УО «МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫ УНИВЕРСИТЕТ ИМЕНИ А.А.КУЛЕШОВА»

СОЦИАЛЬНО-ГУМАНИТАРНЫЙ КОЛЛЕДЖ

Дисциплина

«Конструирование программ и языки программирования»

Разработка программ с использованием операторов цикла

(2 часа)

Методические рекомендации к лабораторной работе №4

Могилев

Понятие «Операторы цикла». Методические указания по лабораторной работе № 4 по дисциплине «Конструирование программ и языки программирования». Для учащихся 3 курса очной формы обучения специальности 2–40 01 01 «Программное обеспечение информационных технологий.

2 Краткие теоретические сведения

Циклы применяются в программах C# для выполнения каких-либо повторяющихся действий.

2.1 Цикл for

Оператор for предназначен для повторного выполнения оператора или группы операторов заданное количество раз. Вот как выглядит этот оператор в общем виде:

```
for([Инициализация];[Условие];[Приращение])  <Оператор>
```

Оператор [Инициализация] выполняется один раз перед началом цикла.

Перед каждой итерацией (т.е. перед каждым выполнением тела цикла `<Оператор>` проверяется `[Условие]`. И наконец, после каждой итерации выполняется оператор `[Приращение]`.

Как правило, в цикле имеется переменная, играющая роль так называемой переменной цикла.

При каждой итерации переменная цикла изменяет свое значение в заданных пределах.

Начальное значение переменной цикла задается в программе до оператора for или в операторе `[Инициализация]`. Предельное значение переменной цикла определяется оператором приращения, а проверка ее текущего значения – в блоке `[Условие]`.

Пример 1:

```
int i;
for (i = 0; i < 10;  i++)
{
    System.Console.Write("{0}",  i);
}
```

Здесь переменная i используется в качестве переменной цикла. Перед началом цикла ей присваивается нулевое значение. Перед каждой итерацией содержимое переменной i сравнивается с числом 10. Если i меньше 10, тело цикла выполняется один раз. В тело цикла помещен вызов метода `Write`, отображающий текущее значение переменной цикла на консоли.

После выполнения тела цикла значение i увеличивается на единицу в блоке приращения. Далее переменная цикла вновь сравнивается с числом 10. Когда значение i превысит 10, цикл завершится.

Таким образом, параметр цикла анализируется перед выполнением тела цикла, а модифицируется после его выполнения.

Вот что выведет на консоль приведенный выше фрагмент программы: 0123456789

2.1.1. Прерывание цикла

С помощью оператора `break` можно в любой момент прервать выполнение цикла. Например, в следующем фрагменте программы прерывается работа цикла, когда значение переменной i становится больше пяти:

```
for (i = 0; i < 10;  i++)
```

```
{
    if (i > 5)
    {
        break;
    }
    System.Console.Write(" {0} ",  i);
}
```

В результате на консоль будут выведены цифры от 0 до 5: 0 1 2 3 4 5

Допускается опускать реализацию любого блока оператора for:

```
int i = 0;
for (; ; )
{
    if (i > 10)
    {
        break;
    }
    System.Console.Write(" {0} ", i);
    i++;
}
```

Это позволяет реализовывать любую необходимую логику циклической обработки.

При создании цикла обязательно нужно предусмотреть условие его завершения. Если же этого не сделать, цикл будет выполняться бесконечно. Программа при этом будет работать вхолостую на одном месте, или, как еще говорят, «зациклится».

Вот пример цикла, из которого нет выхода:

```
for (i = 0; ; )    //   Зацикливание!
{
    i++;
    System.Console.Write("{0} ",  i);
}
```

Здесь не предусмотрели проверку значения переменной цикла, поэтому программа будет постоянно выводить на консоль возрастающие значения, пока вы не прервете ее работу. Кстати, это можно сделать, нажав комбинацию клавиш Control-C или закрыв консольное окно.

