МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Технологии программирования и алгоритмизация»
для студентов направления подготовки
09.03.01 Информатика и вычислительная техника
Направленность (профиль)
« Программное обеспечение средств вычислительной техники и автоматизированных систем»

Ставрополь 2026

Практикум по дисциплине «Технологии программирования и алгоритмизация» для студентов направления подготовки 09.03.01 Информатика и вычислительная техника

Рассмотрены следующие вопросы: разработка программ линейной, разветвляющейся, циклической и смешанной структур; одномерные и двумерные массивы; структуры и строки; функции, стандартные функции языка C++.

В каждой практической работе указывается ее цель, изложены краткие теоретические сведения, описываются этапы выполнения работ и варианты заданий, а также предлагаются контрольные вопросы, которые позволяют студенту произвести самооценку.

СОДЕРЖАНИЕ

Практическое занятие № 1

Программы линейной структуры в языке С++

1. Цель работы: Закрепление знаний об алгоритмах и программах линейной структуры.

2. Основные сведения.

Вначале рассмотрим пример программы линейной структуры на языке С.

Пусть требуется вычислить значение z по формуле

$z = 3(y + 3x) + (x+y)^2$, при этом значения x и y вводятся с клавиатуры. Программа вычисления на языке С с применением форматного ввода-вывода имеет вид:

```
//lab2_1.c  вычисления по формулам
#include<stdio.h>
#include<conio.h>
#include<math.h>
main(){
  float x,y,z;
  printf(«Enter x,y\n»);
  scanf(«%f%f»,&x,&y);
  z = 3*(y+3*x)+(x+y)*(x+y);
  printf("x=%8.2f, y=%8.2f, z=%10.4f",x,y,z);
  getch();
  return 0;
}
```

Дадим краткие пояснения. Первая строка программы – однострочный комментарий, поясняющий назначение программы. Комментарий начинается с двух символов «слэш» подряд и действует до конца строки. Комментарий не влияет на компиляцию программы и предназначен программисту, в том числе и автору программы, т.к. программы довольно быстро забываются.

Хотя комментарий необязателен, настоятельно рекомендуется снабжать комментарием начало программы, чтобы отличить одну программу от другой.

Три следующих строчки программы – директивы включения заголовочных файлов stdio.h (заголовочный файл ввода-вывода), conio.h (от console input-output, заголовочный файл работы с консольным вводом-выводом), math.h (заголовочный файл для работы с математическими функциями).

Следующая строка – описание функции main (главной функции программы). Программа на языке С (и С++) состоит из ряда функций, из которых функция main должна присутствовать обязательно, она служит точкой входа в программу.

Параметры любой функции, в том числе и функции main, заключаются в круглые скобки. Даже если список параметров пуст (как в данном случае), круглые скобки ставятся все равно, чтобы отличить функцию от простой переменной. После списка параметров в круглых скобках записывается тело функции (список выполняемых действий), заключаемое в фигурные скобки.

Строка float x,y,z; содержит описание переменных вещественного типа x, y, z (от float – плавающий, числа с плавающей точкой, могут содержать как целую, так и дробную части). Перед использованием каждой переменной она должна быть описана, т.е. указан ее тип. Переменные типа float (вещественные с одинарной точностью) занимают 4 байта в оперативной памяти и используются в неответственных вычислениях и учебных программах. В ответственных вычислениях используются вещественные переменные с удвоенной точностью типа double, занимающие 8 байтов.

Кроме переменных вещественного типа, в языках С и С++ используются целочисленные переменные типа int. Эти переменные не имеют дробной части и представляются в компьютере точно. Объем занимаемой оперативной памяти для переменных этого типа зависит от реализации. В среде Borland C++ версии 3.1 эти переменные занимают 2

байта, в среде Borland C++ Builder переменные типа int занимают 4 байта. Переменные других типов будут рассмотрены по мере изучения.

Функции printf и scanf содержатся в заголовочном файле stdio.h и служат для вывода на экран и ввода с клавиатуры соответственно. Управляющие символы «\n» (бэкслэш n) в функции printf служат для перевода на новую строку. Последовательности символов, начинающиеся с символа бэкслэш, называются эскейп-последовательностями, так что «\n» – одна из эскейп-последовательностей. Символ «&» (амперсанд) в функции scanf указывает на адрес переменной.

Оператор printf(“Enter x,y”); означает вывод на экран текста «Enter x,y», т.е. приглашения ко вводу x и y, следующая строка scanf(“%f%f”,&x,&y); – ввод с клавиатуры значений x и y и занесение их по адресам этих переменных (для этого используется символ амперсанд).

Оператор z = 3 * (y+3*x) + (x+y)*(x+y); присваивает переменной z значение $3(y + 3x) + (x+y)^2$, звездочка означает знак умножения.

Оператор printf(“x=%8.2f, y=%8.2f, z=%10.4f”,x,y,z); служит для вывода значений x, y, z . В форматной строке, заключенной в кавычки, указываются форматы вывода на экран: %8.2f %8.2f %10.4f переменных x, y, z соответственно, первые 2 переменных выводятся в 8 позициях с двумя знаками после десятичной точки, переменная z в10 позициях с 4-мя знаками после десятичной точки, форматы указываются после символа «процент».

Символ «f» означает вывод в виде числа с плавающей точкой (float). Остальные символы в форматной строке выводятся на экран без изменения. Поэтому вначале будет выведен текст «x=», затем значение x в формате “%8.2f”, далее текст «y=», далее значение y в формате “%8.2f”, затем текст «z=» и наконец значение z в формате “%10.4f”.

Функция getch, содержащаяся в заголовочном файле conio.h, ожидает нажатия любой клавиши, при этом пользователь имеет возможность просмотреть результаты работы программы. Без использования этой функции программа завершит работу, произойдет переход к показу текста

программы, и пользователь не сможет увидеть результаты (экран с результатами покажется лишь на мгновение).

Все функции, кроме функций типа void, возвращают значение определенного типа, для чего служит оператор return a, где a – возвращаемое значение. Если у функции не указан тип (как в данном случае), то по умолчанию функция имеет тип int. Возвращаемое значение 0 означает, что работа программы прошла без ошибок.

Контрольный пример для приведенной программы. Пусть x=3,4; y=2,5. В ответ на приглашение к вводу данных “Enter x,y” вводим 3.4_2.5 и нажимаем ENTER. (Здесь знак подчеркивания означает пробел). Разделителем целой и дробной частей является точка.

Результат работы программы имеет вид:

x=_ _ _ _3.40, _ y=_ _ _ 2.50 , z= _ _ _72.9100

Здесь также знаки подчеркивания означают пробелы.

Этот результат следует проверить с помощью калькулятора Windows.

Все операторы, как выполняющие определенные действия, так и служащие для описания объектов программы (переменных, констант, типов и т.п.), должны завершаться точкой с запятой.

Та же программа на языке C++ с применением потокового ввода-вывода имеет вид:

```
//lab2_2.cpp  вычисления по формулам
#include<iostream.h>
#include<conio.h>
#include<math.h>
main(){
  float x,y,z;
  cout<<”Enter x,y\n”;
  cin>>x>>y;
  z = 3*(y+3*x)+(x+y)*(x+y);
  cout<<”x= “<<x<<”  y=”<<y<<”  z=”<<z<<endl;
```

```
  getch();
  return 0;
}
```

Отличие программы lab2_2.cpp от программы lab2_1.c в том, что в ней применяется потоковый ввод вывод, содержащийся в файле iostream.h. Выходной поток cout – это экран дисплея, входной поток cin – клавиатура. Лексемы “<<” и “>>” указывают направление ввода-вывода: в выходной поток cout из переменных и в переменные из входного потока cin. Потоковый ввод-вывод является бесформатным, компилятор сам определяет формат по типу переменной.

Текстовая константа «endl» (endline, конец строки) означает, как и эскейп-последовательность «\n», переход на новую строку. Если выводится текстовая константа, заканчивающаяся переходом на новую строку, можно применить «\n», если же выводятся значения переменных, после которых следует переход на новую строку, следует применить endl. Ввод-вывод каждого аргумента ввода-вывода должен сопровождаться лексемами “<<” или “>>”.

3. Выполнение работы

3.1. Набрать и откомпилировать оба приведенных варианта программы. Результаты проверить с помощью калькулятора Windows.

3.2. Разработать программы линейной структуры для вычислений по формулам на языках C и C++ согласно вариантам заданий.

4. Варианты заданий

Вычислить значение функции при заданных значениях параметров:

1. $x=2y+3t-z$ при $y=2$; $t=5/(1+y^2)$; $z=4$.
2. $x=3y^2/(4z-2t^2)$ при $t=0.5$; $z=6$; $y=t+2z$.
3. $x=4y^2/(4z-2t^3)$ при $t=1$; $z=3$; $y=2 + t$.
4. $x=4y^3-z/t$ при $t=2$; $z=3$; $y=(t+z)$.
5. $x=6t^2-(z+1)/y^2$ при $y=2$; $z=4$; $t=(2+z)$.
6. $x=(8z^2+1)/(y+t^2)$ при $z=1$; $t=2$; $y=t+z$.

7. $x=6t^3-3z^2/(y+1)$ при t=2; z=t+1; y=3.

8. $x=8z/(t+3)-y^2$ при t=1; z=t+2; y=4.

9. $x=6y^2+3(z-t)^3$ при t=2; z=t+1; y=3.

10. $x=6t^2-(z+1)/(y+1)$ при t=0.5; z=6; y=t+2z.

5. Контрольные вопросы

1. Что такое алгоритм линейной структуры? программа линейной структуры?

2. В каком заголовочном файле содержатся математические функции?

3.Как вывести вещественное число в поле с заданным числом позиций с заданным числом десятичных знаков после точки?

Практическое занятие №2

Условный оператор в языке С++

1. Цель работы: Закрепление знаний о программах разветвляющейся структуры, составление программы с условным оператором и работа с ней.

2. Основные сведения

Алгоритм разветвляющейся структуры - это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия. Программа разветвляющейся структуры реализует такой алгоритм. В программе разветвляющейся структуры имеется один или несколько условных операторов.

Условный оператор в языке С имеет формат:

if (условие) оператор1; else оператор2; (полная форма) или

if (условие) оператор1; (сокращенная форма).

Условие заключается в круглые скобки, при его выполнении исполняется оператор1, при невыполнении - оператор2 (при полной форме условного оператора). Для неполной формы условного оператора при выполнении условия исполняется оператор1, в противном случае оператор1 пропускается и исполняется оператор, следующий за условным оператором.

Оператор1 и оператор2 могут представлять простые операторы (один оператор), в этом случае они не заключаются в скобки. Если же оператор1 и/или оператор2 представляют составной оператор (несколько операторов), то их нужно заключить в фигурные скобки.

В качестве примера приведем программу вычисления функции по различным математическим выражениям на различных интервалах.

```
//razvetvl.c  программа разветвляющейся структуры
#include <stdio.h>
#include <conio.h>
main()
{float x,y;
```

```
printf(”Введите x\n”);
scanf(“%f“,&x);
if (x<=0) {y=x*x+1;goto m;}
else if (x<2) {y=x+1;goto m;}
else y=7-x*x;
m:  printf(“ x=%8.2f y=%8.2f\n”,x,y);
getch();
return 0; }
```

.

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу вычисления заданной функции, исправить выявленные ошибки. Ввести несколько вариантов значений аргумента (в различных интервалах), вычислить функцию вручную и сравнить с полученными по программе результатами.

Составить программу разветвляющейся структуры с применением условного оператора согласно вариантам заданий, откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Варианты заданий

Вычислить значение функции, заданной различными аналитическими выражениями на различных интервалах:

Варианты 1, 6 y =

$$\begin{cases} x^2 \text{ при } x <= -3, \\ 1+x^3 \text{ при } -3 \le x < 5, \\ x^2/(3+4x) \text{ при } x \ge 5. \end{cases}$$

Варианты 2, 7 y =

$$\begin{cases} 3x + 4 \text{ при } x < 1, \\ x^2/(6+x) \text{ при } 1 \le x < 4, \\ 3x^3 + 8 \text{ при } x \ge 5. \end{cases}$$

Варианты 3, 8 y =

$$\begin{cases} (6 - x^2) / 3 \text{ при } x \le 2, \\ x^2 / (3 + 4x^2) \text{ при } 2 < x \le 4, \\ 6x^3 - 4x \text{ при } x > 4. \end{cases}$$

Варианты 4 , 9 . y =

$$\begin{cases} 6x^2 + 3 / (4 - x) \text{ при } x < -2, \\ x^2 / (3 + x^2) \text{ при } -2 \le x < 3, \\ (2x^3 + 4) / (5x^2 - 2) \text{ при } x \ge 3. \end{cases}$$

Варианты 5, 10 y =

$$\begin{cases} 4x^2 + 6 \text{ при } x <– 1, \\ x^3 / (5 + x) \text{ при } – 1 \le x < 2, \\ 4x^2 – 3 \text{ при } x \ge 2. \end{cases}$$

5. Контрольные вопросы

1. Какой алгоритм является алгоритмом разветвляющейся структуры?

2. По программе раздела 2 составьте выражения для функции на интервалах.

3. Что такое условный оператор? Полная и сокращенная формы.

4. Что такое составной оператор? Формат его записи.

5. Что такое оператор безусловного перехода?

6. Найдите операторы безусловного перехода в примере программы.

7. Можно ли обойтись без операторов безусловного перехода?

Практическое занятие №3

Переключатель в языке С++

1. Цель работы: Закрепление знаний о переключателе, составление программы с переключателем и работа с ней.

2. Основные сведения

В ряде случаев разветвление программы в зависимости от значения некоторой переменной требуется произвести не на 2 , а на большее число ветвей. Например, определим время года по введенному номеру месяца.

```
//switch.c   множественный выбор
 #include<stdio.h>
 #include<conio.h>
 int n;
 main() {
 printf(“Введите номер месяца\n”);
 scanf(“%d”,&n);
 printf(“\nВремя года: “);
 switch(n){
 case 1:case 2:case 12:printf(“Зима\n”);break;
 case 3:case 4:case 5:printf(“Весна\n”);break;
 case 6:case 7:case 8:printf(“Лето\n”);break;
 case 9:case 10:case 11:printf(“Осень\n”);break;
 default:printf(“\nНомер месяца неверен\n”);}
 getch();
 return 0;}
```

В приведенном примере программы при вводе номера месяца от 1 до 12 на экране печатается соответствующее время года, если же номер месяца превышает 12, выводится сообщение о неверном вводе месяца, для чего служит зарезервированное слово языка default. После каждой из констант, перечисляющих значение переключателя, ставится двоеточие. Заключительный для каждой ветви оператор break служит для прерывания

цикла проверки и перехода в конец переключателя. В случае отсутствия break переключатель работает неверно - происходит переход на следующую ветвь.

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу с применением переключателя. Исправить выявленные ошибки. Ввести несколько значений номера месяца в этой программе, убедиться в правильности работы программы.

Составить программу с переключателем согласно вариантам заданий, откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Варианты заданий

1. Вводится число экзаменов N<=20. Напечатать фразу ”Мы успешно сдали N экзаменов”, согласовав слово «экзамен» с числом N.

2. Вводится число - номер месяца. Вывести количество дней в месяце (год невисокосный).

3. Вводится число лет (N<=25).Напечатать фразу “Мне N лет”, согласовав слово «лет» с числом N.

4. Вводится число книг N<=10. Вывести фразу “Я взял из библиотеки N книг”, согласовав слово «книга» с числом N.

5. Вводится число карандашей N<=10. Вывести фразу “Я купил N карандашей”, согласовав слово «карандаш» с числом N.

6. Вводится число версий N<=10.Вывести фразу “Следователь проверил N версий”, согласовав слово «версия» с числом N.

7. Вводится число программ N<=10.Напечатать фразу “Я разработал N программ”, согласовав слово «программа» с числом N.

8. Вводится целое число -9<=c<=9.Вывести величину числа в словесной форме с учетом знака.

9. Вводится число вершин N<=10. Вывести фразу “Мы покорили N вершин”, согласовав слово «вершина» с числом N.

10. Вводится число групп N<=10. Вывести фразу “На факультете N групп”, согласовав слово «группа» с числом N.

5. Контрольные вопросы

1. Что такое множественный выбор?

2. В каких случаях применяется переключатель?

3. Зачем ставится в переключателе оператор break?

4. Зачем в переключателе употребляется зарезервированное слово default?

5. Блок-схема переключателя.

Практическое занятие № 4

Циклы. Цикл с параметром

1. Цель работы: Закрепление знаний о программах циклической структуры, составление программы цикла с параметром и работа с ней.

2. Основные сведения

Алгоритм циклической структуры - это алгоритм, в котором происходит многократное повторение одного и того же участка программы. Такие повторяемые участки вычислительного процесса называются циклами.

Программа циклической структуры содержит один или несколько циклов. Различают детерминированные циклы с заранее известным числом повторений и итерационные циклы, в которых число повторений заранее неизвестно.

Для организации цикла необходимо выполнить следующие действия:

1) задать перед циклом начальное значение переменной, изменяющейся в цикле;

2) изменять переменную перед каждым новым повторением цикла;

3) проверять условие повторения цикла;

4) управлять циклом, т.е. переходить к его началу, если он не закончен, или выходить из него по окончании. Изменяющаяся в цикле переменная называется параметром цикла, в одном цикле возможны несколько параметров.

В языке С существует 3 вида циклов: 1) цикл с параметром или цикл типа **for**, 2) цикл с предусловием или цикл типа **while**, 3) цикл с постусловием или цикл типа **do ... while**. В цикле типа for число повторений известно заранее, в циклах типа while и do ... while число повторений цикла заранее неизвестно, производится проверка условия повторения цикла: в цикле типа while - перед циклом, в цикле типа do ... while - после его окончания.

В циклах типов for и while повторяющаяся часть (тело цикла) состоит из одного оператора, если требуется выполнить в цикле несколько операторов, они заключаются в фигурные скобки, образуя составной оператор. В цикле типа do ... while тело цикла помещается между зарезервированными словами языка (лексемами) do и while, фигурные скобки также требуются, в названии цикла его тело условно обозначается тремя точками.

Во всех типах циклов условие продолжения цикла заключается в круглые скобки. Для цикла типа for заголовок цикла состоит из трех разделов: инициализации (присваивания начальных значений), проверки условия повторения, модификации (изменения параметров). Разделителем между разделами заголовка цикла типа for служит точка с запятой.

С помощью цикла типа for удобно находить суммы, произведения, искать максимальные и минимальные значения и т.п. При нахождении суммы некоторой переменной, например S присваивается значение 0, затем в цикле к этой переменной прибавляется соответствующий член заданной последовательности. Рассмотрим пример вычисления суммы квадратов натурального ряда чисел от 1 до n.

$S = 1^2 + 2^2 + \ldots + n^2$.

Программа имеет вид:

```
//sum_sq.c  сумма квадратов натурального ряда
 #include<stdio.h>
 #include<conio.h>
 main()
 {   int S,n,i;
   clrscr();
   printf(“Введите n\n”);
   scanf(“%d”,&n);
   for(S=0,i=1;i<=n;i++) S+=i*i;
   printf(“n=%d S=%d\n”,n,S);
```

```
getch();
return 0; }
```

В программе определяются целые переменные S, n, i, в отличие от других языков в языках C и C++ учитывается регистр при определении идентификатора (имени) переменной, так что S и s - разные переменные. Функция clrscr производит очистку экрана, содержится в заголовочном файле conio.h. После ввода переменной n следует цикл типа for. В разделе инициализации присваиваются начальные значения переменным S (в которой накапливается сумма квадратов) и i - параметру цикла, присваивания разделяются запятой, раздел инициализации отделяется точкой с запятой. В разделе проверки условия значение i сравнивается с n, при i меньшем или равном n цикл повторяется, в противном случае происходит выход из цикла. Раздел модификации состоит из оператора инкремента (увеличения на 1) значения параметра цикла i. Запись i++ эквивалентна i=i+1.Аналогично в операции декремента имеем i-- ⇔ i=i - 1.

В цикле выполняется один оператор S+=i*i; это оператор составного присваивания, эквивалентный оператору S=S+i*i, поэтому фигурные скобки для тела цикла не требуются. Операторы инкремента, декремента и составного присваивания придают языку C++ лаконичность , позволяют уменьшить объем исходного текста (C++ - самый лаконичный из языков высокого уровня).

После окончания цикла производится печать результата (оператор printf). Функция getch, содержащаяся в заголовочном файле conio.h, ожидает нажатия любой клавиши, при этом пользователь имеет возможность просмотреть результаты работы программы, не прибегая к просмотру экрана пользователя с помощью комбинации [ALT]+[F5]. Как указывалось выше, при записи функций без параметров clrscr и getch следует ставить круглые скобки даже с пустым списком параметров.

Вычисление произведений производится аналогично, но перед циклом переменной, например S, в которой накапливается значение произведения,

присваивается значение не 0, а 1. Здесь также возможен оператор составного присваивания, который при вычислении факториала имеет вид S*=i; что эквивалентно S=S*i. Вообще, оператор составного присваивания применяется и для других бинарных операций (операций с двумя операндами).

3. Выполнение работы

Набрать и откомпилировать приведенные выше программы с применением цикла типа FOR, исправить выявленные ошибки. Ввести несколько вариантов значений аргумента в этих программах, вычислить результат вручную и сравнить с полученными по программе результатами.

Составить программы циклической структуры согласно вариантам заданий, откомпилировать их, ввести исходные данные, проверить полученный результат.

4. Варианты заданий

1. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(1+1^2)}{(2+1)} + \frac{(1+2^2)}{(2+2)} + \ldots + \frac{(1+N^2)}{(2+N)} .$$

2. Вводится целое число N>0. Вычислить произведение

$$P = \frac{1}{(1+1^2)} * \frac{1}{(1+2^2)} * \ldots * \frac{1}{(1+N^2)} .$$

3. Вводится целое число N>0. Вычислить произведение

$$P = \frac{(1+1)}{(1+1^2)} * \frac{(1+2)}{(1+2^2)} * \ldots * \frac{(1+N)}{(1+N^2)} .$$

4. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(1+1)}{(2+1^2)} + \frac{(1+2)}{(2+2^2)} + \ldots + \frac{(1+N)}{(2+N^2)} .$$

5. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(1+1^2)}{(2+1^3)} + \frac{(1+2^2)}{(2+2^3)} + \ldots + \frac{(1+N^2)}{(2+N^3)} .$$

6. Вводится целое число N>0. Вычислить произведение

$$P = \frac{2}{(1+1^2)} * \frac{2}{(1+2^2)} * \ldots * \frac{2}{(1+N^2)} .$$

7. Вводится целое число N>0. Вычислить произведение

$$P = \frac{(1+1^3)}{(1+1^2)} * \frac{(1+2^3)}{(1+2^2)} * \ldots * \frac{(1+N^3)}{(1+N^2)} .$$

8. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(2+1^2)}{(1+1^3)} + \frac{(2+2^2)}{(1+2^3)} + \ldots + \frac{(2+N^2)}{(1+N^3)} .$$

9. Вводится целое число N>0. Вычислить произведение

$$P = \frac{(3+1^3)}{(2+1^2)} * \frac{(3+2^3)}{(2+2^2)} * \ldots * \frac{(3+N^3)}{(2+N^2)} .$$

10. Вводится целое число N>0.Вычислить сумму N членов последовательности

$$S = \frac{(3+1^2)}{(2+1^3)} + \frac{(3+2^2)}{(2+2^3)} + \ldots + \frac{(3+N^2)}{(2+N^3)} .$$

5. Контрольные вопросы

1. Какой алгоритм является алгоритмом циклической структуры?

2. Типы циклов в языке С.

3. Какие три раздела записываются в круглых скобках для оператора цикла типа for? Какой разделитель между разделами?

4. Что такое вложенные циклы? Примеры.

Практическое занятие № 5

Цикл с предусловием

1. Цель работы: Закрепление знаний о программах цикла с предусловием, составление программы цикла с предусловием и работа с ней.

2. Основные сведения

Не всегда число повторений цикла известно заранее, в этих случаях применяются циклы с предусловием (проверка перед циклом) или с постусловием (проверка после цикла).Приведем примеры.

Программа вычисления факториала числа n с применением цикла с предусловием использует формулу

n! = 1*2* . . . *n

т.е. факториал числа n есть произведение всех целых чисел от 1 до n.

```
//fact.c  вычисление факториала
 #include<stdio.h>
 #include<conio.h>
 main()
 {   int F,n,i;
   clrscr();
   printf(“Введите n\n”);
   scanf(“%d”,&n);
   F=1; i=1;
   while(i<n) {F*=i; i++}
   printf(“n=%d n!=%d\n”,n,F);
   getch();
   return 0;}
```

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу с применением цикла типа WHILE, исправить выявленные ошибки. Ввести несколько вариантов значений аргумента в этой программе, вычислить результат вручную и сравнить с полученным по программе результатом.

Составить программу циклической структуры с применением цикла типа WHILE согласно вариантам заданий лабораторной работы №5 откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Контрольные вопросы

1. Типы циклов в языке С.

2. Как записывается условие продолжения цикла в цикле типа while ?

3. Что такое вложенные циклы? Примеры.

4. Может ли цикл с предусловием не исполниться ни разу? В каком случае это произойдет?

Практическое занятие № 6

Цикл с постусловием

1. Цель работы: Закрепление знаний о программах цикла с постусловием, составление программы цикла с постусловием и работа с ней.

2. Основные сведения

В программах с применением цикла с постусловием проверка условия повторения цикла производится после цикла, поэтому цикл с постусловием выполнится по крайней мере один раз.

Программа вычисления двойного факториала числа n с применением цикла с постусловием использует формулу

n!!=1*3*5* . . . *n при n нечетном или n!!=2*4*6* . . . *n при n четном.

```
//dfact.c  вычисление двойного факториала
 #include<stdio.h>
 #include<conio.h>
 main()
 {int F,n,i;
   clrscr();
   printf("Введите n\n");
   scanf("%d",&n);
   F=1;
   if(n%2==0) i=0; else i=-1;//условие четности
   do   { i=i+2;
 F*=i;}
   while(i<n) ;
   printf("n=%d n!!=%d\n",n,F);
   getch();
   return 0;}
```

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу с применением цикла типа DO…WHILE, исправить выявленные ошибки.

Ввести несколько вариантов значений аргумента в этой программе, вычислить результат вручную и сравнить с полученным по программе результатом.

Составить программу циклической структуры с применением цикла типа DO…WHILE согласно вариантам заданий лабораторной работы №5 откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Контрольные вопросы

1. Какой алгоритм является алгоритмом циклической структуры?
2. Типы циклов в языке С.
3. Как записывается условие продолжения цикла в цикле типа do ... while?
4. Что такое вложенные циклы? Примеры.
5. Может ли цикл с постусловием не исполниться ни разу? Почему?

Практическое занятие № 7

Одномерные массивы

1. Цель работы: Закрепление знаний о массивах, составление программ с одномерными массивами.

2. Основные сведения

Массивы - структурированный тип данных с элементами одного типа. Массивы в языке С описываются так же, как простые переменные, но после имени массива в квадратных скобках указывается число его элементов. Например, массив составляют заработные платы сотрудников подразделения предприятия, здесь число элементов равно числу сотрудников, массив образуют максимальные дневные температуры в течение года, число элементов массива - число дней в году. Номер элемента массива называется его индексом, в языках С и С++ индекс элемента массива начинается с нуля, поэтому индекс последнего элемента массива на 1 меньше числа элементов в данном массиве.

Приведем пример программы для нахождения наибольшего элемента в массиве и его индекса.

```
//poiskmax.c  поиск макс. эл-та в массиве.
 #include<stdio.h>
 #include<conio.h>
 #define n 10
 int i,max,k,a[n];//размер массива n
 main() {
 max=a[0];k=0;
 for(i=0;i<n;i++) scanf("%d",&a[i]);
 i=0;
 for(i=0;i<n;i++)
  {if(max<a[i])
   {k=i;max=a[i];}}
 printf("Максимальное число в массиве %d, его индекс %d\n",max,k);
```

```
getch();
return 0;}
```

Здесь директива #define n 10 определяет константу n=10, далее мы описываем массив a с числом элементов n. Индексы элементов массива изменяются от 0 до 9.При задании элемента массива в операторе используется имя массива и значение индекса в квадратных скобках. Первый цикл по i в программе - поэлементный ввод массива a. Затем следует поиск максимального элемента в массиве, также в цикле. Переменная max сравнивается с элементами массива, и если элемент массива больше max, переменной max присваивается значение элемента массива, а переменной k - индекс этого элемента. По окончании цикла переменная max будет иметь значение, равное максимальному элементу массива, а переменная k - значение индекса этого элемента (на 1 меньше его номера в массиве).

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу, исправить выявленные ошибки. Ввести элементы массива, убедиться в правильности нахождения максимального элемента.

Составить программу с применением массивов согласно вариантам заданий, откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Варианты заданий

1. Ввести массив A из 10 элементов, найти наибольший элемент и переставить его с первым элементом. Преобразованный массив вывести.

2. Ввести массив A из 10 элементов, найти наименьший элемент и переставить его с последним элементом. Преобразованный массив вывести.

3. Ввести массив A из 10 элементов, найти произведение положительных элементов и вывести его на экран.

4. Ввести массив A из 10 элементов, найти произведение отрицательных элементов и вывести его на экран.

5. Ввести массив A из 10 элементов, найти сумму положительных элементов и вывести ее на экран.

6. Ввести массив A из 10 элементов, найти сумму отрицательных элементов и вывести ее на экран.

7. Ввести массив A из 10 элементов, найти сумму элементов, больших 3 и меньших 8 и вывести ее на экран.

8. Ввести массив A из 10 элементов, найти сумму элементов, меньших по модулю 5 и вывести ее на экран.

9. Ввести массив A из 10 элементов, найти сумму элементов, больших 2 и меньше 5, вывести ее на экран.

10. Ввести массив A из 10 элементов, найти сумму элементов, больших 4 и меньших 7 и вывести ее на экран.

5. Контрольные вопросы

1. Что такое массив?
2. Что такое индекс элемента массива?
3. В каких пределах изменяются индексы элементов массива?
4. Для чего нужна директива define?
5. Как ввести и вывести элемент массива?

Практическое занятие № 8

Многомерные массивы

1. Цель работы: Закрепление знаний о многомерных массивах, составление программ с многомерными массивами.

2. Основные сведения

Массив может иметь не один, а большее число индексов. Число индексов называется размерностью массива, например, массив с двумя индексами называется двумерным массивом. Таким двумерным массивом является, в частности, матрица системы n линейных алгебраических уравнений с n неизвестными. В то же время столбец свободных членов этой системы является одномерным массивом.

По каждому из индексов устанавливается размер массива, например , описание двумерного массива из четырех строк и трех столбцов имеет вид: float a[4][3]. Такой массив имеет 3*4=12 элементов, первый индекс изменяется от 0 до 3, второй индекс - от 0 до 2.Таким образом, элементы массива суть: a[0][0] , a[0][1], a[0][2] , a[1][0] , ... , a[3][2].

Как правило, при обработке многомерных массивов используются вложенные циклы, т.е. цикл по столбцам располагается внутри цикла по строкам. Таким образом производится ввод и вывод элементов массива, поиск максимальных элементов в строках и т.д.

Приведем пример работы с многомерными массивами. В примере вводится матрица 2x3, элементы 2-го и 3-го столбцов умножаются на 2, преобразованная матрица выводится на экран.

```
//mnog_mas.c многомерные массивы
#include<stdio.h>
 #include<math.h>
 main() {
 int a[2][3],i,j;
 for(i=0;i<2;i++) {
 for(j=0;j<3;j++) {
```

```
printf(“Введите a[%d][%d]=”,i,j);
scanf(“%d”,&a[i][j]);}}
for(i=0;i<2;i++)
for(j=1;j<3;j++) a[i][j]*=2;
for(i=0;i<2;i++) {
for(j=0;j<3;j++)
printf(“%6d”,a[i][j]);
printf(“\n”);}
return 0;}
```

В данном примере при вводе цикл по i является внешним, цикл по j - внутренним, вложенным в цикл по i. При вводе параметр цикла i изменяется от 0 до 2 , внутренний параметр j изменяется от 0 до 3. При умножении i изменяется от 0 до 2 , а параметр j от 1 до 3 , т.к. первый столбец умножать на 2 не нужно.

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу , исправить выявленные ошибки. Ввести элементы массива, убедиться в правильности его обработки.

Составить программу с применением многомерных массивов согласно вариантам заданий, откомпилировать ее, ввести исходные данные , проверить полученный результат.

4. Варианты заданий

1. Ввести матрицу A из 3x3 элементов и вектор B из 3 элементов. Переставить на главную диагональ матрицы элементы вектора, а в вектор - элементы главной диагонали. Преобразованные массивы вывести на экран.

2. Ввести матрицу A из 3x3 элементов, найти наименьший элемент и от элементов всех столбцов, за исключением первого, вычесть наименьший элемент. Преобразованную матрицу вывести.

3. Ввести массив B из 3x4 элементов, найти произведения элементов каждого столбца и вывести их на экран.

4. Ввести матрицу A из 3x3 элементов, найти наибольший элемент и от элементов всех строк, за исключением третьей, вычесть наибольший элемент. Преобразованную матрицу вывести.

5. Ввести массив B из 3x4 элементов, найти суммы элементов каждой строки и вывести их на экран.

6. Ввести матрицы A и B из 3x3 элементов, найти матрицу C, элементы которой равны суммам соответствующих элементов матриц A и B и вывести ее на экран.

7. Ввести матрицы A и B из 2x3 элементов, найти матрицу C, элементы которой равны разностям соответствующих элементов матриц A и B и вывести ее на экран.

8. Ввести массив A из 4 элементов, расположить эти элементы на главной диагонали матрицы B размером 4x4,остальные элементы матрицы B положить равными 1. Вывести матрицу B на экран.

9. Ввести матрицу A из 3x3 элементов, найти наименьший элемент и от элементов всех столбцов, за исключением третьего, вычесть наименьший элемент. Преобразованную матрицу вывести.

10. Ввести массив B из 3x3 элементов, найти суммы элементов каждого столбца и вывести их на экран.

5. Контрольные вопросы

1. Что такое многомерный массив?

2. Сколько индексов может быть у элемента многомерного массива?

3. В каких пределах изменяются индексы элементов массива?

4. Как ввести элементы многомерного массива? Сколько нужно циклов?

5. Где применяются многомерные массивы?

