программирования: Энциклопедия программиста. – Киев: ДиаСофт, 2002. – 656 с.
9. Майо, Дж. C# Builder. Быстрый старт. – М.: Бином, 2005. – 384 с.
10. Мартынов, Н.Н. C# для начинающих – М.: КУДИЦ-ПРЕСС, 2007, – 272 с.
11. Микелсен, К. Язык программирования C#. Лекции и упражнения: Учебник. – Киев: ДиаСофт, 2002. – 656 с.
12. Павловская, Т.А. C#. Программирование на языке высокого уровня. Учебник для вузов – СПб.: Питер, 2007. – 432 с.
13. Уотсон К., Нейгел К. Visual C# 2010. Полный курс – М.: Вильямс, 2011 – 955 с.
14. Эндрю Троелсен Язык программирования C# 2010 и платформа . NET 4 – М.: Вильямс, 2011 – 1392 с.
15. Мэтью Мак-Дональд WindowsPresentationFoundation в .NET 4.0 с примерами на C# 2010 – М.: Вильямс, 2011 – 1020 с.
16. Н. КультинMicrosoftVisualC# в задачах и примерах – СПб.: БХВ-Петербург, 2009 – 314 с.
17. Рихтер Дж. CLRviaC#. Программирование на платформе Microsoft. NETFramework 2.0 на языке C# : – СПб.: Питер, 2007. – 636 с.

ПРИЛОЖЕНИЯ

Приложение 1. Основная таблица ASCII

	0	1	2	3	4	5	6	7
0		►		0	@	P	`	p
1	☺	◄	!	1	A	Q	a	q
2	☻	↕	"	2	B	R	b	r
3	♥	‼	#	3	C	S	c	s
4	♦	¶	$	4	D	T	d	t
5	♣	§	%	5	E	U	e	u
6	♠	▬	&	6	F	V	f	v
7		↨	'	7	G	W	g	w
8		↑	(	8	H	X	h	x
9	○	↓	)	9	I	Y	i	y
10		→	*	:	J	Z	j	z
11	♂	←	+	;	K	[	k	{
12	♀	∟	,	<	L	\	l	|
13		↔	-	=	M	]	m	}
14	♫	▲	.	>	N	^	n	~
15	☼	▼	/	?	O	_	o	⌂

Приложение 2. Встроенные типы в C#

Встроенные типы не требуют предварительного определения. Они однозначно соответствуют стандартным классам библиотеки .NET, определенным в пространстве имен System.

Целые типы, а также символьный, вещественные и финансовый типы можно объединить под названием *арифметических типов.*

Название	Ключевое слово	Тип .NET	Диапазон значений	Описание	Размер битов
Логический тип	bool	Boolean	true, false		8
Целые типы	sbyte	SByte	От -128 до 127	Со знаком	8
	byte	Byte	От 0 до 255	Без знака	8
	short	Intl6	От -32768 до 32767	Со знаком	16

Название	Ключевое слово	Тип .NET	Диапазон значений	Описание	Размер битов
	ushort	UIntl6	От 0 до 65535	Без знака	16
	int	Int32	От $-2\cdot10^9$ до $2\cdot10^9$	Со знаком	32
	uint	UInt32	От 0 до $4\cdot10^9$	Без знака	32
	long	Int64	От $-9\cdot10^{18}$ до $9\cdot10^{18}$	Со знаком	64
	ulong	UInt64	От 0 до $18\cdot10^{18}$	Без знака	64
Символьный тип	char	Char	От U+0000 до U+FFFF	Unicode-символ	16
Вещественные	float	Single	От $-3.4\cdot10^{38}$до $3.4\cdot10^{38}$	7 цифр	32
	double	Double	От $\pm5.0\cdot10^{-324}$ до $\pm1.7\cdot10^{308}$	15-16 цифр	64
Финансовый тип	Decimal	Decimal	От $\pm1.0\cdot10^{-28}$ до $\pm7.9\cdot10^{28}$	28-29 цифр	128
Строковый тип	string	String	Длина ограничена объемом доступной памяти	Строка из Unicode-символов	
Тип object	object	Object	Можно хранить все что угодно	Всеобщий предок	

