**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ
И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**
Северо-Кавказский филиал
ордена Трудового Красного Знамени федерального государственного
бюджетного образовательного учреждения высшего образования
«Московский технический университет связи и информатики»

КАФЕДРА ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Методические указания к

ЛАБОРАТОРНЫМ РАБОТАМ 1-4

по дисциплине

Модуль 2. Информационные технологии и программирование

(Основы алгоритмизации и программирования)

Ростов-на-Дону

2021 г.

Методические указания к лабораторным работам 1-4

по дисциплине

Модуль 2. Информационные технологии и программирование

(Основы алгоритмизации и программирования)

Для студентов очной и заочной форм обучения

Направление подготовки - **09.03.01 «**Информатика и вычислительная техника»

Составитель: П.В. Лобзенко, доцент кафедры ИВТ

Рассмотрено и одобрено
на заседании кафедры ИВТ
Протокол от «30» августа 2021 г. № 1

Методические указания к лабораторной работе 1

Исследование технологий создания и использования массивов в языке ТП 7.0

2.1.1 Цели занятия

Выработать навыки составления программ для создания и обработки массивов данных.

2.1.2 Теоретические основы и пример выполнения

Массив – набор переменных, имеющих одно и то же базовое имя и отличающихся друг от друга числовым признаком (индексом).

Таким образом, массив – средство, позволяющее удобно размещать в памяти большое количество однородной информации.

Описание массива:

VAR Имя массива : ARRAY[нач.индекс .. конеч.индекс] OF тип данных ;

Индексы – любой дискретный тип, обычно Integer.

Обращение к элементам массива – по имени и индексу:

```
Mes: Array [1..12] of integer;
Mes[1]:=31;  mes[2]:=28;  mes[3]:=31;  …  mes[12]:=31;
```

Примеры описания массивов:

```
Type    T1 = array[1..100] of real;
          Day=array[1..365] of integer;
          Table=array [char] of boolean;
Var     temp: T1;
        S: table;
```

Примеры обращения к элементам массива:

Temp[1]:=2.5;

temp[i]:=(temp [i-1]+temp[i+1])/2;

S[‘a’]:= true;

Свойства массивов:

- Каждый компонент массива может быть явно обозначен и к нему имеется прямой доступ.
- Число компонентов массива определяется при его описании и в дальнейшем не меняется.
- В качестве индекса можно использовать любое выражение, имеющее тип, совместимый с типом индекса.
- Элементы массива размещаются в памяти последовательно, друг за другом. Каждому элементу отводится в памяти столько места, сколько простой переменной того же типа: arr: array[1..3] of integer;

Многомерные массивы

В качестве элементов массива может быть значения любого типа. В частности, ими могут быть другие массивы.

Пример описания многомерного массива:

Var V2: array[1..10] of array[1..20] of integer;

V2: array[1..10, 1..20] of integer;

Таким образом, массив V2 можно рассматривать двояко:

массив, каждый элемент которого представляет, в свою очередь, массив типа integer;

один двумерный массив (матрица).

обращения к элементам массива будут разными:

A) V2: array[1..10] of array[1..20] of byte;

V2[1] (первый массив с элементами от 1 до 20)

V2[5] (пятый массив с элементами от 1 до 20)

И т.д.

Для адресации, например, к 5-му элементу массива V2[5] , нужно записать V2[5][5].

Б). V2: array[1..10, 1..20] of integer;

Паскаль допускает единственное возможное действие над массивом в целом: использование его в операторе присваивания. При этом типы обоих массивов в этом случае д.б. одинаковы.

```
Type    Vector = array[1..10] of word;
Var     Vect1, Vect2: Vector;
Vect1:=Vect2;
Vect1[i+1]:=100;
```

Элемент массива считается переменной. Он может получать значения в операторе присваивания, а также участвовать в выражениях, допустимых для данного типа переменных.

Типы массивов. Особенности использования массивов в программах.

Типичные ошибки при использовании массивов

Пример 1:

```
Program pr1;
Var     vect: array [1..5] of integer;
        Sum, i : integer;
Begin
 For i:=0 to 5 DO  sum:= vect[i]+sum  {не существует элем. Vect[0]} End.
```

Компилятор выдаст: Error201: Range check error.

Пример 2: Просуммировать все элементы массива.

```
Program error2;
Var     Vect: array[1..5] of integer;
        Sum, i : integer;
Begin
 I:=1;
 While i<5 DO begin      sum:=sum+vect[i];
                         I:=i+1;
              End;
End.
```

Ошибка: переменная i не «добегает» до верхней границы диапазона массива, т.е. Vect[5] не будет участвовать в суммировании. Нужно: I<=5.

В качестве примера выполнения задания приводится программа для вычисления суммы элементов одномерного массива (листинг 2.1).

Листинг 2.1 – Программа вычисления суммы элементов одномерного массива

```
program msg1;
const n=15; {число элементов массива}
var         a : array [1..n] of real;
              summa: real;
begin
  summa := 0;
     for i:=1 to n do
       begin
       readln (a[i]);
       summa := summa+a[i]
          end;
writeln ('сумма ', n, ' элементов массива равна ', summa)
end.
```

2.1.3 Порядок проведения исследований

Алгоритм выполнения заданий ЛР следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов задач, то перейти в его начало);
- составить программы решения всех задач;
- оформить отчет для всех задач вцелом, включив в него задание, блок-схему алгоритма (в электронном виде), тексты программ и skrin-shert результатов выполнения каждой задачи и представить его на проверку.