Практическое занятие № 9

Символьные данные

1. Цель работы: Закрепление знаний о символьных данных, составление программ с применением символьных данных.

2. Основные сведения

Как известно, данные в компьютере представляются с помощью комбинаций двоичных цифр: нуля и единицы. Как же представить символы, например, буквы русского или английского алфавитов?

Для решения этой задачи разработаны специальные таблицы, называемые кодовыми таблицами. В такой таблице определяется, какой комбинацией единиц и нулей задается определенный символ алфавита.

В США в качестве кодовой таблицы была принята таблица ASCII (American Standard Code for Information Interchange, Американский стандартный код для обмена информацией). В качестве основной единицы представления символьной информацией был принят 1 байт, 8 двоичных разрядов. В этом представлении можно определить $2^8 = 256$ различных символов.

Таблица ASCII состоит из двух частей. В первой части (символы с кодами 0 – 127) представлены цифры, символы латиницы (большие и малые буквы латинского алфавита), знаки препинания, управляющие символы (с кодами 0 – 31), например, переход на новую строку, и другие символы.

Во второй части таблицы ASCII (символы с кодами 128 – 255) представлены символы псевдографики (для графления таблиц), некоторые другие символы, например, символы суммы и интеграла, буквы с диакритическими знаками.

Первая половина кодовой таблицы ASCII используется в качестве международной, вторая половина кодовой таблицы в разных странах используется для представления символов национальных алфавитов, поэтому вторая половина кодовой таблицы может быть различной в разных странах.

В России для представления символов кириллицы (букв русского алфавита) использовались две кодировки: кодировка ГОСТ и альтернативная кодировка. В большинстве случаев применяется альтернативная кодировка. В таблице 1 дана международная часть кодовой таблицы ASCII.

Таблица 1. Международная часть таблицы ASCII.

Код	Символ	Код	Символ	Код	Символ	Код	Символ
32		56	8	80	P	104	h
33	!	57	9	81	Q	105	i
34	“	58	:	82	R	106	j
35	#	59	;	83	S	107	k
36	$	60	<	84	T	108	l
37	%	61	=	85	U	109	m
38	&	62	>	86	V	110	n
39	‘	63	?	87	W	111	o
40	(	64	@	88	X	112	p
41	)	65	A	89	Y	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[	115	s
44	,	68	D	92	\	116	t
45	-	69	E	93	]	117	u
46	.	70	F	94	^	118	v
47	/	71	G	95	_	119	w
48	0	72	H	96	`	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	{
52	4	76	L	100	d	124	|

53	5	77	M	101	e	125	}
54	6	78	N	102	f	126	~
55	7	79	O	103	g	127	

Символы с кодами 128 – 255 в альтернативной кодировке, включая символы кириллицы, приведены в таблице 2.

Таблица 2. Символы альтернативной кодировки.

Код	Символ	Код	Символ	Код	Символ	Код	Символ
128	А	160	а	192	└	224	р
129	Б	161	б	193	┴	225	с
130	В	162	в	194	┬	226	т
131	Г	163	г	195	├	227	у
132	Д	164	д	196	─	228	ф
133	Е	165	е	197	┼	229	х
134	Ж	166	ж	198	╞	230	ц
135	З	167	з	199	╟	231	ч
136	И	168	и	200	╚	232	ш
137	Й	169	й	201	╔	233	щ
138	К	170	к	202	╩	234	ъ
139	Л	171	л	203	╦	235	ы
140	М	172	м	204	╠	236	ь
141	Н	173	н	205	═	237	э
142	О	174	о	206	╬	238	ю
143	П	175	п	207	╧	239	я
144	Р	176	▒	208	╨	240	Ё
145	С	177	▓	209	╤	241	ё
146	Т	178	▓	210	╥	242	≥
147	У	179	│	211	╙	243	≤
148	Ф	180	┤	212	╘	244	⌠
149	Х	181	╡	213	╒	245	⌡
150	Ц	182	╢	214	╓	246	÷
151	Ч	183	╖	215	╫	247	≈
152	Ш	184	╕	216	╪	248	⎝
153	Щ	185	╣	217	┘	249	·
154	Ъ	186	║	218	┌	250	□
155	Ы	187	╗	219	▓	251	√
156	Ь	188	╝	220	▄	252	№
157	Э	189	╜	221	▌	253	¤
158	Ю	190	╛	222	▐	254	▪

159	Я	191	╗	223	■	255	

В операционной системе Windows отсутствуют символы псевдографики, т.к. начертание линий производится другими средствами. В этой операционной системе принята кодовая таблица Windows 1251. Символы кириллицы имеют здесь коды 192 –255 (без букв ё), Ё имеет код 168, ё – код 184:

Таблица 3. Символы кириллицы в кодировке Windows 1251.

Код	Символ	Код	Символ	Код	Символ	Код	Символ
168	Ё	207	П	224	а	241	с
184	ё	208	Р	225	б	242	т
192	А	209	С	226	в	243	у
193	Б	210	Т	227	г	244	ф
194	В	211	У	228	д	245	х
195	Г	212	Ф	229	е	246	ц
196	Д	213	Х	230	ж	247	ч
197	Е	214	Ц	231	з	248	ш
198	Ж	215	Ч	232	и	249	щ
199	З	216	Ш	233	й	250	ъ
200	И	217	Щ	234	к	251	ы
201	Й	218	Ъ	235	л	252	ь
202	К	219	Ы	236	м	253	э
203	Л	220	Ь	237	н	254	ю
204	М	221	Э	238	о	255	я
205	Н	222	Ю	239	п		
206	О	223	Я	240	р		

В кодовой таблице ASCII (и в других кодовых таблицах, в которых на символ отводится 1 байт) можно представить 256 различных символов. Этого достаточно для буквенных алфавитов, но недостаточно для азиатских языков

с иероглифической письменностью (количество иероглифов составляет несколько тысяч). Поэтому была введена кодировка Unicode, в которой на каждый символ выделяется не 1 байт, а 2 байта, 16 двоичных разрядов. В кодировке Unicode можно представить $2^{16} = 65536$ различных символов, что достаточно для всех языков, в том числе и с иероглифической письменностью.

В языке C++ символьные данные имеют типы char или unsigned char и занимают 1 байт оперативной памяти. Кодировка зависит от реализации компилятора.

В среде программирования Borland C++ Builder в режиме консольного приложения используется кодировка ASCII, русские буквы записываются в альтернативной кодировке (таблица 2). В режиме оконного приложения используется кодировка Windows 1251.

Ввод символьных данных в C++ Builder производится в кодировке Windows 1251, а вывод в консольном приложении – в альтернативной кодировке. Поэтому при вводе-выводе символьных и строковых данных в режиме консольного приложения будет иметь место несоответствие введенных и выводимых данных.

Символы латиницы (латинские буквы) имеют одни и те же коды как в альтернативной кодировке, так и в кодировке Windows 1251, поэтому латинские буквы в режиме консольного приложения вводятся и выводятся правильно. Русские же буквы имеют в этих таблицах разные коды, поэтому при воде русских букв на экран выводятся символы псевдографики.

В режиме оконных приложений Windows и ввод и вывод осуществляется по кодовой таблице Windows 1251, поэтому несоответствия вводимых и выводимых данных нет.

В языке C строка определяется как массив символов, в конце строки ставится символ с кодом 0. Такие строки называются нуль-терминальными строками, т.е. строками, в которых признак конца строки – символ с кодом 0. Этот символ добавляет сам компилятор при формировании строки.

В приводимой ниже программе рассматриваются три строки, из которых одна постоянная, задается в программе, и две вводимых строки. Для постоянной строки, определяемой в программе, требуется перекодировка из кодовой таблицы Windows 1251 в альтернативную кодировку. Для вводимых строк и ввод и вывод осуществляется в альтернативной кодовой таблице, перекодировка не требуется.

```
//lab10.cpp  символьные данные
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
char* cyr(char* S){
unsigned char* r=S;
while(*r){if (*r==168) *r=240;
else if(*r==184) *r=241; else
if (*r>=192) if(*r<=239) *r-=64;
else *r-=16; r++;}
return S;} //cyr
main(){
int i,k;
unsigned char j,s0[]="Постоянная строка",s1[20],s2[20],s00[18];
cout<<cyr("s0 без преобразования")<<endl;
cout<<s0<<endl;
strcpy(s00,s0);
cout<<cyr("s0 с преобразованием")<<endl;
cout<<cyr(s00)<<endl;
cout<<cyr("коды s0 без преобразования")<<endl;
strcpy(s00,s0);
for(j=0;j<=strlen(s00);j++)
printf("%u ",s00[j]);printf("\n");
cout<<cyr("коды s0 с преобразованием")<<endl;
```

```
strcpy(s00,s0); cyr(s00);
for(j=0;j<=strlen(s0);j++)
printf("%u ",s00[j]);printf("\n");
cout<<cyr("Введите строку")<<endl;
gets(s1);
cout<<cyr("Вывод введенной строки")<<endl;
cout<<s1<<endl;
cout<<cyr("Вывод кодов введенной строки")<<endl;
for(j=0;j<=strlen(s1);j++)
printf("%u ",s1[j]);printf("\n");
cout<<cyr("Введите строку")<<endl;
gets(s2);
cout<<cyr("Вывод введенной строки")<<endl;
printf("%s\n",s2);
cout<<cyr("Вывод кодов введенной строки")<<endl;
for(j=0;j<=strlen(s2);j++)
printf("%u ",s2[j]);printf("\n");
getch();
return 0;}
```

В приведенной программе для перекодировки из таблицы Windows 1251 в альтернативную кодировку применяется функция cyr (от кириллица). Параметром функции является указатель на строку, после применения этой функции строка оказывается перекодированной.

Из сравнения таблиц 2 и 3 вытекает алгоритм перекодировки в функции cyr: буквы Ё и ё, имеющие в кодировке 1251 коды 168 и 184, заменяются кодами 240 и 241 соответственно, все большие буквы русского алфавита и первая половина маленьких букв (коды 192 – 239) заменяются кодами, меньшими на 64, вторая половина маленьких букв (коды 240 – 255) заменяется кодами, меньшими на 16.

Строка s0, определенная в программе как «Постоянная строка», при непосредственном выводе (без перекодировки) превращается в набор бессмысленных символов. Если же строку s0 скопировать в строку s00 с помощью стандартной функции strcpy, произвести перекодировку строки s00 с помощью функции cyr и вывести s00 на экран, получим правильный вывод «Постоянная строка».

Выводим коды преобразованной и не преобразованной строки s0, убеждаемся, что не преобразованная строка имеет кодировку 1251, преобразованная – альтернативную кодировку.

Теперь введем строки s1 и s2 с клавиатуры, например, «Моя строка1» и «Моя строка2» с помощью функции gets. Эта функция вводит строку целиком, вместе с пробелами. Выводим строки s1 и s2 с помощью оператора вывода cout и функции.printf. Далее выводим коды символов этих строк. Убеждаемся, что и ввод, и вывод осуществляются в альтернативной кодировке, поэтому применять функцию cyr для преобразования нет необходимости.

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу, исправить выявленные ошибки. Основное внимание обратить на выяснение типа кодовой таблицы при различных определениях строки. Ввести несколько вариантов строк и выяснить полученную кодировку.

Модифицировать приведенную выше программу с заданием различных постоянных и вводимых строк. Описать полученные результаты.

4. Контрольные вопросы

1. Как представляются в компьютере символьные данные?
2. Что такое кодовая таблица?
3. Какие типы кодовых таблиц вы знаете?
4. Для чего нужна перекодировка? В каких случаях она необходима?
5. Принципы построения функции перекодировки.

6. Зачем при выводе на экран сообщения «Вывод введенной строки» применяется функция cyr?

7. Почему при выводе введенной строки не применятся функция cyr?

8. Как вывести коды символов введенной строки?

9. Зачем строка s0 копируется в строку s00?

Практическое занятие № 10

Строки символов

1. Цель работы: Закрепление знаний о строках, составление программ с применением строк.

2. Основные сведения

Строка - это одномерный массив символов, содержащий на один символ больше требуемого числа элементов в строке. Дело в том, что в языке С строка может содержать до 32767 символов, признаком окончания строки является символ с нулевым кодом, входящий в строку.

Строка, как массив символов (тип char *имя* [n]), заканчивающийся символом с нулевым кодом (нуль-терминальная строка, ASCIIZ строка, С-строка) унаследована в языке С++ от языка С. Кроме С-строк, в языке С++ имеется класс string, который рассматривается позднее.

Работа с нуль-терминальными строками проще, чем работа со строками класса string, но следует помнить, что для нуль-терминальных строк отсутствует проверка выхода за границы диапазона (сообщение out of range). Поэтому программист должен позаботиться о том, чтобы не выйти за границы диапазона.

Заголовочный файл string.h в стандартной библиотеке С++ содержит функции работы с С-строками. В то же время в визуальных средах Visual C++ и Borland C++ Builder заголовочный файл string.h содержит функции для работы с классом string. Во избежание путаницы в этих средах заголовочный файл для работы с С-строками носит название cstring.h.

В заголовочном файле cstring.h содержатся следующие функции:

strcmp сравнивает две строки;

strcpy копирует одну строку в другую;

strlen возвращает длину строки;

strstr ищет подстроку в строке.

В приведенном ниже примере программы вводится строка длиной не более 80 символов (strlen = 81 с учетом нуль-символа), содержащая слова, разделяемые пробелами. Требуется подсчитать общее количество слов.

```
//slova.cpp подсчет числа слов
#include<stdio.h>
#include<conio.h>
#include<cstring.h>
main(){ const len=81;
int i,j,k;
char stroka[len];
printf("Enter string\n");
gets(stroka);
k=1; //счетчик слов
j=strlen(stroka);
printf("%s\n",stroka);
for(i=0;i<j;i++) if( stroka[i]= =' ' &&
stroka[i]!=' ') k++ ; // между апострофами пробел
printf("The number of the words %d\n", k);
getch();
return 0;}
```

Обратим внимание, что для ввода строки мы воспользовались функцией gets из заголовочного файла stdio.h. Эта функция вводит строку целиком, включая пробельные символы (пробел, знаки табуляции), в то время как функция scanf вводит строку только до первого пробельного символа. Убеждаемся в этом, заменив строку программы gets(stroka); на строку scanf("%s",&stroka);.

Аналогичный результат получим, применяя входной поток cin и операцию ввода >>, будет ввод до первого пробельного символа.

В программе производится перебор всех символов до конца строки. Если текущий символ есть пробел, а следующий символ – не пробел, то количество слов k увеличивается на 1 (закончилось текущее слово).

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу исправить выявленные ошибки. Ввести строку со словами, разделяемыми пробелами, убедиться в правильности работы программы. Заменить строку с функцией gets на строку с функцией scanf, убедиться, что в этом случае ввод осуществляется до первого пробельного символа.

Разработать и исполнить программу согласно вариантам заданий.

4. Варианты заданий

1. Ввести строку из слов, разделенных пробелами. Подсчитать количество слов, начинающихся с приставки, вводимой с клавиатуры.
2. Ввести строку из слов, разделенных пробелами. Подсчитать количество слов, начинающихся с заданного символа.
3. Ввести строку из слов, разделенных запятыми. Подсчитать количество слов.
4. Ввести строку из слов, разделенных пробелами. Подсчитать количество слов, содержащих цифры.
5. Ввести строку из слов, разделенных пробелами. Подсчитать количество слов, содержащих буквы ‘a’, ‘b’, ‘c’, ‘d’.
6. Ввести строку из слов, разделенных пробелами. Подсчитать количество слов, содержащих 3 буквы.
7. Ввести строку из слов, разделенных пробелами. Подсчитать количество слов, содержащих 2 или 3 буквы.
8. Ввести строку из слов, разделенных пробелами. Подсчитать количество слов, содержащих буквы ‘d’, ‘r’, ‘t’.
9. Ввести строку из слов, разделенных пробелами. Подсчитать количество слов, начинающихся с буквы ‘z’.

10. Ввести строку из слов, разделенных пробелами. Подсчитать количество слов, содержащих 4 буквы.

5. Контрольные вопросы

1. Что такое нуль-терминальная строка?
2. Как описать нуль-терминальную строку?
3. Почему при вводе строки целесообразно применять функцию gets а не scanf?
4. В каком заголовочном файле интегрированной среды Bjrland C++ Builder содержатся функции работы с нуль-терминальными строками?
5. Какая функция определяет, содержится ли подстрока в строке?
6. Какая функция сравнивает строки?
7. Какая функция копирует строки?

Практическое занятие № 11

Структуры в языке С++

1. Цель работы: Закрепление знаний о структурах, составление программ с применением структур.

2. Основные сведения

В отличие от массивов, в которых содержатся элементы одного типа, в структурах могут содержаться элементы как одного, так и разных типов. В качестве примера можно привести сведения о товарах : наименование товара (текст), количество товара, его цена. Наименование товара относится к строковому типу данных, количество и цена товара - к вещественному типу (float).

Поскольку одна структура может объединять разнородные типы данных, с помощью структур удобно описываются базы данных в самых различных предметных областях. Например, структура «Студент» может содержать следующие сведения: № зачетной книжки, фамилия, имя, отчество, курс, группа, стипендия, домашний адрес, оценки по предметам.

В приведенном ниже примере описана структура tovar с полями:
naim[20] - массив символов из 19 элементов - наименование; kol, cena - вещественные числа (количество, цена), t1, t2, t3 - переменные типа структуры tovar .Наименование, как строковая переменная, вводится в формате %s, количество и цена товара вводятся в обычном для вещественных чисел формате %f.

Для обращения к элементу структуры вначале следует записать имя структуры, затем следует точка, затем - имя поля.

```
//struct.c работа со структурами
 #include <stdio.h>
 struct tovar{char naim[20];float kol,cena;} t1,t2,t3;
float r1,r2,r3,r;//стоимости товаров 1, 2, 3 и общая стоимость
 main() {
 printf("Введите наименование товара 1\n");
```

```
scanf(“%s”,&t1.naim);//имя структуры,точка,имя поля-наименование
// %s - форматный ввод строки символов
printf(“Введите количество товара 1, кг\n”);
scanf(“%f”,&t1.kol);//имя структуры,точка,имя поля-количество
printf(“Введите цену товара 1, руб/кг\n”);
scanf(“%f”,&t1.cena);//имя структуры,точка,имя поля-цена
r1=t1.kol*t1.cena;
printf(“Стоимость товара %s равна %8.2f руб.\n”,t1.naim,r1);
/*Аналогично вводятся данные для товаров 2 и 3 и под-
считывается общая стоимость r=r1+r2+r3 */
return 0;}
```

Со структурами, как и с массивами, можно работать с помощью указателей. Указатель на структуру вводится с помощью описания:

*имя_структуры *имя_указателя*;

Например, для структуры tovar описание указателя имеет вид:

```
struct tovar{char naim[20];float kol,cena;}t1, t2, *p;
```

Здесь t1, t2 – структуры типа tovar, p – указатель на структуру этого типа. Указатель может быть инициализирован адресом конкретной структуры, например:

```
p = &t1;
```

При работе со структурами с помощью указателей вводится операция «стрела», лексема, состоящая из знака «минус» и знака «больше», т.е. “->”. Стрела заменяет точку при обращении к полю структуры и является разделителем между именем указателя на структуру и именем поля структуры. Например, после описанной инициализации указателя p для обращения к полю naim структуры t1 вместо t1.naim следует записать p -> naim.

3. Выполнение работы

Набрать и откомпилировать приведеную выше программу, исправить выявленные ошибки. Ввести элементы структуры, убедиться в правильности работы программы.

Составить программу с применением структур согласно вариантам заданий, откомпилировать ее, ввести исходные данные , проверить полученный результат.

4. Варианты заданий

1. Ввести 3 структуры типа student с полями :фамилия ,стипендия; и вывести сумму стипендий студентов.
2. Ввести 2 структуры типа kniga с полями :автор ,название, год издания; и вывести количество книг издания до 1998 года.
3. Ввести 3 структуры типа VUZ с полями :наименование вуза, количество студентов и вывести общее число студентов.
4. Ввести 2 структуры типа computer с полями: наименование, тактовая частота, стоимость и вывести общую стоимость компьютеров с тактовой частотой не ниже 100 МГц.
5. Ввести 3 структуры типа scool с полями: город, номер школы, число поступивших в вузы; и вывести общее число поступивших.
6. Ввести 2 структуры типа sport с полями: вид спорта, количество секций, количество спортсменов в секции; и вывести общее число спортсменов.
7. Ввести 3 структуры типа bus с полями: марка автобуса, скорость, вместимость; и вывести общую вместимость автобусов со скоростью не ниже 80 км/ч.
8. Ввести 2 структуры типа fabric с полями: наименование фабрики, год основания, число работающих; и вывести общее число работающих на фабриках основания до 1990 года.
9. Ввести 2 структуры типа kafedra с полями :ФИО ,должность, год рождения; и вывести количество сотрудников с годом рождения до 1978 года.

10. Ввести 3 структуры типа computer с полями: наименование, объем оперативной памяти, объем внешней памяти и вывести число компьютеров с объемом внешней памяти более 60 Гб .

5. Контрольные вопросы

1. Что такое структура?

2. Какой разделитель ставится между именем структуры и именем поля?

3. Как ввести структуру?

4. Может ли существовать массив структур? Можно ли в вашей программе использовать массив структур?

5. Как определить указатель на структуру?

6. Какой разделитель ставится между именем указателя и именем поля структуры?

Практическое занятие № 12

Объединения. Битовые поля

1. Цель работы: Закрепление знаний об объединениях и битовых полях, составление программ с применением объединений и битовых полей.

2. Основные сведения

Для экономии места в оперативной памяти в языках Си/Си++ вводится структурированный тип данных – объединение (union). Пример записи объединения:

```
union a {
int c;
char d;
double r;
};
```

Далее объявляются переменные данного типа:

```
union a t1, t2, t3;
```

Для каждого из экземпляров t1, t2, t3 объединения a будет выделено в оперативной памяти столько места, сколько занимает самое большое поле, в данном случае поле r типа double (8 байтов). Для обращения к какому-нибудь полю пишем t1.c = 5; t2.d = ‘R’ и т.д., как и для структуры. Аналог объединения в языке Паскаль – записи с вариантами.

Приведем пример программы с применением объединений для расшифровки формата вещественного числа с удвоенной точностью.

```
//lab11.cpp объединения
#include<stdio.h>
#include<conio.h>
main(){
int i,j;
union {
unsigned char a[8];
double b;}t1,t2;
```

```
t1.b=1; t2.b=-1;
for(i=0;i<8;i++)
printf("%X ",t1.a[7-i]);printf("\n");
for(i=0;i<8;i++)
printf("%X ",t2.a[7-i]);printf("\n");
getch();
return 0; }
```

В программе описывается шаблон объединения, содержащий массива из 8 беззнаковых байтов и вещественного числа. Вводятся 2 переменных типа объединения t1 и t2. Вещественному полю объединения t1 присваивается значение 1, вещественному полю объединения t2 – значение –1.

Далее выводятся в шестнадцатеричной форме побайтно оба объединения, в виде двух шестнадцатеричных цифр для каждого байта с пробелом между байтами. Таким образом числа 1 и –1 будут иметь представления:

3F F0 00 00 00 00 00 00

BF F0 00 00 00 00 00 00.

Перевод из 16-ричного представления в двоичное не представляет труда: каждая 16-ричная цифра представляется как 4 двоичных цифры.

Битовые поля представляют возможность обращения к каждому отдельному биту (двоичному разряду), что характерно для языков низкого уровня. Двухбайтовое машинное слово (16 двоичных разрядов) описывается как совокупность знаковых (signed int) или беззнаковых (unsigned int) битовых полей. Пример описания битового поля:

struct hk{ int a:5, b:3; unsigned c:4, d:4) r1,r2;

Здесь два знаковых поля a и b из пяти и трех разрядов (спецификатор signed опущен) и два беззнаковых поля c и d по 4 разряда в каждом (спецификатор int опущен). После шаблона битовых полей описаны две переменных этого типа r1 и r2.

С помощью битовых полей можно экономить память, если в программе требуется много переменных малой длины (несколько битов). Кроме того, можно упаковывать несколько данных малого размера.

3. Выполнение работы

Набрать и откомпилировать приведенную выше программу, исправить выявленные ошибки.

Составить программу с применением объединений согласно вариантам заданий, откомпилировать ее, ввести исходные данные, проверить полученный результат.

4. Варианты заданий

1. Составить программу для получения побитового представления целых чисел 384 и –1187 с помощью объединений.

2. Составить программу для получения побитового представления целых чисел 314 и –987 с помощью объединений.

3. Составить программу для получения побитового представления целых чисел 137 и –1082 с помощью объединений.

4. Составить программу для получения побитового представления целых чисел 312 и –874 с помощью объединений.

5. Составить программу для получения побитового представления целых чисел 104 и –176 с помощью объединений.

6. Составить программу для получения побитового представления целых чисел 231 и –687 с помощью объединений.

7. Составить программу для получения побитового представления целых чисел 95 и –271 с помощью объединений.

8. Составить программу для получения побитового представления целых чисел 82 и –563 с помощью объединений.

9. Составить программу для получения побитового представления целых чисел 529 и –688 с помощью объединений.

10. Составить программу для получения побитового представления целых чисел 614 и –391 с помощью объединений.

5. Контрольные вопросы

1. Что такое объединение?

2. Какой разделитель ставится между именем объединения и именем поля?

3. Какая разница между объединением и структурой?

4. Какая память выделяется для объединения?

5. Что такое битовое поле?

6. Каковы возможные типы битовых полей?

7. Когда удобно применить битовое поле?

Практическое занятие № 13

Функции в языке С++

1. Цель работы: закрепление знаний о функциях, составление программ с применением функций, освоение функций перевода данных из одного типа в другой.

2. Основные сведения

Довольно часто в программе требуется повторить определенную последовательность операторов в разных частях программы. Для того, чтобы описывать эту последовательность один раз, а применять многократно, в языках программирования применяются подпрограммы. В языке Паскаль имеется 2 вида подпрограмм: процедуры и функции, в языке С - один вид подпрограмм - функции. Если же по смыслу требуется именно процедура, применяется функция, не возвращающая значения (типа void).

В программе на языке С обязательно должна содержаться функция main, которая служит точкой входа в программу, наличие других функций необязательно.

Форма записи функции в программе следующая:

<тип> <имя> (<список формальных параметров>) {<тело функции>}

Здесь <тип> - тип возвращаемого функцией значения, если тип не указан, по умолчанию принимается тип int, <имя> - имя функции, за ним следует список формальных параметров в круглых скобках, даже если список параметров пуст, круглые скобки ставятся все равно, за списком параметров в круглых скобках следует тело функции в фигурных скобках.

Для каждого формального параметра указывается тип, формальные параметры разделяются запятыми. В теле функции должен быть по крайней мере один оператор

return <выражение>; или

return;

где выражение определяет возвращаемое функцией значение, если функция типа void (не возвращает значение), то применяется вторая форма возврата из функции, без указания <выражения>.

Описание функции должно предшествовать ее использованию. Если же по каким-либо причинам этого сделать нельзя, следует поместить перед использованием функции ее прототип - заголовок функции, содержащий тип возвращаемого функцией значения, имя функции, типы формальных параметров в круглых скобках, после круглых скобок в прототипе вместо тела функции ставится точка с запятой. Если указан прототип, описание функции, содержащее тело функции, можно поместить после использования функции.

В среде Borland C++ Builder имеется ряд заголовочных файлов, содержащих стандартные функции, которые не нужно описывать в программе. Однако обязательно использование директивы #include<имя файла>, где указывается имя заголовочного файла, содержащего данную функцию. Например, функции ввода-вывода содержатся в файле stdio.h, функции для работы со строками - в файле cstring.h, математические функции - в файле math.h.

Функции в языке С не могут быть вложены друг в друга, каждая функция описывается отдельно, но в теле одной функции может быть обращение к другой функции.

Стандартные функции, содержащиеся в заголовочных файлах, в программе не описываются. Чаще других используются математические функции, содержащиеся в заголовочном файле math.h . Как правило, аргументы и возвращаемые функциями значения имеют тип double, другие случаи отмечаются особо. В программе переменные могут иметь и другой вещественный тип, например, float, при обращении к стандартной функции будет производиться неявное преобразование типов float -> double и double -> float. Предпочтительнее употребление типа double, т.к. вычисления проводятся с двойной точностью и меньше сказываются ошибки округления.

Для ввода и вывода переменных типа double в функциях printf и scanf применяется формат %lf .

Приведем пример программы для составления таблицы функции

$y = x^2 – 3x +4$

при изменении x от 0 до 1 с шагом 0,1.

```
//lab14_1.cpp  функции
#include<stdio.h>
#include<conio.h>
#include<math.h>
double y(double x){
return x*x-3*x+4;}
main(){int i; double x;
printf("============================\n");
printf("    x   |      y\n");
printf("============================\n");
for(i=0;i<=10;i++){x=(double)0.1*i;
printf("%7.1lf  | %9.4lf\n",x,y(x));}
printf("============================\n");
getch();
return 0;}
```

После директив включения заголовочных файлов следует описание функции. В описании используются формальные параметры, в данном случае один вещественный параметр x. В главной функции main производится несколько вызовов функции y (обращений к функции). При обращении используются фактические параметры функции.

Аналогично производится программирование функций с несколькими параметрами. При этом в описании функции каждый из формальных параметров описывается со своим типом. При обращении к функции число и типы фактических параметров должны совпадать с числом и типами формальных параметров.

Примечание. Возможно описание функций с параметрами по умолчанию. Параметры по умолчанию описываются последними. Тогда число фактических параметров может быть меньше числа формальных параметров, причем часть или все параметры по умолчанию могут быть опущены.

Приведем пример программы для составления таблицы функции двух переменных

$z = 2x^2 + 5y$

при изменении x от 0,5 до 1,5 с шагом 0,5 и изменении y от 1 до 2 с шагом 0,5:

```
//lab14_2.cpp  функция двух переменных
#include<stdio.h>
#include<conio.h>
#include<math.h>
double z(double x, double y){
return 2*x*x+5*y;}
main(){int i,j; double x,y;
printf("============================\n");
printf("   x  |   y  |    z\n");
printf("============================\n");
for(i=5;i<=15;i+=5){x=(double)0.1*i;
for(j=10;j<=20;j+=5){y=(double)0.1*j;
printf("%5.1lf |%5.1lf | %9.4lf \n",x,y,z(x,y));}
printf("============================\n");
}
getch();
return 0;}
```

В результате работы программы будет выведена функция двух переменных. В этой и предыдущей программах мы использовали

упрощенное графление таблицы с помощью знаков равенства и вертикальных черточек.

3. Выполнение работы

Набрать и откомпилировать приведенные выше программы, исправить выявленные ошибки.

Составить программу с применением функций одной и двух переменных, проверить полученный результат.

В работе требуется выполнить два заданий: задание 1 – вычисление значения функции y=f(x) по заданным начальным значениям аргумента x_n, конечным значениям аргумента x_k , шагу значения аргумента dx;

задание 2 – вычисление функции z= g(x,y) двух переменных при начальных значениях x_n и y_n ,конечных значениях x_k и y_k , шагах dx и dy. Функции, начальные, конечные значения и шаги даются в вариантах заданий.

4. Варианты заданий.

Номер варианта	Задание 1			
	y=f(x)	x_n	x_k	dx
1	$\sqrt{x}+3$	0	1	0,2
2	$X^2 + 1$	0	2	0,2
3	$2x +7 – x^3$	1	2	0,1
4	$7x + 8 – x^2$	0	1	0,1
5	$(x+3)^2 – 5$	1	2	0,25
6	$x^2 + 4 – x$	0	2	0,2
7	x/(x + 1)	1	3	0,2
8	x/(3+x)	0	2	0,2
9	$1,5*x^2$	0	1	0,1
10	3/(x+2)	1	2	0,1

Номер варианта	Задание 2						
	z=g(x,y)	x_n	x_k	dx	y_n	y_k	dy
1	$3x-y^2$	1	2	0,5	0	1,5	0,5
2	$4x+3y^2$	0	1,5	0,5	1	2	0,5
3	$2x^2-3y$	1	2	0,5	0	1,5	0,5
4	$5x-y^2$	0	2	0,5	1	2	0,5
5	2x+3y	1	2	0,5	0	1	0,5
6	2x+ y	0	1	0,5	1	2	0,5
7	$2x^2-y$	0,5	1,5	0,5	1	3	1
8	$5x+y^2$	0	2	1	1	2	0,5
9	3x/(y+2)	0	1	0,5	1	3	1
10	(5x+4)/(y+1)	1	2	0,5	0	1	0,5

5. Контрольные вопросы

1.Что такое подпрограмма? Виды подпрограмм в языке С++.

2. Что такое формальные параметры функции?

3. Как записывается заголовок функции? Тело функции?

4. Фактические параметры функции.

5. Что такое прототип функции?

6. Можно ли обращаться к функции, без предварительного ее описания?

7. Нужно ли описывать стандартные функции?

8. Как записать в программе показательную функцию?

Практическое занятие № 14

Рекурсия. Рекурсивные функции

1. Цель работы: закрепление знаний о рекурсивных функциях, составление программ с применением рекурсивных функций.

2. Основные сведения. Возможно обращение функции к самой себе. Такое обращение называется рекурсией, а функции – рекурсивными. Классический пример рекурсии – вычисление факториала целого числа.

Факториал целого числа n – произведение чисел от 1 до n:

n! = 1*2 ***3…***n

Из выражения факториала можно вывести рекуррентное соотношение

n! = n * (n – 1) !

Это соотношение и служит основой рекурсивной функции нахождения факториала:

```
//lab15.cpp  факториал
#include<stdio.h>
#include<conio.h>
int fact(int x){
if (x==0) return 1;
else return x*fact(x-1);}
main(){int n;
printf("Enter n\n");
scanf("%d",&n);
printf("n=%d  n!=%d\n",n,fact(n));
getch();
return 0;}
```

В основной программе main вводится целое число n, затем производится обращение к функции нахождения факториала fact. В этой функции имеется нерекурсивный выход if (x==0) return 1; т.е. при n = 0 факториал равен 1, в противном случае (т.е. при n ≠ 0) производится обращение к той же функции fact, но с аргументом, меньшим на 1.

Если и в этом случае аргумент не равен нулю, вновь производится обращение к той же функции, но с аргументом, меньшим еще на 1, и т. д. В конечном итоге аргумент станет равным нулю, и произойдет нерекурсивный выход из функции, а результат будет равен произведению всех целых чисел от 1 до n.