2.1.2. Возобновление цикла

В отличие от оператора **break**, прерывающего цикл, оператор **continue** позволяет возобновить выполнение цикла с самого начала.

Вот как он используется:

```
for (i = 0; ; i++)
{
    System.Console.Write("{0} ", i);
    if (i < 9)
                    {
```

```
        continue;
    }
    else
    {
        break;
    }
```

Если в ходе выполнения цикла значение переменной i не достигло девяти, цикл возобновляет свою работу с самого, начала (т. е. с вывода значения переменной цикла на консоль). Когда указанное значение будет достигнуто, выполнение цикла прервется оператором `break`.

2.2 Цикл while

Оператор while проверяет условие завершения цикла перед выполнением тела цикла:

```
i = 0;
while (i < 10)
{
    System.Console.Write("{0} ", i); i++;
}
```

В отличие от оператора `for` оператор `while` никак не изменяет значения переменной цикла, поэтому мы должны позаботиться об этом сами.

Перед тем как приступить к выполнению цикла, устанавливается начальное значение параметра цикла i, равное нулю. После выполнения тела цикла необходимо самостоятельно изменять значение параметра цикла, увеличивая его на единицу.

Цикл будет прерван, как только значение переменной i превысит 10.

В цикле `while` можно использовать описанные ранее операторы прерывания цикла `break` и возобновления цикла `continue`.

Следующий цикл будет выполняться бесконечно:

```
while (true)
{
  System.Console.Write("{0}", i);
   i++;
}
```

2.3 Цикл do while

Оператор `do` используется вместе с ключевым словом `while`. При этом условие завершения цикла проверяется после выполнения его тела:

```
i = 0;
do
{
    System.Console.Write("{0}", i);
    i++;
} while (i < 10);
```

Как только это значение достигнет 10, цикл будет прерван.

Аналогично циклу while цикл do допускает прерывание оператором break и возобновление оператором continue.

2.4 Цикл foreach

Для обработки таких типов данных, как массивы и контейнеры, язык C# предлагает очень удобный оператор `foreach`, для которого нет аналога в языках программирования C и C++.

Так как изучение этих типов данных еще впереди, то вернемся к изучению этого цикла позже.

Примеры использования циклов.

Листинг 1

```
using System;
namespace Iteration
{
    class IterationApp
    {
        static void Main(string[] args)
        {
            Console.Write("Цикл for\n");
            int i;
            for (i = 0; i < 10; i++)
            {
                Console.Write(" {0} ", i);
            }
Console.WriteLine("\n\nЦикл for (вариант2)"); for (i = 0; i < 10; i++)
            {
                if (i > 5)
                {
                    break;
                }
                Console.Write("{0} ", i);
            }
            Console.WriteLine("\n\nЦикл for (вариант3)");
            for (i = 0; ; i++)
            {
                Console.Write("{0} ", i);
                if (i < 9)
                {
                    continue;
                }
                else
                {
                    break;
                }
            }
            Console.WriteLine("\n\nЦикл while");
```

```
            i = 0;
            while (i < 10)
            {
                Console.Write("{0} ", i);
                i++;
            }
            Console.WriteLine("\n\nЦикл do");
            i = 0;
            do
            {
                Console.Write("{0} ", i);
                i++;
            } while (i < 10);
            Console.ReadKey();
        }
    }
}
```

2.5 Методы

Метод представляет собой блок кода, содержащий набор инструкций. В C# все инструкции выполняются в контексте метода. В этом разделе описываются именованные методы. Другой тип методов, называемых анонимными функциями, описан в других разделах документации.

Методы объявляются в классе или в структуре путем указания уровня доступа, возвращаемого значения, имени метода и списка параметров этого метода. Все вместе эти элементы образуют подпись метода. Параметры заключаются в круглые скобки и разделяются запятыми. Пустые скобки указывают на то, что у метода нет параметров.