2.1.4 Варианты заданий

1. Найти N элементов массива X, в котором $X_1 = X_2 = X_3 = 2$; а все последующие элементы вычисляются по формуле: $X_k = X_{k-2} - X_{k-3} + 1/K$.
2. Вычислить значения элементов массива Z по формуле :
   Z= cos X + tg X, где X меняется на отрезке [1;15] с шагом 0,92
3. Вычислить и напечатать значения функции $Y = A_k^2 + A_k - \sin A_k$ где элементы массива A вводятся с клавиатуры .
4. Найти сумму положительных значений элементов массива W, вводимого с клавиатуры.
5. Составить массив из положительных значений функции Z= cos X * sin X для X, изменяющегося на отрезке [-5,10] с шагом 0,67.
6. Ввести с клавиатуры информацию о температуре воздуха за 2 недели. Записать в массив. Определить, сколько раз за это время она была ниже ноля.
7. Рост студентов представить в виде массива. Рост девушек закодировать со знаком "-",а рост юношей со знаком "+". Определить средний рост мальчиков.
8. Составить массив B из отрицательных значений функции Z=cos(x)/sin(x-2) для x, изменяющегося на отрезке [5;-10] с шагом 0,67.
9. Вычислить последовательность N чисел Фибоначчи и записать ее в массив $F_0 = F_1 = 1$; $F_{i+1} = F_i + F_{i-1}$

10. Вычислить N элементов массива X:

$X_k = X_{k-1} + (1/2)X_{k-2}$, где первые элементы уже известны $X_1 = 0$, $X_2 = 0{,}25$.

11. Написать программу нахождения N элементов массивов X и Y, пользуясь формулами: $X_k = 3X_{k-1} + k$, $Y_k = X_{k-1} + Y_{k-1}$, $X_0 = 1$, $Y_0 = 2$.

12. Найти N элементов массива $X_1 = X_2 = X_3 = 1$; $X_k = X_{k-1} + X_{k-3} - 1/K$.

13. Найти сумму N элементов массива $X_1 = X_2 = X_3 = 2$; $X_k = X_{k-2} - X_{k-3} + 1/K$

14. Вычислить значения элементов массива Z по формуле :

Z= cos X + ln X, где X меняется на отрезке [1;15] с шагом 0,92 и найти их сумму

15. Вычислить сумму значений функции $Y_k = A_k^2 + A_k - \text{SIN } A_k$

где элементы массива A вводятся с клавиатуры .

16. Рассчитать сумму N значений элементов массива B, формуле :

17. Найти сумму отрицательных значений элементов массива W, вводимого с клавиатуры.

18. Найти сумму значений элементов массива W с четными индексами вводимого с клавиатуры.

19. Ввести с клавиатуры информацию о температуре воздуха за 2 недели. Определить, сколько раз за это время она была ниже ноля.

20. Найти сумму значений элементов массива A с нечетными индексами вводимого с клавиатуры.

21. Рассчитать сумму N значений элементов массива B, заполненного рандомно.

22. Составить массив B из отрицательных значений функции Z=COS(X)/SIN(X-2) для X, изменяющегося на отрезке [5;-10] с шагом 0,67 и найти его сумму.

23. Вычислить последовательность N чисел Фибоначчи:

$F_0 = F_1 = 1$; $F_{i+1} = F_i + F_{i-1}$ и записать ее в массив. Найти сумму чисел с нечетными номерами.

24. Написать программу нахождения элементов массивов X и Y,

пользуясь формулами: $X_k = 3X_{k-1} + K$, $Y_k = X_{k-1} + Y_{k-1}$, $X_0 = Y_0 = 1$ и найти их сумму.

25. Найти N элементов массива $X_1 = X_2 = X_3 = 1$; $X_k = X_{k-1} + X_{k-3} - 1/K$ и найти их сумму.

Методические указания к лабораторной работе 2

Выявление закономерностей использования процедур и функций в решении прикладных задач

2.2.1 Цели занятия

Исследовать особенности составления методов пользователя. Изучить правила составления методов пользователя.

2.2.2 Теоретические основы и пример выполнения

Покажем основы составления пользовательских методов на примере языка C# [5].

Вначале рассмотрим процедуры и функции - методы класса.

Долгое время процедуры и функции играли не только функциональную, но и архитектурную роль. Весьма популярным при построении программных систем был метод функциональной декомпозиции "сверху вниз", и сегодня еще играющий важную роль. Но с появлением ООП архитектурная роль функциональных модулей отошла на второй план.

Для объектно-ориентированных языков, к которым относится и язык C#, в роли архитектурного модуля выступает класс. Программная система строится из модулей, роль которых играют классы, но каждый из этих модулей имеют содержательную начинку, задавая некоторую абстракцию данных.

Синтаксически в описании метода различают две части - описание заголовка и описание тела метода:

заголовок_метода

тело_метода

Рассмотрим синтаксис заголовка метода:

[атрибуты][модификаторы]{void| тип_результата_функции}

имя_метода([список_формальных_аргументов])

Имя метода и список формальных аргументов составляют сигнатуру метода. Заметьте, в сигнатуру не входят имена формальных аргументов - здесь важны типы аргументов. В сигнатуру не входит и тип возвращаемого результата.

Квадратные скобки (метасимволы синтаксической формулы) показывают, что атрибуты и модификаторы могут быть опущены при описании метода. У метода есть модификатор доступа и у него четыре возможных значения, из которых пока рассмотрим только два - public и private. Модификатор public показывает, что метод открыт и доступен для вызова клиентами и потомками класса. Модификатор private говорит, что метод предназначен для внутреннего использования в классе и доступен для вызова только в теле методов самого класса. Заметьте, если модификатор доступа опущен, то по умолчанию предполагается, что он имеет значение private и метод является закрытым для клиентов и потомков класса.

Обязательным при описании заголовка является указание типа результата, имени метода и круглых скобок, наличие которых необходимо и в том случае, если сам список формальных аргументов отсутствует. Формально тип результата метода указывается всегда, но значение void однозначно определяет, что метод реализуется процедурой. Тип результата, отличный от void, указывает на функцию. Вот несколько простейших примеров описания методов:

```
void A() {...};
int B(){...);
public void C(){...};
```

Методы A и B являются закрытыми, а метод C - открыт. Методы A и C реализованы процедурами, а метод B - функцией, возвращающей целое значение.