Механизм работы рекурсивной функции, например, функции вычисления факториала, следующий. При обращении к такой функции, как и к любой функции, в стек заносятся параметры данной функции и адрес возврата из функции, т.е. адрес оператора, следующего за вызовом функции.

Если функция не рекурсивная, больше обращений к ней не проводится, а при достижении конца функции по оператору return или return <выражение> производится выход в основную программу по адресу возврата, сохраненному в стеке.

Если же функция рекурсивная, то в ее теле содержится обращение к самой себе. По этому обращению в стек вновь заносятся параметры и адрес возврата и т.д. Таким образом, для рекурсивной функции в стек будут при каждом новом обращении к функции из тела этой же функции заноситься новые параметры и адреса возврата. Так будет продолжаться до тех пор, пока не встретится нерекурсивный выход.

В случае вычисления факториала по рекурсивной функции в стек заносится сначала параметр n, затем n-1, и т.д., пока параметр не станет равным нулю и произойдет нерекурсивный выход. При этом все запомненные в стеке параметры будут перемножены между собой и получен правильный результат.

Если в рекурсивной функции отсутствует нерекурсивный выход, обращения к функции будут повторяться бесконечно, и в стек будут заноситься все новые параметры, пока не произойдет переполнение стека. В этом случае компилятор выдает специфическое сообщение «stack overflow», переполнение стека. Если выдано такое сообщение, следует искать в одной из рекурсивных функций отсутствие нерекурсивного выхода.

Не следует думать, что факториал можно вычислить только с помощью рекурсивной функции. С равным успехом его можно вычислить с помощью цикла с параметром, в котором сомножитель увеличивается за каждый шаг на единицу. Вообще, для любой рекурсивной функции найдется метод ее вычисления без помощи рекурсии.

Зачем же в этом случае применяют рекурсию? Дело заключается в исключительной простоте рекурсивных алгоритмов, в чем можно убедиться на примере факториала. Сформулировать же алгоритм без помощи рекурсии в некоторых случаях может оказаться весьма затруднительным. Поэтому применение рекурсивных алгоритмов оправдано.

При вычислении факториала глубина рекурсии равна единице, т.е. функция обращается к самой себе с параметром, меньшим на единицу. Имеются случаи большей глубины рекурсии. Например, при вычислении чисел Фибоначчи глубина рекурсии равна двум, функция обращается к самой себе с параметрами, меньшим на единицу и на два. Рассмотрим вычисление чисел Фибоначчи более подробно.

В XIII веке Леонардо Фибоначчи (сын Боначчи) из Пизы поставил и решил задачу о размножении кроликов. Имеется пара кроликов, которая через месяц приносит приплод. Еще через месяц приплод приносит как старая, так и новая пара кроликов. Число пар кроликов через n месяцев и есть n-е число Фибоначчи F_n.

Легко убедиться, что числа Фибоначчи описываются рекуррентной формулой:

$F_1 = 1$, $F_2 = 1$, $F_n = F_{n-1} + F_{n-2}$.

Например, $F_3 = F_2 + F_1 = 1 + 1 = 2$, $F_4 = F_3 + F_2 = 2 + 1 = 3$ и т.д.

Программа вычисления чисел Фибоначчи имеет вид:

```
//lab15_2.cpp числа Фибоначчи
#include<stdio.h>
#include<conio.h>
int Fib(int x){if(x==1||x==2)
```

```
return 1; else return
Fib(x-1)+Fib(x-2);}
main(){int n;
printf("Enter n\n");
scanf("%d",&n);
printf("n=%d F(n)=%d\n",n, Fib(n));
getch();
return 0;}
```

Приведенная программа работает верно, но оптимальна ли она? Можно убедиться в том, что эта программа тратит много времени на повторные вычисления. Например, при вычислении F_5 вычисляются F_4 и F_3, при вычислении F_4 вычисляются F_3 и F_2, т.е. вновь вычисляется F_3 и т.д. Чем больше номер числа Фибоначчи, тем больше повторных вычислений.

Таким образом, использовать рекурсивную функцию вычисления чисел Фибоначчи нецелесообразно. Более оптимальной является функция с циклическими вычислениями. Причина этого заключается в том, что глубина рекурсии в рекуррентной формуле для чисел Фибоначчи равна двум, т.е. больше единицы.

Кроме прямой рекурсии, как при вычислении факториала, имеется еще косвенная рекурсия. Такой случай наблюдается, если в программе присутствуют по крайней мере две функции, обращающиеся друг к другу. В этом случае следует воспользоваться прототипом одной из связанных функций.

При использовании прототипа одна из связанных функций описывается после ее использования. Это стало возможным благодаря прототипу, в котором указывается тип возвращаемого функцией значения и типы всех параметров. (Совокупность типов параметров и типа функции называются сигнатурой функции).

3. Выполнение работы

Набрать и откомпилировать приведенные выше программы, исправить выявленные ошибки.

Составить программу с применением рекурсивной функции согласно вариантам заданий.

4. Варианты заданий

1. Последовательность вещественных чисел задается правилами:

$R_1 = 1,2$; $R_2 = 0,9$; $R_n = 0,9*R_{n-1} + 1,1*R_{n-2}$.

Найти пятый член последовательности.

2. Последовательность вещественных чисел задается правилами:

$R_1 = 1,25$; $R_2 = 0,8$; $R_n = 0,95*R_{n-1} + 1,0*R_{n-2}$.

Найти четвертый член последовательности.

3. Последовательность вещественных чисел задается правилами:

$R_1 = 1,1$; $R_2 = 0,9$; $R_n = 0,86*R_{n-1} + 1,15*R_{n-2}$.

Найти шестой член последовательности.

4. Последовательность вещественных чисел задается правилами:

$R_1 = 1,05$; $R_2 = 0,9$; $R_n = 0,95*R_{n-1} + 1,05*R_{n-2}$.

Найти седьмой член последовательности.

5. Последовательность вещественных чисел задается правилами:

$R_1 = 1,0$; $R_2 = 0,82$; $R_n = 0,9*R_{n-1} + 1,15*R_{n-2}$.

Найти четвертый член последовательности.

6. Последовательность вещественных чисел задается правилами:

$R_1 = 1,12$; $R_2 = 0,89$; $R_n = 0,91*R_{n-1} + 1,05*R_{n-2}$.

Найти восьмой член последовательности.

7. Последовательность вещественных чисел задается правилами:

$R_1 = 1,22$; $R_2 = 0,93$; $R_n = 0,94*R_{n-1} + 1,16*R_{n-2}$.

Найти пятый член последовательности.

8. Последовательность вещественных чисел задается правилами:

$R_1 = 1,3$; $R_2 = 0,91$; $R_n = 0,89*R_{n-1} + 1,17*R_{n-2}$.

Найти девятый член последовательности.

9. Последовательность вещественных чисел задается правилами:

$R_1 = 1{,}18$; $R_2 = 0{,}83$; $R_n = 0{,}9*R_{n-1} + 1{,}08*R_{n-2}$.

Найти шестой член последовательности.

10. Последовательность вещественных чисел задается правилами:

$R_1 = 1{,}08$; $R_2 = 0{,}92$; $R_n = 0{,}84*R_{n-1} + 1{,}19*R_{n-2}$.

Найти девятый член последовательности.

5. Контрольные вопросы

1. Что такое рекурсия? Рекурсивная функция?

2. Что такое нерекурсивный выход?

3. К чему может привести отсутсвие нерекурсивного выхода в рекурсивной функции?

4. Можно ли вычислить факториал без рекурсии?

5. Что такое прототип функции?

6. Что такое сигнатура функции?

7. Всегда ли целесообразно использовать рекурсивные функции?

Список рекомендуемых источников

Перечень основной литературы:

1. Тюльпинова, Н. В. Алгоритмизация и программирование Электронный ресурс: Учебное пособие / Н. В. Тюльпинова. - Саратов : Вузовское образование, 2019. - 200 c. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4487-0470-3, экземпляров неограничено
2. Бедердинова, О. И Основы алгоритмизации и структурного программирования Электронный ресурс / Бедердинова О. И. : учебное пособие. - Архангельск: САФУ, 2017. - 88 c. - ISBN 978-5-261-01227-6, экземпляров неограничено
3. Забихуллин, Ф. З. Структурное программирование на С++ Электронный ресурс / Забихуллин Ф. З. : учебное пособие. - Уфа : БГПУ имени М. Акмуллы, 2019. - 45 c. - ISBN 978-5-907176-11-9, экземпляров неограничено
4. Конова, Е. А. Алгоритмы и программы. Язык С++ Электронный ресурс / Конова Е. А., Поллак Г. А. : учебное пособие для впо. - 5-е изд., стер. - Санкт-Петербург: Лань, 2020. - 384 c. - Допущено УМО по образованию в области прикладной информатики в качестве учебного пособия для студентов, обучающихся по направлению «Прикладная информатика». - ISBN 978-5-8114-5431-0, экземпляров неограничено

Белева, Л.Ф. Программирование на языке С++ Электронный ресурс: учебное пособие / Л.Ф. Белева. - Саратов : Ай Пи Эр Медиа, 2018. - 81 c. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4486-0253-5, экземпляров неограниченно

Перечень дополнительной литературы:

1. Белоцерковская, И. Е. Алгоритмизация. Введение в язык программирования С++ / И.Е. Белоцерковская; Н.В. Галина; Л.Ю. Катаева. - 2-е изд., испр. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 197 c., экземпляров неограничено
2. Седжвик, Р. Алгоритмы на С++ / Р. Седжвик. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 1773 c., экземпляров неограниченно
3. Лубашева, Т. В. Основы алгоритмизации и программирования: учебное пособие / Т.В. Лубашева, Б.А. Железко. - Минск : РИПО, 2016. - 378 с.: ил. - http://biblioclub.ru/. - Библиогр. в кн. - ISBN 978-985-503-625-9, экземпляров неограничено
4. Зоткин, С. П. Программирование на языке высокого уровня С/С++ Электронный ресурс / Зоткин С. П. : конспект лекций. - 3-е изд. - Москва: МИСИ – МГСУ, 2018. - 140 c. - ISBN 978-5-7264-1810-0, экземпляров неограничено
5. Каширская, Е. Н. Процедурное программирование: Практикум Электронный ресурс / Каширская Е. Н. - Москва : РТУ МИРЭА, 2020. - 75 c., экземпляров неограничено

6. Баженова, И.Ю. Введение в программирование Электронный ресурс : учебное пособие / В.А. Сухомлин / И.Ю. Баженова. - Введение в программирование,2020-07-28. - Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. - 327 c. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4487-0073-6, экземпляров неограниченно

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по организации и проведению самостоятельной работы
по дисциплине «Технологии программирования и алгоритмизация»
для студентов направления подготовки
09.03.01 Информатика и вычислительная техника
Направленность (профиль)
«Программное обеспечение робототехнических и интеллектуальных систем»

Ставрополь 2025

ОБЩАЯ ХАРАКТЕРИСТИКА ДИСЦИПЛИНЫ

Цель и задачи дисциплины

Целью дисциплины «Технологии программирования и алгоритмизация» является формирование из набора компетенций будущего бакалавра. Изучение дисциплины обеспечивает реализацию требований Федерального Государственного образовательного стандарта высшего образования по направлению 09.03.01 Информатика и вычислительная техника

Задачи дисциплины: изучить основы алгоритмизации и программирование на примере языка программирования С++.

Место дисциплины в структуре образовательной программы

Дисциплина относится обязательной части. Ее освоение происходит в 1,2 семестрах.

Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	**Код, формулировка индикатора**	**Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов**
ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения	ИД-1ОПК-8 применяет языки программирования и работы с базами данных, современные программные среды разработки информационных систем и технологий для автоматизации бизнес-процессов, решения прикладных задач различных классов, ведения баз данных и информационных хранилищ.	Самостоятельно осваивает новые для себя современные языки программирования и языки работы с базами данных, среды разработки информационных систем и технологий.
	ИД-2ОПК-8 разрабатывает алгоритмы и программы пригодные для практического применения, принимает участие в отладке и тестировании прототипов программно-технических комплексов задач.	Разрабатывает алгоритмы и компьютерные программы, пригодные для практического применения. Читает коды программных продуктов, написанных на освоенных языках программирования, и вносит требуемые изменения.
ОПК-9 Способен осваивать методики использования программных средств для решения практических задач	ИД-2ОПК-9 использует программные средства для решения практических задач	Решает практические задачи с использованием программных средств и технологий программирования

2. ХАРАКТЕРИСТИКА САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Все виды самостоятельной работы по дисциплине «Технологии программирования и алгоритмизация» приведены в таблице «Технологическая карта самостоятельной работы». В основу самостоятельной работы студентов по дисциплине положено конспектирование лекций и рекомендуемых источников, подготовка к компьютерному тестированию знаний, оформление и подготовка к защите отчетов по лабораторным работам, выполнению разноуровневых индивидуальных заданий в ходе лабораторных работ. По каждому виду самостоятельной работы предусмотрены определённые формы отчетности, которые студенты предъявляют преподавателю в ходе лабораторных занятий или в часы индивидуальных консультаций.

Таблица 1 - Технологическая карта самостоятельной работы

Код оцениваемой компетенции, индикатора (ов)	Этап формирования компетенции (№ темы) *(в соответствии с рабочей программой дисциплины)*	Средства и технологии оценки	Вид контроля, аттестация *(текущий/промежуточный)*	Тип контроля *(устный, письменный или с использованием технических средств)*	Наименов оценочно средства
ИД-1ОПК-8 ИД-2ОПК-8 ИД-2ОПК-9	Тема 1-21	Вопросы для собеседования по лабораторным работам	текущий	устный	Вопросы собеседов по лаборатор работам
ИД-1ОПК-8 ИД-2ОПК-8 ИД-2ОПК-9	Тема 1-21	Вопросы для собеседования	текущий	устный	Вопросы собеседов

Для успешного освоения дисциплины, необходимо выполнить указанные в таблице виды самостоятельной работы, используя рекомендуемые источники информации.

1. Перечень основной литературы:

5. Тюльпинова, Н. В. Алгоритмизация и программирование Электронный ресурс: Учебное пособие / Н. В. Тюльпинова. - Саратов : Вузовское образование, 2019. - 200 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4487-0470-3, экземпляров неограничено

6. Бедердинова, О. И Основы алгоритмизации и структурного программирования Электронный ресурс / Бедердинова О. И. : учебное пособие. - Архангельск: САФУ, 2017. - 88 с. - ISBN 978-5-261-01227-6, экземпляров неограничено

7. Забихуллин, Ф. З. Структурное программирование на С++ Электронный ресурс / Забихуллин Ф. З. : учебное пособие. - Уфа : БГПУ имени М. Акмуллы, 2019. - 45 с. - ISBN 978-5-907176-11-9, экземпляров неограничено

8. Конова, Е. А. Алгоритмы и программы. Язык С++ Электронный ресурс / Конова Е. А., Поллак Г. А. : учебное пособие для впо. - 5-е изд., стер. - Санкт-Петербург: Лань, 2020. - 384 с. - Допущено УМО по образованию в области прикладной информатики в качестве учебного пособия для студентов, обучающихся по направлению «Прикладная информатика». - ISBN 978-5-8114-5431-0, экземпляров неограничено

Белева, Л.Ф. Программирование на языке С++ Электронный ресурс: учебное пособие / Л.Ф. Белева. - Саратов : Ай Пи Эр Медиа, 2018. - 81 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4486-0253-5, экземпляров неограниченно

2. Перечень дополнительной литературы:

7. Белоцерковская, И. Е. Алгоритмизация. Введение в язык программирования С++ / И.Е. Белоцерковская; Н.В. Галина; Л.Ю. Катаева. - 2-е изд., испр. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 197 с., экземпляров неограничено

8. Седжвик, Р. Алгоритмы на С++ / Р. Седжвик. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 1773 с., экземпляров неограниченно

9. Лубашева, Т. В. Основы алгоритмизации и программирования: учебное пособие / Т.В. Лубашева, Б.А. Железко. - Минск : РИПО, 2016. - 378 с.: ил. - http://biblioclub.ru/. - Библиогр. в кн. - ISBN 978-985-503-625-9, экземпляров неограничено

10. Зоткин, С. П. Программирование на языке высокого уровня С/С++ Электронный ресурс / Зоткин С. П. : конспект лекций. - 3-е изд. - Москва: МИСИ – МГСУ, 2018. - 140 с. - ISBN 978-5-7264-1810-0, экземпляров неограничено

11. Каширская, Е. Н. Процедурное программирование: Практикум Электронный ресурс / Каширская Е. Н. - Москва : РТУ МИРЭА, 2020. - 75 с., экземпляров неограничено

12. Баженова, И.Ю. Введение в программирование Электронный ресурс : учебное пособие / В.А. Сухомлин / И.Ю. Баженова. - Введение в программирование,2020-07-28. - Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. - 327 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4487-0073-6, экземпляров неограниченно

3. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА

При самостоятельном изучении тем лекционных занятий и заданной литературы студентам рекомендуется:

- в день проведения занятий проработать, изученный на занятии учебный материал по конспекту. При этом необходимо выделить вопросы, на которые следует обратить большее внимание при дальнейшем изучении текущей темы. После проработки учебного материала необходимо ответить на, рекомендуемые контрольные вопросы;
- с целью качественного закрепления изученного материала в последующем следует более детально проработать учебный материал с использованием рекомендованной литературы. Студентам рекомендуется выделить вопросы, требующие более детальной проработки с помощью преподавателя;
- накануне проведения лекции студенту рекомендуется закрепить ранее изученный теоретический материал, подготовить вопросы, требующие уточнения у преподавателя.

Рекомендации по организации работы с литературой

Для овладения, закрепления и систематизации знаний студентам рекомендуется самостоятельная работа с рекомендованной литературой и конспектом лекций. При самостоятельной работе с рекомендованной литературой студентам рекомендуется проводить конспектирование учебного материала, изложенного в рекомендованных источниках.

Конспектирование источников проводится при детальном изучении темы, при этом студентам рекомендуется:

- дополнять конспект лекционного занятия путем его расширения по наиболее важным вопросам, рассмотренным на занятии.

Контроль этого вида самостоятельной работы осуществляется на учебных занятиях путем проверки преподавателем конспекта лекций и собеседования.

При конспектировании источников студенты должны исходить из рационального объема, конспектируемого материала. Конспект должен быть кратким и понятным при его использовании после изучения текущей темы.

Для успешного освоения дисциплины, необходимо самостоятельно детально изучить представленные темы по рекомендуемым источникам информации.

Работа по конспектированию источников оценивается удовлетворительно если:

- конспект имеет достаточный объем детализации по, изучаемой теме, обеспечивающий заданный уровень освоения теоретического материала дисциплины;

- конспект оформлен аккуратно, изучаемый материал изложен последовательно в соответствии с порядком изучения дисциплины, содержит обобщающие выводы по каждой теме.

4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

При выполнении практикума по дисциплине (лабораторные работы) для достижения базового уровня сформированности компетенций студент должен уметь применять базовые знания по разделам дисциплины, владеть навыками самостоятельного решения учебных задач базового уровня.

При выполнении практикума по дисциплине (лабораторные работы) для достижения повышенного уровня сформированности компетенций студент должен уметь применять повышенные знания по разделам дисциплины, владеть навыками самостоятельного решения учебных задач повышенного уровня.

Методические указания при подготовке к лабораторной работе

Подготовку к лабораторной работе рекомендуется проводить в следующей последовательности:

- уяснить тему и цель, предстоящей лабораторной работы;
- изучить теоретический материал в соответствии с темой лабораторной работы (рекомендуется использовать рекомендованную литературу, конспект лекций, учебное пособие (практикум по лабораторным работам);
- ознакомиться с оборудованием и материалами, используемыми на лабораторной работе (при использовании специализированного оборудования или программного обеспечения необходимо изучить порядок и правила его использования).

Методические указания при выполнении лабораторной работы

При выполнении лабораторной работы студенты должны строго соблюдать, установленные требования безопасности.

При выполнении лабораторной работы студентам рекомендуется:

- уяснить цель, выполняемых заданий и способы их решения;
- задания, указанные в лабораторной работе выполнять в той последовательности, в которой они указаны в лабораторном практикуме;
- при выполнении практического задания и изучении теоретического материала использовать помощь преподавателя;
- оформить отчет по лабораторной работе;
- ответить на контрольные вопросы.

Методические указания при подготовке к защите лабораторной работы

При подготовке к защите лабораторной работы студентам рекомендуется:

- подготовить отчет по лабораторной работе;
- подготовить обоснование, сделанных выводов;

- закрепить знания теоретического материала по теме лабораторной работы (рекомендуется использовать контрольные вопросы);
- знать порядок проведения расчетов (проводимых исследований);
- уметь показать и пояснить порядок исследований при использовании специализированного оборудования или программного обеспечения.

Защита лабораторной работы осуществляется путем собеседования студента с преподавателем. При собеседовании студент представляет на проверку отчет по лабораторной работе. Собеседование со студентом, претендующим на оценку «отлично» проводится по вопросам и практическим заданиям повышенного уровня обучения. Собеседование со студентом, не претендующим на оценку «отлично» проводится по вопросам и практическим заданиям базового уровня обучения.

5. ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ ПО САМОСТОЯТЕЛЬНОМУ ИЗУЧЕНИЮ ЛИТЕРАТУРЫ

Вопросы для собеседования

Вопросы (задача, задание) для проверки уровня обученности

1. Для чего нужны диаграммы деятельности UML?
2. Что такое состояние действия и состояние деятельности?
3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?
4. Какой алгоритм является алгоритмом разветвляющейся структуры?
5. Чем отличается разветвляющийся алгоритм от линейного?
6. Что такое условный оператор? Какие существуют его формы?
7. Что такое составной оператор? Каков формат его записи?
8. Какие операторы сравнения используются в Си?
9. Что называется простым условием? Приведите примеры.
10. Что такое составное условие? Приведите примеры.
11. Какие логические операторы допускаются при составлении сложных условий?
12. Может ли оператор ветвления содержать внутри себя другие ветвления?
13. Что такое множественный выбор?
14. В каких случаях применяется переключатель?
15. Зачем ставится в переключателе оператор break?
16. Зачем в переключателе употребляется зарезервированное слово default?
17. Какие UML-диаграммы показывают работу переключателя?
18. Какой алгоритм является алгоритмом циклической структуры?
19. Типы циклов в языке Си.

20. Назовите основные параметры цикла.

21. Что представляют собой операторы инкремента и декремента, в каких формах они используются?

22. Как записывается условие продолжения цикла в циклах типа while и do ... while?

23. Основная форма цикла do … while.

24. Какие три раздела записываются в круглых скобках для оператора цикла типа for? Какой разделитель между разделами?

25. Что такое вложенные циклы? Примеры.

26. Как образуется бесконечный цикл и как выйти из него?

27. Схема приведения цикла while к эквивалентному ему циклу for.

28. Назовите операторы, способные изменять естественный порядок выполнения вычислений.

29. Для чего нужен оператор break?

30. Где употребляется оператор continue и для чего он используется?

31. Для чего используется оператор return?

32. Какой тип должно иметь выражение?

33. Как объявляются одномерные массивы в языке C++?

34. Какими должны быть размерности при описании статического массива в языке C++?

35. Каков диапазон изменения индекса массива в языке C++?

36. Каким образом производится инициализация массива в языке C++?

37. Чем является идентификатор массива?

38. Для чего используется управляющая последовательность \t?

39. Как представляется в C++ двумерный массив?

40. Где и каким образом хранится двумерный массив?

41. В каких пределах изменяются индексы двумерного массива?

42. Какими способами можно описать двумерный массив?

43. Какие действия выполняются для каждой строки при вычислении количества положительных элементов?

44. В каком месте программы следует записывать операторы инициализации накапливаемых в цикле величин?

45. Каким образом в динамической области памяти можно создавать двумерные массивы?

46. Способы создания динамического массива.

47. Каким образом производится обращение к элементам динамических массивов?

48. Что представляет собой функция в C++? Что нужно для ее использования?

49. Приведите пример заголовка функции.

50. Что включает в себя определение функции?

51. В чем отличие функции от других программных объектов?

52. Каким образом происходит вызов функции?

53. Охарактеризуйте существующие способы решения проблемы получения из подпрограммы признака ее аварийного завершения.

54. Каков механизм передачи параметров в функцию?
55. Способы передачи входных данных.
56. Что представляет собой средство C++, называемое значениями параметров по умолчанию?
57. Каким образом происходит передача в функцию имени функции?
58. Каким образом происходит передача одномерных массивов в функцию?
59. Каким образом происходит передача строк в функцию?
60. Каким образом происходит передача двумерных массивов в функцию?
61. Каким образом происходит передача структур в функцию?
62. Какая функция называется рекурсивной? Преимущества и недостатки рекурсии.
63. В чем смысл использования шаблонов?
64. Как записать параметр шаблона?
65. Перечислите основные свойства параметризованных классов?
66. Что такое ассоциация?
67. Что такое наследование?
68. Что такое агрегация?
69. Что такое зависимость?
70. Назовите и охарактеризуйте три основных принципа в использовании классов.

1. Критерии оценивания компетенций (*в соответствии с результатами освоения дисциплины***)**

Оценка «**отлично**» выставляется студенту, если студент дал полный, развернутый ответ на поставленные вопросы, при этом показана совокупность осознанных знаний по дисциплине, доказательно раскрыты основные положения вопросов; в ответе прослеживается четкая структура, логическая последовательность, отражающая сущность раскрываемых понятий. Знание демонстрируется на фоне понимания его в системе дисциплины. Ответ изложен литературным языком с использованием технической терминологии. Могут быть допущены недочеты в определении понятий, исправленные студентом самостоятельно в процессе ответа, продемонстрировал высокое умение применять полученные знания на практике через решение конкретной задачи, свободно без затруднений справился с поставленной задачей, показав владение разносторонними приемами и навыками ее выполнения, не допустил ошибок и неточностей.

Оценка «**хорошо**» выставляется студенту, если студент дал полный, развернутый ответ на поставленные вопросы, при этом показано умение выделить существенные и несущественные признаки, причинно-следственные связи. Ответ четко структурирован, логичен, изложен литературным языком с использованием технической терминологии. Могут быть допущены 2-3 неточности или незначительные ошибки, исправленные студентом с помощью преподавателя, продемонстрировал умение применять

полученные знания на практике через решение конкретной задачи, справился с поставленной задачей, показав владение необходимыми приемами и навыками ее выполнения, при этом допустил не более одной ошибки.

Оценка «**удовлетворительно**» выставляется студенту, если студент дал недостаточно полные и недостаточно развернутые ответы по вопросам билета. Логика и последовательность изложения имеют нарушения. Допущены ошибки в раскрытии понятий, употреблении терминов. Студент не способен самостоятельно выделить существенные и несущественные признаки и причинно-следственные связи. В ответе отсутствуют выводы. Умение раскрыть значение обобщенных знаний не показано. Речевое оформление требует поправок, коррекции, продемонстрировал посредственное умение применять полученные знания на практике через решение конкретной задачи, с трудом справился с поставленной задачей, при этом допустил не более двух ошибок.

Оценка «**неудовлетворительно**» выставляется студенту, если ответ представляет собой разрозненные знания с существенными ошибками по вопросу. Присутствуют фрагментарность, нелогичность изложения. Студент не осознает связь обсуждаемых вопросов с другими объектами дисциплины. Отсутствуют выводы, конкретизация и доказательность изложения. Речь неграмотная, гистологическая терминология не используется. Дополнительные и уточняющие вопросы преподавателя не приводят к коррекции ответа студента, ответ на вопросы полностью отсутствует или студент отказался от ответа, не продемонстрировал умение применять полученные знания на практике через решение конкретной задачи, не справился с поставленной задачей или допустил при ее решении три и более серьезные ошибки.

2. Описание шкалы оценивания

Максимально возможный балл за весь текущий контроль устанавливается равным **55.** Текущее контрольное мероприятие считается сданным, если студент получил за него не менее 60% от установленного для этого контроля максимального балла. Рейтинговый балл, выставляемый студенту за текущее контрольное мероприятие, сданное студентом в установленные графиком контрольных мероприятий сроки, определяется следующим образом:

Уровень выполнения контрольного задания	Рейтинговый балл (в % от максимального балла за контрольное задание)
Отличный	**100**
Хороший	**80**
Удовлетворительный	**60**
Неудовлетворительный	**0**

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих

этапы формирования компетенций

Процедура проведения данного оценочного мероприятия включает в себя:

- отдельное собеседование по каждой из указанных тем;
- составление конспекта.

Предлагаемые задания позволяют проверить компетенции **ОПК-1**

Для подготовки к каждой теме следует изучить и законспектировать рекомендованные источники и научную литературу и на их основании подготовиться к указанным в методических материалах вопросам для обсуждения.

При проверке задания, оцениваются

- увереное владение материалом;
- умение четко и логично излагать свои мысли:
- умение творчески подходить к решению основных вопросов темы;
- самостоятельность мышления,
- полное соответствие конспекта установленным требованиям;
- самостоятельная устная речь, полностью раскрывающая суть суть проблематики собседования.

7. ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ ПРИ ПРОВЕРКЕ ГОТОВНОСТИ К ЛАБОРАТОРНЫМ РАБОТАМ

Вопросы для защиты лабораторных работ

Тема: Форматы представления констант и данных, виды операций, ввод-вывод в языке С++

1. Форматы представления констант и данных, виды операций, ввод-вывод в языке С++

Тема: Исследование программирования алгоритмов разветвляющейся структуры

1. Операторы if и switch языков Си и С++.

Тема: Исследование программирования алгоритмов циклической структуры

1. Операторы while, do … while и for языков Си и С++.

Тема: Исследование программирования алгоритмов с использованием статических массивов

1. Определение и использование статических массивов в языках программирования Си и С++.

Тема: Исследование программирования алгоритмов с использованием динамических массивов

1. Связь между массивами и указателями в языках Си и C++.
2. Выделение и освобождение динамической памяти в языках Си и C++.
3. Обращение к элементам динамического массива.

Тема: Исследование программирования алгоритмов с использованием многомерных массивов

1. Определение и использование статических и динамических многомерных массивов в языках Си и C++.

Тема: Исследование средств обработки строк в C++

1. Определение и операции со строками стандартной библиотеки C++.

Тема: Исследование программирования алгоритмов обработки ASCIIZ строк

1. Определение и использование ASCIIZстрок в языках Си и C++.
2. Исследование средств стандартной библиотеки для обработки ASCIIZстрок.

Тема: Исследование возможностей повторного использования кода в структурном программировании

1. Определение и использование обычных и встраиваемых функций в языках Си и C++.

Тема: Исследование средств препроцессора

1. Средства препроцессора в языках Си и C++.
2. Определение и использование макроопределений в языках Си и C++.
3. Сравнение макроопределений и функций.

Тема: Исследование средств для создания многомодульных программ

1. Директивы условной компиляции.
2. Создание подключаемых файлов.
3. Статических и динамические библиотеки.
4. Компиляция многомодульных программ с использованием компиляторов GCCи Clang.

Тема: Исследования средств создания классов и объектов в языке C++

1. Определение класса.
2. Использование класса.
3. Определение методов класса.
4. Вложенные классы.

Тема: Исследование средств инициализации объектов и перегрузка операций в языке C++

1. Перегрузка операций.
2. Конструкторы и деструктор.
3. Константы в классе.

4. Поля-массивы в классе.
5. Статические элементы класса.

Тема: Наследование классов в языке C++

1. Простое открытое наследование.
2. Конструкторы и деструкторы при наследовании.
3. Поля и методы при наследовании.
4. Статические элементы класса при наследовании.
5. Вложенные классы и наследование.
6. Операторы присваивания и принцип подстановки.
7. Функции-операции преобразования.
8. Закрытое и защищенное наследование.
9. Виртуальные и абстрактные методы.
10. Абстрактные классы.

Тема: Исследование средств организации ввода-вывода в языке C++

1. Классификация потоков.
2. Подключение потоков.
3. Операции ввода-вывода.
4. Состояние потока.
5. Форматирование ввода-вывода.
6. Файловые потоки.
7. Буферизация.
8. Строковые потоки.
9. Позиционирование в потоке.
10. Широкие потоки.

1. Критерии оценивания компетенций

оценка «отлично» выставляется, если студент продемонстрировал высокое умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, свободно без затруднений справился с поставленной задачей, показав владение разносторонними приемами и навыками ее выполнения, не допустил ошибок и неточностей;

оценка «хорошо» выставляется, если студент продемонстрировал умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, справился с поставленной задачей, показав владение необходимыми приемами и навыками ее выполнения, при этом допустил не более одной ошибки;

оценка «удовлетворительно» выставляется, если студент продемонстрировал посредственное умение применять полученные знания на

практике через решение конкретной задачи с применением алгоритмов и языков программирования, с трудом справился с поставленной задачей, при этом допустил не более двух ошибок;

оценка «неудовлетворительно» выставляется, если студент не продемонстрировал умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, не справился с поставленной задачей или допустил при ее решении три и более серьезные ошибки.

2. Описание шкалы оценивания

Максимально возможный балл за весь текущий контроль устанавливается равным **55.** Текущее контрольное мероприятие считается сданным, если студент получил за него не менее 60% от установленного для этого контроля максимального балла. Рейтинговый балл, выставляемый студенту за текущее контрольное мероприятие, сданное студентом в установленные графиком контрольных мероприятий сроки, определяется следующим образом:

Уровень выполнения контрольного задания	Рейтинговый балл (в % от максимального балла за контрольное задание)
Отличный	**100**
Хороший	**80**
Удовлетворительный	**60**
Неудовлетворительный	**0**

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения данного оценочного мероприятия включает в себя:

- собеседование;
- защита лабораторных работ.

Предлагаемые задания позволяют проверить компетенции **ОПК-1**.

К лабораторному занятию студент должен подготовить ответы на вопросы, проработать дополнительные научные источники, сделать конспект теоретического материала (если это предусмотрено).

Максимальное количество баллов студент получает, если он выполнит Практическое задание, владеет материалом, умеет логично и четко излагать мысли, показвает самостоятельность выполнения задания.

Основанием для снижением оценки являются:

- слабое знание темы и неполностью выполненное задание;
- невыполненное задание к лабораторной работе;
- отсутствие умения применить теоретические знания для решения лабораторных задач.

При проверке задания, оцениваются

- активное участие в работе;

- увереное владение материалом;
- умение четко и логично излагать свои мысли:
- умение творчески подходить к решению заданий;
- самостоятельность мышления,
- самостоятельная устная речь, полностью раскрывающая суть суть проблематики собседования.