Методы имеют смысл подпрограмм. В случае, если метод возвращает какоелибо значение, его использование схоже с использованием функции. Если метод не возвращает никаких значений - его применение аналогично процедуре.

Методы могут возвращать значения вызывающим их объектам. Если тип возвращаемого значения, указываемый перед именем метода, не равен **void**, для возвращения значения используется ключевое слово **return**. В результате выполнения инструкции с ключевым словом **return**, после которого указано значение нужного типа, вызвавшему метод объекту будет возвращено это значение. Кроме того, ключевое слово return останавливает выполнение метода. Если тип возвращаемого значения **void**, инструкцию **return** без значения все равно можно использовать для завершения выполнения метода. Если ключевое слово **return** отсутствует, выполнение метода завершится, когда будет достигнут конец его блока кода. Для возврата значений методами с типом возвращаемого значения отличным от **void** необходимо обязательно использовать ключевое слово **return.**

Чтобы использовать возвращаемое методом значение в вызываемом методе, вызов метода можно поместить в любое место кода, где требуется значение соответствующего типа.

2.5.1. Описание методов.Синтаксис

Если переменные хранят некоторые значения, то методы содержат собой набор операторов, которые выполняют определенные действия. По сути метод - это именованный блок кода, который выполняет некоторые действия.

Общее определение методов выглядит следующим образом:

```
[модификаторы] тип_возвращаемого_значения название_метода
([параметры])
{
    // тело метода
}
```

Модификаторы и параметры необязательны.

Пример и использованием возвращаемого значения:

```
static string my_func()
{ //данный метод возвращает строку
return "Hello world!";
}
static void Main(string[] args)
{ //"Главный" метод
string s = "*** "+my_func()+" ***";
Console.WriteLine(s);
}
```

Пример с использованием void:

```
static void my_proc()
{
Console.WriteLine( "Hello world!");
}
static void Main(string[] args)
{
my_proc();
}
```

```
    static int Sum(int x, int y)
    {
        return x + y;
    }
}
```

Если параметрами метода передаются значения переменных, которые представляют базовые примитивные типы (за исключением типа object), то таким переменным должно быть присвоено значение. Например, следующая программа не скомпилируется:

```
1   class Program
2   {
3       static void Main(string[] args)
4       {
5           int a;
6           int b = 9;
7           Sum(a, b);  // Ошибка - переменной a не присвоено значение
8
9           Console.ReadKey();
10      }
11      static int Sum(int x, int y)
12      {
13          return x + y;
14      }
15  }
```

При передаче значений параметрам важно учитывать тип параметров: между аргументами и параметрами должно быть соответствие по типу. Например:

```
1   class Program
2   {
3       static void Main(string[] args)
4       {
5           Display("Tom", 24); // Name: Tom  Age: 24
6
7           Console.ReadKey();
8       }
9       static void Display(string name, int age)
10      {
11          Console.WriteLine($"Name: {name}  Age: {age}");
12      }
13  }
```

В данном случае первый параметр метода Display представляет тип string, поэтому мы должны передать этому параметру значение типа string, то есть строку. Второй параметр представляет тип int, поэтому должны передать ему целое число, которое соответствует типу int.

Другие данные параметрам мы передать не можем. Например, следующий вызов метода Display будет ошибочным:

```
1   Display(45, "Bob"); // Ошибка! несоответствие значений типам параметров
```

2.5.3. Необязательные параметры

По умолчанию при вызове метода необходимо предоставить значения для всех его параметров. Но C# также позволяет использовать необязательные параметры. Для таких параметров нам необходимо объявить значение по умолчанию. Также следует учитывать, что после необязательных параметров все последующие параметры также должны быть

2.5.2. Параметры методов

Параметры позволяют передать в метод некоторые входные данные. Например, определим метод, который складывает два числа:

```
1   static int Sum(int x, int y)
2   {
3       return x + y;
4   }
```