Список формальных аргументов метода может быть пустым, и это довольно типичная ситуация для методов класса. Список может содержать фиксированное число аргументов, разделяемых символом запятой.

Рассмотрим теперь синтаксис объявления формального аргумента:

[ref|out|params]тип_аргумента имя_аргумента

Обязательным является указание типа и имени аргумента. Заметьте, никаких ограничений на тип аргумента не накладывается. Он может быть любым скалярным типом, массивом, классом, структурой, интерфейсом, перечислением, функциональным типом.

Несмотря на фиксированное число формальных аргументов, есть возможность при вызове метода передавать ему произвольное число фактических аргументов. Для реализации этой возможности в списке формальных аргументов необходимо задать ключевое слово params.

Оно задается один раз и указывается только для последнего аргумента списка, объявляемого как массив произвольного типа. При вызове метода этому формальному аргументу соответствует произвольное число фактических аргументов.

Содержательно, все аргументы метода разделяются на три группы: входные, выходные и обновляемые. Аргументы первой группы передают информацию методу, их значения в теле метода только читаются. Аргументы второй группы представляют собой результаты метода, они получают значения в ходе работы метода. Аргументы третьей группы выполняют обе функции. Их значения используются в ходе вычислений и обновляются в результате работы метода. Выходные аргументы всегда должны сопровождаться ключевым словом out, обновляемые - ref. Что же касается входных аргументов, то, как правило, они задаются без ключевого слова, хотя иногда их полезно объявлять с параметром ref.

Заметьте, если аргумент объявлен как выходной с ключевым словом out, то в теле метода обязательно должен присутствовать оператор присваивания, задающий значение этому аргументу. В противном случае возникает ошибка еще на этапе компиляции.

Процедуры и функции связываются теперь с классом, они обеспечивают функциональность данных класса и называются методами класса. Главную роль в программной системе играют данные, а функции лишь служат данным.

В C# процедуры и функции существуют только как методы некоторого класса, они не существуют вне класса.

2.2.3 Порядок проведения исследований

Алгоритм выполнения заданий ЛР следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов задач, то перейти в его начало);
- составить методы (функции) решения всех задач, поместить их в класс Form1;
- оформить отчет для всего приложения в целом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrin-shert результата выполнения каждой задачи и представить его на проверку.

2.2.4 Варианты заданий

Составить процедуры и функции, или методы возвращающие и не возвращающие значения для следующих заданий.

1. Найти среднее арифметическое положительных чисел, введенных с клавиатуры. Всего ввести N различных чисел.

2. Ввести с клавиатуры N чисел. Найти сумму тех из них, которые принадлежат интервалу (2;9).

3. Для N введенных с клавиатуры чисел найти сумму положительных кратных 3.

4. Для арифметической прогрессии 4, 9, 14, 19... найти первые n членов этой прогрессии.

5. Найти сумму отрицательных значений функции Z=sin(5-x)/cos(x-2) для x, изменяющегося на отрезке [-5,12] с шагом 1,23.

6. Найти среднее арифметическое отрицательных чисел, введенных с клавиатуры. Всего ввести N различных чисел.

7. Найти среднее арифметическое чисел, принадлежащих отрезку [2,184], кратных 2 и введенных с клавиатуры. Всего ввести N различных чисел.

8. Найти сумму значений функции, больших 2 $Z=\sin(1/x)+5\cos(1/(x-3))+x$ для x, изменяющегося на отрезке [-3,8] с шагом 1,123.

9. Найти n членов последовательности $x_1 = x_2 = x_3 = 1$; $x_k = x_{k-1} + x_{k-3}$.

10. Вычислить последовательность N чисел $A_0 = x$, $A_1 = 2$, $A_k = A_{k-1} - A_{k-2}$.

11. Для $x_1 = 0{,}3$ и $x_2 = -0{,}3$ найти $x_k = k + \sin(x_{k-2})$ для k, изменяющегося следующим образом: k = 3,4,...,14.

12. Составить таблицу перевода дюймов в сантиметры для расстояний от 1 до 13 дюймов с шагом 1.

13. Вывести на печать значения функции, меньшие 2, $Z=\sin(1/x)+5\cos(x-3)+x$ для x, изменяющегося на отрезке [-7,4] с шагом 1.

14. Напечатать таблицу значений функции $Y = tg(x/b)+x/(b-2)$ для x, изменяющегося от 0 до 10 с шагом 1 (b - произвольное число).

15. Вычислить N -й член последовательности $x_k = x_{k-2} - x_{k-1}$, $x_0 = 2{,}4$ $x_1 = 3{,}8$.

16. Составить таблицу перевода суток (от 1 до 7) в часы, минуты, секунды.

17. Вычислить N-й член последовательности $x_k = x_{k-1} + (2/3)x_{k-2} + 1$, $x_1 = -1$, $x2 = 1{,}38$.

18. Напечатать значения функции $z = 1/(x-2)+1/(x-5)+\ln(12{,}8-X)$ для x, изменяющегося на отрезке [-4,14] с шагом 1,152.

19. Вывести на печать отрицательные значения функции $z=\sin(5-x)/\cos(x-2)$ для x, изменяющегося на отрезке [-6,13] с шагом 1,541 (учесть область допустимых значений функции).

20. Из N введенных с клавиатуры чисел напечатать кратные 3 и меньшие 58.

21. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (1,11) и являются четными.

22. Из N введенных с клавиатуры чисел напечатать положительные, кратные 3.

23. Вывести на печать значения функции z = sin(x/(x-2)), находящиеся в интервале (-0,4;0,8) для x, изменяющегося от 8 до -6 с шагом 1,235.

24. Ввести с клавиатуры N чисел. Напечатать те из них, которые принадлежат интервалу (2;9).

25. Для геометрической прогрессии 2, 6, 18, 54, 162 ... определить первые n членов этой прогрессии.

26. Ввести с клавиатуры N чисел. Напечатать те из них, которые не принадлежат интервалу (1;5).