Часть 2. Программирование на C#

Содержание

ПРАКТИЧЕСКАЯ РАБОТА 1 .ИЗУЧЕНИЕ СРЕДЫ РАЗРАБОТКИ VISUAL STUDIO.

Цель работы: изучить среду быстрой разработки приложений Visual Studio. Научится размещать и настраивать внешний вид элементов управления на форме.

1.1. Интегрированная среда разработчика Visual Studio

Среда Visual Studio визуально реализуется в виде одного окна с несколькими панелями инструментов. Количество, расположение, размер и вид панелей может меняться программистом или самой средой разработки в зависимости от текущего режима работы среды или пожеланий программиста, что значительно повышает производительность работы.

При запуске Visual Studio появляется начальная страница со списком последних проектов, а также командами *Создать проект* и *Открыть проект*. Нажмите ссылку *Создать проект* или выберите в меню *Файл* команду *Создать проект*, на экране появится диалог для создания нового проекта (рис. 1).

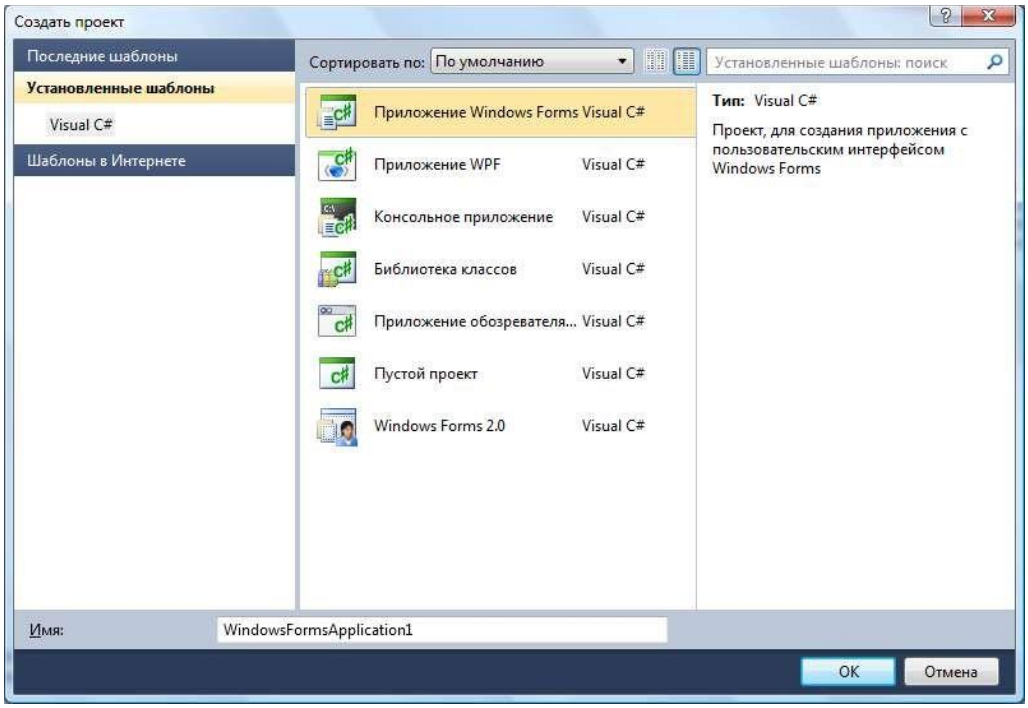

Рис 1. Диалог создания нового проекта

Слева в списке шаблонов приведены языки программирования, которые поддерживает данная версия Visual Studio: убедитесь, что там выделен раздел Visual C#. В средней части приведены типы проектов, которые можно создать. В наших лабораторных работах будут использоваться два типа проектов:

Приложение Windows Forms – данный тип проекта позволяет создать полноценное приложение с окнами и элементами управления (кнопками, полями ввода и пр.) Такой вид приложения наиболее привычен большинству пользователей.

Консольное приложение – в этом типе проекта окно представляет собой текстовую консоль, в которую приложение может выводить тексты или ожидать ввода информации пользователя. Консольные приложения часто используются для вычислительных задач, для которых не требуется сложный или красивый пользовательский интерфейс.

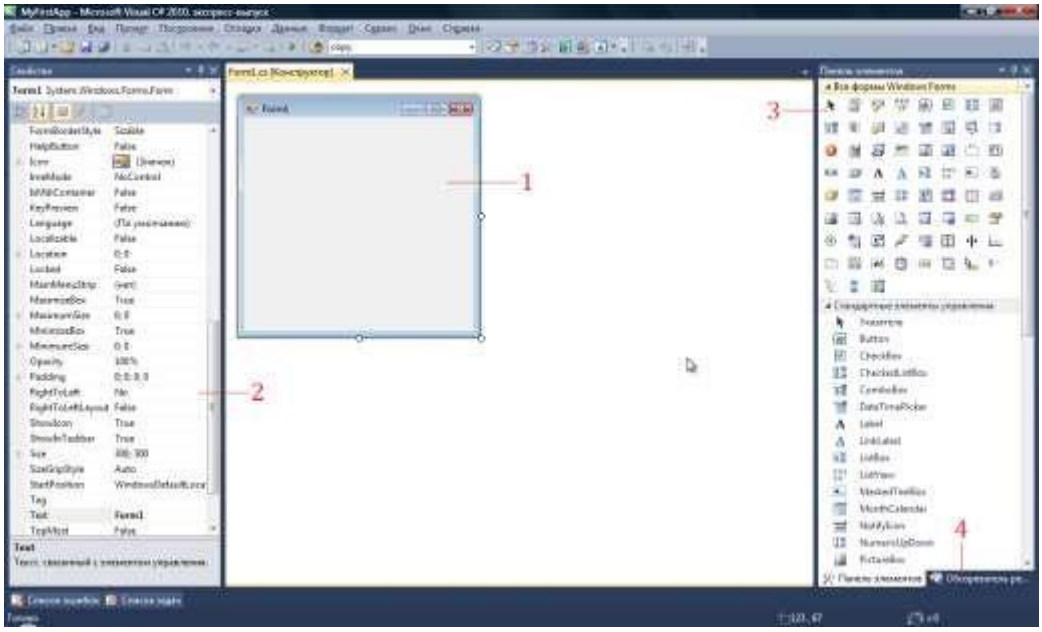

Рис 2. Главное окно Visual Studio

Выберите в списке тип проекта «Приложение Windows Forms», в поле «имя» внизу окна введите желаемое имя проекта (например, MyFirstApp) и нажмите кнопку ОК. Через несколько секунд Visual Studio создаст проект и Вы сможете увидеть на экране картинку, подобную представленной на рис. 2.

В главном окне Visual Studio присутствует несколько основных элементов, которые будут помогать нам в работе. Прежде всего, это **форма** (1) – будущее окно нашего приложения, на котором будут размещаться элементы управления. При выполнении программы помещенные элементы управления будут иметь тот же вид, что и на этапе проектирования.

Второй по важности объект – это **окно свойств** (2), в котором приведены все основные свойства выделенного элемента управления или окна. С помощью кнопки можно просматривать свойства элемента управления, а кнопка переключает окно в режим просмотра событий. Чтобы было удобнее искать нужные свойства, можно отсортировать их по алфавиту с помощью кнопки . Если этого окна на экране нет, его можно активировать в меню *Вид → Окно свойств* (иногда этот пункт вложен в подпункт *Другие окна*).

Сами элементы управления можно брать на **панели элементов** (3). Все элементы управления разбиты на логические группы, что облегчает поиск нужных элементов. Если панели нет на экране, её нужно активировать командой *Вид → Панель элементов*.

Наконец, **обозреватель решений** (4) содержит список всех файлов, входящих в проект, включая добавленные изображения и служебные файлы. Активируется командой *Вид →*

Обозреватель решений.

Указанные панели могут уже присутствовать на экране, но быть скрытыми за другими панелями или свёрнуты к боковой стороне окна. В этом случае достаточно щёлкнуть на соответствующем ярлычке, чтобы вывести панель на передний план.

Окно текста программы предназначено для просмотра, написания и редактирования текста программы. Переключаться между формой и текстом программы можно с помощью команд *Вид → Код* и *Вид → Конструктор*. При первоначальной загрузке в окне текста программы находится текст, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows-окна. При помещении элемента управления в окно формы, текст программы автоматически дополняется описанием необходимых для его работы библиотек стандартных программ (раздел using) и переменных для доступа к элементу управления (в скрытой части класса формы).

Программа на языке C# составляется как описание алгоритмов, которые необходимо выполнить, если возникает определенное событие, связанное с формой (например, щелчок «мыши» на кнопке – событие Click, загрузка формы – Load). Для каждого обрабатываемого в форме события, с помощью окна свойств, в тексте программы организуется метод, в котором программист записывает на языке C# требуемый алгоритм.

1.2. Настройка формы

Настройка формы начинается с настройки размера формы. С помощью мыши, «захватывая» одну из кромок формы или выделенную строку заголовка отрегулируйте нужные размеры формы.

Для настройки будущего окна приложения задаются свойства формы. Для задания любых свойств формы и элементов управления на форме используется окно свойств.

Новая форма имеет одинаковые имя (Name) и заголовок (Text) – Form1.

Для изменения заголовка щелкните кнопкой мыши на форме, окне свойств найдите и щелкните мышкой на строчке с названием Text. В выделенном окне наберите «*Лаб. раб. N1. Ст. гр. 7А62 Иванов А. А*». Для задания цвета окна используйте свойство BackColor.

1.3. Размещение элементов управления на форме

Для размещения различных элементов управления на форме используется панель элементов. Панель элементов содержит элементы управления, сгруппированные по типу. Каждую группу элементов управления можно свернуть, если она в настоящий момент не нужна. Для выполнения лабораторных работ потребуются элементы управления из группы *Стандартные элементы управления*.

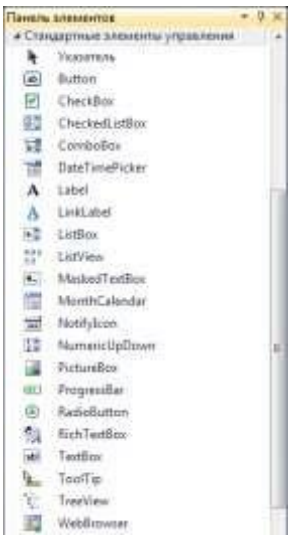

Рис.3 Панель элементов

Щёлкните на элементе управления, который хотите добавить, а затем щёлкните в нужном месте формы – элемент появится на форме. Элемент можно перемещать по форме схватившись за него левой кнопкой мышки (иногда это можно сделать лишь за появляющийся при нажатии на элемент квадрат со стрелками). Если элемент управления позволяет изменять размеры, то на соответствующих его сторонах появятся белые кружки, ухватившись за которые и можно изменить размер. После размещения элемента управления на форме, его можно выделить щелчком мыши и при этом получить доступ к его свойствам в окне свойств.

1.4. Размещение строки ввода

Если необходимо ввести из формы в программу или вывести на форму информацию, которая вмещается в одну строку, используют окно однострочного редактора текста, представляемого элементом управления TextBox.

В данной программе с помощью однострочного редактора будет вводиться имя пользователя.

Выберите на панели элементов пиктограмму с названием TextBox, щелкните мышью в том месте формы, где вы хотите ее поставить. Захватив его мышкой, отрегулируйте размеры элемента управления и его положение. Обратите внимание на то, что теперь в тексте программы можно использовать переменную textBox1, которая соответствуют добавленному элементу управления. В этой переменной в свойстве Text будет содержаться строка символов (тип string) и отображаться в соответствующем окне TextBox.

С помощью окна свойств установите шрифт и размер символов отражаемых в строке TextBox (свойство Font).

1.5. Размещение надписей

На форме могут размещаться пояснительные надписи. Для нанесения таких надписей на форму используется элемент управления Label. Выберите на панели элементов пиктограмму с названием Label, щелкните на ней мышью. После этого в нужном месте

формы щелкните мышью, появится надпись label1. Щелкнув на ней мышью, отрегулируйте размер и, изменив свойство Text в окне свойств, введите строку, например «*Введите своё имя:*», а также выберите размер символов (свойство Font).

Обратите внимание, что в тексте программы теперь можно обращаться к новой переменной типа Label. В ней хранится пояснительная строка, которую можно изменять в процессе работы программы.

1.6. Написание программы обработки события

С каждым элементом управления на форме и с самой формой могут происходить события во время работы программы. Например, с кнопкой может произойти событие – нажатие кнопки, а с окном, которое проектируется с помощью формы, может произойти ряд событий: создание окна, изменение размера окна, щелчок мыши на окне и т. п. Эти события могут быть обрабатываться в программе. Для обработки таких событий необходимо создать обработчики события – специальный *метод*. Для создания обработчика события существует два способа.

Первый способ – создать обработчик для события по умолчанию (обычно это самое часто используемое событие данного элемента управления). Например, для кнопки таким образом создаётся обработчик события нажатия.

1.7. Написание программы обработки события нажатия кнопки

Поместите на форму кнопку, которая описывается элементом управления Button. С помощью окна свойств измените заголовок (Text) на слово «*Привет*» или другое по вашему желанию. Отрегулируйте положение и размер кнопки.

После этого два раза щелкните мышью на кнопке, появится текст программы:

```
private void button1_Click(object sender, EventArgs e)
{

}
```

Это и есть обработчики события нажатия кнопки. Вы можете добавлять свой код между скобками { }. Например, наберите:

```
MessageBox.Show("Привет, " + textBox1.Text + "!");
```

1.8. Написание программы обработки события загрузки формы

Второй способ создания обработчика события заключается в выборе соответствующего

события для выделенного элемента на форме. При этом используется окно свойств и его закладка  . Рассмотрим этот способ. Выделите форму щелчком по ней, чтобы вокруг неё появилась рамка из точек. В окне свойств найдите событие Load. Щелкнете по данной строчке дважды мышкой. Появится метод:

```
private void Form1_Load(object sender, EventArgs e)
{

}
```

Между скобками { } вставим текст программы:

```
BackColor = Color.AntiqueWhite;
```

Каждый элемент управления имеет свой набор обработчиков событий, однако некоторые из них присуши большинству элементов управления. Наиболее часто применяемые события:

Activated	Форма получает это событие при активации
Load	Возникает при загрузке формы. В обработчике данного события следует задавать действия, которые должны происходить в момент создания формы, например установка начальных значений
KeyPress	Возникает при нажатии кнопки на клавиатуре. Параметр e.KeyChar имеет тип char и содержит код нажатой клавиши (клавиша *Enter* клавиатуры имеет код #13, клавиша *Esc* – #27 и т. д.). Обычно это событие используется в том случае, когда необходима реакция на нажатие одной из клавиш

KeyDown	Возникает при нажатии клавиши на клавиатуре. Обработчик этого события получает информацию о нажатой клавише и состоянии клавиш Shift, Alt и Ctrl, а также о нажатой кнопке мыши. Информация о клавише передается параметром e.KeyCode, который представляет собой перечисление Keys с кодами всех клавиш, а информацию о клавишахмодификаторах Shift и др. можно узнать из параметра e.Modifiers
KeyUp	Является парным событием для KeyDown и возникает при отпускании ранее нажатой клавиши
Click	Возникает при нажатии кнопки мыши в области элемента управления
DoubleClick	Возникает при двойном нажатии кнопки мыши в области элемента управления

Важное примечание! Если какой-то обработчик был добавлен по ошибке или больше не нужен, то для его удаления нельзя просто удалить программный код обработчика! Сначала нужно удалить строку с именем обработчика в окне свойств на закладке . В противном случае программа может перестать компилироваться и даже отображать форму в дизайнере Visual Studio.

1.9. Запуск и работа с программой

Запустить программу можно выбрав в меню *Отладка* команду *Начать отладку*. При этом происходит трансляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением .exe. На экране появляется активное окно программы.

Если в программе есть ошибки, то в окне Список ошибок появятся найденные ошибки, а программа обычно не запускается. Иногда, однако, может быть запущена предыдущая версия программы, в которой ещё нет последних изменений: чтобы этого не происходило, нужно в настройках Visual Studio установить опцию показа окна ошибок и запретить запуск при наличии ошибок (рис. 4 и 5).

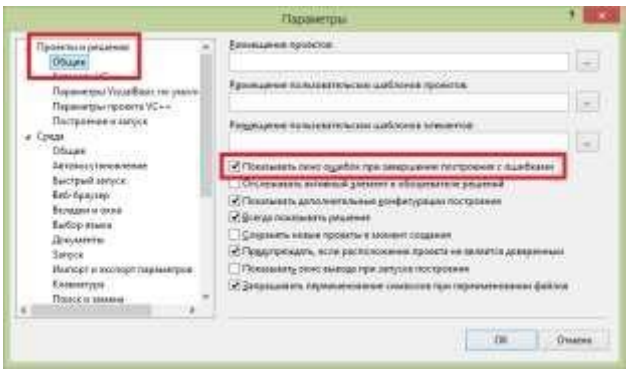

Рис. 4. Опция показа сообщений об ошибках

Для завершения работы программы и возвращения в режим проектирования формы не забудьте закрыть окно программы!

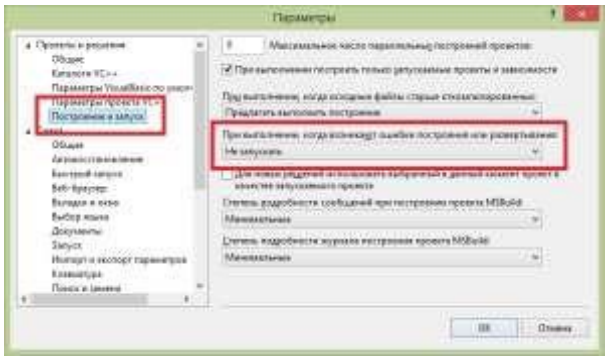

Рис. 5. Опция отключения запуска предыдущей версии при ошибках

1.10. Динамическое изменение свойств

Свойства элементов на окне могут быть изменены динамически во время выполнения программы. Например, можно изменить текст надписи или цвет формы. Изменение свойств происходит внутри обработчика события (например, обработчика события нажатия на кнопку). Для этого используют оператор присвоения вида:

<имя элемента>.<свойство> = <значение>;

Например:

label1.Text = "Привет";

<Имя элемента> определяется на этапе проектирования формы, при размещении элемента управления на форме. Например, при размещении на форме ряда элементов TextBox, эти элементы получают имена textBox1, textBox2, textBox3 и т. д. Эти имена могут быть замены

в окне свойств в свойстве (Name) для текущего элемента. Допускается использование латинских или русских символов, знака подчеркивания и цифр (цифра не должна стоять в начале идентификатора). Список свойств для конкретного элемента можно посмотреть в окне свойств, а также в приложении к данным методическим указаниям.

Если требуется изменить свойства формы, то никакое имя элемента перед точкой вставлять не нужно, как и саму точку. Например, чтобы задать цвет формы, нужно просто написать:

```
BackColor = Color.Green;
```

1.11. Выполнение индивидуального задания

Ниже приведено 15 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. Прочтите в Приложении 1 описание свойств и описание элементов управления Form, Label, TextBox, Button. С помощью окна свойств установите первоначальный цвет формы, шрифт выводимых символов.

Индивидуальные задания

1. Разместите на форме четыре кнопки (Button). Сделайте на кнопках следующие надписи: «красный», «зеленый», «синий», «желтый». Создайте четыре обработчика события нажатия на данные кнопки, которые будут менять цвет формы в соответствии с текстом на кнопках.
2. Разместите на форме две кнопки (Button) и одну метку (Label). Сделайте на кнопках следующие надписи: «привет», «до свидания». Создайте обработчики события нажатия на данные кнопки, которые будут менять текст метки на слова, написанные на кнопках. Создайте обработчик события создания формы (Load), который будет устанавливать цвет формы и менять текст метки на строку «Начало работы».
3. Разместите на форме две кнопки (Button) и одну метку (Label). Сделайте на кнопках следующие надписи: «скрыть», «показать». Создайте обработчики события нажатия на данные кнопки, которые будут скрывать или показывать метку. Создайте обработчик события создания формы (Load), который будет устанавливать цвет формы и менять текст метки на строку «Начало работы».
4. Разместите на форме три кнопки (Button) и одно поле ввода (TextBox). Сделайте на кнопках следующие надписи: «скрыть», «показать», «очистить». Создайте обработчики события нажатия на данные кнопки, которые будут скрывать или показывать поле ввода. При нажатии на кнопку «очистить» текст из поля ввода должен быть удален.
5. Разместите на форме две кнопки (Button) и одно поле ввода (TextBox). Сделайте на кнопках следующие надписи: «заполнить», «очистить». Создайте обработчики события нажатия на данные кнопки, которые будут очищать или заполнять поле ввода знаками «******». Создайте обработчик события создания формы (Load), который будет устанавливать цвет формы и менять текст в поле ввода на строку «+++++».
6. Разработайте игру, которая заключается в следующем. На форме размещены пять

кнопок (Button). При нажатии на кнопку некоторые кнопки становятся видимыми, а другие – невидимыми. Цель игры – скрыть все кнопки.

7. Разработайте игру, которая заключается в следующем. На форме размещены четыре кнопки (Button) и четыре метки (Label). При нажатии на кнопку часть надписей становится невидимыми, а часть наоборот становятся видимыми. Цель игры – скрыть все надписи.
8. Разместите на форме ряд кнопок (Button). Создайте обработчики события нажатия на данные кнопки, которые будут делать неактивными текущую кнопку. Создайте обработчик события изменение размера формы (Resize), который будет устанавливать все кнопки в активный режим.
9. Разместите на форме ряд кнопок (Button). Создайте обработчики события нажатия на данные кнопки, которые будут делать неактивными следующую кнопку. Создайте обработчик события нажатия кнопки мыши на форме (Click), который будет устанавливать все кнопки в активный режим.
10. Разместите на форме три кнопки (Button) и одно поле ввода
11. (TextBox). Сделайте на кнопках следующие надписи: «*****», «+++++», «00000». Создайте обработчики события нажатия на данные кнопки, которые будут выводить текст, написанный на кнопках, в поле ввода. Создайте обработчик события создания формы (Load), который будет устанавливать цвет формы и менять текст в поле ввода на строку «Готов к работе».
12. Разместите на форме ряд полей ввода (TextBox). Создайте обработчики события нажатия кнопкой мыши на данные поля ввода, которые будут выводить в текущее поле ввода его номер. Создайте обработчик события изменение размера формы (Resize), который будет очищать все поля ввода.
13. Разместите на форме поле ввода (TextBox), метку (Label) и кнопку (Button). Создайте обработчик события нажатия на кнопку, который будет копировать текст из поля ввода в метку. Создайте обработчик события нажатия кнопки мышки на форме (Click), который будет устанавливать цвет формы и менять текст метки на строку «Начало работы» и очищать поле ввода.
14. Разместите на форме поле ввода (TextBox), и две кнопки
15. (Button) с надписями: «блокировать», «разблокировать». Создайте обработчики события нажатия на кнопки, которые будут делать активным или неактивным поле ввода. Создайте обработчик события нажатия кнопки мышки на форме (Click), который будет устанавливать цвет формы и делать невидимыми все элементы.
16. Реализуйте игру минер на поле 3x3 из кнопок (Button). Первоначально все кнопки не содержат надписей. При попытке нажатия на кнопку на ней либо показывается количество мин, либо надпись «мина!» и меняется цвет окна.
17. Разместите на форме четыре кнопки (Button). Напишите для каждой обработчик события, который будет менять размеры и местоположение на окне других кнопок.

ПРАКТИЧЕСКАЯ РАБОТА 2. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА C#

Цель работы: Изучить основные конструкции языка C#.

Краткие теоретические сведения

Структура программы

Инструкции

Базовым строительным блоком программы являются **инструкции** (statement). Инструкция представляет некоторое действие, например, арифметическую операцию, вызов метода, объявление переменной и присвоение ей значения. В конце каждой инструкции в C# ставится точка с запятой (;). Данный знак указывает компилятору на конец инструкции. Например:

```
1    Console.WriteLine("Привет");
```

Данная строка представляет вызов метода Console.WriteLine, который выводит на консоль строку. В данном случае вызов метода является инструкцией и поэтому завершается точкой с запятой.

Набор инструкций может объединяться в блок кода. Блок кода заключается в фигурные скобки, а инструкции помещаются между открывающей и закрывающей фигурными скобками:

```
1    {
2        Console.WriteLine("Привет");
3        Console.WriteLine("Добро пожаловать в C#");
4    }
```

В данном блоке кода две инструкции, которые выводят на консоль определенную строку.

Одни блоки кода могут содержать другие блоки:

```
1    {
2        Console.WriteLine("Первый блок");
3        {
4            Console.WriteLine("Второй блок");
5        }
6    }
```

Метод Main

Точкой входа в программу на языке C# является метод Main. При создании проекта консольного приложения в Visual Studio, например, создается следующий метод Main:

```
1    class Program
2    {
```

```
3       static void Main(string[] args)
4       {
5           // здесь помещаются выполняемые инструкции
6       }
7   }
```

По умолчанию метод Main размещается в классе Program. Название класса может быть любым. Но метод Main является обязательной частью консольного приложения. Если мы изменим его название, то программа не скомпилируется.

По сути и класс, и метод представляют своего рода блок кода: блок метода помещается в блок класса. Внутри блока метода Main располагаются выполняемые в программе инструкции.

Регистрозависимость

C# является регистрозависимым языком. Это значит, в зависимости от регистра символов какое-то определенные названия может представлять разные классы, методы, переменные и т.д. Например, название обязательного метода Main начинается именно с большой буквы: "Main". Если мы назовем метод "main", то программа не скомпилируется, так как метод, который представляет стартовую точку в приложении, обязательно должен называться "Main", а не "main" или "MAIN".

Комментарии

Важной частью программного кода являются комментарии. Они не являются собственно частью программы, при компиляции они игнорируются. Тем не менее комментарии делают код программы более понятным, помогая понять те или иные его части.

есть два типа комментариев: однострочный и многострочный. Однострочный комментарий размещается на одной строке после двойного слеша //. А многострочный комментарий заключается между символами /* текст комментария */. Он может размещаться на нескольких строках. Например:

```
1   using System;
2
3   namespace HelloApp
4   {
5       /*
```

```
6           программа, которая спрашивает у пользователя имя
7           и выводит его на консоль
8       */
9       class Program
10      {
11          // метод Main - стартовая точка приложения
12          static void Main(string[] args)
13          {
14              Console.Write("Введите свое имя: ");
15              string name = Console.ReadLine();       // вводим имя
16          }
17      }
18  }
```

Программы верхнего уровня

Начиная с версии C# 9.0 (.NET 5) добавлена возможность создавать программы верхнего уровня. То есть, если у нас программа состоит из одного метода Main, то мы можем убрать из определения программы объявление пространства имен (namespace HelloApp), объявление класса (class Program) и объявление метода Main (static void Main(string[] args)) и оставить только директивы using с подключаемыми пространствами имен и собственно исполняемые инструкции.

Например, выше в предыдущем листинге кода программа запрашивала ввод имени пользователя. Фактически все тело программы состоит из метода Main, поэтому мы ее можем сократить в C# 9.0 следующим образом:

```
1   using System;
2
3   Console.Write("Введите свое имя: ");
4   string name = Console.ReadLine();       // вводим имя
5   Console.WriteLine($"Привет {name}");    // выводим имя на консоль
```

Результат работы программы будет тот же, что и в предыдущем случае.

Переменные

Для хранения данных в программе применяются **переменные**. Переменная представляет именованную область памяти, в которой хранится значение определенного типа. Переменная имеет тип, имя и значение. Тип определяет, какого рода информацию может хранить переменная.

Перед использованием любую переменную надо определить. Синтаксис определения переменной выглядит следующим образом:

```
тип имя_переменной;
```

Вначале идет тип переменной, потом ее имя. В качестве имени переменной может выступать любое произвольное название, которое удовлетворяет следующим требованиям:

- имя может содержать любые цифры, буквы и символ подчеркивания, при этом первый символ в имени должен быть буквой или символом подчеркивания
- в имени не должно быть знаков пунктуации и пробелов
- имя не может быть ключевым словом языка C#. Таких слов не так много, и при работе в Visual Studio среда разработки подсвечивает ключевые слова синим цветом.

Хотя имя переменой может быть любым, но следует давать переменным описательные имена, которые будут говорить об их предназначении.

Например, определим простейшую переменную:

```
string name;
```

В данном случае определена переменная name, которая имеет тип **string**. то есть переменная представляет строку. Поскольку определение переменной представляет собой инструкцию, то после него ставится точка с запятой.

При этом следует учитывать, что C# является регистрозависимым языком, поэтому следующие два определения переменных будут представлять две разные переменные:

```
string name;
string Name;
```

После определения переменной можно присвоить некоторое значение:

```
string name;
name = "Tom";
```

Так как переменная name представляет тип string, то есть строку, то мы можем присвоить

ей строку в двойных кавычках. Причем переменной можно присвоить только то значение, которое соответствует ее типу.

В дальнейшем с помощью имени переменной мы сможем обращаться к той области памяти, в которой хранится ее значение.

Также мы можем сразу при определении присвоить переменной значение. Данный прием называется инициализацией:

```
string name = "Tom";
```

Отличительной чертой переменных является то, что в программе можно многократно менять их значение. Например, создадим небольшую программу, в которой определим переменную, поменяем ее значение и выведем его на консоль:

```
using System;

namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = "Tom";  // определяем переменную и инициализируем ее

            Console.WriteLine(name);    // Tom

            name = "Bob";       // меняем значение переменной
            Console.WriteLine(name);    // Bob

            Console.Read();
        }
    }
}
```

Консольный вывод программы:

```
Tom
Bob
```

Литералы

Литералы представляют неизменяемые значения (иногда их еще называют константами). Литералы можно передавать переменным в качестве значения. Литералы бывают логическими, целочисленными, вещественными, символьными и строчными. И отдельный

литерал представляет ключевое слово null.

Логические литералы

Есть две логических константы - **true** (истина) и **false** (ложь):

```
1  Console.WriteLine(true);
2  Console.WriteLine(false);
```

Целочисленные литералы

Целочисленные литералы представляют положительные и отрицательные целые числа, например, 1, 2, 3, 4, -7, -109. Целочисленные литералы могут быть выражены в десятичной, шестнадцатеричной и двоичной форме.

С целыми числами в десятичной форме все должно быть понятно, так как они используются в повседневной жизни:

```
1  Console.WriteLine(-11);
2  Console.WriteLine(5);
3  Console.WriteLine(505);
```

Числа в двоичной форме предваряются символами 0b, после которых идет набор из нулей и единиц:

```
1  Console.WriteLine(0b11);       // 3
2  Console.WriteLine(0b1011);     // 11
3  Console.WriteLine(0b100001);   // 33
```

Для записи числа в шестнадцатеричной форме применяются символы 0x, после которых идет набор символов от 0 до 9 и от A до F, которые собственно представляют число:

```
1  Console.WriteLine(0x0A);   // 10
2  Console.WriteLine(0xFF);   // 255
3  Console.WriteLine(0xA1);   // 161
```

Вещественные литералы

Вещественные литералы представляют вещественные числа. Этот тип литералов имеет две формы. Первая форма - вещественные числа с фиксированной запятой, при которой дробную часть отделяется от целой части точкой. Например:

```
1  3.14
2  100.001
3  -0.38
```

Также вещественные литералы могут определяться в экспоненциальной форме МЕр, где М — мантисса, Е - экспонента, которая фактически означает "*10^" (умножить на десять в степени), а р — порядок. Например:

```
1  Console.WriteLine(3.2e3);   // по сути равно 3.2 * 10<sup>3</sup> = 3200
2  Console.WriteLine(1.2E-1);  // равно 1.2 * 10<sup>-1</sup> = 0.12
```

Символьные литералы

Символьные литералы представляют одиночные символы. Символы заключаются в одинарные кавычки.

Символьные литералы бывают нескольких видов. Прежде всего это обычные символы:

```
1  '2'
2  'A'
3  'T'
```

Специальную группу представляют **управляющие последовательности** Управляющая последовательность представляет символ, перед которым ставится обратный слеш. И данная последовательность интерпретируется определенным образом. Наиболее часто используемые последовательности:

- '\n' - перевод строки
- '\t' - табуляция
- '\' - обратный слеш

И если компилятор встретит в тексте последовательность \t, то он будет воспринимать эту последовательность не как слеш и букву t, а как табуляцию - то есть длинный отступ.

Также символы могут определяться в виде шестнадцатеричных кодов, также заключенный в одинарные кавычки.

Еще один способ определения символов представляет использования шестнадцатеричных кодов ASCII. Для этого в одинарных кавычках указываются символы '\x', после которых идет шестнадцатеричный код символа из таблицы ASCII. Коды символов из таблицы ASCII можно посмотреть здесь.

Например, литерал '\x78' представляет символ "x":

```
1  Console.WriteLine('\x78');    // x
2  Console.WriteLine('\x5A');    // Z
```

И последний способ определения символьных литералов представляет применение кодов из таблицы символов Unicode. Для этого в одинарных кавычках указываются символы '\u', после которых идет шестнадцатеричный код Unicode. Например, код '\u0411' представляет кириллический символ 'Б':

```
1  Console.WriteLine('\u0420');    // P
2  Console.WriteLine('\u0421');    // C
```

Строковые литералы

Строковые литералы представляют строки. Строки заключаются в двойные кавычки:

```
1  Console.WriteLine("hello");
2  Console.WriteLine("фыва");
3  Console.WriteLine("hello word");
```

Если внутри строки необходимо вывести двойную кавычку, то такая внутренняя кавычка предваряется обратным слешем:

```
1  Console.WriteLine("Компания \"Рога и копыта\"");
```

Также в строках можно использовать управляющие последовательности. Например, последовательность '\n' осуществляет перевод на новую строку:

```
1  Console.WriteLine("Привет \nмир");
```

При выводе на консоль слово "мир" будет перенесено на новую строку:

```
Привет
мир
```

null

null представляет ссылку, которая не указывает ни на какой объект. То есть по сути отсутствие значения.

Типы данных

Как и во многих языках программирования, в C# есть своя система типов данных, которая используется для создания переменных. Тип данных определяет внутреннее представление данных, множество значений, которые может принимать объект, а также допустимые действия, которые можно применять над объектом.