Метод Sum имеет два параметра: x и y. Оба параметра представляют тип int. Поэтому при вызове данного метода нам обязательно надо передать на место этих параметров два числа.

```
1    class Program
2    {
3        static void Main(string[] args)
4        {
5            int result = Sum(10, 15);
6            Console.WriteLine(result);  // 25
7
8            Console.ReadKey();
9        }
10       static int Sum(int x, int y)
11       {
12           return x + y;
13       }
14   }
```

При вызове метода Sum значения передаются параметрам по позиции. Например, в вызове Sum(10, 15) число 10 передается параметру x, а число 15 - параметру y. Значения, которые передаются параметрам, еще называются **аргументами**. То есть передаваемые числа 10 и 15 в данном случае являются аргументами.

Иногда можно встретить такие определения как **формальные параметры** и **фактические параметры**. Формальные параметры - это собственно параметры метода (в данном случае x и y), а фактические параметры - значения, которые передаются формальным параметрам. То есть фактические параметры - это и есть аргументы метода.

Передаваемые параметру значения могут представлять значения переменных или результат работы сложных выражений, которые возвращают некоторое значение:

```
class Program
{
    static void Main(string[] args)
    {
        int a = 25;
        int b = 35;
        int result = Sum(a, b); Console.WriteLine(result);  // 60
        result = Sum(b, 45);
        Console.WriteLine(result);  // 80
        result = Sum(a + b + 12, 18);//a + b + 12 представляет значение параметра x
        Console.WriteLine(result);  // 90
        Console.ReadKey();
    }
```

необязательными:

```
1   static int OptionalParam(int x, int y, int z=5, int s=4)
2   {
3       return x + y + z + s;
4   }
```

Так как последние два параметра объявлены как необязательные, то мы можем один из них или оба опустить:

```
1   static void Main(string[] args)
2   {
3       OptionalParam(2, 3);
4
5       OptionalParam(2,3,10);
6
7       Console.ReadKey();
8   }
```

2.5.4. Именованные параметры

В предыдущих примерах при вызове методов значения для параметров передавались в порядке объявления этих параметров в методе. Но мы можем нарушить подобный порядок, используя именованные параметры:

```
1    static int OptionalParam(int x, int y, int z=5, int s=4)
2    {
3        return x + y + z + s;
4    }
5    static void Main(string[] args)
6    {
7        OptionalParam(x:2, y:3);
8
9        //Необязательный параметр z использует значение по умолчанию
10       OptionalParam(y:2, x:3, s:10);
11
12       Console.ReadKey();
13   }
```

2.5.5. Передача параметров в метод по ссылке. Операторы ref и out

В C# значения переменных по-умолчанию передаются по значению (в метод передается локальная копия параметра, который используется при вызове). Это означает, что мы не можем внутри метода изменить параметр из вне:

```
public static void ChangeValue(object a)
{

   a = 2;
}


static void Main(string[] args)
{

   int a = 1;
   ChangeValue(a);
   Console.WriteLine(a); // 1
   Console.ReadLine();
```

```
}
```

Чтобы передавать параметры по ссылке, и иметь возможность влиять на внешнюю переменную, используются ключевые слова ref и out.

2.5.5.1.Ключевое слово ref

Чтобы использовать ref, это ключевое слово стоит указать перед типом параметра в методе, и перед параметром при вызове метода:

```
public static void ChangeValue(ref int a)
{

   a = 2;
}

static void Main(string[] args)
{

   int a = 1;
   ChangeValue(ref
   a);
   Console.WriteLine(a); // 2
   Console.ReadLine();
}
```

В этом примере мы изменили значение внешней переменной внутри метода.

Особенностью ref является то, что переменная, которую мы передаем в метод, обязательно должна быть проинициализирована значением, иначе компилятор выдаст ошибку «Use of unassigned local variable 'a'». Это является главным отличием ref от out.