27. Найти n членов последовательности $x_1 = x_2 = x_3 = 1$; $x_k = x_{k-1} - 2x_{k-3}$.

28. Вычислить последовательность N чисел $A_0 = x, A_1 = 2$, $A_k = A_{k-1} + A_{k-2}$

29. Составить таблицу перевода килограммов (от 1 до 13) в граммы с шагом

30. Найти сумму значений функции Y=COS(x/A)+x/(A-2) для x, изменяющегося от 2 до 13 с шагом 1 (A - произвольное число).

Методические указания к лабораторной работе 3

Исследование управляющих операторов C#

2.3.1 Цели занятия

Выработать умения и навыки по составлению программ с использованием базовых операторов языка C#.

2.3.2 Теоретические основы и пример выполнения

2.3.2.1 Консольный ввод и вывод. Преобразования типов

Для вывода данных различных типов используется метод Write() класса Console.

Можно выводить данные потоком:

```
int x=12547;
Console.WriteLine("x="+x);
```

И будет напечатан текст x= и значение этой переменной:

x=12547

Возможен также форматный вывод, когда используются, так называемые, метки-заполнители (листинг 2.2)

Листинг 2.2- Форматный вывод данных

```
void frmt()
        {
            int x=12547;
            double y = 529465797.45963;
            Console.WriteLine("{0:d}   \n{1:f5}",x,y);
        }
 Результат:
12547
529465797.45963
```

Получение системного времени:

System.DateTime dat = Convert.ToDateTime("15.03.2003").

Ввод и вывод в консоли осуществляется всегда в типе string. Чтобы ввести число какого-то типа (int, double, float) нужно использовать метод Parse, пара-

метром которого должен быть метод ReadLine(), который прочитывает введенную строку, т.е. используется вот такая конструкция:

ТИП Переменная=ТИП.Parse(Console.ReadLine());

Пример ввода с клавиатуры:

int peremennayA=int.Parse(Console.ReadLine());

double peremennayX= double. Parse(Console.ReadLine());

Запись переменных и выражений в языке C# имеет определенную структуру (листинг 2.3).

Листинг 2.3- Инициализация констант и переменных

```
public void SimpleVars()
{
//Объявления локальных переменных
int x, s; //без инициализации
int y =0, u = 77; //обычный способ инициализации
//допустимая инициализация
float w1=0f, w2 = 5.5f, w3 =w1+ w2 + 125.25f;
```

Листинг 2.3- Продолжение

```
//допустимая инициализация в объектном стиле
int z= new int();
//Недопустимая инициализация.
//Конструктор с параметрами не определен
//int v = new int(77);
x=u+y; //теперь x инициализирована
if(x> 5) s = 4;
for (x=1; x<5; x++)s=5;
//Инициализация в if и for не рассматривается,
//поэтому s считается неинициализированной переменной
//Ошибка компиляции:использование неинициализированной переменной
//Console.WriteLine("s= {0}",s);
} //SimpleVars
//Инициализация переменных. Глобальные переменные
//fields
```

```
int x,y; //координаты точки
string name; //имя точки
//конструктор с параметрами
public Testing(int x, int y, string name)
{
this.x = x; this.y = y; this.name = name;
}
//Локальные переменные
/// <summary>
/// Анализ области видимости переменных
/// </summary>
/// <param name="x"></param>
public void ScopeVar(int x)
{
//int x=0;
int y =77; string s = name;
if (s=="Точка1")
{
```

Листинг 2.3- Продолжение

```
int u = 5; int v = u+y; x +=1;
Console.WriteLine("y= {0}; u={1};
v={2}; x={3}", y,u,v,x);
}
else
{
int u= 7; int v= u+y;
Console.WriteLine("y= {0}; u={1}; v={2}", y,u,v);
}
//Console.WriteLine("y= {0}; u={1}; v={2}",y,u,v);
//Локальные переменные не могут быть статическими.
//static int Count = 1;
//Ошибка: использование sum до объявления
//Console.WriteLine("x= {0}; sum ={1}", x,sum);
int i;long sum =0;
for(i=1; i<x; i++)
```

```
{
//ошибка: коллизия имен: y
//float y = 7.7f;
sum +=i;
}
Console.WriteLine("x= {0}; sum ={1}", x,sum);
}//ScopeVar
//Константы
/// <summary>
/// Константы
/// </summary>
public void Constants()
{
const int SmallSize = 38, LargeSize =58;
const int MidSize = (SmallSize + LargeSize)/2;
const double pi = 3.141593;
//LargeSize = 60; //Значение константы нельзя изменить.
Console.WriteLine("MidSize= {0}; pi={1}",
```

Листинг 2.3- Продолжение

```
MidSize, pi);
}
```

2.3.2.2 Операции отношения и условный оператор

Операции отношения можно просто перечислить - в объяснениях они не нуждаются. Всего операций 6 (==, !=, <, >, <=, >=):

== - «равно»;

!= - «неравно»;

&& - «И»;

|| - «ИЛИ»;

++x; или x++; --x; или x--; -инкремент, декремент.

В C#, как и в C++, разрешены условные выражения. Конечно, без них можно обойтись, заменив их условным оператором. Вот простой пример их использования, поясняющий синтаксис их записи (Листинг 2.4).

Листинг 2.4-Условное выражение

```
int a = 7, b= 9, max;
max= (a>b) ? a:b;
Console.WriteLine("a = " + a + "; b= " + b +
"; max(a,b) = " + max);
```

Условное выражение начинается с условия, заключенного в круглые скобки, после которого следует знак вопроса и пара выражений, разделенных двоеточием " : ". Условием является выражение типа bool. Если оно истинно, то из пары выражений выбирается первое, в противном случае результатом является значение второго выражения. В данном примере переменная max получит значение 9.

Более длинная форма записи (листинг 2.5).