В языке C# есть следующие примитивные типы данных:

- **bool**: хранит значение true или false (логические литералы). Представлен системным типом System.Boolean

```
1  bool alive = true;
2  bool isDead = false;
```

- **byte**: хранит целое число от 0 до 255 и занимает 1 байт. Представлен системным типом System.Byte

```
1  byte bit1 = 1;
2  byte bit2 = 102;
```

- **sbyte**: хранит целое число от -128 до 127 и занимает 1 байт. Представлен системным типом System.SByte

```
1  sbyte bit1 = -101;
2  sbyte bit2 = 102;
```

- **short**: хранит целое число от -32768 до 32767 и занимает 2 байта. Представлен системным типом System.Int16

```
1  short n1 = 1;
2  short n2 = 102;
```

- **ushort**: хранит целое число от 0 до 65535 и занимает 2 байта. Представлен системным типом System.UInt16

```
1  ushort n1 = 1;
2  ushort n2 = 102;
```

- **int**: хранит целое число от -2147483648 до 2147483647 и занимает 4 байта. Представлен системным типом System.Int32. Все целочисленные литералы по умолчанию представляют значения типа int:

```
1  int a = 10;
2  int b = 0b101;  // бинарная форма b =5
3  int c = 0xFF;   // шестнадцатеричная форма c = 255
```

- **uint**: хранит целое число от 0 до 4294967295 и занимает 4 байта. Представлен системным типом System.UInt32

```
1  uint a = 10;
2  uint b = 0b101;
3  uint c = 0xFF;
```

- **long**: хранит целое число от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 и занимает 8 байт. Представлен системным типом System.Int64

```
1  long a = -10;
2  long b = 0b101;
3  long c = 0xFF;
```

- **ulong**: хранит целое число от 0 до 18 446 744 073 709 551 615 и занимает 8 байт. Представлен системным типом System.UInt64

```
1  ulong a = 10;
2  ulong b = 0b101;
3  ulong c = 0xFF;
```

- **float**: хранит число с плавающей точкой от $-3.4*10^{38}$ до $3.4*10^{38}$ и занимает 4 байта. Представлен системным типом System.Single
- **double**: хранит число с плавающей точкой от $\pm5.0*10^{-324}$ до $\pm1.7*10^{308}$ и занимает 8 байта. Представлен системным типом System.Double
- **decimal**: хранит десятичное дробное число. Если употребляется без десятичной запятой, имеет значение от $\pm1.0*10^{-28}$ до $\pm7.9228*10^{28}$, может хранить 28 знаков после запятой и занимает 16 байт. Представлен системным типом System.Decimal
- **char**: хранит одиночный символ в кодировке Unicode и занимает 2 байта. Представлен системным типом System.Char. Этому типу соответствуют

символьные литералы:

```
1  char a = 'A';
2  char b = '\x5A';
3  char c = '\u0420';
```

- **string**: хранит набор символов Unicode. Представлен системным типом System.String. Этому типу соответствуют символьные литералы.

```
1  string hello = "Hello";
2  string word = "world";
```

- **object**: может хранить значение любого типа данных и занимает 4 байта на 32-разрядной платформе и 8 байт на 64-разрядной платформе. Представлен системным типом System.Object, который является базовым для всех других типов и классов .NET.

```
1  object a = 22;
2  object b = 3.14;
3  object c = "hello code";
```

Например, определим несколько переменных разных типов и выведем их значения на консоль:

```
1  using System;
2
3  namespace HelloApp
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              string name = "Tom";
10             int age = 33;
```

```
11          bool isEmployed = false;
12          double weight = 78.65;
13
14          Console.WriteLine($"Имя: {name}");
15          Console.WriteLine($"Возраст: {age}");
16          Console.WriteLine($"Вес: {weight}");
17          Console.WriteLine($"Работает: {isEmployed}");
18      }
19  }
20  }
```

Для вывода данных на консоль здесь применяется интерполяция: перед строкой ставится знак $ и после этого мы можем вводить в строку в фигурных скобках значения переменных. Консольный вывод программы:

```
Имя: Tom
Возраст: 33
Вес: 78,65
Работает: False
```

Использование суффиксов

При присвоении значений надо иметь в виду следующую тонкость: все вещественные литералы рассматриваются как значения типа **double**. И чтобы указать, что дробное число представляет тип **float** или тип **decimal**, необходимо к литералу добавлять суффикс: F/f - для float и M/m - для decimal.

```
1  float a = 3.14F;
2  float b = 30.6f;
3
4  decimal c = 1005.8M;
5  decimal d = 334.8m;
```

Подобным образом все целочисленные литералы рассматриваются как значения типа **int**. Чтобы явным образом указать, что целочисленный литерал представляет значение типа uint, надо использовать суффикс **U/u**, для типа **long** - суффикс **L/l**, а для типа **ulong** -

суффикс **UL/ul**:

```
1  uint a = 10U;
2  long b = 20L;
3  ulong c = 30UL;
```

Использование системных типов

Выше при перечислении всех базовых типов данных для каждого упоминался системный тип. Потому что название встроенного типа по сути представляет собой сокращенное обозначение системного типа. Например, следующие переменные будут эквивалентны по типу:

```
1  int a = 4;
2  System.Int32 b = 4;
```

Неявная типизация

Ранее мы явным образом указывали тип переменных, например, int x;. И компилятор при запуске уже знал, что x хранит целочисленное значение.

Однако мы можем использовать и модель неявной типизации:

```
1  var hello = "Hell to World";
2  var c = 20;
3
4  Console.WriteLine(c.GetType().ToString());
5  Console.WriteLine(hello.GetType().ToString());
```

Для неявной типизации вместо названия типа данных используется ключевое слово var. Затем уже при компиляции компилятор сам выводит тип данных исходя из присвоенного значения. В примере выше использовалось выражение Console.WriteLine(c.GetType().ToString());, которое позволяет нам узнать выведенный тип переменной c. Так как по умолчанию все целочисленные значения рассматриваются как значения типа int, то поэтому в итоге переменная c будет иметь тип int или System.Int32

Эти переменные подобны обычным, однако они имеют некоторые ограничения.

Во-первых, мы не можем сначала объявить неявно типизируемую переменную, а затем инициализировать:

```
1 // этот код работает
2 int a;
3 a = 20;
4
5 // этот код не работает
6 var c;
7 c= 20;
```

Во-вторых, мы не можем указать в качестве значения неявно типизируемой переменной null:

```
1 // этот код не работает
2 var c=null;
```

Так как значение null, то компилятор не сможет вывести тип данных.

double или decimal

Из выше перечисленного списка типов данных очевидно, что если мы хотим использовать в программе числа до 256, то для их хранения мы можем использоват переменные типа byte. При использовании больших значений мы можем взять тип short, int, long. То же самое для дробных чисел - для обычных дробных чисел можно взять тип float, для очень больших дробных чисел - тип double. Тип decimal здесь стоит особняком в том плане, что несмотря на большую разрядность по сравнению с типом double, тип double может хранить большее значение. Однако значение decimal может содержать до 28 знаков после запятой, тогда как значение типа double - 15-16 знаков после запятой.

Decimal чаще находит применение в финансовых вычислениях, тогда как double - в математических операциях. Общие различия между этими двумя типами можно выразить следующей таблицей:

	Decimal	**Double**
Наибольшее значение	$\sim 10^{28}$	$\sim 10^{308}$
Наименьшее значение (без учета нуля)	10^{-28}	$\sim 10^{-323}$
Знаков	28	15-16

после запятой		
Разрядность	16 байт	8 байт
Операций в секунду	сотни миллионов	миллиарды

Консольный ввод-вывод

Консольный вывод

Для вывода информации на консоль мы уже использовали встроенный метод **Console.WriteLine**. То есть, если мы хотим вывести некоторую информацию на консоль, то нам надо передать ее в метод Console.WriteLine:

```
1  using System;
2
3  namespace HelloApp
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              string hello = "Привет мир";
10             Console.WriteLine(hello);
11             Console.WriteLine("Добро пожаловать в C#!");
12             Console.WriteLine("Пока мир...");
13             Console.WriteLine(24.5);
14
15             Console.ReadKey();
16         }
17     }
18 }
```

Консольный вывод:

```
Привет мир!
```

```
Добро пожаловать в C#!
Пока мир...
24,5
```

Нередко возникает необходимость вывести на консоль в одной строке значения сразу нескольких переменных. В этом случае мы можем использовать прием, который называется интерполяцией:

```
1  using System;
2
3  namespace HelloApp
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              string name = "Tom";
10             int age = 34;
11             double height = 1.7;
12             Console.WriteLine($"Имя: {name}  Возраст: {age}  Рост: {height}м");
13
14             Console.ReadKey();
15         }
16     }
17 }
```

Для встраивания отдельных значений в выводимую на консоль строку используются фигурные скобки, в которые заключается встраиваемое значение. Это можем значение переменной ({name}) или более сложное выражение (например, операция сложения {4 + 7}). А перед всей строкой ставится знак доллара $.

При выводе на консоль вместо помещенных в фигурные скобки выражений будут выводиться их значения:

```
Имя: Tom  Возраст: 34  Рост: 1,7м
```

Есть другой способ вывода на консоль сразу нескольких значений:

```
1  using System;
2
3  namespace HelloApp
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              string name = "Tom";
10             int age = 34;
11             double height = 1.7;
12             Console.WriteLine("Имя: {0}  Возраст: {2}  Рост: {1}м", name, height, age);
13
14             Console.ReadKey();
15         }
16     }
17 }
```

Этот способ подразумевает, что первый параметр в методе Console.WriteLine представляет выводимую строку ("Имя: {0} Возраст: {2} Рост: {1}м"). Все последующие параметры представляют значения, которые могут быть встроенны в эту строку (name, height, age). При этом важен порядок подобных параметров. Например, в данном случае вначале идет name, потом height и потом age. Поэтому у name будет представлять параметр с номером 0 (нумерация начинается с нуля), height имеет номер 1, а age - номер 2. Поэтому в строке "Имя: {0} Возраст: {2} Рост: {1}м" на место плейсхолдеров {0}, {2}, {1} будут вставляться значения соответствующих параметров.

Кроме Console.WriteLine() можно также использовать метод **Console.Write()**, он работает точно так же за тем исключением, что не осуществляет переход на следующую строку.

Консольный ввод

Кроме вывода информации на консоль мы можем получать информацию с консоли. Для этого предназначен метод **Console.ReadLine()**. Он позволяет получить введенную строку.

```
1  using System;
2
3  namespace HelloApp
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.Write("Введите свое имя: ");
10             string name = Console.ReadLine();
11             Console.WriteLine($"Привет {name}");
12
13             Console.ReadKey();
14         }
15     }
16 }
```

В данном случае все, что вводит пользователь, с помощью метода Console.ReadLine передается в переменную name.

Пример работы программы:

```
Введите свое имя: Том
Привет Том
```

Таким образом мы можем вводить информацию через консоль. Однако минусом этого метода является то, что Console.ReadLine считывает информацию именно в виде строки. Поэтому мы можем по умолчанию присвоить ее только переменной типа string. Как нам быть, если, допустим, мы хотим ввести возраст в переменную типа int или другую информацию в переменные типа double или decimal? По умолчанию платформа .NET предоставляет ряд методов, которые позволяют преобразовать различные значения к типам int, double и т.д. Некоторые из этих методов:

- **Convert.ToInt32()** (преобразует к типу int)
- **Convert.ToDouble()** (преобразует к типу double)
- **Convert.ToDecimal()** (преобразует к типу decimal)

Пример ввода значений:

```
using System;


namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Введите имя: ");
            string name = Console.ReadLine();

            Console.Write("Введите возраст: ");
            int age = Convert.ToInt32(Console.ReadLine());

            Console.Write("Введите рост: ");
            double height = Convert.ToDouble(Console.ReadLine());

            Console.Write("Введите размер зарплаты: ");
            decimal salary = Convert.ToDecimal(Console.ReadLine());

            Console.WriteLine($"Имя: {name}  Возраст: {age}  Рост: {height}м  Зарплата: {salary}$");

            Console.ReadKey();
        }
    }
```

26	}

При вводе важно учитывать текущую операционную систему. В одних культурах разделителем между целой и дробной частью является точка (США, Великобритания...), в других - запятая (Россия, Германия...). Например, если текущая ОС - русскоязычная, значит, надо вводить дробные числа с разделителем запятой. Если локализация англоязычная, значит, разделителем целой и дробной части при вводе будет точка.

Пример работы программы:

```
Введите имя: Том
Введите возраст: 25
Введите рост: 1,75
Введите размер зарплаты: 300,67
Имя: Том  Возраст: 25  Рост: 1,75м  Зарплата: 300,67$
```

ПРАКТИЧЕСКАЯ РАБОТА 3. ВВЕДЕНИЕ В WINDOWS FORMS. СОЗДАНИЕ ГРАФИЧЕСКОГО ПРИЛОЖЕНИЯ

Цель работы: Изучение методов построения форм Windows и получение навыков по настройке форм, созданию непрямоугольных и наследуемых (производных) форм.

Упражнение *1.* Настройка прямоугольной формы *Windows*

Формы Windows — это основной компонент пользовательского интерфейса. Формы предоставляют контейнер, который содержит элементы управления, меню и позволяет отображать приложение в уже привычной и единообразной модели. Формы могут реагировать на события мыши и клавиатуры, поступающие от пользователя, и выводить на экран данные для пользователя с помощью элементов управления, которые содержатся в форме.

Формы Windows содержат множество свойств, позволяющих настраивать их внешний вид и поведение. Просматривать и изменять эти свойства можно в окне **Properties** конструктора при разработке, а также программно во время выполнения приложения.

В следующей таблице перечислены некоторые свойства форм

Windows, отвечающие за внешний вид и поведение приложения:

Свойство	Описание

Name	Задает имя классу **Form**, показанному в конструкторе. Данное свойство задается исключительно во время разработки
BackColor	Указывает цвет фона формы
Enabled	Указывает, может ли форма принимать ввод от пользователя. Если свойству **Enabled** задано значение **False**, все элементы управления формы также блокируются
ForeColor	Указывает цвет переднего плана формы, то есть цвет выводимого текста. Если отдельно не указать значение свойства **ForeColor** элементов управления формы, они примут то же значение
FormBorderStyle	Указывает вид и поведение границы и строки заголовка формы Значения свойства:
	None - Форма не имеет границы, не может быть минимизирована или развернута до максимальных размеров и у нее нет экранной кнопки управления окном и кнопки справки
	FixedSingle - Форма имеет тонкую границу, и размеры формы нельзя изменить во время выполнения. Форма может быть минимизирована, развернута до максимальных размеров, и иметь кнопку справки или кнопку управления окном, что определяется остальными свойствами
	Fixed3D - Форма имеет объемную границу, и размеры формы нельзя изменить во время выполнения. Форма может быть минимизирована, развернута до максимальных размеров, и иметь кнопку справки или кнопку управления окном, что определяется остальными свойствами
	FixedDialog - Форма имеет тонкую границу, и размеры формы нельзя изменить во время выполнения. У формы нет экранной кнопки управления окном, но может быть кнопка справки, что определяется остальными свойствами. Форму можно минимизировать и развернуть до максимальных размеров
	Sizable - Форма имеет настройки по умолчанию, но они могут изменяться пользователем. Форма может быть минимизирована, развернута до максимальных размеров, и иметь кнопку справки, что определяется остальными свойствами

	FixedToolWindow - Форма имеет тонкую границу, и размеры формы нельзя изменить во время выполнения. Форма содержит только кнопку закрытия
	SizableToolWindow - Форма имеет тонкую границу, и размеры формы могут быть изменены пользователем. Форма содержит только кнопку закрытия
Location	Когда свойству **StartPosition** задано значение **Manual**, это свойство указывает исходное положение формы относительно верхнего левого угла экрана
MaximizeBox	Указывает, есть ли у формы кнопка **MaximizeBox**
MaximumSize	Устанавливает максимальный размерформы. Если задать этому свойству размер 0; 0, у формы не будет верхнего ограничения размера
MinimizeBox	Указывает, есть ли у формы кнопка **MinimizeBox**
MinimumSize	Устанавливает минимальный размер формы, который пользователь может задать
Opacity	Устанавливает уровень непрозрачности или прозрачности формы от 0 до 100%. Форма, непрозрачность которой составляет 100%, полностью непрозрачна, а форма, имеющая 0 % непрозрачности, наоборот, полностью прозрачна
Size	Принимает и устанавливает исходный размер формы
StartPosition	Указывает положение формы в момент ее первого выведения на экран
Text	Указывает заголовок формы
TopMost	Указывает, всегда ли форма отображается поверх всех остальных форм, свойству **TopMost** которых не задано значение **True**
Visible	Указывает, видима ли форма во время работы
WindowState	Указывает, является ли форма минимизированной, развернутой до максимальных размеров, или же при первом появлении ей задан размер, указанный в свойстве **Size**

Создание нового проекта

Откройте Visual Studio и создайте новый проект Windows Forms. Проект откроется с формой по умолчанию с именем **Form1** в конструкторе.

Выберите форму в конструкторе. Свойства формы отображаются в окне **Properties**.

В окне **Properties** задайте свойствам значения, как указано ниже:

Свойство	Значение
Text	Trey Research
FormBorderStyle	Fixed3D
StartPosition	Manual
Location	100; 200
Opacity	75%

Перетащите три кнопки из **Toolbox** в форму и разместите их так, как вам будет удобно.

Поочередно выберите каждую кнопку и в окне **Properties** задайте свойству кнопок **Text** значения **Border Style, Resize** и **Opacity.**

Для кнопки **Border Style** задайте свойство **Anchor** – **Top, Left.**

Реализация обработчиков событий

В конструкторе дважды щелкните кнопку **Border Style**, чтобы открыть окно с кодом обработчика события **Buttonl Click.** Добавьте в этот метод следующую строку кода:

this.FormBorderStyle = FormBorderStyle.Sizable;

Возвратитесь в окно конструктора, дважды щелкните кнопку Resize

и добавьте следующую строку:

this.Size = new Size(300, 500);

Возвратитесь в окно конструктора, дважды щелкните кнопку

Opacity и добавьте следующую строку:

this.Opacity = 1;

Запуск готового решения

Для построения решения выберите меню **Build (**Построение**)**, далее команду **Build Solution (**Построить решение**)**. При наличии ошибок исправьте их и снова постройте решение. В дальнейшем при необходимости выбора последовательности действий очередность команд будет описываться, например, так: **Build | Build Solution.**

Нажмите Ctrl + F5 или выберите **Debug (**Отладка**) | Start Without Debugging (**Запуск без отладки**)**, чтобы запустить приложение. Щелкайте каждую кнопку и наблюдайте, как изменяется вид формы.

Измените поочередно расположение левой и верхней границ формы и сравните поведение кнопок внутри формы. Обратите внимание, что расстояние до этих границ от кнопки **Border Style** остается постоянным. Почему?

Упражнение ***2.*** Создание непрямоугольной формы ***Windows***

В этом упражнении вы создадите треугольную форму Windows.

Откройте Visual Studio и создайте новый проект Windows Forms. Проект откроется с

формой по умолчанию с именем **Form1** в конструкторе.

В окне Properties задайте свойству **FormBorderStyle** значение **None,** а свойству **BackColor** значение **Red.** В этом случае форму легче будет увидеть при тестировании приложения.

Перетащите кнопку из Toolbox в левый верхний угол формы. Задайте свойству **Text** кнопки значение **Close Form.**

Дважды щелкните кнопку Close Form и добавьте в обработчик события **Button1 Click** следующий код:

this.Close();

В конструкторе дважды щелкните форму, чтобы открыть обработчик события **Form1 Load.** Добавьте в этот метод следующий код (он задает области формы треугольную форму указанием многоугольника с тремя углами):

System.Drawing.Drawing2D.GraphicsPath myPath = new System.Drawing.Drawing2D.GraphicsPath();

myPath.AddPolygon(new Point[] { new Point(0, 0), new Point(0, this.Height),

new Point(this.Width, 0) });

Region myRegion = new Region(myPath); this.Region = myRegion;

Постройте и запустите приложение. Появится треугольная форма.

Упражнение ***3.*** Создание наследуемой формы

Если у вас имеется уже готовая форма, которую вы собираетесь использовать в нескольких приложениях, удобно создать наследуемую (производную) форму. В этом упражнении вы создадите новую форму и унаследуете ее от существующей базовой формы, а затем измените производную форму, настроив ее для конкретной работы.

Откройте проект из предыдущего упражнения. Базовой формой для создания производной будет треугольная форма.

Для кнопки **Close Form** задайте свойство **Modifiers** как **protected**.

Добавьте производную форму: меню **Project (**Проект**) | Add Windows Form…(**Добавить форму **Windows)**, в окне **Categories (**Категории**)** укажите **Windows Form**, в окне **Templates (**Шаблоны**)** выберите **Inherited Form (**Наследуемая форма**).**

В окне **Add New Item** в поле **Name** укажите название формы: nForm.cs и нажмите **Add** для добавления формы.

В появившемся окне **Inheritance Picker**, в котором отображаются все формы текущего проекта, выберите базовую форму Form1 и нажмите OK.

Постройте проект.

Откройте форму nForm в режиме конструктора. Проверьте, что она имеет треугольную форму и свойства базовой формы и элемента управления наследованы.

Настройте свойства производной формы:

для кнопки:

свойство **Text** – Hello!!!

свойство **BackColor** – Brown

для формы: свойство **BackColor** – Blue

Постройте проект.

Задайте производную форму в качестве стартовой, указав в функции **Main** следующий код:

Application.Run(new nForm());

Постройте и запустите приложение. Должна открыться производная форма со своими свойствами. Проверьте, наследуется ли закрытие формы кнопкой.

Упражнение ***4.*** Создание ***MDI*****-**приложения

В этом упражнении Вы создадите MDI-приложение с родительской формой, загружающей и организующей дочерние формы. Также Вы познакомитесь с элементом управления **MenuStrip**, который позволяет создать меню формы.

Создание нового проекта с базовой формой

Создайте новый проект Windows Forms, укажите имя

MdiApplication.

Переименуйте файл Form1.cs на ParentForm.cs.

Для формы задайте следующие свойства:

Name	**ParentForm**
Size	420; 320
Text	Parent Form

Проверьте, что произошли изменения в функции Main так, чтобы форма ParentForm стала стартовой.

Откройте файл ParentForm.cs в режиме конструктора.

Для свойства формы **IsMdiContainer** задайте значение **True.**

Таким способом эта форма будет определена как родительская форма MDI.

Создание меню для работы с формами

Создайте пункт меню **File:**

Откройте ПИ **Toolbox**, добавьте на форму ЭУ **MenuStrip** и задайте для его свойства **Name** значение **MdiMenu**.

Выделите меню в верхней части формы и задайте имя первого пункта меню **&File**.

Для свойства **Name** пунктаменю **File** задайте значение

FileMenuItem.

Раскройте меню **File**.

Выделите элемент, появившейся под элементом **File,** и задайте его как **&New**.

Для свойства **Name** пунктаменю **New** задайте значение

NewMenuItem.

Выделите элемент, появившийся под элементом **New,** и задайте его как **&Exit.**

Для свойства **Name** пунктаменю **Exit** задайте значение

ExitMenuItem.

Дважды кликните левой кнопкой мыши по пункту меню **Exit**

для создания обработчика события **Click**.

В обработчик события **Click** для пункта меню **Exit** добавьте следующий код:

this.Close();

Создайте пункт меню **Window:**

Переключитесь в режим конструктора.

Выделите второй пункт меню справа от **File** и задайте его значением **&Window**.

Для свойства **Name** пункта меню **Window** задайте значение

WindowMenuItem.

Раскройте меню **Window**.

Выделите элемент, появившейся под элементом **Window,** и задайте для его свойства **Text** значение **&Cascade**.

Для свойства **Name** пункта меню **Cascade** задайте значение

WindowCascadeMenuItem.

Выделите элемент, появившийся под элементом **Cascade,** и задайте для его свойства **Text** значение **&Tile.**

Для свойства **Name** пунктаменю **Tile** задайте значение

WindowTileMenuItem.

Дважды кликните левой кнопкой мыши по пункту меню

Cascade для создания обработчика события **Click**:

this.LayoutMdi (System.Windows.Forms.MdiLayout.Cascade);

j. Вернитесь в режим конструктора и дважды кликните левой кнопкой мыши по пункту меню **Tile.**

k. В обработчик события **Click** для пункта меню **Tile** добавьте следующий код:

this.LayoutMdi(System.Windows.Forms.MdiLayout.TileHorizontal);

Реализуйте список открытых окон в меню **Window**:

В конструкторе выберите компонент **Mdimenu**. Укажите в свойстве **MdiWindowListItem** имя пункта, созданного для этого – **WindowMenuItem.**

Создание дочерней формы

Создайте дочернюю форму:

Выберите пункт меню **Project | Add Windows Form.**

Задайте имя формы **ChildForm.cs**.

Для свойства **Text** формы задайте значение **Child Form**.

На ПИ **Toolbox** дважды кликните левой кнопкой мыши по ЭУ **RichTextBox** и задайте для его свойства **Name** значение **ChildTextBox**.

Для свойства **Dock** ЭУ **RichTextBox** задайте значение **Fill**.

Удалите существующий текст (если он есть) для свойства **Text**

ЭУ **RichTextBox** и оставьте его пустым.

На ПИ **Toolbox** дважды кликните левой кнопкой мыши по ЭУ

MenuStrip.

Для свойства **Name** ЭУ **MenuStrip** задайте значение

ChildWindowMenu.

Выделите меню в верхней части формы и наберите текст

F&ormat.

Для свойства **Name** пункта меню **Format** задайте значение **FormatMenuItem**, для свойства **MergeAction** установите значение **Insert**, а свойству **MergeIndex – 1**. В этом случае меню **Format** будет располагаться после **File** при объединении базового и дочерних меню.

a. Выделите элемент, появившийся под элементом **Format,** и наберите текст **&Toggle Foreground**.
b. Для свойства **Name** пункта меню **Toggle Foreground** задайте значение **ToggleMenuItem**.

c. Дважды кликните левой кнопкой мыши по пункту меню **Toggle Foreground** и добавьте следующий код в обработчик события **Click**:

```
if (ToggleMenuItem.Checked)
{
 ToggleMenuItem.Checked = false; ChildTextBox.ForeColor =
 System.Drawing.Color.Black;
}
 else
{
        ToggleMenuItem.Checked = true;
    ChildTextBox.ForeColor = System.Drawing.Color.Blue;
   }
```

Отображение дочерней формы

11. Отобразите дочернюю форму в родительской форме:
    a. Откройте ParentForm.cs в режиме конструктора.
    b. Дважды кликните левой кнопкой мыши по кнопке **New** в меню **File** для создания обработчика события **Click.**
    c. Добавьте следующий код для обработчика события **Click** для пункта меню **New:**

```
ChildForm newChild = new ChildForm();
newChild.MdiParent = this; newChild.Show();
```

Работа с приложением

12. Проверьте работу приложения:
    a. Постройте и запустите приложение.
    b. Когда появится родительская форма, выберите пункт меню **File | New**.
    c. В родительском окне появится новая дочерняя форма. Обратите внимание на то, дочернее меню сливается с родительским и пункты меню упорядочиваются в соответствие со свойством **MergeIndex**, установленным ранее.
    d. Наберите какой-нибудь текст в дочернем окне и воспользуйтесь пунктом меню **Format** для изменения цвета шрифта текста.
    e. Откройте еще несколько дочерних окон.
    f. Выберите пункт меню **Window | Tile**. Обратите внимание на то, что дочерние окна выстраиваются в упорядоченном порядке.
    g. Закройте все дочерние окна.
    h. Обратите внимание на то, что, когда закроется последнее дочернее окно, меню родительской формы изменится, и оттуда исчезнет пункт **Format**.
    i. Для закрытия приложения выберите пункт меню **File | Exit**.

13. Обратите внимание, что заголовок у дочерних окон одинаковый. При создании нескольких документов, например в Microsoft Word, они называются ДокументN, где N — номер документа. Реализуйте эту возможность:

a. Откройте код родительской формы и в классе ParentForm объявите переменную openDocuments:

```
private int openDocuments = 0;
```

b. К свойству **Text** дочерней формы добавьте счетчик числа открываемых документов (в коде обработчика события **Click** для пункта меню **New)**:

```
newChild.Text = newChild.Text+" "+ ++openDocuments;
```

14. Запустите приложение. Теперь заголовки новых документов содержат порядковый номер.

Дополнительное упражнение

Для углубления знаний о добавлении и настройке форм Windows выполните следующие задания.

Задание **1.** Создайте пользовательскую форму, которая во время выполнения будет иметь овальное очертание. Данная форма должна содержать функциональность, дающую возможность пользователю закрывать ее во время выполнения.

Рекомендация: при разработке формы в виде эллипса используйте следующий код:

```
// Добавление эллипса, вписанного в прямоугольную форму
// заданной ширины и высоты
myPath.AddEllipse(0, 0, this.Width, this.Height);
```

Задание **2**. Создайте приложение с двумя формами и установите вторую форму как стартовую. Сделайте так, чтобы при запуске стартовая форма разворачивалась до максимальных размеров и содержала функциональность, дающую возможность пользователю открыть первую форму, отображающуюся в виде ромба зеленого цвета с кнопкой (в центре ромба) закрытия формы с надписью GREENPEACE.

ПРАКТИЧЕСКАЯ РАБОТА 4. РАЗРАБОТКА ОКОННОГО ПРИЛОЖЕНИЯ С ЛИНЕЙНЫМ ВЫЧИСЛИТЕЛЬНЫМ ПРОЦЕССОМ

Цель работы: научиться составлять каркас простейшей программы в среде Visual Studio. Написать и отладить программу линейного алгоритма.

Краткие теоретические сведения

Структура приложения

Перед началом программирования необходимо создать проект. *Проект* содержит все исходные материалы для приложения, такие как файлы исходного кода, ресурсов, значки, ссылки на внешние файлы, на которые опирается программа, и данные конфигурации, такие как параметры компилятора.

Кроме понятия проект часто используется более глобальное понятие – *решение (solution).*

Решение содержит один или несколько проектов, один из которых может быть указан как стартовый проект. Выполнение решения начинается со стартового проекта.

Таким образом, при создании простейшей C# программы в Visual Studio создается папка решения, в которой для каждого проекта создается подпапка проекта, а уже в ней будут создаваться другие подпапки с результатами компиляции приложения.

Проект – это основная единица, с которой работает программист. При создании проекта можно выбрать его тип, а Visual Studio создаст каркас проекта в соответствии с выбранным типом.

В предыдущей работе мы попробовали создавать оконные приложения, или иначе *Приложения Windows Forms*. Примером другого типа проекта является привести проект *консольного* приложения.

По своим "внешним" проявлениям консольные напоминают приложения DOS, запущенные в Windows. Тем не менее, это настоящие Win32-приложения, которые под DOS работать не будут. Для консольных приложений доступен Win32 API, а кроме того, они могут использовать консоль – окно, предоставляемое системой, которое работает в текстовом режиме и в которое можно вводить данные с клавиатуры. Особенность консольных приложений в том, что они работают не в графическом, а в текстовом режиме.

Проект в Visual Studio состоит из файла проекта (файл с расширением .csproj), одного или нескольких файлов исходного текста (с расширением .cs), файлов с описанием окон формы (с расширением .designer.cs), файлов ресурсов (с расширением .resx), а также ряда служебных файлах.

В *файле проекта* находится информация о модулях, составляющих данный проект, входящих в него ресурсах, а также параметров построения программы. Файл проекта автоматически создается и изменяется средой Visual Studio и не предназначен для ручного редактирования.

Файл исходного текста – программный модуль предназначен для размещения текстов программ. В этом файле программист размещает текст программы, написанный на языке C#. Модуль имеет следующую структуру:

```
// Раздел подключенных пространств имен using System;


// Пространство имен нашего проекта
namespace MyFirstApp
{
        // Класс окна
        public partial class Form1 : Form
```

```
        {
        // Методы окна                public Form1()
            {
                InitializeComponent();
            }
        }
}
```

В разделе подключения пространств имен (каждая строка которого располагается в начале файла и начинается ключевым словом using) описываются используемые пространства имён. Каждое пространство имён включает в себя классы, выполняющие определённую работу, например, классы для работы с сетью располагаются в пространстве System.Net, а для работы с файлами – в System.IO. Большая часть пространств, которые используются в обычных проектах, уже подключена при создании нового проекта, но при необходимости можно дописать дополнительные пространства имён.

Для того чтобы не происходило конфликтов имён классов и переменных, классы нашего проекта также помещаются в отдельное пространство имен. Определяется оно ключевым словом namespace, после которого следует имя пространства (обычно оно совпадает с именем проекта).

Внутри пространства имен помещаются наши классы – в новом проекте это класс окна, который содержит все методы для управления поведением окна. Обратите внимание, что в определении класса присутствует ключевое слово partial, это говорит о том, что в исходном тексте представлена только часть класса, с которой мы работаем непосредственно, а служебные методы для обслуживания окна скрыты в другом модуле (при желании их тоже можно посмотреть, но редактировать вручную не рекомендуется).

Наконец, внутри класса располагаются переменные, методы и другие элементы программы. Фактически, основная часть программы размещается внутри класса при создании обработчиков событий.

При компиляции программы Visual Studio создает исполняемые .exe-файлы в каталоге bin.

Работа с проектом

Проект в Visual Studio состоит из многих файлов, и создание сложной программы требует хранения каждого проекта в отдельной папке. При создании нового проекта Visual Studio по умолчанию сохраняет его в отдельной папке. Рекомендуется создать для себя свою папку со своей фамилией внутри папки своей группы, чтобы все проекты хранились в одном месте. После этого можно запускать Visual Studio и создавать новый проект (как это сделать показано в предыдущей работе).

Сразу после создания проекта рекомендуется сохранить его в подготовленной папке: *Файл*

→ *Сохранить всё*. При внесении значительных изменений в проект следует еще раз сохранить проект той же командой, а перед запуском программы на выполнение среда обычно сама сохраняет проект на случай какого-либо сбоя. Для открытия существующего проекта используется команда *Файл → Открыть проект*, либо можно найти в папке файл проекта с расширением .sln и сделать на нём двойной щелчок.

Описание данных

Типы данных имеют особенное значение в C#, поскольку это *строго типизированный язык*. Это означает, что все операции подвергаются строгому контролю со стороны компилятора на соответствие типов, причем недопустимые операции не компилируются. Такая строгая проверка типов позволяет предотвратить ошибки и повысить надежность программ. Для обеспечения контроля типов все переменные, выражения и значения должны принадлежать к определенному типу (рис. 2.1). Такого понятия, как *бестиповая* переменная, допустимая в ряде скриптовых языков, в C# вообще не существует. Более того, тип значения определяет те операции, которые разрешается выполнять над ним. Операция, разрешенная для одного типа данных, может оказаться недопустимой для другого.