2.5.5.2.Ключевое слово out

Out используется точно таким же образом как и ref, за исключением того, что параметр не обязан быть проинициализирован перед передачей, но при этом в методе переданному параметру обязательно должно быть присвоено новое значение:

```
public static void ChangeValue(out int a)
{

   a = 2;
}

static void Main(string[] args)
{

   int a;
   ChangeValue(out
   a);
   Console.WriteLine(a); // 2
   Console.ReadLine();
```

```
}
```

Если не присвоить новое значение параметру out, мы получим ошибку «The out parameter 'a'
must be assigned to before control leaves the current method»

2.5.5.3.Массив параметров. Ключевое слово params

Во всех предыдущих примерах мы использовали постоянное число параметров. Но, используя ключевое слово **params**, мы можем передавать неопределенное количество параметров:

```
1   static void Addition(params int[] integers)
2   {
3       int result = 0;
4       for (int i = 0; i < integers.Length; i++)
5       {
6           result += integers[i];
7       }
8       Console.WriteLine(result);
9   }
10
11  static void Main(string[] args)
12  {
13      Addition(1, 2, 3, 4, 5);
14
15      int[] array = new int[] { 1, 2, 3, 4 };
16      Addition(array);
17
18      Addition();
19      Console.ReadLine();
20  }
```

Причем, как видно из примера, на место параметра с модификатором params мы можем передать как отдельные значения, так и массив значений, либо вообще не передавать параметры.

Если же нам надо передать какие- то другие параметры, то они должны указываться до параметра с ключевм словом params:

```
1   //Так работает
2   static void Addition( int x, string mes, params int[] integers,)
3   {}
```

Вызов подобного метода:

```
1   Addition(2, "hello", 1, 3, 4);
```

Однако после параметра с модификатором params мы НЕ можем указывать другие параметры. То есть следующее определение метода недопустимо:

```
1   //Так НЕ работает
2   static void Addition(params int[] integers, int x, string mes)
3   {}
```

4 Задания

1. Выполнить задания по вариантам. Дано натуральное n. Вычислить значение суммы.(с помощью циклов FOR, WHILE)

1. $\frac{1}{1}+\frac{1}{2}+\cdots;$

2. $\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)\ldots;$

3. $\frac{1}{1^5}+\frac{1}{2^5}+\cdots+;$

4. $-\frac{1}{3}+\frac{1}{5}+\cdots+;$

5. $1+\frac{1}{2}+\frac{1}{4}+\cdots;$

6. $\frac{1}{1*2}-\frac{1}{2*3}+\cdots;$

7. $\frac{1}{3^2}+\frac{1}{5}+\cdots;$

8. $\frac{1!}{1}+\frac{2!}{1+\frac{1}{2}}+\cdots;$

9. $n\left\{\sqrt{2+\sqrt{2+\cdots+\sqrt{2}}};\right.$

10. $\frac{1}{\sin 1}+\frac{1}{\sin 1+\sin 2}+\cdots;$

11. $1-\frac{1}{2}+\frac{1}{4}-\frac{1}{8}+\cdots;$

12. $\sin 1+\sin 2+\cdots;$

13. $\frac{1}{3}+\frac{1}{8}+\cdots;$

14. $1+3+6+\cdots;$

15. $(-1)^n.$

2. Составьте программу табулирования функции y(x), выведите на экран значения x и y(x). Нужный вариант задания выберите из нижеприведенного списка по указанию преподавателя. Откорректируйте элементы управления в форме в соответствии со своим вариантом задания.