Листинг 2.5- Классическая форма записи условного оператора

```
if(условие) {прямая ветка}
else {побочная ветка}
```

Листинг 2.5- Продолжение

```
if(x<=0 || y>=4 && d!=m || s==5) { Con-
sole.WriteLine("g="+g);}
else { Console.WriteLine("x="+x);}
```

При необходимости требуемый тип данных можно указать явно перед переменной или перед выражением:

```
//cast
int p;
p = (int)x;
//b = (bool)x;
```

В классе Math и содержаться различные математические функции:

- тригонометрические функции - Sin, Cos, Tan;
- обратные тригонометрические функции - ASin, ACos, ATan, ATan2 (sinx, cosx);

- гиперболические функции - Tanh, Sinh, Cosh;
- экспоненту и логарифмические функции - Exp, Log, Log10;
- модуль, корень, знак - Abs, Sqrt, Sign;
- функции округления - Ceiling, Floor, Round;
- минимум, максимум, степень, остаток - Min, Max, Pow, IEEEReminder.

В листинге 2.6 демонстрируется использование этих функций.

Листинг 2.6 – Математические конструкции языка C#

```
public void MathFunctions()
{
double a, b,t,t0,dt,y;
string NameFunction;
Console.WriteLine("Введите имя F(t)исследуемой функции
a*F(b*t)" + " (sin, cos, tan, cotan)");
NameFunction = Console.ReadLine();
Console.WriteLine("Введите параметр a (double)");
a= double.Parse(Console.ReadLine());
Console.WriteLine("Введите параметр b (double)");
```

Листинг 2.6 – Продолжение

```
b= double.Parse(Console.ReadLine());
Console.WriteLine("Введите начальное время t0(double)");
t0= double.Parse(Console.ReadLine());
const int points = 10;
dt = 0.2;
for(int i = 1; i<=points; i++)
{
t = t0 + (i-1)* dt;
switch (NameFunction)
{
case ("sin"): y = a*Math.Sin(b*t); break;
case ("cos"): y = a*Math.Cos(b*t); break;
case ("tan"): y = a*Math.Tan(b*t); break;
case ("cotan"): y = a/Math.Tan(b*t); break;
case ("ln"): y = a*Math.Log(b*t); break;
```

```
case ("tanh"): y = a*Math.Tanh(b*t); break;
default: y=1; break;
}//switch
Console.WriteLine ("t = " + t + "; " + a +"*" +
NameFunction +"(" + b + "*t)= " + y + ";");
}//for
double u = 2.5, v = 1.5, p,w;
p= Math.Pow(u,v);
w = Math.IEEERemainder(u,v);
Console.WriteLine ("u = " + u + "; v= " + v +
"; power(u,v)= " + p + "; reminder(u,v)= " + w);
}//MathFunctions
```

2.3.2.3 Операторы языка C#

Блок или составной оператор выглядит следующим образом:

{оператор_1 ...

оператор_N}

Это нужно, если под одним условием или в одном цикле, или по еще какой-либо причине должны выполняться сразу несколько операторов. Тогда их берут в фигурные скобки и управление сначала передается этому блоку операторов, а потом только выполняются все операторы, следующие за ним.

Оператор if

if(выражение_1) оператор_1

else if(выражение_2) оператор_2

...

else if(выражение_K) оператор_K

else оператор_N

if(выражение1) if(выражение2) if(выражение3) ...

Оператор switch

switch(выражение)

{

case константное_выражение_1: [операторы_1 оператор_перехода_1]

...

case константное_выражение_K: [операторы_K оператор_перехода_K]

[default: операторы_N оператор_перехода_N]

}

Ветвь default может отсутствовать.

В качестве примера выбора из множества альтернатив приведем программу, которая определяет период жизни в зависимости от возраста – age (листинг 2.7).

Листинг 2.7- Пример использования оператора switch

```
public void SetPeriod()
{
if ((age > 0)&& (age <7))period=1;
else if ((age >= 7)&& (age <17))period=2;
```

Листинг 2.6- Продолжение

```
else if ((age >= 17)&& (age <22))period=3;
else if ((age >= 22)&& (age <27))period=4;
```

Листинг 2.7- Продолжение

```
else if ((age >= 27)&& (age <37))period=5;
else period =6;
}
public void SetStatus()
{
switch (period)
{
case 1: status = "child"; break;
case 2: status = "schoolboy"; break;
case 3: status = "student"; break;
case 4: status = "junior researcher"; break;
case 5: status = "senior researcher"; break;
case 6: status = "professor"; break;
default : status = "не определен"; break;
}
Console.WriteLine("Имя = {0}, Возраст = {1}, Статус = {2}",
```

```
name, age, status);}
```

Оператор goto имеет простой синтаксис и семантику:

goto [метка|case константное_выражение|default];

Далее рассмотрим операторы цикла. Первый из них- это цикл со счетчиком:

for(инициализаторы; условие; список_выражений) оператор

Для демонстрации этого цикла покажем программу, которая в передаваемом ей тексте определяет палиндромы. Палиндромом называется симметричная строка текста, читающаяся одинаково слева направо и справа налево (листинг 2.8).

Листинг 2.8- Определение палиндромов. Демонстрация цикла for

```
public bool Palindrom(string str)
{
for (int i =0,j =str.Length-1; i<j; i++,j--)
if(str[i]!=str[j]) return(false);
return(true);}
```

Далее, рассмотрим цикл с предусловием:

while(выражение)

{тело цикла}

и цикл с постусловием:

do

{тело цикла}

while(выражение);

Продемонстрируем эти типы циклов в программе, которая работает следующим образом (листинг 2.9):

- Внешний цикл - образец многократно решаемой задачи.
- Завершение цикла определяется в диалоге с пользователем.