Приложение 3. Таблица явных числовых преобразований в C#

Из	Целевой тип
sbyte	byte, ushort, uint, ulong, char
byte	Sbyte, char
short	Sbyte, byte, ushort, uint, ulong, char
ushort	sbyte, byte, short, char
Int	sbyte, byte, short, ushort, uint, ulong, char
uint	sbyte, byte, short, ushort, int, char
long	sbyte, byte, short, ushort, int, uint, ulong, char
ulong	sbyte, byte, short, ushort, int, uint, long, char
char	sbyte, byte,short
float	sbyte, byte, short, ushort, int, uint, long, ulong, char, decimal
double	sbyte, byte, short, ushort, int, uint, long, ulong, char, float, decimal
decimal	sbyte, byte, short, ushort, int, uint, long, ulong, char, float,double

Приложение 4. Основные методы пространства имен System

4.1. Основные поля и статические методы класса Math

Имя	Описание	Результат	Пояснения
Abs	Модуль	Перегружен	записывается как Abs(x)
Acos	Арккосинус[2]	double	Acos(double x)
Asin	Арксинус	double	Asin(double x)
Atari	Арктангенс	double	Atan(double x)
Atan2	Арктангенс	double	Atan2(double x, double y)
Ceiling	Округление до большего целого	double	Ceiling(double x)
Cos	Косинус	double	Cos(double x)
Cosh	Гиперболический косинус	double	Cosh(double x)
DivRem	Деление и остаток	Перегружен	DivRem(x, y, rem)
E	База натурального логарифма	double	2,71828182845905
Exp	Экспонента	double	e^xзаписывается как Exp (x)
Floor	Округление до меньшего целого	double	Floor(double x)
IEEERemainder	Остаток от деления	double	IEEERemainder(double x, double y)
Log	Натуральный логарифм	double	$\log_e$x записывается как Log(x)
Logl0	Десятичный логарифм	double	$\log_{10}$x записывается как Logl0(x)
Max	Максимум из двух чисел	Перегружен	Max(x, y)
Min	Минимум из двух чисел	Перегружен	Min(x, y)
PI	Значение числа p	double	3,14159265358979
Pow	Возведение в степень	double	$\mathbf{x^y}$
Round	Округление	Перегружен	Округление до ближайшего целого.
Sign	Знак числа	int	Аргументы перегружены
Sin	Синус	double	Sin(double x)
Sinn	Гиперболический синус	double	Sinh(double x)
Sqrt	Квадратный корень	double	√x записывается как Sqrt(x)
Tan	Тангенс	double	Tan(double x)
Tanh	Гиперболический тангенс	double	Tanh(double x)

4.2. Основные элементы класса Array

Элемент	Вид	Описание
Length	Свойство	Количество элементов массива (по всем размерностям)
Rank	Свойство	Количество размерностей массива
BinarySearch	Статический метод	Двоичный поиск в отсортированном массиве
Clear	Статический метод	Присваивание элементам массива значений по умолчанию
Copy	Статический метод	Копирование заданного диапазона элементов одного массива в другой массив
CopyTo	Метод	Копирование всех элементов текущего одномерного массива в другой одномерный массив
GetValue	Метод	Получение значения элемента массива
IndexOf	Статический метод	Поиск первого вхождения элемента в одномерный массив
LastIndexOf	Статический метод	Поиск последнего вхождения элемента в одномерный массив
Reverse	Статический метод	Изменение порядка следования элементов на обратный
SetValue	Метод	Установка значения элемента массива
Sort	Статический метод	Упорядочивание элементов одномерного массива