1) $y = 10^{-2}bc / x + \cos\sqrt{a^3 x},$
$x_0 = -1.5; x_k = 3.5; dx = 0.5;$
$a = -1.25; b = -1.5; c = 0.75;$

2) $y = 1.2(a-b)^3 e^{x^2} + x,$
$x_0 = -0.75; x_k = -1.5; dx = -0.05;$
$a = 1.5; b = 1.2;$

3) $y = 10^{-1} ax^3 tg(a - bx),$
$x_0 = -0.5; x_k = 2.5; dx = 0.05;$
$a = 10.2; b = 1.25;$

4) $y = ax^3 + \cos^2(x^3 - b),$
$x_0 = 5.3; x_k = 10.3; dx = 0.25;$
$a = 1.35; b = -6.25;$

5) $y = x^4 + \cos(2 + x^3 - d)$,
$x_0 = 4.6; x_k = 5.8; dx = 0.2;$
$d = 1.3;$

6) $y = x^2 + tg(5x + b/x)$,
$x_0 = -1.5; x_k = -2.5; dx = -0.5;$
$b = -0.8;$

7) $y = 9(x + 15\sqrt{x^3 + b^3})$,
$x_0 = -2.4; x_k = 1; dx = 0.2;$
$b = 2.5;$

8) $y = 9x^4 + \sin(57.2 + x)$,
$x_0 = -0.75; x_k = -2.05; dx = -0.2;$

9) $y = 0.0025bx^3 + \sqrt{x + e^{0.82}}$,
$x_0 = -1; x_k = 4; dx = 0.5;$
$b = 2.3;$

10) $y = x \cdot \sin(\sqrt{x + b - 0.0084})$,
$x_0 = -2.05; x_k = -3.05; dx = -0.2;$
$b = 3.4;$

11) $y = x + \sqrt{|x^3 + a - be^x|}$,
$x_0 = -4; x_k = -6.2; dx = -0.2;$
$a = 0.1;$

12) $y = 9(x^3 + b^3)tgx$,
$x_0 = 1; x_k = 2.2; dx = 0.2;$
$b = 3.2;$

13) $y = |x - b|^{1/2} / |b^3 - x^3|^{3/2} + \ln|x - b|$,
$x_0 = -0.73; x_k = -1.73; dx = -0.1;$
$b = -2;$

14) $y = (x^{5/2} - b)\ln(x^2 + 12.7)$,
$x_0 = 0.25; x_k = 5.2; dx = 0.3;$
$b = 0.8;$

15) $y = 10^{-3}|x|^{5/2} + \ln|x + b|$,
$x_0 = 1.75; x_k = -2.5; dx = -0.25;$
$b = 35.4;$

16) $y = 15.28|x|^{-3/2} + \cos(\ln|x| + b)$,
$x_0 = 1.23; x_k = -2.4; dx = -0.3;$
$b = 12.6;$

17) $y = 0.00084(\ln|x|^{5/4} + b)/(x^2 + 3.82)$,
$x_0 = -2.35; x_k = -2; dx = 0.05;$
$b = 74.2;$

18) $y = 0.8 \cdot 10^{-5}(x^3 + b^3)^{7/6}$,
$x_0 = -0.05; x_k = 0.15; dx = 0.01;$
$b = 6.74;$

19) $y = (\ln(\sin(x^3 + 0.0025)))^{3/2} + 0.8 \cdot 10^{-3}$,
$x_0 = 0.12; x_k = 0.64; dx = 0.2;$

20) $y = a + x^{2/3}\cos(x + e^x)$,
$x_0 = 5.62; x_k = 15.62; dx = 0.5;$
$a = 0.41$

Задание 3.

1. Сколькими способами можно отобрать команду в составе 5 человек из 8 кандидатов, из 10 кандидатов, из 11 кандидатов? Вычисления оформить в виде функции. $C_n^k = \frac{n!}{k!(n-k)!}$

2. В порт в среднем заходит 3 корабля в день. Какова вероятность того, что в порт придет 2 корабля; 4 корабля? Вычисления оформить в виде функции. $P(k) = \frac{3^k e^{-3}}{k!}$

3. Два спортсмена начинают одновременно движение из одной точки. Первый начинает движение со скоростью 10 км/час и равномерно увеличивает скорость на 1 км за каждый час. Второй начинает движение со скоростью 9 км/час и равномерно увеличивает скорость на 1,6 км за каждый час. Выяснить какой из спортсменов преодолеет больший путь через 1 час, 4 часа. Вычисления путей оформить в виде функции.