Листинг 2.9- Пример циклов с пред- и постусловием

```
public void Loop()
{
string answer, text;
```

```
do
{
Console.WriteLine("Введите слово");
text = Console.ReadLine();
int i =0, j = text.Length-1;
while ((i<j) && (text[i] == text[j]))
{i++; j--;}
if (text[i] == text[j])
Console.WriteLine(text +" - это палиндром!");
else
Console.WriteLine(text +" - это не палиндром!");
Console.WriteLine("Продолжим? (yes/no)");
answer = Console.ReadLine();
}
while(answer =="yes");
}
```

И завершим рассмотрение циклов циклом foreach. Новым видом цикла, не унаследованным от C++, является цикл foreach, удобный при работе с массивами, коллекциями и другими подобными контейнерами данных.

Его синтаксис:

foreach(тип идентификатор in контейнер) оператор

В приведенном ниже примере показана работа с трехмерным массивом, заполненным случайными элементами. Массив создается с использованием циклов типа for, а при нахождении суммы его элементов, минимального и максимального значения используется цикл foreach (листинг 2.10).

Листинг 2.10- Использование цикла foreach

```
public void SumMinMax()
{
int [,,] arr3d = new int[10,10,10];
Random rnd = new Random();
for (int i =0; i<10; i++)
for (int j =0; j<10; j++)
for (int k =0; k<10; k++)
```

```
arr3d[i,j,k]= rnd.Next(100);
long sum =0; int min=arr3d[0,0,0], max=arr3d[0,0,0];
foreach(int item in arr3d) {
sum +=item;
if (item > max) max = item;
else if (item < min) min = item;
}
Console.WriteLine("sum = {0}, min = {1}, max = {2}",
sum, min, max);
}
```

2.3.3 Порядок проведения исследований

Алгоритм выполнения задач практического занятия следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов задач, то перейти в его начало);
- составить методы (функции) решения всех задач;
- оформить отчет для всего приложения в целом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrinshert результата выполнения каждой задачи и представить его на проверку.

2.3.4 Варианты заданий

1. Составить программу для перевода длины в метрах в длину в сантиметрах, определив функцию, выполняющую это преобразование и передав длину в метрах в качестве параметра.

2. Составить программу для нахождения суммы элементов каждого из трех массивов, введенных с клавиатуры, определив функцию, выполняющую это действие, и передавая массивы в качестве параметра.

3. Даны числа S, T. Получить с использованием функции пользователя F(T,-2S,1.17)+F(2.2,T,S-T) где F(A, B, C) = (2A-B-sin(C))/(5+C)

4. Составить программу перевода двоичной записи натурального числа в десятичную, описав соответствующую функцию с параметром. Перевод осуществлять для чисел, вводимых с клавиатуры. Признак конца ввода - число 0.

5. Даны числа S, T. Получить с использованием функции пользователя с параметрами:

G(1,sin(S))+2G(T*S,24)-G(5,-S), где G(A,B)=(2A+B*B)/(A*B*2+B*5).

6. Составить программу для расчета значений гипотенузы треугольника, определив функцию, выполняющую этот расчет. Катеты передаются в качестве параметров.

7. Найти периметр десятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, которые передаются функции в качестве параметров из основной программы.

8. Найти периметр шестиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами. Координаты передаются функции в качестве параметров из основной программы.

9. Найти площадь пятиугольника, координаты вершин которого заданы. Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами, и процедуру вычисления площади треугольника по трем сторонам. Описать функции с соответствующими формальными параметрами.

10. Составить программу вывода на экран всех натуральных чисел, не превосходящих N и делящихся на каждую из своих цифр. Описать соответствующую функцию, получающую из основной программы в качестве параметра натуральное число и возвращающую TRUE, если оно удовлетворяет указанному условию.

11. Используя подпрограмму - функцию, составить программу для нахождения максимального из трех чисел. Числа передаются функции в качестве параметров.

12. Используя подпрограмму - функцию, составить программу для печати знаков трех чисел, введенных с клавиатуры и передаваемых функции в качестве параметра.

13. Используя подпрограмму - функцию, составить программу для возведения чисел в целую положительную степень. Число передаются функции в качестве параметра из основной программы. Расчет вести для чисел, пока не будет введено число, равное 0.

14. Используя подпрограмму - функцию, составить программу для вычисления функции Z=(X1+Y1)/(X1*Y1), где X1 - первый корень уравнения X^2-4*X-1=0; Y1 - первый корень уравнения $2*Y^2 + A*Y - A^2 = 0$ (A - произвольное).

15. Задав функцию, вывести на печать средние арифметические двух массивов, введенных с клавиатуры. Массив передается функции в качестве параметра.

16. Задав функцию, рассчитать и вывести на печать максимальные значения в трех парах чисел, вводимых с клавиатуры. Пара чисел передается функции в качестве параметра.

17. Найти периметр восьмиугольника, координаты вершин которого заданы. Определить функцию вычисления расстояния между двумя точками, заданными своими координатами. Координаты передать функции в качестве параметров.

18. Даны четыре пары чисел. Получить с использованием функции пользователя наибольший общий делитель для каждой пары.

19. Даны числа A, B, C. Получить с использованием функции пользователя наименьшее значение. Числа передаются функции из основной программы в качестве параметров.

20. Даны числа x = 1,2,...,N. Получить с использованием функции пользователя значения 3*P(X+3)*P(X) для заданных x, где $P(X) = 10*X^3 - 14*X^2 + 12*X - 2$.

21. Составить программу для расчета значений катета треугольника, определив функцию, выполняющую этот расчет. Гипотенуза и второй катет передаются в качестве параметров.

22. Даны целые числа a,b,c,d. Проверить с использованием функции пользователя их четность. Число для проверки передается в функцию в качестве параметра из основной программы.

23. Для каждого из 10 введенных с клавиатуры чисел напечатать сообщение: является ли оно простым или нет, описав функцию логического типа, возвращающую значение “ИСТИНА”, если число, переданное ей в качестве параметра, является простым.

24. Даны числа S, T. Получить с использованием функции пользователя Y(T,S)=G(12,S)+G(T,S)-G(2S-1,S*T), где G(A,B)=(2*A+B*B)/(A*B*2+B*5).

25. Определите функцию, определяющую, какой целой степенью числа 2 является ее аргумент (если число не является степенью двойки - выдать соответствующее сообщение).