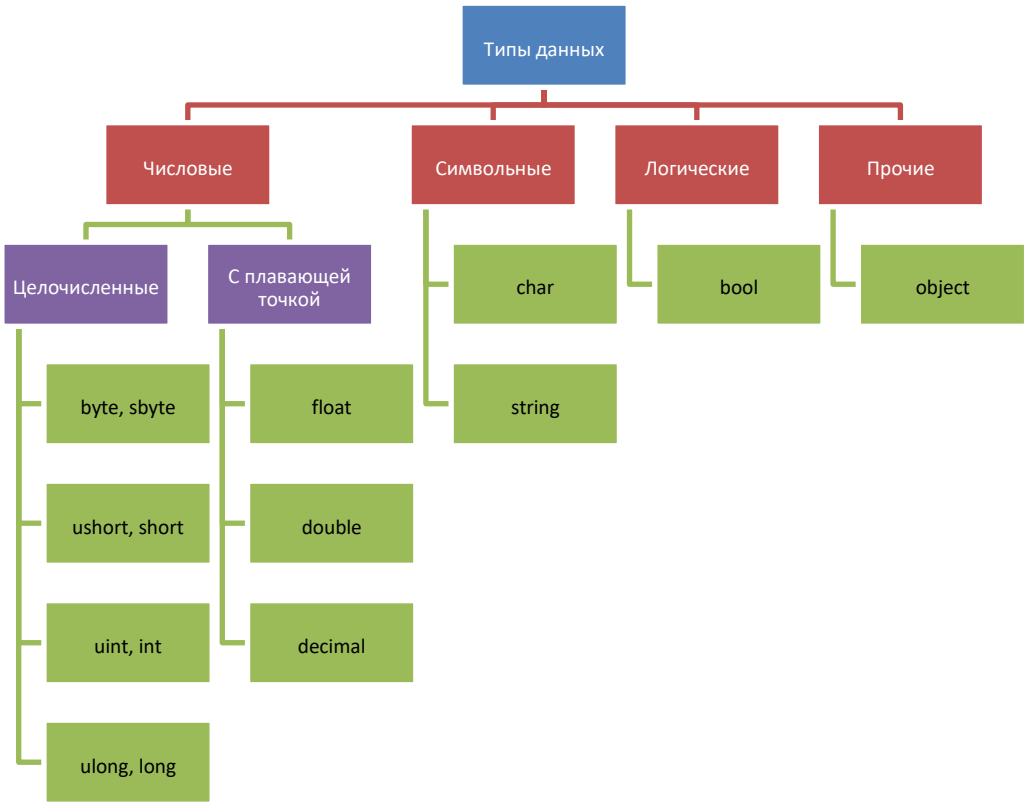

Рис.2.1 Структура типов данных

Целочисленные типы

В C# определены девять целочисленных типов: char, byte, sbyte, short, ushort, int, uint, long и ulong. Тип char может хранить числа, но чаще используется для представления символов. Остальные восемь целочисленных типов предназначены для числовых расчетов.

Некоторые целочисленные типы могут хранить как положительные, так и отрицательные

значения (sbyte, short, int и long), другие же – только положительные (char, byte, ushort, uint и ulong).

Типы с плавающей точкой

Такие типы позволяют представлять числа с дробной частью. В C# имеются три разновидности типов данных с плавающей точкой: float, double и decimal. Первые два типа представляют числовые значения с одинарной и двойной точностью, вычисления над ними выполняются аппаратно и поэтому быстро. Тип decimal служит для представления чисел с плавающей точкой высокой точности без округления, характерного для типов float и double. Вычисления с использованием этого типа выполняются программно и поэтому более медленны.

Числа, входящие в выражения, C# по умолчанию считает целочисленными. Поэтому следующее выражение будет иметь значение 0, ведь если 1 нацело разделить на 2 то получится как раз 0:

```
double x = 1 / 2;
```

Чтобы этого не происходило, в подобных случаях нужно явно указывать тип чисел с помощью символов-модификаторов: f для float и d для double. Приведённый выше пример правильно будет выглядеть так:

```
double x = 1d / 2d;
```

Иногда в программе возникает необходимость записать числа в экспоненциальной форме. Для этого после мантиссы числа записывается символ «e» и сразу после него – порядок. Например, число $2{,}5 \cdot 10^{-2}$ будет записано в программе следующим образом:

```
2.5e-2
```

Символьные типы

В C# символы представлены не 8-разрядным кодом, как во многих других языках программирования, а 16-разрядным кодом, который называется юникодом (Unicode). В юникоде набор символов представлен настолько широко, что он охватывает символы практически из всех естественных языков на свете.

Основным типом при работе со строками является тип string, задающий строки переменной длины.

Тип string представляет последовательность из нуля или более символов в кодировке Юникод. По сути, текст хранится в виде последовательной доступной только для чтения коллекции объектов char.

Логический тип данных

Тип bool представляет два логических значения: «истина» и «ложь». Эти логические

значения обозначаются в C# зарезервированными словами true и false соответственно. Следовательно, переменная или выражение типа bool будет принимать одно из этих логических значений.

Рассмотрим самые популярные данные – *переменные* и *константы*. Переменная – это ячейка памяти, которой присвоено некоторое имя и это имя используется для доступа к данным, расположенным в данной ячейке. Для каждой переменной задаётся *тип данных* – диапазон всех возможных значений для данной переменной. Объявляются переменные непосредственно в тексте программы. Лучше всего сразу присвоить им начальное значение с помощью знака присвоения «=» (*переменная = значение*):

```
int a;  // Только объявление int b = 7; // Объявление и инициализация
```

Для того чтобы присвоить значение символьной переменной, достаточно заключить это значение (т. е. символ) в одинарные кавычки:

```
char ch;            // Только объявление char symbol = 'Z'; // Объявление и инициализация
```

Частным случаем переменных являются *константы*. Константы – это переменные, значения которых не меняются в процессе выполнения программы. Константы описываются как обычная переменная, только с ключевым словом const впереди:

```
const int c = 5;
```

Ввод/вывод данных в программу

Рассмотрим один из способов ввода данных через элементы, размещенные на форме. Для ввода данных чаще всего используют элемент управления TextBox, через обращение к его свойству Text. Свойство Text хранит в себе строку введенных символов. Поэтому данные можно считать таким образом:

```
private void button1_Click(object sender,EventArgs e)
{
        string s = textBox1.Text;
}
```

Однако со строкой символов трудно производить арифметические операции, поэтому лучше всего при вводе числовых данных перевести строку в целое или вещественное число. Для этого у типов, или int и double существуют методы Parse для преобразования строк в числа. С этими числами можно производить различные арифметические действия. Таким образом, предыдущий пример можно переделать следующим образом:

```
private void button1_Click(object sender, EventArgs e)
```

```
{       string s = textBox1.Text;       int a = int.Parse(s);
                int b = a * a;
}
```

Важное примечание! В языках программирования в дробных числах чаще всего используется точка, например: «15.7». Однако в C# методы преобразования строк в числа

(вроде double.Parse() или Convert.ToFloat()) учитывают региональные настройки Windows, в которых в качестве десятичной точки используется символ *запятой* (например, «15,7»). Поэтому в полях TextBox в формах следует вводить дробные числа с *запятой*, а не с точкой. В противном случае преобразование не выполнится, а программа остановится с ошибкой.

Перед выводом числовые данные следует преобразовать назад в строку. Для этого у каждой переменной существует метод ToString(), который возвращает в результате строку с символьным представлением значения. Вывод данных можно осуществлять в элементы TextBox или Label, использую свойство Text. Например:

```
private void button1_Click(object sender, EventArgs e)
{       string s = textBox1.Text;       int a = int.Parse(s);     int b = a * a;    label1.Text =
b.ToString();
}
```

Арифметические действия и стандартные функции

При вычислении выражения стоящего в правой части оператора присвоения могут использоваться арифметические операции:

* – умножение;

+ – сложение;

– – вычитание;

/ – деление;

% – остаток от деления.

Для задания приоритетов операций могут использоваться круглые скобки ( ). Также могут использоваться стандартные математические функции, представленные методами класса Math:

Math.Sin(a) – синус;

Math.Sinh(a) – гиперболический синус;

Math.Cos(a) – косинус (аргумент задается в радианах);

Math.Atan(a) – арктангенс (аргумент задается в радианах);

Math.Log(a) – натуральный логарифм;

Math.Exp(a) – экспонента;

Math.Pow(x, y) – возводит переменную x в степень y;

Math.Sqrt(a) – квадратный корень;

Math.Abs(a) – модуль числа;

Math.Truncate(a) – целая часть числа; Math.Round(a) – округление числа;

В тригонометрических функциях все аргументы задаются в радианах.

Пример написания программы

Задание: составить программу вычисления для заданных значений x, y, z арифметического выражения

$u = tg^2(x+y)e^{y-z}\cos x^2 - \sin\sqrt{z^2}$.

Панель диалога программы организовать в виде, представленном на рис:

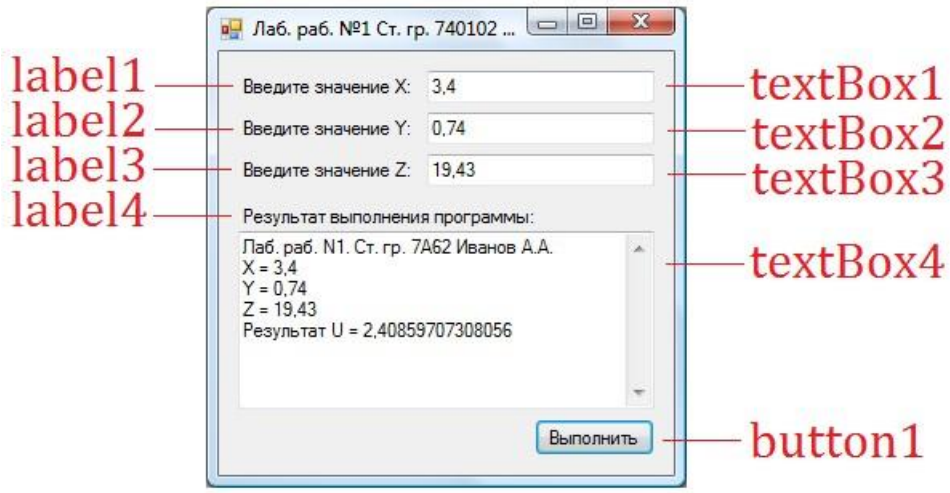

Рис 2.1. Внешний вид программы

Для вывода результатов работы программы в программе используется текстовое окно, которое представлено обычным элементом управления. После установки свойства Multiline в True появляется возможность растягивать элемент управления не только по горизонтали, но и по вертикали. А после установки свойства ScrollBars в значение Both в окне появится вертикальная, а при необходимости и горизонтальная полосы прокрутки.

Информация, которая отображается построчно в окне, находится в массиве строк Lines, каждая строка которого имеет тип string. Однако нельзя напрямую обратиться к этому

свойству для добавления новых строк, поскольку размер массивов в C# определяется в момент их инициализации. Для добавления нового элемента используется свойство Text, к текущему содержимому которого можно добавить новую строку:

```
textBox4.Text += Environment.NewLine + "Привет";
```

В этом примере к текущему содержимому окна добавляется символ перевода курсора на новую строку (который может отличаться в разных операционных системах, и потому представлен свойством класса Environment) и сама новая строка. Если добавляется числовое значение, то его предварительно нужно привести в символьный вид методом ToString().

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) кнопку «*Выполнить*». В окне textBox4 появляется результат. Измените исходные значения x, y, z в окнах textBox1– textBox3 и снова нажмите кнопку «*Выполнить*» – появится новые результаты.

Полный текст программы имеет следующий вид:

```
using System;
using System.Windows.Forms;

namespace MyFirstApp
{
        public partial class Form1 : Form
        {
                public Form1()
                {
                        InitializeComponent();
                }

                private void Form1_Load(object sender,
                        EventArgs e)
                {
                // Начальное значение X                textBox1.Text = "3,4";
```

```
            // Начальное значение Y                    textBox2.Text = "0,74";
        // Начальное значение Z                    textBox3.Text = "19,43";

                }


                private void button1_Click(object sender,

                    EventArgs e)

                {

                    // Считывание значения X

                double x = double.Parse(textBox1.Text);     // Вывод значения X в окно
                textBox4.Text += Environment.NewLine +

                        "X = " + x.ToString();

                    // Считывание значения Y

                double y = double.Parse(textBox2.Text); // Вывод значения Y в окно
            textBox4.Text += Environment.NewLine +

                    "Y = " + y.ToString();  // Считывание значения Z

                double z = double.Parse(textBox3.Text);     // Вывод значения Z в окно
                textBox4.Text += Environment.NewLine + "Z = " + z.ToString();

                    // Вычисляем арифметическое выражение

                double a = Math.Tan(x + y) *  Math.Tan(x + y);

                double b = Math.Exp(y - z);

double c = Math.Sqrt(Math.Cos(x*x) + Math.Sin(z*z));

                double u = a -b * c;                    // Выводим результат в окно
            textBox4.Text += Environment.NewLine +

                        "Результат U = " + u.ToString();

                }

            }

}
```

Если просто скопировать этот текст и заменить им то, что было в редакторе кода Visual Studio, то программа не заработает. Правильнее будет создать обработчики событий Load у формы и Click у кнопки и уже в них вставить соответствующий код. Это замечание относится и ко всем последующим лабораторным работам.

Выполнение индивидуального задания

Ниже приведено 15 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с этим установите необходимое количество окон TextBox, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов. Для проверки правильности программы после задания приведен контрольный пример: тестовые значения переменных, используемых в выражении, и результат, который при этом получается.

Индивидуальные задания

1. $$t = \frac{2\cos\left(x - \frac{\pi}{6}\right)}{0.5 + \sin^2 y}\left(1 + \frac{z^2}{3 - z^2/5}\right).$$

При x=14.26, y=-1.22, z=3.5×10^{-2} t=0.564849.

2. $$u = \frac{\sqrt[3]{8 + |x - y|^2 + 1}}{x^2 + y^2 + 2} - e^{|x-y|}\left(tg^2 z + 1\right)^x.$$

При x=-4.5, y=0.75×10^{-4},z=0.845×10^{2} u=-55.6848.

3. $$v = \frac{1 + \sin^2(x + y)}{\left|x - \frac{2y}{1 + x^2 y^2}\right|} x^{|y|} + \cos^2\left(arctg\frac{1}{z}\right).$$

При x=3.74×10^{-2}, y=-0.825, z=0.16×10^{2}, v=1.0553.

4. $$w = \left|\cos x - \cos y\right|^{\left(1 + 2\sin^2 y\right)}\left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4}\right).$$

При x=0.4×10^{4}, y=-0.875, z=-0.475×10^{-3} w=1.9873.

5. $\alpha = \ln\left(y^{-\sqrt{|x|}}\right)\left(x - \frac{y}{2}\right) + \sin^2 arctg(z).$

При x=-15.246, y=4.642×10^{-2}, z=20.001×10^{2} α=-182.036.

6. $\beta = \sqrt{10\left(\sqrt[3]{x} + x^{y+2}\right)}\left(\arcsin^2 z - |x-y|\right).$

При x=16.55×10^{-3}, y=-2.75, z=0.15 β=-38.902.

7. $\gamma = 5arctg(x) - \frac{1}{4}\arccos(x)\frac{x + 3|x-y| + x^2}{|x-y|z + x^2}.$

При x=0.1722, y=6.33, z=3.25×10^{-4} γ=-172.025.

8. $\varphi = \frac{e^{|x-y|}|x-y|^{x+y}}{arctg(x) + arctg(z)} + \sqrt[3]{x^6 + \ln^2 y}.$

При x=-2.235×10^{-2}, y=2.23, z=15.221 φ=39.374.

9. $\psi = \left|x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}}\right| + (y-x)\frac{\cos y - \frac{z}{(y-x)}}{1 + (y-x)^2}.$

При x=1.825×10^{2}, y=18.225, z=-3.298×10^{-2} ψ=1.2131.

10. $a = 2^{-x}\sqrt{x + \sqrt[4]{|y|}}\sqrt[3]{e^{x-1/\sin z}}.$

При x=3.981×10^{-2}, y=-1.625×10^{3}, z=0.512 a=1.26185.

11. $b = y^{\sqrt[3]{|x|}} + \cos^3(y)\frac{|x-y|\left(1 + \frac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{|x-y|} + \frac{x}{2}}.$

При x=6.251, y=0.827, z=25.001 b=0.7121.

12. $c = 2^{(y^x)} + (3^x)^y - \frac{y\left(arctgz - \frac{\pi}{6}\right)}{|x| + \frac{1}{y^2+1}}.$

При x=3.251, y=0.325, z=0.466×10^{-4} c=4.025.

13. $$f=\frac{\sqrt[4]{y+\sqrt[3]{x-1}}}{|x-y|\left(\sin^2 z+tgz\right)}.$$

При x=17.421, y=10.365×10^{-3}, z=0.828×10^{5} f=0.33056.

14. $$g=\frac{y^{x+1}}{\sqrt[3]{|y-2|}+3}+\frac{x+\frac{y}{2}}{2|x+y|}(x+1)^{-1/\sin z}.$$

При x=12.3×10^{-1}, y=15.4, z=0.252×10^{3} g=82.8257.

15. $$h=\frac{x^{y+1}+e^{y-1}}{1+x|y-tgz|}\left(1+|y-x|\right)+\frac{|y-x|^2}{2}-\frac{|y-x|^3}{3}.$$

При x=2.444, y=0.869×10^{-2}, z=-0.13×10^{3} h=-0.49871.

ПРАКТИЧЕСКАЯ РАБОТА 5. РАЗРАБОТКА ОКОННОГО ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

Цель работы: научиться пользоваться элементами управления для организации переключений (RadioButton). Написать и отладить программу разветвляющегося алгоритма.

Краткие теоретические сведения

Логические переменные и операции над ними

Переменные логического типа описываются посредством служебного слова bool. Они могут принимать только два значения – False (ложь) и True (истина). Результат False (ложь) и True (истина) возникает при использовании операций сравнения > (больше), < (меньше), != (не равно), >= (больше или равно), <= (меньше или равно), == (равно). Описываются логические переменные следующим образом:

```
bool b;
```

В языке C# имеются логические операции, применяемые к переменным логического типа. Это операции *логического отрицания* (!), *логическое И* (&&) и *логическое ИЛИ* (||). Операция логического отрицания является *унарной операцией*. Результат операции ! есть

False, если операнд истинен, и True, если операнд имеет значение ложь. Так,

!True → False (неправда есть ложь)

!False → True (не ложь есть правда)

Результат операции *логическое И* (&&) есть истина, только если оба ее операнда истинны, и ложь во всех других случаях. Результат операции *логическое ИЛИ* (||) есть истина, если какой-либо из ее операндов истинен, и ложен только тогда, когда оба операнда ложны.

3.2. Условные операторы

Операторы ветвления позволяют изменить порядок выполнения операторов в программе. К операторам ветвления относятся условный оператор if и оператор выбора switch.

Условный оператор if используется для разветвления процесса обработки данных на два направления. Он может иметь одну из форм: сокращенную или полную.

Форма сокращенного оператора if:

```
if (B) S;
```

где B – логическое или арифметическое выражение, истинность ко-

торого проверяется; S – оператор.

При выполнении сокращенной формы оператора if сначала вычисляется выражение B, затем проводится анализ его результата: если B истинно, то выполняется оператор S; если B ложно, то оператор S пропускается. Таким образом, с помощью сокращенной формы оператора if можно либо выполнить оператор S, либо пропустить его. Форма полного оператора if:

```
if (B) S1; else S2;
```

где B – логическое или арифметическое выражение, истинность ко-

торого проверяется; S1, S2 – операторы.

При выполнении полной формы оператора if сначала вычисляется выражение B, затем анализируется его результат: если B истинно, то выполняется оператор S1, а оператор S2 пропускается; если B ложно, то выполняется оператор S2, а S1 – пропускается. Таким образом, с помощью полной формы оператора if можно выбрать одно из двух альтернативных действий процесса обработки данных. Для примера, вычислим значение функции:

$$y(x) = \begin{cases} \sin(x), \text{если } x < a \\ \cos(x), \text{если } a < x < b \\ tg(x), \text{если } x > b \end{cases}$$

Указанное выражение может быть запрограммировано в виде

```
if (x <= a)                y = Math.Sin(x);
if ((x > a) && (x < b))                    y = Math.Cos(x);
if (x >= b)
        y = Math.Sin(x) / Math.Cos(x);
```

или с помощью оператора else:

```
if (x <= a)      y = Math.Sin(x);
else
    if (x < b)              y = Math.Cos(x);      else
                    y = Math.Sin(x) / Math.Cos(x);
```

Важное примечание! В С-подобных языках программирования, к которым относится и C#, операция сравнения представляется двумя знаками равенства, например: if (a == b)

Оператор выбора switch предназначен для разветвления процесса вычислений по нескольким направлениям. Формат оператора:

```
switch (<выражение>)
{       case <константное_выражение_1>:
                [<оператор 1>];
        <оператор перехода>;      case <константное_выражение_2>:
        [<оператор 2>];           <оператор перехода>;      ...      case
<константное_выражение_n>:
        [<оператор n>];           <оператор перехода>;      [default:
                <оператор>;]
}
```

Замечание. Выражение, записанное в квадратных скобках, является необязательным элементом в операторе switch. Если оно отсутствует, то может отсутствовать и оператор перехода.

Выражение, стоящее за ключевым словом switch, должно иметь арифметический, символьный или строковый тип. Все константные выражения должны иметь разные значения, но их тип должен совпадать с типом выражения, стоящим после switch или приводиться к нему. Ключевое слово case и расположенное после него константное выражение называют также *меткой* case.

Выполнение оператора начинается с вычисления выражения, расположенного за ключевым словом switch. Полученный результат сравнивается с меткой case. Если результат выражения соответствует метке case, то выполняется оператор, стоящий после этой метки, за которым обязательно должен следовать оператор перехода: break, goto, return и т. д. При использовании оператора break происходит выход из switch и управление передается оператору, следующему за switch. Если же используется оператор goto, то управление передается оператору, помеченному меткой, стоящей после goto. Наконец, оператор return завершает выполнение текущего метода.

Если ни одно выражение case не совпадает со значением оператора switch, управление передается операторам, следующим за необязательной подписью default. Если подписи default нет, то управление передается за пределы оператора switch. Пример использования оператора switch:

```
int dayOfWeek = 5; switch (dayOfWeek)
{
        case 1:
            MessageBox.Show("Понедельник");
        break;      case 2:
            MessageBox.Show("Вторник");
        break;      case 3:
            MessageBox.Show("Среда");
        break;      case 4:
            MessageBox.Show("Четверг");
        break;      case 5:
            MessageBox.Show("Пятница");
        break;      case 6:
```

```
                MessageBox.Show("Суббота");
        break;          case 7:
                MessageBox.Show("Воскресенье");
        break;          default:
                MessageBox.Show("Неверное значение!");
                break;
}
```

Поскольку на момент выполнения оператора switch в этом примере переменная dayOfWeek равнялась 5, то будут выполнены операторы из блока case 5.

В ряде случаев оператор switch можно заменить несколькими операторами if, однако не всегда такая замена будет легче для чтения. Например, приведённый выше пример можно переписать так:

```
int dayOfWeek = 5; if (dayOfWeek == 1)
        MessageBox.Show("Понедельник");
else
        if (dayOfWeek == 2)
        MessageBox.Show("Вторник");     else
        if (dayOfWeek == 3)                     MessageBox.Show("Среда");
                else
                if (dayOfWeek == 4)
    MessageBox.Show("Четверг");
                        else
                        if (dayOfWeek == 5)
    MessageBox.Show("Пятница");
                                else
                                if (dayOfWeek == 6)
    MessageBox.Show("Суббота");
                                        else
                                        if (dayOfWeek == 7)
                MessageBox.Show("Воскресенье");
                                                else
```

MessageBox.Show("Неверное значение!");

Кнопки-переключатели

При создании программ в Visual Studio для организации разветвлений часто используются элементы управления в виде кнопокпереключателей (RadioButton). Состояние такой кнопки (включено – выключено) визуально отражается на форме, а в программе можно узнать его с помощью свойства Checked: если кнопка включена, это свойство будет содержать True, в противном случае False. Если пользователь выбирает один из вариантов переключателя в группе, все остальные автоматически отключаются.

Группируются радиокнопки с помощью какого-либо контейнера – часто это бывает элемент GroupBox. Радиокнопки, размещённые в разных контейнерах, образуют независимые группы.

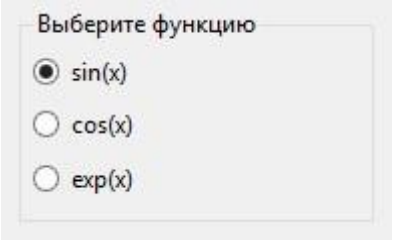

Рис. 3.1. Группа радиокнопок

```
if (radioButton1.Checked)
    MessageBox.Show("Выбрана функция: синус"); else if (radioButton2.Checked)
    MessageBox.Show("Выбрана функция: косинус"); else if (radioButton1.Checked)
        MessageBox.Show("Выбрана функция: экспонента");
```

Пример написания программы

Задание: ввести три числа – x, y, z. Вычислить

$$U = \begin{cases} y \cdot \sin(x) + z, \text{ при } z - x = 0 \\ y \cdot e^{\sin(x)} - z, \text{ при } z - x < 0 \\ y \cdot \sin(\sin(x)) + z, \text{ при } z - x > 0 \end{cases}$$

Создание формы

Создайте форму, в соответствии с рис. 3.2.

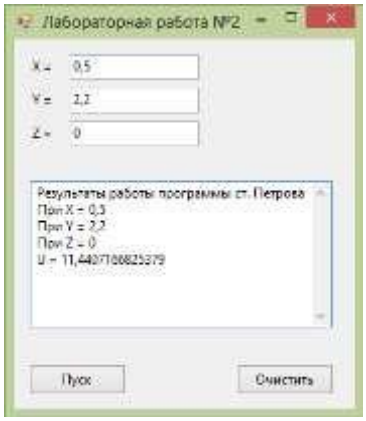

Рис 2. Окно работы

Разместите на форме элементы Label, TextBox и Button. Поле для вывода результатов также является элементом TextBox с установленным в True свойством Multiline и свойством ScrollBars установленным в Both.

Создание обработчиков событий

Обработчики событий создаются аналогично тому, как и в предыдущих лабораторных работах. Текст обработчика события нажатия на кнопку «Пуск» приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
{
        // Получение исходных данных из TextBox         double x =
Convert.ToDouble(textBox2.Text);  double y = Convert.ToDouble(textBox1.Text);       double z
= Convert.ToDouble(textBox3.Text);          // Ввод исходных данных в окно результатов
                textBox4.Text = "Результаты работы программы " +
                "ст. Петрова И.И. " +                    Environment.NewLine;
                textBox4.Text += "При X = " + textBox2.Text +
                                                Environment.NewLine;
                textBox4.Text += "При Y = " + textBox1.Text +
                                                Environment.NewLine;
                textBox4.Text += "При Z = " + textBox3.Text +
                                                Environment.NewLine;
        // Вычисление выражения   double u;        if ((z - x) == 0)
                u = y * Math.Sin(x) * Math.Sin(x) + z;        else              if ((z - x) < 0)
```

```
                u = y * Math.Exp(Math.Sin(x)) - z;
            else
                u = y * Math.Sin(Math.Sin(x)) + z;
        // Вывод результата
        textBox4.Text += "U = " + u.ToString() +
            Environment.NewLine;
}
```

Запустите программу и убедитесь в том, что все ветви алгоритма выполняются правильно.

Индивидуальные задания По указанию преподавателя выберите индивидуальное задание из нижеприведенного списка. В качестве *f(x)* использовать по выбору: *sh(x)*, x^2, e^x. Отредактируйте вид формы и текст программы, в соответствии с полученным заданием.

Усложнённый вариант задания для продвинутых студентов: с помощью радиокнопок (RadioButton) дать пользователю возможность во время работы программы выбрать одну из трёх приведённых выше функций.

1. $$a=\begin{cases}(f(x)+y)^2-\sqrt{f(x)y}, & xy\rangle 0\\ (f(x)+y)^2+\sqrt{|\mathrm{f(x)\ y}|}, & xy\langle 0\\ (f(x)+y)^2+1, & xy=0.\end{cases}$$

2. $$b=\begin{cases}\ln(f(x))+(f(x)^2+y)^3, & x/y\rangle 0\\ \ln|f(x)/y|+(f(x)+y)^3, & x/y\langle 0\\ (f(x)^2+y)^3, & x=0\\ 0, & y=0.\end{cases}$$

3. $$c=\begin{cases}f(x)^2+y^2+\sin(y), & x-y=0\\ (f(x)-y)^2+\cos(y), & x-y\rangle 0\\ (y-f(x))^2+tg(y), & x-y\langle 0.\end{cases}$$

4. $$d=\begin{cases}(f(x)-y)^3+arctg(f(x)), & x\rangle y\\ (y-f(x))^3+arctg(f(x)), & y\rangle x\\ (y+f(x))^3+0.5, & y=x.\end{cases}$$

5. $$e=\begin{cases}i\sqrt{f(x)}, & i-\text{нечетное},\ x\rangle 0\\ i/2\sqrt{|f(x)|}, & i-\text{четное},\ x\langle 0\\ \sqrt{|if(x)|}, & \text{иначе}.\end{cases}$$

6. $$g=\begin{cases}e^{f(x)-|b|}, & 0.5\langle xb\langle 10\\ \sqrt{|f(x)+b|}, & 0.1\langle xb\langle 0.5\\ 2f(x)^2, & \text{иначе}.\end{cases}$$

7. $$s=\begin{cases}e^{f(x)}, & 1\langle xb\langle 10\\ \sqrt{|f(x)+4*b|}, & 12\langle xb\langle 40\\ bf(x)^2, & \text{иначе}.\end{cases}$$

8. $$j=\begin{cases}\sin(5f(x)+3m|f(x)|), & -1\langle m\langle x\\ \cos(3f(x)+5m|f(x)|), & x\rangle m\\ (f(x)+m)^2, & x=m.\end{cases}$$

9. $$l=\begin{cases}2f(x)^3+3p^2, & x\rangle|p|\\ |f(x)-p|, & 3\langle x\langle|p|\\ (f(x)-p)^2, & x=|p|.\end{cases}$$

10. $$k=\begin{cases}\ln(|f(x)|+|q|), & |xq|\rangle 10\\ e^{f(x)+q}, & |xq|\langle 10\\ f(x)+q, & |xq|=10\end{cases}$$

11. $$m=\frac{\max(f(x),y,z)}{\min(f(x),y)}+5.$$

12. $$n=\frac{\min(f(x)+y,y-z)}{\max(f(x),y)}.$$

13. $$p=\frac{|\min(f(x),y)-\max(y,z)|}{2}.$$

14. $$q=\frac{\max(f(x)+y+z,xyz)}{\min(f(x)+y+z,xyz)}.$$

15. $$r=\max(\min(f(x),y),z).$$

ПРАКТИЧЕСКАЯ РАБОТА 6. РАЗРАБОТКА ОКОННОГО ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

Цель работы: изучить простейшие средства отладки программ в среде Visual Studio. Составить и отладить программу циклического алгоритма.

Краткие теоретические сведения

Операторы организации циклов

Под *циклом* понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной форме.

К операторам цикла относятся: *цикл с предусловием* while, *цикл с постусловием* do while, *цикл с параметром* for и *цикл перебора* foreach. Рассмотрим некоторые из них.

Цикл с предусловием

Оператор цикла while организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Формат цикла while:

while (B) S;

где B – выражение, истинность которого проверяется (условие завершения цикла); S – тело цикла – оператор (простой или составной).

Перед каждым выполнением тела цикла анализируется значение выражения B: если оно истинно, то выполняется тело цикла, и управление передается на повторную проверку условия B; если значение B ложно – цикл завершается и управление передается на оператор,

следующий за оператором S.

Если результат выражения B окажется ложным при первой проверке, то тело цикла не выполнится ни разу. Отметим, что если условие B во время работы цикла не будет изменяться, то возможна ситуация зацикливания, то есть невозможность выхода из цикла. Поэтому внутри тела должны находиться операторы, приводящие к изменению значения выражения B так, чтобы цикл мог корректно завершиться.

В качестве иллюстрации выполнения цикла while рассмотрим программу вывода целых чисел от 1 до n по нажатию кнопки на форме:

```
private void button1_Click(object sender, EventArgs e) {
        int n = 10; // Количество повторений цикла          int i = 1;  // Начальное значение
                while (i <= n) // Пока i меньше или равно n
                {
                MessageBox.Show(i.ToString()); // Показываем i                    i++;    //
Увеличиваем i на 1
                }
}
```

Цикл с постусловием

Оператор цикла do while также организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Однако в отличие от цикла while условие завершения цикла проверяется после выполнения тела цикла. Формат цикла do while:

```
do S while (B);
```

где B – выражение, истинность которого проверяется (условие за-

вершения цикла); S – тело цикла – оператор (простой или блок).

Сначала выполняется оператор S, а затем анализируется значение выражения B: если оно истинно, то управление передается оператору **S**, если ложно – цикл завершается, и управление передается на оператор, следующий за условием B. Так как условие B проверяется после выполнения тела цикла, то в любом случае тело цикла выполнится хотя бы один раз.

В операторе do while, так же как и в операторе while, возможна ситуация зацикливания в случае, если условие B всегда будет оставаться истинным.

Цикл с параметром

Цикл с параметром имеет следующую структуру:

for (<инициализация>; <выражение>; <модификация>)

<оператор>;

Инициализация используется для объявления и/или присвоения начальных значений величинам, используемым в цикле в качестве параметров (счетчиков). В этой части можно записать несколько операторов, разделенных запятой. Областью действия переменных, объявленных в части инициализации цикла, является цикл и вложенные блоки.

Инициализация выполняется один раз в начале исполнения цикла.

Выражение определяет условие выполнения цикла: если его результат истинен, цикл выполняется. Истинность выражения проверяется перед каждым выполнением тела цикла, таким образом, цикл с параметром реализован как *цикл с предусловием*. В блоке выражение через запятую можно записать несколько логических выражений, тогда запятая равносильна операции логическое И (&&).

Модификация выполняется после каждой итерации цикла и служит обычно для изменения параметров цикла. В части модификация можно записать несколько операторов через запятую.

Оператор (простой или составной) представляет собой тело цикла.

Любая из частей оператора for (инициализация, выражение, модификация, оператор) может отсутствовать, но точку с запятой, определяющую позицию пропускаемой части, надо оставить.

Пример формирования строки состоящей из чисел от 0 до 9 разделенных пробелами:

string s = ""; // Инициализация строки

for (int i = 0; i <= 9; i++) // Перечисление всех чисел s += i.ToString() + " "; // Добавляем число и пробел MessageBox.Show(s.ToString()); // Показываем результат

Данный пример работает следующим образом. Сначала вычисляется начальное значение переменной i. Затем, пока значение i меньше или равно 9, выполняется тело цикла и затем повторно вычисляется значение i. Когда значение i становится больше 9, условие становится ложным и управление передается за пределы цикла.

Средства отладки программ

Практически в каждой вновь написанной программе после запуска обнаруживаются ошибки.

Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические, синтаксические). При обнаружении ошибок компилятор формирует список, который отображается по завершению компиляции (рис. 4.1). При этом возможен только запуск программы, которая была успешно скомпилирована для предыдущей версии программы.

При выявлении ошибок компиляции в нижней части экрана появляется текстовое окно (рис 4.1), содержащее сведения обо всех ошибках, найденных в проекте. Каждая строка этого окна содержит имя файла, в котором найдена ошибка, номер строки с ошибкой и характер ошибки. Для быстрого перехода к интересующей ошибке необходимо дважды щелкнуть на строке с ее описанием. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому следует исправлять ошибки последовательно, сверху вниз и, после исправления каждой ошибки компилировать программу снова.

Рис. 4.1. Окно со списком ошибок компиляции

Ошибки второго уровня (ошибки выполнения) связаны с ошибками выбранного алгоритма решения или с неправильной программной реализацией алгоритма. Эти ошибки проявляются в том, что результат расчета оказывается неверным либо происходит переполнение, деление на ноль и др. Поэтому перед использованием отлаженной программы ее надо протестировать, т. е. сделать просчеты при таких комбинациях исходных данных, для которых заранее известен результат. Если тестовые расчеты указывают на ошибку, то для ее поиска следует использовать встроенные средства отладки среды программирования.

В простейшем случае для локализации места ошибки рекомендуется поступать следующим образом. В окне редактирования текста установить точку останова перед подозрительным участком, которая позволит остановить выполнение программы и далее более детально следить за ходом работы операторов и изменением значений переменных. Для этого достаточно в окне редактирования кода щелкнуть слева от нужной строки. В результате чего данная строка будет выделена красным (рис. 4.2).

Рис. 4.2. Фрагмент кода с точкой останова

При выполнении программы и достижения установленной точки, программа будет остановлена, и далее можно будет выполнять код по шагам с помощью команд *Отладка → Шаг с обходом* (без захода в методы) или *Отладка → Шаг с заходом* (с заходом в методы) (рис 4.3).

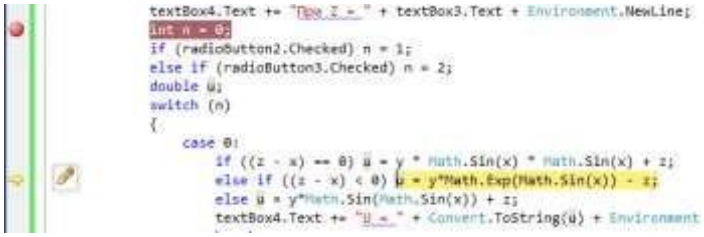

Рис. 4.3. Отладка программы

Желтым цветом выделяется оператор, который будет выполнен. Значение переменных во время выполнения можно увидеть, наведя на них курсор. Для прекращения отладки и остановки программы нужно выполнить команду меню *Отладка → Остановить отладку*.

Для поиска алгоритмических ошибок можно контролировать значения промежуточных переменных на каждом шаге выполнения подозрительного кода и сравнивать их с результатами, полученными вручную.

Порядок выполнения задания

Задание: Вычислить и вывести на экран таблицу значений функции $y=a \cdot ln(x)$ при x, изменяющемся от x_0 до x_k с шагом dx, a – константа.

Панель диалога представлена на рис 4.4. Текст обработчика нажатия кнопки **Вычислить** приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
{
        // Считывание начальных данных          double x0 =
Convert.ToDouble(textBox1.Text);  double xk = Convert.ToDouble(textBox2.Text);     double
dx = Convert.ToDouble(textBox3.Text);     double a = Convert.ToDouble(textBox4.Text);

                textBox5.Text = "Работу выполнил ст. Иванов М.А." +
```

```
                Environment.NewLine;
            // Цикл для табулирования функции
            double x = x0;
            while (x <= (xk + dx / 2))
            {
            double y = a * Math.Log(x);                textBox5.Text += "x=" +
Convert.ToString(x) +
                                                "; y=" + Convert.ToString(y) +
                 Environment.NewLine;
                x = x + dx;
            }
}
```

После отладки программы следует проверить правильность работы программы с помощью контрольного примера, приведенного на рис. 4.4. Установите точку останова на оператор перед циклом и запустите программу. После попадания на точку остановки, выполните пошагово программу и проследите, как меняются все переменные в процессе выполнения.

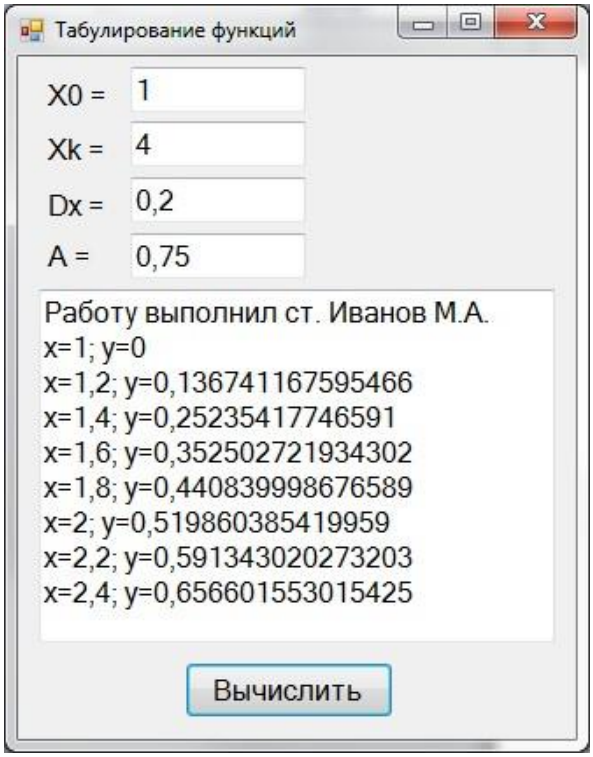

Рис. 4.4. Окно программы для табулирования функции

Индивидуальные задания

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

21) $y = x^{b^b} + \cos(x^{3/2} + b^{3/4})$,
$x_0 = 13.7; x_k = 19.1; dx = 0.4;$
$b = 2;$

22) $y = 10^{-2}(a + bx) - e^{x^3 + b}$,
$x_0 = -3.4; x_k = -1.4; dx = 0.1;$
$a = 5; b = 4;$

23) $y = ax^3 + b^{5/4}xe^{-x}$,
$x_0 = 2.51; x_k = 10.59; dx = 1.01;$
$a = 4; b = 2;$

24) $y = a|x|^{5/2} + \cos(\sqrt{e^x})$,
$x_0 = -0.31; x_k = 0.61; dx = 0.3;$
$a = 8;$

25) $y = 3.1\sqrt{ax^2} - |a + b|x,$

$x_0 = -2.35; x_k = -5.55; dx = -0.05;$

$a = 2; b = -5;$

ПРАКТИЧЕСКАЯ РАБОТА 7. РАЗРАБОТКА ОКОННОГО ПРИЛОЖЕНИЯ ДЛЯ ОБРАБОТКИ СТРОК

Цель работы: изучить правила работы с элементом управления ListBox. Написать программу для работы со строками.

Краткие теоретические сведения

Строковый тип данных

Для хранения строк в языке C# используется тип string. Чтобы объявить (и, как правило, сразу инициализировать) строковую переменную, можно написать следующий код:

```
string a = "Текст";
string b = "строки";
```

Над строками можно выполнять операцию сложения – в этом случае текст одной строки будет добавлен к тексту другой:

```
string c = a + " " + b; // Результат: Текст строки
```

Тип string на самом деле является псевдонимом для класса String, с помощью которого над строками можно выполнять ряд более сложных операций. Например, метод IndexOf может осуществлять поиск подстроки в строке, а метод Substring возвращает часть строки указанной длины, начиная с указанной позиции:

```
string a = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int index = a.IndexOf("OP"); // Результат: 14 (счёт с 0) string b = a.Substring(3, 5); // Результат: DEFGH
```

Если требуется добавить в строку специальные символы, это можно сделать с помощью escape-последовательностей, начинающихся с обратного слэша:

\" Кавычка

\\ Обратная косая черта

\n Новая строка

\r Возврат каретки

\t Горизонтальная табуляция

Более эффективная работа со строками

Строки типа string представляют собой неизменяемые объекты:

после того, как строка инициализирована, изменить её уже нельзя. Рассмотрим для примера следующий код:

```
string s = "Hello, "; s += "world!";
```

Здесь компилятор создаёт в памяти строковый объект и инициализирует его строкой «*Hello,* », а затем создаёт другой строковый объект и инициализирует его значением первого объекта и новой строкой «*world!*», а затем заменяет значение переменной s на новый объект. В результате строка s содержит именно то, что хотел программист, однако в памяти остаётся и изначальный объект со строкой «*Hello,* ». Конечно, со временем сборщик мусора уничтожит этот бесхозный объект, однако если в программе идёт интенсивная работа со строками, то таких бесхозных объектов может оказаться очень много. Как правило, это негативно сказывается на производительности программы и объеме потребляемой ею памяти.

Чтобы компилятор не создавал каждый раз новый строковый объект, разработчики языка C# ввели другой строковый класс: StringBuilder. Приведённый выше пример с использованием этого класса будет выглядеть следующим образом:

```
StringBuilder s = new StringBuilder("Hello, ");
s.Append("world!");
```

Конечно, визуально этот код выглядит более сложным, зато при активном использовании строк в программе он будет гораздо эффективнее. Помимо добавления строки к существующему объекту (Append) класс StringBuilder имеет ещё ряд полезных методов:

Insert Вставляет указанный текст в нужную позицию исходной строки

Remove Удаляет часть строки

Replace Заменяет указанный текст в строке на другой

Если нужно преобразовать объект StringBuilder в обычную строку, то для этого можно использовать метод ToString():

```
StringBuilder s = new StringBuilder("Яблоко"); string a = s.ToString();
```

Элемент управления ListBox

Элемент управления ListBox представляет собой список, элементы которого выбираются при помощи клавиатуры или мыши. Список элементов задается свойством Items. Items – это элемент, который имеет свои свойства и свои методы. Методы Add, RemoveAt и Insert используются для добавления, удаления и вставки элементов.

Объект Items хранит объекты, находящиеся в списке. Объект может быть любым классом – данные класса преобразуются для отображения в строковое представление методом

ToString(). В нашем случае в качестве объекта будут выступать строки. Однако, поскольку объект Items хранит объекты, *приведённые* к типу object, перед использованием необходимо *привести* их обратно к изначальному типу, в нашем случае string:

```
string a = (string)listBox1.Items[0];
```

Для определения номера выделенного элемента используется свойство SelectedIndex.

Порядок выполнения индивидуального задания

Задание: Написать программу подсчета числа слов в произвольной строке. В качестве разделителя может быть любое число пробелов. Для ввода строк использовать ListBox. Строки вводятся на этапе проектирования формы, используя окно свойств. Вывод результата организовать в метку Label.

Панель диалога будет иметь вид:

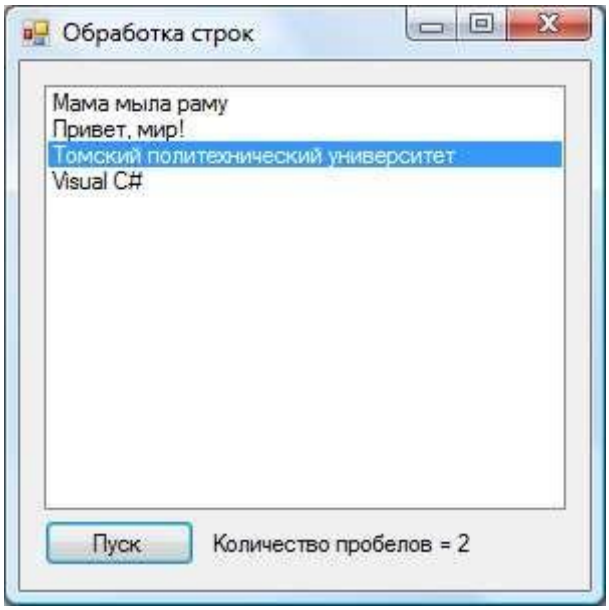

Рис. 6.1. Окно программы обработки строк

Текст обработчика нажатия кнопки «Пуск» приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
{
        // Получаем номер выделенной строки     int index = listBox1.SelectedIndex;  // Считываем строку в переменную str         string str = (string)listBox1.Items[index];    // Узнаем количество символов в строке      int len = str.Length;

                // Считаем, что количество пробелов равно 0

                int count = 0;

                // Устанавливаем счетчик символов в 0
```

```
        int i = 0;
        // Организуем цикл перебора всех символов в строке
        while (i < len)
        {
        // Если нашли пробел, то увеличиваем          // счетчик пробелов на 1
            if (str[i] == ' ')
            count++;              i++;
        }
        label1.Text = "Количество пробелов = " +
            count.ToString();
}
```

Индивидуальные задания

1. Во всех заданиях исходные данные вводить с помощью ListBox. Строки вводятся на этапе проектирования формы, используя окно свойств. Вывод результата организовать в метку Label.

2. Дана строка, состоящая из групп нулей и единиц. Посчитать количество нулей и единиц.
3. Посчитать в строке количество слов.
4. Найти количество знаков препинания в исходной строке.
5. Дана строка символов. Вывести на экран цифры, содержащиеся в строке.
6. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести количество четных чисел в этой строке.
7. Дана строка символов. Вывести на экран количество строчных русских букв, входящих в эту строку.
8. Дана строка символов. Вывести на экран только строчные русские буквы, входящие в эту строку.
9. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. В каждом слове заменить первую букву на прописную.
10. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Удалить первую букву в каждом слове.
11. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами i- и jю буквы. Для ввода i и j на форме добавить свои поля ввода.
12. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Поменять местами первую и последнюю буквы каждого слова.
13. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Заменить все буквы латинского алфавита на знак «+».
14. Дана строка символов, содержащая некоторый текст на русском языке. Заменить все большие буквы «А» на символ «*».
15. Дана строка символов, содержащая некоторый текст. Разработать программу, которая определяет, является ли данный текст палиндромом, т. е. читается ли он

слева направо так же, как и справа налево (например, «А роза упала на лапу Азора»).

16. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Сформировать новую строку, состоящую из чисел длин слов в исходной строке.

ПРАКТИЧЕСКАЯ РАБОТА 8. РАЗРАБОТКА ОКОННОГО ПРИЛОЖЕНИЯ ДЛЯ ОБРАБОТКИ ОДНОМЕРНЫХ МАССИВОВ

Цель работы: Изучить способы получения случайных чисел. Написать программу для работы с одномерными массивами.

Краткие теоретические сведения

Работа с массивами

Массив – набор элементов одного и того же типа, объединенных общим именем. Массивы в C# можно использовать по аналогии с тем, как они используются в других языках программирования. Однако C#массивы имеют существенные отличия: они относятся к ссылочным типам данных, более того – реализованы как объекты. Фактически имя массива является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе инициализации массива. А за освобождением памяти следит система сборки мусора – неиспользуемые массивы автоматически утилизируются данной системой.

Рассмотрим в данной работе одномерные массивы. *Одномерный массив* – это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в C# начинается с нуля, то есть, если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одномерный массив в C# реализуется как объект, поэтому его создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив, затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяется размер массива.

В общем случае процесс объявления переменной типа массив, и выделение необходимого объема памяти может быть разделено. Кроме того на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

```
тип[] имя_массива;
```

В этом случае описывается ссылка на одномерный массив, которая в дальнейшем может быть использована для адресации на уже существующий массив. Размер массива при таком объявлении не задаётся. Пример, в котором объявляется массив целых чисел с именем $_a$:

```
int[] a;
```

Другая форма объявления массива включает и его инициализацию указанным количеством элементов:

```
тип[] имя_массива = new тип[размер];
```

В этом случае объявляется одномерный массив указанного типа и выделяется память под указанное количество элементов. Адрес данной области памяти записывается в ссылочную переменную. Элементы массива инициализируются значениями, которые по умолчанию приняты для данного типа: массивы числовых типов инициализируются нулями, строковые переменные – пустыми строками, символы – пробелами, объекты ссылочных типов – значением null. Пример такого объявления:

```
int[] a = new int[10];
```

Здесь выделяется память под 10 элементов типа $_{int}$.

Наконец, третья форма записи даёт возможность сразу инициализировать массив конкретными значениями:

```
тип[] имя_массива = {список инициализации};
```

При такой записи выделяется память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации. Пример:

```
int[] a = { 0, 1, 2, 3 };
```

В данном случае будет создан массив a, состоящий из четырёх элементов, и каждый элемент будет инициализирован очередным значением из списка.

Обращение к элементам массива происходит с помощью индекса: для этого нужно указать имя массива и в квадратных скобках его номер. Например: a[0], b[10], c[i]. Следует помнить, что нумерация элементов начинается с нуля!

Так как массив представляет собой набор элементов, объединенных общим именем, то

обработка массива обычно производится в цикле. Например:

```
int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

for (int i = 0; i < 10; i++)        MessageBox.Show(myArray[i]);
```

Случайные числа

Одним из способов инициализации массива является задание определение элементов через случайные числа. Для работы со случайными числами используются класс Random. Метод Random.Next создает случайное число в диапазоне значений от нуля до максимального значения типа int (его можно узнать с помощью свойства Int32.MaxValue). Для создания случайного числа в диапазоне от нуля до какого-либо другого положительного числа используется перегрузка метода

Random.Next(Int32) – единственный параметр метода указывает верхнюю границу диапазона, сама граница в диапазон не включается. Для создания случайного числа в другом диапазоне используется перегрузка метода Random.Next(Int32, Int32) – первый аргумент задаёт нижнюю границу диапазона, а второй – верхнюю.

Порядок выполнения индивидуального задания

Создайте форму с элементами управления как приведено на рис. 7.1. Опишите одномерный массив. Создайте обработчики события для кнопок (код приведен ниже). Данная программа заменяет все отрицательные числа нулями. Протестируйте правильность выполнения программы. Модифицируйте программу в соответствии с индивидуальным заданием.

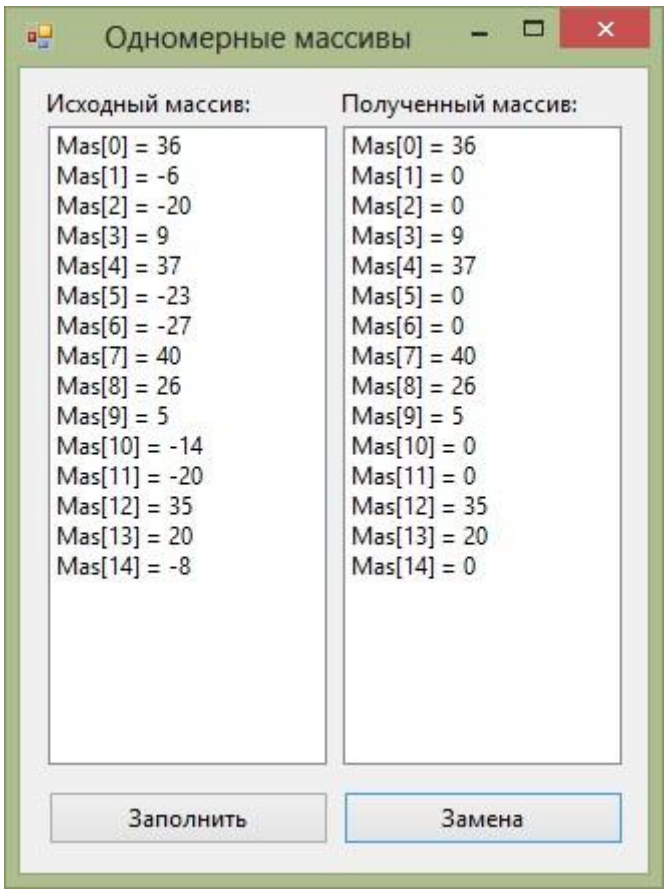

Рис. 7.1. Окно программы для работы с одномерными массивами

```
// Глобальная переменная видна всем методам
int[] Mas = new int[15];

// Заполнение исходного массива
private void button1_Click(object sender, EventArgs e) {
        // Очищаем элемент управления    listBox1.Items.Clear();
                // Инициализируем класс случайных чисел
        Random rand = new Random();        // Генерируем и выводим 15 элементов
                for (int i = 0; i < 15; i++)
                {
                        Mas[i] = rand.Next(-50, 50);
                        listBox1.Items.Add("Mas[" + i.ToString() +
                                "] = " + Mas[i].ToString());
                }
```

```
}


// Замена отрицательных элементов нулями
private void button2_Click(object sender, EventArgs e)
{
        // Очищаем элемент управления    listBox2.Items.Clear();          // Обрабатываем все элементы       for (int i = 0; i < 15; i++)
                {
                if (Mas[i] < 0)                                 Mas[i] = 0;
                        listBox2.Items.Add("Mas[" + Convert.ToString(i)
                                + "] = " + Mas[i].ToString());
                }
}
```

Индивидуальные задания

1. В массиве из 20 целых чисел найти наибольший элемент и поменять его местами с первым элементом.
2. В массиве из 10 целых чисел найти наименьший элемент и поменять его местами с последним элементом.
3. В массиве из 15 вещественных чисел найти наибольший элемент и поменять его местами с последним элементом.
4. В массиве из 25 вещественных чисел найти наименьший элемент и поменять его местами с первым элементом.
5. Дан массив X, содержащий 27 элементов. Вычислить и вывести элементы нового массива Y, где $y_i = 6.85x_i^2 - 1.52$. Если $y_i<0$, то вычислить и вывести $a=x_i^3-0.62$ и продолжить вычисления; если y_i 0, то вычислить и вывести $b=1/x_i^2$ и продолжить вычисления.
6. Дан массив X, содержащий 16 элементов. Вычислить и
7. $$d_i = \frac{e^{x_i} + 2e^{-x_i}}{\sqrt{5 + \sin x_i}}$$
8. и значения $d_i>0.1$. вывести значения d_i, где
9. Дан массив Y, содержащий 25 элементов. Записать в массив R и вывести значения элементов, вычисляемые по формуле
10. $$r_i = \frac{5y_i + \cos^2 y_i}{2.35}$$, i=1,2,…,25.
11. Дан массив F, содержащий 18 элементов. Вычислить и вывести элементы нового массива $p_i=0.13f_i^3-2.5f_i+8$. Вывести отрицательные элементы массива P.
12. Вычислить и вывести элементы массива Z, где $z_i=i^2+1$, если i – нечетное, и $z_i=2i-1$, если i – четное. Сформировать и вывести массив D: $d_i=2.5z_i$, если $z_i>=2.5$ и $d_i=z_i/2.5$, если z_i 2.5.

13. Заданы массивы D и E. Вычислить и вывести значения $f_i=(2d_i+\sin e_i)/d_i$, где i=1,2,…,16; вывести $1<f_i<3$.
14. В массиве R, содержащем 25 элементов, заменить значения отрицательных элементов квадратами значений, значения положительных увеличить на 7, а нулевые значения оставить без изменения. Вывести массив R.
15. Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые кратны 5.
16. Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые нечетны и отрицательны.
17. Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести сумму тех элементов, которые удовлетворяют условию $\square a_i \square \square i^2$.
18. Дан массив A целых чисел, содержащий 30 элементов. Вычислить и вывести количество и сумму тех элементов, которые делятся на 5 и не делятся на 7.
19. Дан массив A вещественных чисел, содержащий 25 элементов. Вычислить и вывести число отрицательных элементов и число членов, принадлежащих отрезку[1,2].
20. Дан массив C, содержащий 23 элемента. Вычислить и вывести среднее арифметическое всех значений $c_i>3.5$.
21. Дан массив Z целых чисел, содержащий 35 элементов. Вычислить и вывести R=S+P, где S – сумма четных элементов, меньших 3, P – произведение нечетных элементов, больших 1.
22. Дан массив Q натуральных чисел, содержащий 20 элементов. Найти и вывести те элементы, которые при делении на 7 дают остаток 1,2 или 5.
23. Дан массив Q натуральных чисел, содержащий 20 элементов. Найти и вывести те элементы, которые обладают тем свойством, что корни уравнения $q_i^2+3q_i-5=0$ действительны и положительны.
24. Дан массив, содержащий 10 элементов. Вычислить произведение элементов, стоящих после первого отрицательного элемента. Вывести исходный массив и результат вычислений.
25. Дан массив, содержащий 14 элементов. Вычислить сумму элементов, стоящих до первого отрицательного элемента. Вывести исходный массив и результат вычислений.
26. Дан массив содержащий 12 элементов. Все четные элементы сложить, вывести массив и результат.
27. Дан массив, содержащий 15 элементов. Все положительные элементы возвести в квадрат, а отрицательные умножить на 2. Вывести исходный и полученный массив.
28. Дан массив, содержащий 14 элементов. Все отрицательные элементы заменить на 3. Вывести исходный и полученный массив.
29. Дан массив из 15 целых чисел. Найти количество нечетных положительных элементов.
30. Массив задан датчиком случайных чисел на интервале [-33, 66]. Найти наименьший нечетный элемент.
31. Разработать программу, выводящую количество максимальных элементов в массиве из пятидесяти целочисленных элементов.
32. Разработать программу, циклически сдвигающую элементы целочисленного массива влево. Нулевой элемент массива ставится на последнее место, остальные элементы сдвигаются влево на одну позицию. Запрещается использовать второй массив.
33. Дано два массива с неубывающими целыми числами. Напишите программу нахождения элементов содержащихся в каждом массиве.
34. Дано два массива с неубывающими целыми числами. Напишите программу, формирующую новый массив их элементов первых двух. В результирующем массиве не должно быть одинаковых элементов.

35. Дан массив целых чисел из 30 элементов. Найдите все локальные максимумы. (Элемент является локальным максимумом, если он не имеет соседей, больших, чем он сам).

ПРАКТИЧЕСКАЯ РАБОТА 9. РАЗРАБОТКА ОКОННОГО ПРИЛОЖЕНИЯ ДЛЯ ОБРАБОТКИ ДВУМЕРНЫХ МАССИВОВ

Цель работы: изучить свойства элемента управления DataGridView. Написать программу с использованием двухмерных массивов.

Краткие теоретические сведения

Двухмерные массивы

Многомерные массивы имеют более одного измерения. Чаще всего используются двумерные массивы, которые представляют собой таблицы. Каждый элемент такого массива имеет два индекса, первый определяет номер строки, второй – номер столбца, на пересечении которых находится элемент. Нумерация строк и столбцов начинается с нуля. Объявить двумерный массив можно одним из предложенных способов:

```
тип[,] имя_массива;
```

```
тип[,] имя_массива = new тип[размер1, размер2];
```

```
тип[,] имя_массива =
        {{элементы 1-ой строки},
        ...,
        {элементы n-ой строки}};
```

```
тип[,] имя_массива = new тип[,]
        {{элементы 1-ой строки},
        ...,
        {элементы n-ой строки}};
```

В качестве примера рассмотрим код, который строит «таблицу умножения» – каждая ячейка будет содержать значение, равное произведению номера строки и номера столбца:

```
// Объявление двухмерного массива int[,] mul = new int[10,10]; // Заполнение массива for (int i = 0; i < 10; i++)  for (int j = 0; j < 10; j++)  mul[i, j] = i * j;
```

Элемент управления DataGridView

При работе с двухмерными массивами ввод и вывод информации на экран удобно организовывать в виде таблиц. Элемент управления DataGridView может быть использован для отображения информации в виде двумерной таблицы. Для обращения к ячейке в этом элементе необходимо указать номер строки и номер столбца. Например:

```
dataGridView1.Rows[2].Cells[7].Value = "*";
```

Этот код запишет во вторую строку и седьмой столбец знак звёздочки.

Порядок выполнения задания

В ходе выполнения задания нужно создать программу для определения целочисленной матрицы 15×15. Разработать обработчик кнопки, который будет искать максимальный элемент на дополнительной диагонали матрицы. Результат вывести в текстовое поле. Окно программы приведено на рис. 8.1.

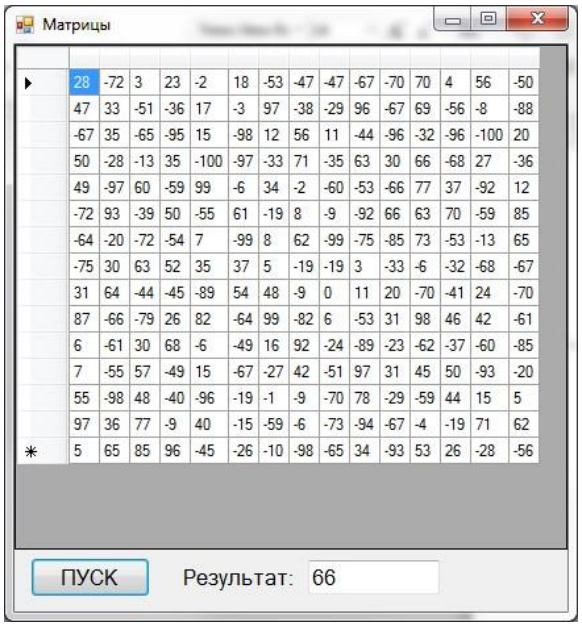

Рис. 8.1. Окно программы для работы с двухмерным массивом

Текст обработчика события нажатия на кнопку приведен ниже.