4. Два спортсмена начинают одновременно движение из одной точки. Первый начинает движение со скоростью 10 км/час и равномерно увеличивает скорость на 1 км за каждый час. Второй начинает движение со скоростью 9 км/час и равномерно увеличивает скорость на 1,6 км за каждый час. Определить, когда 2-й спортсмен догонит первого. Вычисления оформить в виде функции.

5. В партии из K изделий имеется L дефектных. Для контроля выбираются R изделий. Определить вероятность того, что S изделий будут дефектными. Решить задачу для K=10, L=5, R=4, S=2; и K=10, L=4, R=5, S=3. Вычисления оформить в виде функции. $P = \frac{C_l^s * C_{k-l}^{r-s}}{C_k^r}, C_n^k = \frac{n!}{k!(n-k)!}$

6. Два треугольника заданы координатами своих вершин A, B, C. Вычислить площади треугольников с помощью формулы Герона и определить какой из них имеет большую площадь. Данные для первого треугольника: A(1;1), B(4;2), C(2;3,5). Для второго треугольника: A(1;2), B(4;1), C(3;3,5). Вычисления оформить в виде функции.

$$S = \sqrt{p(p-a)(p-b)(p-c)}, a = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

7. Футболист ударом ноги посылает мяч вертикально вверх с высоты 1м и с начальной скоростью 20м/сек. На какой высоте мяч будет через 1 с, 3 с, 4 с. Вычисления оформить в виде функции.

8. Круг задан координатами центра Q и координатами одной из точек окружности (точка Z). Внутри круга содержится квадрат, заданный координатами трех своих вершин A, B, C. Произвольно выбирается точка внутри круга. Найти вероятность того, что эта точка попадет в квадрат. Задачу решить для Q(4,5), Z(7,5), A(2,4), B(4,6),C(2,6) и для Q(5,4), Z(5,7), A(5,3), B(3,5;4,5),C(5,6). Вычисления оформить в виде функции. Искомая вероятность равна $p = \frac{S_{\text{кв}}}{S_{\text{кр}}}$.

9. Футболист ударом ноги посылает мяч вертикально вверх с высоты 1м и с начальной скоростью 20м/сек. Определить, когда мяч будет на высоте 5м, 10м. Вычисления оформить в виде функции.

10. Определить вероятности того, что среди 5 детей одной семьи нет ни одной девочки, две девочки, три девочки, четыре девочки, 5 девочек. Вероятность

рождения девочки и мальчика одинакова (p=0,5, q=1-p). $P = C_n^m p^m q^{n-m}$(из n детей m девочек). Вычисления оформить в виде функции.

$$P = \frac{C_l^s * C_{k-l}^{r-s}}{C_k^r}, C_n^k = \frac{n!}{k!\,(n-k)!}$$

11. Футболист ударом ноги посылает мяч вертикально вверх с высоты 1м и с начальной скоростью 20м/сек. Определить с точностью до 0,25 с, когда и на какой максимальной высоте окажется мяч в течение 4 с. Вычисления оформить в виде функции.

12. Круг задан координатами центра Q и координатами одной из точек окружности (точка Z). Внутри круга содержится треугольник, заданный координатами своих вершин A, B, C. Произвольно выбирается точка внутри круга. Найти вероятность того, что эта точка попадет в треугольник. Задачу решить для Q(4,5), Z(7,5), A(2,4), B(5,5),C(3,6) и для Q(5,4), Z(5,7), A(3,3), B(7,5),C(5,6). Вычисления оформить в виде функции.