26. Определите функцию, подсчитывающую сумму N первых элементов целочисленного массива A. N и массив A передать в качестве параметров.

27. Вычислить количество простых чисел, не превосходящих заданного N. Описать функцию логического типа, возвращающую значение true, если число простое и false в противном случае.

28. Используя подпрограмму - функцию с параметрами, составить программу для вычисления функции $F(X,Y) = (2X^3-4*X^2+X+1)/(9*Y^3+Y+4) + 3*Y^2+5*Y$.

29. Составить программу для перевода веса в граммах в вес в килограммах, определив функцию, выполняющую это преобразование. Вес в граммах передается функции в качестве параметра.

30. Даны числа S, T. Получить с использованием функции пользователя G(12, S)+G(T, S)-G(2S-1, S*T) где G(A, B) = (2*A+B*B)/(A*B*2+B*5).

Методические указания к лабораторной работе 4

Исследование управляющих операторов Java

2.4.1 Цели занятия

Выработать умения и навыки составлять типовые программы решения задач на языке программирования Java с использованием основных управляющих операторов.

2.4.2 Теоретические основы и пример выполнения

Рассмотрим условный оператор.

if … else - после служебного слова if должно располагаться логическое выражение. Логическое выражение должно быть взято в круглые скобки. В противном случае, либо выполняются операторы, стоящие за служебным словом else, либо, если нет служебного слова else, выполнение оператора if прекращается.

Оператор If дает возможность в зависимости от условия выполнять ту или иную ветвь программы. Синтаксис оператора следующий:

If условие выражение1 else выражение2;

Условие должно давать результат в виде логического значения истинности или ложно-сти. Выражение1 будет выполняться если условие истинно. Выражение2 будет выпол-няться если условие ложно.

Существует сокращенный вариант оператора:

If условие выражение1

Пример. Определить, является ли введенное число днем недели, т.е. входит ли число в диапазон от 1 до 7 (листинг 2.11).

Листинг 2.11- Пример написания условий

```
package lr;
import java.util.Scanner;
public class ll {
public static void main(String[] args) {
int a;
 System.out.println("введите число");
```

```
     Scanner scan=new Scanner(System.in);
     a=scan.nextInt();
     if ((a<1)||(a>7)){
     System.out.println("введенное число не является днем недели
");
     }else{
          System.out.println("введенное число является днем недели
");
    }}}
```

Выражение условия (A<1) || (A>7) будет давать TRUE, если выполняется A<1 или A>7 - в этом случае выполняется ветка printf('Error ',A);, иначе ветка printf('OK ',A);.

Далее перейдем к циклическому алгоритму (цикл с параметром - FOR).

Оператор for организует цикл:

for(выражение 1; выражение 2; выражение 3)

оператор;

выражение 3 (приращение) вычисляется после каждого прогона цикла;

выражение 1 (инициализация) вычисляется перед началом цикла;

выражение 2 (условие) - до и после каждого прогона цикла.

Оператор тела цикла выполняется до тех пор, пока истинно выражение 2.

Любое из выражений, или же все три, могут отсутствовать, но при этом должны сохраняться все точки с запятыми. Если выражение 2 опущено, то считается, что оно всегда истинно (листинг 2.12).

Листинг 2.12- Фрагмент программы вычисления суммы чисел от 0 до 9

```
package lr;
public class Lr3 {
 public static void main(String[] args) {
      int addCounter=0;
      for(int counter=0; counter<10;counter++){
           System.out.println(" counter="+counter );
            addCounter+=counter;
}
```

```
 System.out.println(" Сумма чисел от 0 до 9="+addCounter );
 }
```

Далее, рассмотрим оператор цикла с предусловием.

Для организации цикла с предусловием используется оператор while:

while(выражение)

оператор;

Оператор тела цикла выполняется до тех пор, пока истинно выражение, записанное в скобках.

Выражение вычисляется до начала и после каждого прогона цикла. Цикл не выполняется ни разу, если выражение ложно (равно 0) (листинг 2.13).

Листинг 2.13- Фрагмент программы вычисления суммы чисел от 0 до 9

```
package lr;
public class Lr4prim {
 public static void main(String[] args) {
      int counter=0, addCounter=0;
      while(counter++<9) {
           System.out.println(" counter="+counter );
            addCounter+=counter;
                                  }
 System.out.println(" Сумма чисел от 0 до 9="+addCounter );
 }
}
```

Завершим рассмотрение циклов оператором цикла с постусловием.

Цикл с пост-условием do while имеет вид:

do

оператор;

while(условие);

Оператор тела цикла выполняется до тех пор, пока истинно условие. Условие проверяется после выполнения оператора тела цикла (листинг 2.14).

Листинг 2.14- Цикл с предусловием

```
package lr;
import java.math.BigInteger;
import java.util.Scanner;
public class Lr5prim {
 public static void main(String[] args) {
      int n;
      long x=1;
       System.out.println("Расчет    фактериала    числа.Ведите
число= " );
      Scanner scan=new Scanner(System.in);
      n=scan.nextInt();
      int Z=n;
 do {
           x*=n;
```

Листинг 2.14- Продолжение

```
 System.out.println("число= " + n);

 } while (--n > 0);
      System.out.println("Факториал  " + Z+"="+x);
 }
}
```

Оператор continue возвращает управление на начало цикла, пропуская стоящие после него операторы цикла. Оператором break можно завершить цикл.

Оператор безусловного перехода goto метка; передает управление на оператор, перед которым стоит метка. Метка представляет собой идентификатор с двоеточием и может стоять перед любым выполняемым оператором.

Как пример выполнения задания лабораторной работы покажем программу для решения одного из вариантов заданий, где нужно вывести на печать положительные значения функции z=sin(x)+5cos(x-2) для x изменяющегося на отрезке [5,-10] с шагом 1,2 (листинг 2.15).