4.3. Основные методы класса System.Char

Метод	Описание
GetNumericValue	Возвращает числовое значение символа, если он является цифрой, и -1 в противном случае
GetUnicodeCategory	Возвращает категорию Unicode-символа
IsControl	Возвращает true, если символ является управляющим
IsDigit	Возвращает true, если символ является десятичной цифрой
IsLetter	Возвращает true, если символ является буквой
IsLetterOrDigit	Возвращает true, если символ является буквой или цифрой
IsLower	Возвращает true, если символ задан в нижнем регистре
IsNumber	Возвращает true, если символ является числом (десятичным или шестнадцатеричным)
IsPunctuation	Возвращает true, если символ является знаком препинания
IsSeparator	Возвращает true, если символ является разделителем
IsUpper	Возвращает true, если символ записан в верхнем регистре

Метод	Описание
IsWhiteSpace	Возвращает true, если символ является пробельным (пробел, перевод строки и возврат каретки)
Parse	Преобразует строку в символ (строка должна состоять из одного символа)
ToLower	Преобразует символ в нижний регистр
ToUpper	Преобразует символ в верхний регистр
MaxValue, MinValue	Возвращают символы с максимальным и минимальным кодами (эти символы не имеют видимого представления)

4.4. Основные элементы класса System.String

Название	Вид	Описание
Concat	Статический метод	Конкатенация строк. Метод допускает сцепление произвольного числа строк
Copy	Статический метод	Создание копии строки
IndexOf(string str), LastIndexOf, IndexOfAny, LastlndexOfAny	Методы	Определение индексов первого и последнего вхождения заданной подстроки str или любого символа из заданного набора
Insert	Метод	Вставка подстроки в заданную позицию
Length	Свойство	Длина строки (количество символов)
PadLeft, PadRight	Методы	Выравнивание строки по левому или правому краю путем вставки нужного числа пробелов в начале или в конце строки
Remove(№, n)	Метод	Удаление подстроки из n символов с заданной позиции №
Replace	Метод	Замена всех вхождений заданной подстроки или символа новыми подстрокой или символом
StartsWith, EndsWith	Методы	Возвращает true или false в зависимости от того, начинается или заканчивается строка заданной подстрокой
Substring	Метод	Выделение подстроки, начиная с заданной позиции
ToCharArray	Метод	Преобразование строки в массив символов
ToLower, ToUpper	Методы	Преобразование символов строки к нижнему или верхнему регистру
Trim, TrimStart, TrimEnd	Методы	Удаление пробелов в начале и конце строки или только с одного ее конца (обратные по отношению к методам PadLeft и PadRight действия)

4.5. Основные методы класса System.Random

Название	Описание
Next ()	Возвращает целое положительное число во всем положительном диапазоне типа int
Next(макс)	Возвращает целое положительное число в диапазоне [0, макс]
Next(мин, макс)	Возвращает целое положительное число в диапазоне [мин, макс]
NextBytes(массив)	Возвращает массив чисел в диапазоне [0, 255]
NextDouble()	Возвращает вещественное положительное число в диапазоне [0, 1]

Учебное издание

ПОТАПОВА Людмила Евгеньевна
АЛЕЙНИКОВА Татьяна Григорьевна

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ C#

Методические рекомендации
к выполнению лабораторных работ

Технический редактор *Г.В. Разбоева*
Компьютерный дизайн *И.В. Волкова*

Подписано в печать 14.01.2014. Формат 60x84 $^{1}/_{16}$. Бумага офсетная.
Усл. печ. л. 2,91. Уч.-изд. л. 1,70. Тираж 65 экз. Заказ1.

Издатель и полиграфическое исполнение – учреждение образования
«Витебский государственный университет имени П.М. Машерова».
ЛИ № 02330/110 от 30.01.2013.

Отпечатано на ризографе учреждения образования
«Витебский государственный университет имени П.М. Машерова».
210038, г. Витебск, Московский проспект, 33.