Искомая вероятность равна $p = \frac{S_{mp}}{S_{\text{кр}}}$.

13. Футболист ударом ноги посылает мяч вертикально вверх с высоты 1м и начальной скоростью 20м/сек. Определить с точностью до 0,25 с, когда и на какой высоте будет максимальная скорость мяча в течение 4 с. Вычисления оформить в виде функции.

14. Два спортсмена начинают движение из одной точки. Первый начинает движение со скоростью 10 км/час и равномерно увеличивает скорость на 1 км за каждый час. Второй начинает движение со скоростью 9 км/час и равномерно увеличивает скорость на 1,6 км за каждый час. Определить с точность до 0,25 часа, когда и каким окажется максимальное расстояние между спортсменами в течение 5 часов. Вычисления оформить в виде функции.

15. Стрелок производит по мишени 5 выстрелов. Вероятность попадания в мишень при каждом выстреле 0,6. Вычислить вероятность попадания в мишень : ни разу, 1 раз, 2 раза, 3 раза, 4 раза, 5 раз. Определить когда будет максимальная вероятность. Вычисления оформить в виде функции.

$$P = \frac{C_l^s * C_{k-l}^{r-s}}{C_k^r}, C_n^k = \frac{n!}{k!\,(n-k)!}$$

Задание 4

1. Составить программу нахождения наибольшего общего делителя и наименьшего общего кратного двух натуральных чисел ($\text{НОК}(\text{A},\text{B}) = \frac{\text{A}*\text{B}}{\text{НОД}(\text{A},\text{B})}$).

2. На плоскости заданы своими координатами nточек. Составить программу, определяющую между какими из пар точек самое большое расстояние. Указание. Координаты точек занести в массив.

3. Проверить, являются ли данные три числа взаимно простыми.

4. Даны две дроби $\frac{A}{B}$ и $\frac{C}{D}$(A, B, C,D— натуральные числа). Составить программу: деления дроби на дробь; умножения дроби на дробь; сложения этих дробей. Ответ должен быть несократимой дробью.

5. Составить программу вычисления суммы факториалов всех четных чисел от mдоn.

6. Дан массив A(N). СформироватьмассивB(M), элементами которого являются большие из двух рядом стоящих в массиве A чисел. (Например, массив A состоит из элементов 1, 3, 5, -2, 0, 4, 0. Элементами массива B будут 3, 5, 4.)

7. Дано простое число. Составить функцию, которая будет находить следующее за ним простое число.

8. Составить программу, определяющую, в каком из данных двух чисел больше цифр.

9. Два натуральных числа называются «дружественными», если каждое из них равно сумме всех делителей (кроме его самого) другого (например, числа 220 и 284). Найти все пары «дружественных чисел», которые не больше данного числа N.

10. Два простых числа называются «близнецами», если они отличаются друг от друга на 2 (например, 41 и 43). Напечатать все пары «близнецов» из отрезка [n, 2n], гдеn— заданное натуральное число больше 2.

11. Из заданного числа вычли сумму его цифр. Из результата вновь вычли сумму его цифр и т.д. Через сколько таких действий получится нуль?

12. Составить функцию для нахождения наименьшего нечетного натурального делителя k(k≠1) любого заданного натурального числаn.

13. Найти все простые натуральные числа, не превосходящие n, двоичная запись которых представляет собой палиндром, т.е. читается одинаково слева направо и справа налево.

14. Найти все натуральные n-значные числа, цифры в которых образуют строго возрастающую последовательность (например, 1234, 5789).

15. Найти все натуральные числа, не превосходящие заданного n, которые делятся на каждую из своих цифр.

5 Контрольные вопросы

1. Для чего применяются циклы?
2. Как записывается и как работает оператор FOR?
3. Для организации каких циклов применим оператор FOR?
4. Как записывается и как работает оператор WHILE? DO … WHILE?
5. В чем отличие оператора WHILE от оператора DO … WHILE?