Листинг 2.15 – Пример выполнения варианта лабораторной работы

```
package lr;
public class Lr5laba {
```

```
     public static void main(String[] args) {
    System.out.println("Вывести на печать положительные значения функции z=sin(x)+5cos(x-2) для x изменяющегося на отрезке[5;-10] с шагом 1,2.");
          double x=5.0;
          double z;
          do
          {
          z=Math.sin(x)+5*Math.cos(x-2);
          if (z > 0){
     System.out.print("Положительное значение функции Z="+z+ " при x="); System.out.print(x+"\n");}
          x = x - 1.2;
          }while (x>=-10);}}
```

2.4.3 Порядок проведения исследований

Алгоритм выполнения заданий лабораторной работы следующий:

- выбрать 5 задач по следующему правилу: номер по журналу- первая задача; номер каждой последующей задачи определяется прибавлением цифры 3 к номеру первой задачи, который только что вычислили (если достигнуто окончание списка вариантов задач, то перейти в его начало);
- составить методы (функции) решения всех задач;
- оформить отчет для всего приложения в целом, включив в него задание, блок-схему алгоритма (в электронном виде), текст программы и skrinshert результата выполнения каждой задачи и представить его на проверку.

2.4.4 Варианты заданий

1. Вывести на печать положительные значения функции y=sin(x)+5cos(x-2) для x изменяющегося на отрезке [-5, 12] с шагом 1,2.

2. Вывести на печать значения функции z=tg(2x)-sin(x) для x изменяющегося на отрезке [-3, 3] с шагом 0,3.

3. Ввести с клавиатуры и напечатать модули N чисел; если введено отрицательное число, ввод и печать прекратить.

4. Вывести на печать значения функции z=ln(x)+tg(2x), большие 1, для x изменяющегося на отрезке [3, 8] с шагом 0,9.

5. Определить, является ли натуральное число N степенью числа 5 или нет.

6. Напечатать значения функции y=ln(x+12/x),где значения x вводятся с клавиатуры. При вводе числа, не входящего в область определения функции, вычисления прекратить.

7. Напечатать значения функции y=ln(x-1/x),где значения x вводятся с клавиатуры. При вводе числа,не входящего в область определения функции, вычисления прекратить.

8. Для x из интервала (-2;8) с шагом 0,75 вычислить y=(4x-3x+tg(x))/A, где A вводится с клавиатуры.

9. Вывести на печать значения функции z=sin(x)+cos(x),находящиеся в интервале (-0,2; 0,8) для x изменяющегося на отрезке [4,-6] с шагом 0,91.

10. Дано натуральное число N. Получить наименьшее число вида 4^k, большее N.

11. Для x из интервала (2;8) с шагом 0,75 вычислить y=(4x-3x+cos(x))/A, где A вводится с клавиатуры.

12. Найти первый член последовательности ln(9n)/(n*n), меньший 1, для n изменяющегося следующим образом: n=1,2,3... .

13. Определить, является ли натуральное число N степенью числа 3 или нет.

14. Вывести на печать отрицательные значения функции z=cos(x)-5sin(x-2) для x изменяющегося на отрезке [-3, 11] с шагом 0,9.

15. Ввести с клавиатуры и напечатать квадраты N чисел, если введено кратное 3 положительное число, ввод и печать прекратить.

16. Вывести на печать отрицательные значения функции z=tg(x)+5cos(x-2) для x изменяющегося на отрезке [12, 1] с шагом 1,2.

17. Ввести с клавиатуры и напечатать N чисел, если введено равное нулю или кратное 2 число, ввод и печать прекратить.

18. Вывести на печать значения функции z=ln(|x|)+tg(2x), большие 2 для x изменяющегося на отрезке [3, -8] с шагом 0,9.

19. Найти первый отрицательный член последовательности sin(tg(n/2)) для n изменяющегося на следующим образом: n=1,2,3... .

20. Напечатать значения функции y=ln(x+12/x),где значения x вводятся с клавиатуры. При вводе числа, не входящего в область определения функции, вычисления прекратить.

21. Найти первую цифру в целом положительном числе.

22. Дано натуральное число N. Получить наибольшее число вида 3^k, меньшее N.

23. Вывести на печать значения функции z=sin(x)+cos(x),находящиеся в интервале (-0,3;0,7) для x изменяющегося на отрезке [-4,6] с шагом 0,91.

24. Дано натуральное число N. Получить наименьшее число вида 5^k, большее N.

25. Для x из интервала (-2;8) с шагом 0,75 вычислить y=(4x-3x+tg(x))/A, где A вводится с клавиатуры.

26. Найти первый член последовательности ln(9n/(n*n+1), меньший 0, для n изменяющегося на следующим образом: n=1,2,3... .

27. Определить, является ли натуральное N степенью числа 4 или нет.

28. Вывести на печать положительные значения функции z=sin(x)-5cos(x-2) для x изменяющегося на отрезке [5,-12] с шагом 1,2.

29. Напечатать значения функции $Y = \sqrt{2x^2 - x^3}$ для произвольных *x*, вводимых с клавиатуры. При вводе числа, не входящего в область определения функции, ввод и печать прекратить.

30. Найти первый отрицательный член последовательности cos(ctg(n)) для n изменяющегося на следующим образом: n=1,2,3... .

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Боровский А. Н. Qt4.7+. Практическое программирование на C++. — СПб.: БХВ-Петербург, 2012. — 496 с.: ил. — (Профессиональное программирование).

2 UCE — Кроссплатформенный C++ фреймворк для разработки приложений с пользовательским интерфейсом. [Электронный ресурс] // [сайт] [2018], URL: https://habr.com/post/209956/. (дата обращения: 03.06.2018).

3 Галисеев Г.В. Компоненты в Delphi 7. Профессиональная работа. – М.: Издательский дом «Вильямс», 2008. – 624 с.

4 Иванова Г.С. Технология программирования: Учебник для вузов. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. - 320 с.

5 Фленов М. Е. Ф69 Библия C#. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. — 560 с.