```
private void button1_Click(object sender, EventArgs e)
{       dataGridView1.RowCount = 15; // Кол-во строк    dataGridView1.ColumnCount = 15; // Кол-во столбцов
        int[,] a = new int[15,15]; // Инициализируем массив       int i,j;
        //Заполняем матрицу случайными числами        Random rand = new Random();
        for (i = 0; i < 15; i++)                for (j = 0; j < 15; j++)                        a[i,j] =
```

```
rand.Next(-100, 100);          // Выводим матрицу в dataGridView1       for (i = 0; i < 15; i++)
                for (j = 0; j < 15; j++)

                        dataGridView1.Rows[i].Cells[j].Value =                          a[i,
j].ToString();

        // Поиск максимального элемента          // на дополнительной диагонали    int m =
int.MinValue;          for (i = 0; i < 15; i++)

                        if (a[i, 14 – i] > m) m = a[i, 14 – i];

                // выводим результат

        textBox1.Text = Convert.ToString(m); }
```

Индивидуальные задания

1. Дана матрица A(3,4). Найти наименьший элемент в каждой строке матрицы. Вывести исходную матрицу и результаты вычислений.
2. Дана матрица A(3,3). Вычислить сумму второй строки и произведение первого столбца. Вывести исходную матрицу и результаты вычислений.
3. Дана матрица A(4,4). Найти наибольший элемент в главной диагонали. Вывести матрицу и наибольший элемент.
4. Дана матрица A(3,4). Найти сумму элементов главной диагонали и эту сумму поставить на место последнего элемента. Вывести исходную и полученную матрицу.
5. Дана матрица A(4,3). Вычислить наибольший элемент матрицы. Вывести исходную матрицу и наибольший элемент.
6. Дана матрица A(4,3).Найти количество положительных элементов.
7. Дана матрица A(3,4).Найти количество отрицательных элементов.
8. Даны матрицы X(15,15) и Y(15,15). Вычислить и вывести элементы новой матрицы $z_{ij}=12x_{ij}-0.85y_{ij}^2$.
9. Даны матрицы A(6,6),B(6,6) и C(6,6). Получить матрицу D(6,6), элементы которой вычисляются по формуле $d_{ij}=\max\square a_{ij},(b_{ij}+c_{ij})\square$. Матрицу D(6,6) вывести.
10. Вычислить сумму S элементов главной диагонали матрицы B(10,10). Если S>10, то исходную матрицу преобразовать по формуле $b_{ij}=b_{ij}+13.5$; если S 10, то $b_{ij}=b_{ij}^2-1.5$. Вывести сумму S и преобразованную матрицу.
11. Дана матрица F(15,15). Вывести номер и среднее арифметическое элементов строки, начинающейся с 1. Если такой строки нет, то вывести сообщение «*Строки нет*».
12. Дана матрица F(7,7). Найти наименьший элемент в каждом столбце. Вывести матрицу и найденные элементы.
13. Найти наибольший элемент главной диагонали матрицы A(15,15) и вывести всю строку, в которой он находится.
14. Найти наибольшие элементы каждой строки матрицы Z(16,16) и поместить их на главную диагональ. Вывести полученную матрицу.
15. Вычислить суммы элементов матрицы Y(12,12) по столбцам и вывести их.
16. Найти наибольший элемент матрицы A(10,10) и записать нули в ту строку и столбец, где он находится. Вывести наибольший элемент, исходную и полученную матрицу.
17. Дана матрица R(9,9). Найти наименьший элемент в каждой строке и записать его на место первого элемента строки. Вывести исходную и полученную матрицы.
18. Определить количество положительных элементов каждой строки матрицы A(10,20) и запомнить их в одномерном массиве N. Массив N вывести.

19. Вычислить количество H положительных элементов последнего столбца матрицы X(5,5). Если H<3, то вывести все положительные элементы матрицы, если H 3, то вывести сумму элементов главной диагонали матрицы.
20. Вычислить и вывести сумму элементов матрицы A(12,12), расположенных над главной диагональю матрицы.
21. Двумерный массив 30x20 заполнить случайными символами английского алфавита (заглавными буквами). Вывести на экран сколько раз встречается каждый символ.
22. Найти номер столбца матрицы, в котором находится наименьшее количество положительных элементов.
23. Дан двухмерный массив 20x20 целочисленных элементов. Найдите все локальные максимумы. (Элемент является локальным максимумом, если он не имеет соседей, больших, чем он сам).
24. Дана матрица 7x7. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.
25. Задана матрица, содержащая N строк и M столбцов. Седловой точкой этой матрицы назовем элемент, который одновременно является минимумом в своей строке и максимумом в своем столбце. Найдите количество седловых точек заданной матрицы.
26. Требуется совершить обход квадратной матрицы по спирали так, как показано на рисунке: заполнение происходит с единицы из левого верхнего угла и заканчивается в центре числом N^2, где N – порядок матрицы. Реализуйте программу для матрицы 10x10.

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

27.
28. Требуется заполнить змейкой квадратную матрицу так, как показано на рисунке: заполнение происходит с единицы из левого верхнего угла и заканчивается в правом нижнем числом N^2, где N – порядок матрицы. Реализуйте программу для матрицы 10x10.

1	3	4	10
2	5	9	11
6	8	12	15
7	13	14	16

29. Дана шахматная доска (матрица 8x8). Разработать программу, показывающую последовательность ходов конем с произвольной клетки. Конь ходит в соответствии с шахматными правилами, но в произвольную сторону (сгенерировать случайным образом). В клетку, с которой начинается ход, выводится единица. В клетку, в которую идет далее конь, записывается двойка и т. д. Ходить конем на клетки, на которых уже побывал конь, нельзя. Алгоритм останавливает работу, когда конем ходить некуда. Максимальная последовательность ходов – 64.
30. Дана квадратная матрица 10x10. Реализуйте программу для транспонирования матрицы по главной и побочной диагонали.
31. Реализуйте игру сапер с произвольным числом мин расположенных в случайных местах.
32. Реализуйте игру морской бой с n подводными лодками (подводная лодка занимает

одну клеточку), расположенных в случайных местах.

33. Проверка на симпатичность. Рассмотрим таблицу, содержащую *n* строк и *m* столбцов, в каждой клетке которой расположен ноль или единица. Назовем такую таблицу симпатичной, если в ней нет ни одного квадрата 2 на 2, заполненного целиком нулями или целиком единицами. Так, например, таблица 4 на 4, расположенная слева, является симпатичной, а расположенная справа таблица 3 на 3 – не является.

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

0	0	1
0	0	1
1	1	1

Является ли симпатичной таблица 7x7 заполненная случайным образом?

ПРАКТИЧЕСКАЯ РАБОТА 10. ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИЙ

Цель работы: изучить возможности построения графиков с помощью элемента управления Chart. Написать и отладить программу построения на экране графика заданной функции.

Краткие теоретические сведения

Как строится график с помощью элемента управления Chart

Обычно результаты расчетов представляются в виде графиков и диаграмм. Библиотека .NET Framework имеет мощный элемент управления Chart для отображения на экране графической информации (рис.

9.1).

Построение графика (диаграммы) производится после вычисления таблицы значений функции *y=f(x)* на интервале [Xmin, Xmax] с заданным шагом. Полученная таблица передается в специальный массив Points объекта Series элемента управления Chart с помощью метода DataBindXY. Элемент управления Chart осуществляет всю работу по отображению графиков: строит и размечает оси, рисует координатную сетку, подписывает название осей и самого графика, отображает переданную таблицу в виде всевозможных графиков или диаграмм. В элементе управления Chart можно настроить толщину, стиль и цвет линий, параметры шрифта подписей, шаги разметки координатной сетки и многое другое. В процессе работы программы изменение параметров возможно через обращение к соответствующим свойствам элемента управления Chart. Так, например, свойство AxisX содержит значение максимального предела нижней оси графика и при его изменении во время работы программы автоматически изменяется изображение графика.

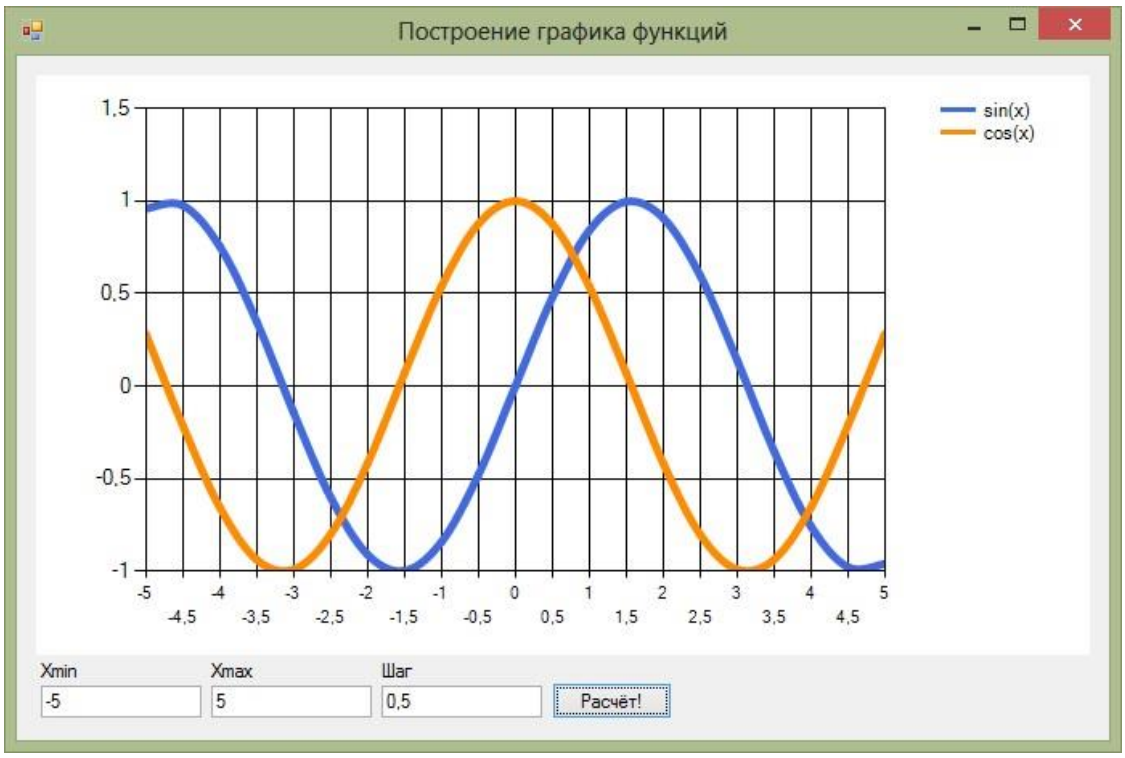

Рис 8.1. Окно программы с элементом управления

Пример написания программы

Задание: составить программу, отображающую графики функций *sin(x)* и *cos(x)* на интервале [Xmin, Xmax]. Предусмотреть возможность изменения разметки координатных осей, а также шага построения таблицы.

Прежде всего, следует поместить на форму сам элемент управления Chart. Он располагается в панели элементов в разделе *Данные*.

Список графиков хранится в свойстве Series, который можно изменить, выбрав соответствующий пункт в окне свойств. Поскольку на одном поле требуется вывести два отдельных графика функций, нужно добавить ещё один элемент. Оба элемента, и существующий и добавленный, нужно соответствующим образом настроить: изменить тип диаграммы ChartType на Spline. Здесь же можно изменить подписи к графикам с абстрактных *Series1* и *Series2* на *sin(x)* и *cos(x)* – за это отвечает свойство Legend. Наконец, с помощью свойства BorderWidth можно сделать линию графика потолще, а затем поменять цвет линии с помощью свойства Color.

Ниже приведён текст обработчика нажатия кнопки «Расчёт!», который выполняет все требуемые настройки и расчёты и отображает графики функций:

```
private void buttonCalc_Click(object sender,
                EventArgs e)
{
```

```
// Считываем с формы требуемые значения double Xmin =
double.Parse(textBoxXmin.Text); double Xmax = double.Parse(textBoxXmax.Text);
double Step = double.Parse(textBoxStep.Text);

// Количество точек графика
int count = (int)Math.Ceiling((Xmax - Xmin) / Step)
+ 1;

// Массив значений X – общий для обоих графиков double[] x = new
double[count];

// Два массива Y – по одному для каждого графика double[] y1 = new
double[count]; double[] y2 = new double[count];

// Расчитываем точки для графиков функции for (int i = 0; i < count; i++)
{
// Вычисляем значение X x[i] = Xmin + Step * i;
// Вычисляем значение функций в точке X
y1[i] = Math.Sin(x[i]); y2[i] = Math.Cos(x[i]);
}

// Настраиваем оси графика chart1.ChartAreas[0].AxisX.Minimum = Xmin;
chart1.ChartAreas[0].AxisX.Maximum = Xmax;

// Определяем шаг сетки
chart1.ChartAreas[0].AxisX.MajorGrid.Interval = Step; // Добавляем вычисленные
значения в графики chart1.Series[0].Points.DataBindXY(x, y1);
chart1.Series[1].Points.DataBindXY(x, y2); }
```

Выполнение индивидуального задания

Постройте графики функций для соответствующих вариантов из работы №2. Таблицу данных получить путём изменения параметра X с шагом h. Самостоятельно выбрать удобные параметры настройки.

ПРАКТИЧЕСКАЯ РАБОТА 11. СОЗДАНИЕ ПРОСТЕЙШИХ ГРАФИЧЕСКИХ ИЗОБРАЖЕНИЙ

Цель работы: изучить возможности Visual Studio по создание простейших графических изображений. Написать и отладить программу построения на экране различных графических примитивов.

Краткие теоретические сведения

Событие Paint

Для форм в C# предусмотрен способ, позволяющий приложению при необходимости перерисовывать окно формы в любой момент времени. Когда вся клиентская область окна формы или часть этой области требует перерисовки, форме передается событие Paint. Все, что требуется от программиста, это создать обработчик данного события (рис.

10.1), наполнив его необходимой функциональностью.

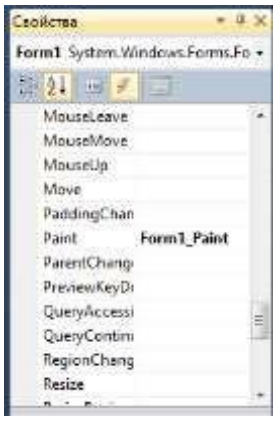

Рис. 10.1. Создание обработчика события Paint

. Объект Graphics для рисования

Для рисования линий и фигур, отображение текста, вывода изображений и т. д. нужно использовать объект Graphics. Этот объект предоставляет поверхность рисования и используется для создания графических изображений. Ниже представлены два этапа работы с графикой.

Создание или получение объекта Graphics

Использование объекта Graphics для рисования

Существует несколько способов создания объектов Graphics. Одним из самых используемых является получение ссылки на объект Graphics через объект PaintEventArgs при обработке события Paint формы или элемента управления:

```
private void Form1_Paint(object sender,
                PaintEventArgs e)
```

```
{
        Graphics g = e.Graphics;
    // Далее вставляется код рисования }
```

Методы и свойства класса Graphics

Имена большого количества методов, определенных в классе Graphics, начинается с префикса Draw* и Fill*. Первые из них предназначены для рисования текста, линий и не закрашенных фигур (таких, например, как прямоугольные рамки), а вторые – для рисования закрашенных геометрических фигур. Ниже рассматривается применение наиболее часто используемых методов, более полную информацию можно найти в документации по Visual Studio.

Метод DrawLine рисует линию, соединяющую две точки с заданными координатами. У метода есть несколько перегруженных версий:

public void DrawLine(Pen, Point, Point); public void DrawLine(Pen, PointF, PointF); public void DrawLine(Pen, int, int, int, int); public void DrawLine(Pen, float, float, float, float);

Первый параметр задает инструмент для рисования линии – перо.

Перья создаются как объекты класса Pen, например:

```
Pen p = new Pen(Brushes.Black, 2);
```

Здесь создаётся черное перо толщиной 2 пиксела. При создании пера можно выбрать его цвет, толщину и тип линии, а также другие атрибуты.

Остальные параметры перегруженных методов DrawLine задают координаты соединяемых точек. Эти координаты могут быть заданы как объекты класса Point и PointF, а также в виде целых чисел и чисел с плавающей десятичной точкой.

В классах Point и PointF определены свойства X и Y, задающие, соответственно, координаты точки по горизонтальной и вертикальной оси. При этом в классе Point эти свойства имеют целочисленные значения, а в классе PointF – значения с плавающей десятичной точкой.

Третий и четвертый вариант метода DrawLine позволяет задавать координаты соединяемых точек в виде двух пар чисел. Первая пара определяет координаты первой точки по горизонтальной и вертикальной оси, а вторая – координаты второй точки по этим же осям. Разница между третьим и четвертым методом заключается в использовании координат различных типов (целочисленных int и с плавающей десятичной точкой float).

Чтобы испытать метод DrawLine в работе, создайте приложение DrawLineApp (аналогично тому, как Вы создавали предыдущее приложение). В этом приложении создайте следующий

обработчик события Paint:

```
private void Form1_Paint(object sender,
        PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        g.Clear(Color.White);
        for (int i = 0; i < 50; i++)
            g.DrawLine(new Pen(Brushes.Black, 2),
                10, 4 * i + 20, 200, 4 * i + 20);
    }
```

Здесь мы вызываем метод DrawLine в цикле, рисуя 50 горизонтальных линий (рис. 10.2).

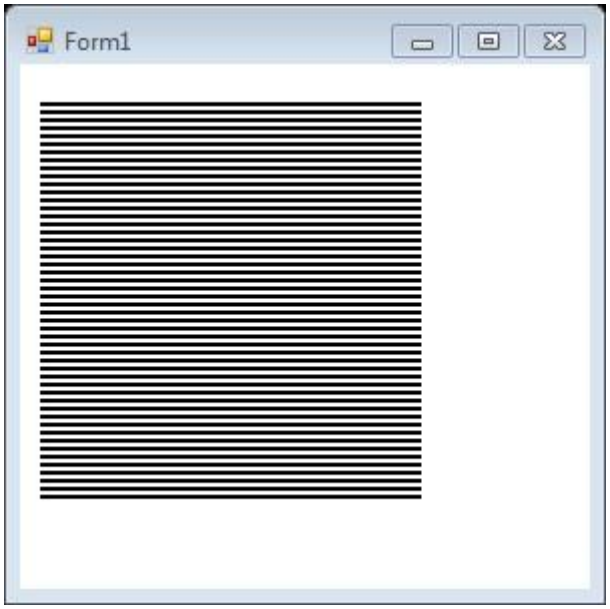

Рис. 10.2. Пример использования DrawLine

Вызвав один раз метод DrawLines, можно нарисовать сразу несколько прямых линий, соединенных между собой. Иными словами, метод DrawLines позволяет соединить между собой несколько точек. Координаты этих точек по горизонтальной и вертикальной оси передаются методу через массив класса Point или PointF:

```
public void DrawLines(Pen, Point[]); public void DrawLines(Pen, PointF[];
```

Для демонстрации возможностей метода DrawLines создайте приложение. Код будет выглядеть следующим образом:

```
Point[] points = new Point[50];
Pen pen = new Pen(Color.Black, 2);

private void Form1_Paint(object sender,
                PaintEventArgs e)
{
                Graphics g = e.Graphics;
                g.DrawLines(pen, points);
}

private void Form1_Load(object sender, EventArgs e)
{
                for (int i = 0; i < 20; i++)
                {
                int xPos;                   if (i % 2 == 0)
                        {
                                xPos = 10;
                        }
                        else
                        {
                                xPos = 400;
                        }
                        points[i] = new Point(xPos, 10 * i);
                }
}
```

Результат работы программы приведен на рис. 10.3.

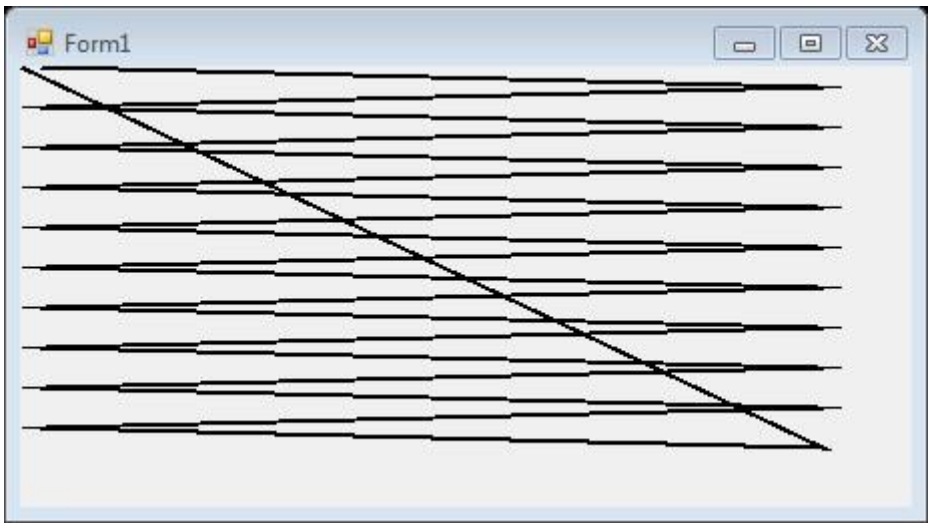

Рис. 10.3. Пример использования массива точек

Для прорисовки прямоугольников можно использовать метод DrawRectangle:

DrawRectangle(Pen, int, int, int, int);

В качестве первого параметра передается перо класса Pen. Остальные параметры задают расположение и размеры прямоугольника.

Для прорисовки многоугольников можно использовать следующий метод:

DrawPolygon(Pen, Point[]);

Метод DrawEllipse рисует эллипс, вписанный в прямоугольную область, расположение и размеры которой передаются ему в качестве параметров. При помощи метода DrawArc программа может нарисовать сегмент эллипса. Сегмент задается при помощи координат прямоугольной области, в которую вписан эллипс, а также двух углов, отсчитываемых в направлении против часовой стрелки. Первый угол Angle1 задает расположение одного конца сегмента, а второй Angle2 – расположение другого конца сегмента (рис. 10.4).

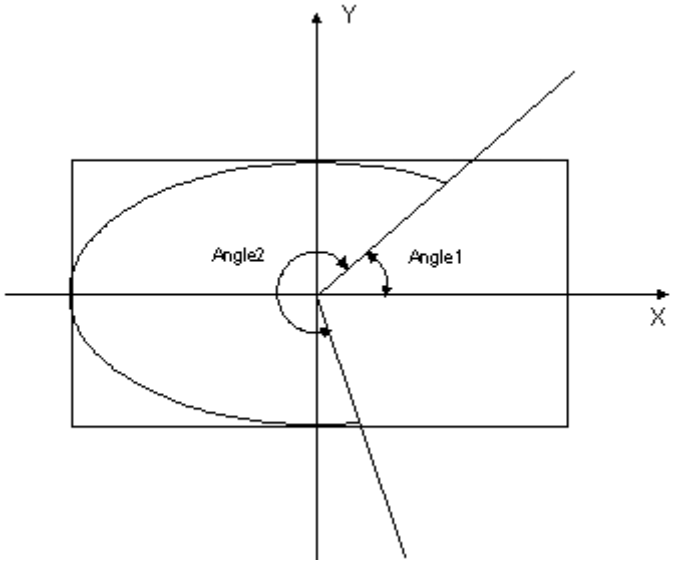

Рис. 10.4. Углы и прямоугольник, задающие сегмент эллипса

В классе Graphics определен ряд методов, предназначенных для рисования закрашенных фигур. Имена некоторых из этих методов, имеющих префикс Fill*:

FillRectangle (рисование закрашенного прямоугольника), FillRectangles (рисование множества закрашенных многоугольников), FillPolygon (рисование закрашенного многоугольника), FillEllipse (рисование закрашенного эллипса) FillPie (рисование закрашенного сегмента эллипса) FillClosedCurve (рисование закрашенного сплайна) FillRegion (рисование закрашенной области типа Region).

Есть два отличия методов с префиксом Fill* от одноименных методов с префиксом Draw*. Прежде всего, методы с префиксом Fill* рисуют закрашенные фигуры, а методы с префиксом Draw* – не закрашенные. Кроме этого, в качестве первого параметра методам с префиксом Fill* передается не перо класса Pen, а кисть класса SolidBrush. Ниже приведем пример выводящий закрашенный прямоугольник:

```
SolidBrush B = new SolidBrush(Color.DeepPink);
g.FillRectangle(B, 0, 0, 100, 100);
```

Индивидуальное задание

Изучите с помощью справки MSDN[1] методы и свойства классов Graphics, Color, Pen и SolidBrush. Создайте собственное приложение выводящий на форму рисунок, состоящий из различных объектов (линий, многоугольников, эллипсов, прямоугольников и пр.), не закрашенных и закрашенных полностью. Используйте разные цвета и стили линий (сплошные, штриховые, штрих-пунктирные).

[1] http://msdn.microsoft.com/ru-ru/library/system.drawing(v=vs.100).aspx

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по организации и проведению самостоятельной работы
по дисциплине «Технологии программирования и алгоритмизация»
для студентов направления подготовки
09.03.01 Информатика и вычислительная техника

Ставрополь 2026

ОБЩАЯ ХАРАКТЕРИСТИКА ДИСЦИПЛИНЫ

Цель и задачи дисциплины

Целью дисциплины «Технологии программирования и алгоритмизация» является формирование из набора компетенций будущего бакалавра. Изучение дисциплины обеспечивает реализацию требований Федерального Государственного образовательного стандарта высшего образования по направлению 09.03.01 Информатика и вычислительная техника

Задачи дисциплины: изучить основы алгоритмизации и программирование на примере языка программирования С++.

Место дисциплины в структуре образовательной программы

Дисциплина относится обязательной части. Ее освоение происходит в 1,2 семестрах.

Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	**Код, формулировка индикатора**	**Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов**
ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения	ИД-1ОПК-8 применяет языки программирования и работы с базами данных, современные программные среды разработки информационных систем и технологий для автоматизации бизнес-процессов, решения прикладных задач различных классов, ведения баз данных и информационных хранилищ.	Самостоятельно осваивает новые для себя современные языки программирования и языки работы с базами данных, среды разработки информационных систем и технологий.
	ИД-2ОПК-8 разрабатывает алгоритмы и программы пригодные для практического применения, принимает участие в отладке и тестировании прототипов программно-технических комплексов задач.	Разрабатывает алгоритмы и компьютерные программы, пригодные для практического применения. Читает коды программных продуктов, написанных на освоенных языках программирования, и вносит требуемые изменения.
ОПК-9 Способен осваивать методики использования программных средств для решения практических задач	ИД-2ОПК-9 использует программные средства для решения практических задач	Решает практические задачи с использованием программных средств и технологий программирования

2. ХАРАКТЕРИСТИКА САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Все виды самостоятельной работы по дисциплине «Технологии программирования и алгоритмизация» приведены в таблице «Технологическая карта самостоятельной работы». В основу самостоятельной работы студентов по дисциплине положено конспектирование лекций и рекомендуемых источников, подготовка к компьютерному тестированию знаний, оформление и подготовка к защите отчетов по лабораторным работам, выполнению разноуровневых индивидуальных заданий в ходе лабораторных работ. По каждому виду самостоятельной работы предусмотрены определённые формы отчетности, которые студенты предъявляют преподавателю в ходе лабораторных занятий или в часы индивидуальных консультаций.

Таблица 1 - Технологическая карта самостоятельной работы

Код оцениваемой компетенции, индикатора (ов)	Этап формирования компетенции (№ темы) *(в соответствии с рабочей программой дисциплины)*	Средства и технологии оценки	Вид контроля, аттестация *(текущий/промежуточный)*	Тип контроля *(устный, письменный или с использованием технических средств)*	Наименование оценочного средства
ИД-1ОПК-8 ИД-2ОПК-8 ИД-2ОПК-9	Тема 1-21	Вопросы для собеседования по лабораторным работам	текущий	устный	Вопросы для собеседования к практическим работам
ИД-1ОПК-8 ИД-2ОПК-8 ИД-2ОПК-9	Тема 1-21	Вопросы для собеседования	текущий	устный	Вопросы для собеседования

Для успешного освоения дисциплины, необходимо выполнить указанные в таблице виды самостоятельной работы, используя рекомендуемые источники информации.

1. Перечень основной литературы:

1. Тюльпинова, Н. В. Алгоритмизация и программирование Электронный ресурс: Учебное пособие / Н. В. Тюльпинова. - Саратов : Вузовское

образование, 2019. - 200 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4487-0470-3, экземпляров неограничено

2. Бедердинова, О. И Основы алгоритмизации и структурного программирования Электронный ресурс / Бедердинова О. И. : учебное пособие. - Архангельск: САФУ, 2017. - 88 с. - ISBN 978-5-261-01227-6, экземпляров неограничено

3. Забихуллин, Ф. З. Структурное программирование на С++ Электронный ресурс / Забихуллин Ф. З. : учебное пособие. - Уфа : БГПУ имени М. Акмуллы, 2019. - 45 с. - ISBN 978-5-907176-11-9, экземпляров неограничено

4. Конова, Е. А. Алгоритмы и программы. Язык С++ Электронный ресурс / Конова Е. А., Поллак Г. А. : учебное пособие для впо. - 5-е изд., стер. - Санкт-Петербург: Лань, 2020. - 384 с. - Допущено УМО по образованию в области прикладной информатики в качестве учебного пособия для студентов, обучающихся по направлению «Прикладная информатика». - ISBN 978-5-8114-5431-0, экземпляров неограничено

Белева, Л.Ф. Программирование на языке С++ Электронный ресурс: учебное пособие / Л.Ф. Белева. - Саратов : Ай Пи Эр Медиа, 2018. - 81 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4486-0253-5, экземпляров неограниченно

2. Перечень дополнительной литературы:

1. Белоцерковская, И. Е. Алгоритмизация. Введение в язык программирования С++ / И.Е. Белоцерковская; Н.В. Галина; Л.Ю. Катаева. - 2-е изд., испр. - Москва: Национальный Открытый Университет «ИНТУИТ», 2016. - 197 с., экземпляров неограничено

2. Седжвик, Р. Алгоритмы на С++ / Р. Седжвик. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 1773 с., экземпляров неограниченно

3. Лубашева, Т. В. Основы алгоритмизации и программирования: учебное пособие / Т.В. Лубашева, Б.А. Железко. - Минск : РИПО, 2016. - 378 с.: ил. - http://biblioclub.ru/. - Библиогр. в кн. - ISBN 978-985-503-625-9, экземпляров неограничено

4. Зоткин, С. П. Программирование на языке высокого уровня С/С++ Электронный ресурс / Зоткин С. П. : конспект лекций. - 3-е изд. - Москва: МИСИ – МГСУ, 2018. - 140 с. - ISBN 978-5-7264-1810-0, экземпляров неограничено

5. Каширская, Е. Н. Процедурное программирование: Практикум Электронный ресурс / Каширская Е. Н. - Москва : РТУ МИРЭА, 2020. - 75 с., экземпляров неограничено

6. Баженова, И.Ю. Введение в программирование Электронный ресурс : учебное пособие / В.А. Сухомлин / И.Ю. Баженова. - Введение в программирование,2020-07-28. - Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. - 327 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4487-0073-6, экземпляров неограниченно

3. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА

При самостоятельном изучении тем лекционных занятий и заданной литературы студентам рекомендуется:

- в день проведения занятий проработать, изученный на занятии учебный материал по конспекту. При этом необходимо выделить вопросы, на которые следует обратить большее внимание при дальнейшем изучении текущей темы. После проработки учебного материала необходимо ответить на, рекомендуемые контрольные вопросы;
- с целью качественного закрепления изученного материала в последующем следует более детально проработать учебный материал с использованием рекомендованной литературы. Студентам рекомендуется выделить вопросы, требующие более детальной проработки с помощью преподавателя;
- накануне проведения лекции студенту рекомендуется закрепить ранее изученный теоретический материал, подготовить вопросы, требующие уточнения у преподавателя.

Рекомендации по организации работы с литературой

Для овладения, закрепления и систематизации знаний студентам рекомендуется самостоятельная работа с рекомендованной литературой и конспектом лекций. При самостоятельной работе с рекомендованной литературой студентам рекомендуется проводить конспектирование учебного материала, изложенного в рекомендованных источниках.

Конспектирование источников проводится при детальном изучении темы, при этом студентам рекомендуется:

- дополнять конспект лекционного занятия путем его расширения по наиболее важным вопросам, рассмотренным на занятии.

Контроль этого вида самостоятельной работы осуществляется на учебных занятиях путем проверки преподавателем конспекта лекций и собеседования.

При конспектировании источников студенты должны исходить из рационального объема, конспектируемого материала. Конспект должен быть кратким и понятным при его использовании после изучения текущей темы.

Для успешного освоения дисциплины, необходимо самостоятельно детально изучить представленные темы по рекомендуемым источникам информации.

Работа по конспектированию источников оценивается удовлетворительно если:

- конспект имеет достаточный объем детализации по, изучаемой теме, обеспечивающий заданный уровень освоения теоретического материала дисциплины;

- конспект оформлен аккуратно, изучаемый материал изложен последовательно в соответствии с порядком изучения дисциплины, содержит обобщающие выводы по каждой теме.

4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

При выполнении практикума по дисциплине (практические работы) для достижения базового уровня сформированности компетенций студент должен уметь применять базовые знания по разделам дисциплины, владеть навыками самостоятельного решения учебных задач базового уровня.

При выполнении практикума по дисциплине (лабораторные работы) для достижения повышенного уровня сформированности компетенций студент должен уметь применять повышенные знания по разделам дисциплины, владеть навыками самостоятельного решения учебных задач повышенного уровня.

Методические указания при подготовке к практической работе

Подготовку к практической работе рекомендуется проводить в следующей последовательности:

- уяснить тему и цель, предстоящей работы;
- изучить теоретический материал в соответствии с темой работы (рекомендуется использовать рекомендованную литературу, конспект лекций, учебное пособие (практикум по практическим работам);
- ознакомиться с оборудованием и материалами, используемыми на практической работе (при использовании специализированного оборудования или программного обеспечения необходимо изучить порядок и правила его использования).

Методические указания при выполнении практической работы

При выполнении работы студенты должны строго соблюдать, установленные требования безопасности.

При выполнении работы студентам рекомендуется:

- уяснить цель, выполняемых заданий и способы их решения;
- задания, указанные в работе выполнять в той последовательности, в которой они указаны в практикуме;
- при выполнении практического задания и изучении теоретического материала использовать помощь преподавателя;
- оформить отчет по практической работе;
- ответить на контрольные вопросы.

Методические указания при подготовке к защите практической работы

При подготовке к защите работы студентам рекомендуется:

- подготовить отчет по л работе;
- подготовить обоснование, сделанных выводов;
- закрепить знания теоретического материала по теме работы (рекомендуется использовать контрольные вопросы);

- знать порядок проведения расчетов (проводимых исследований);
- уметь показать и пояснить порядок исследований при использовании специализированного оборудования или программного обеспечения.

Защита работы осуществляется путем собеседования студента с преподавателем. При собеседовании студент представляет на проверку отчет по работе. Собеседование со студентом, претендующим на оценку «отлично» проводится по вопросам и практическим заданиям повышенного уровня обучения. Собеседование со студентом, не претендующим на оценку «отлично» проводится по вопросам и практическим заданиям базового уровня обучения.

5. ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ ПО САМОСТОЯТЕЛЬНОМУ ИЗУЧЕНИЮ ЛИТЕРАТУРЫ

Вопросы для собеседования

Вопросы (задача, задание) для проверки уровня обученности

1. Для чего нужны диаграммы деятельности UML?
2. Что такое состояние действия и состояние деятельности?
3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?
4. Какой алгоритм является алгоритмом разветвляющейся структуры?
5. Чем отличается разветвляющийся алгоритм от линейного?
6. Что такое условный оператор? Какие существуют его формы?
7. Что такое составной оператор? Каков формат его записи?
8. Какие операторы сравнения используются в Си?
9. Что называется простым условием? Приведите примеры.
10. Что такое составное условие? Приведите примеры.
11. Какие логические операторы допускаются при составлении сложных условий?
12. Может ли оператор ветвления содержать внутри себя другие ветвления?
13. Что такое множественный выбор?
14. В каких случаях применяется переключатель?
15. Зачем ставится в переключателе оператор break?
16. Зачем в переключателе употребляется зарезервированное слово default?
17. Какие UML-диаграммы показывают работу переключателя?
18. Какой алгоритм является алгоритмом циклической структуры?
19. Типы циклов в языке Си.
20. Назовите основные параметры цикла.

21. Что представляют собой операторы инкремента и декремента, в каких формах они используются?

22. Как записывается условие продолжения цикла в циклах типа while и do ... while?

23. Основная форма цикла do … while.

24. Какие три раздела записываются в круглых скобках для оператора цикла типа for? Какой разделитель между разделами?

25. Что такое вложенные циклы? Примеры.

26. Как образуется бесконечный цикл и как выйти из него?

27. Схема приведения цикла while к эквивалентному ему циклу for.

28. Назовите операторы, способные изменять естественный порядок выполнения вычислений.

29. Для чего нужен оператор break?

30. Где употребляется оператор continue и для чего он используется?

31. Для чего используется оператор return?

32. Какой тип должно иметь выражение?

33. Как объявляются одномерные массивы в языке C++?

34. Какими должны быть размерности при описании статического массива в языке C++?

35. Каков диапазон изменения индекса массива в языке C++?

36. Каким образом производится инициализация массива в языке C++?

37. Чем является идентификатор массива?

38. Для чего используется управляющая последовательность \t?

39. Как представляется в C++ двумерный массив?

40. Где и каким образом хранится двумерный массив?

41. В каких пределах изменяются индексы двумерного массива?

42. Какими способами можно описать двумерный массив?

43. Какие действия выполняются для каждой строки при вычислении количества положительных элементов?

44. В каком месте программы следует записывать операторы инициализации накапливаемых в цикле величин?

45. Каким образом в динамической области памяти можно создавать двумерные массивы?

46. Способы создания динамического массива.

47. Каким образом производится обращение к элементам динамических массивов?

48. Что представляет собой функция в C++? Что нужно для ее использования?

49. Приведите пример заголовка функции.

50. Что включает в себя определение функции?

51. В чем отличие функции от других программных объектов?

52. Каким образом происходит вызов функции?

53. Охарактеризуйте существующие способы решения проблемы получения из подпрограммы признака ее аварийного завершения.

54. Каков механизм передачи параметров в функцию?

55. Способы передачи входных данных.
56. Что представляет собой средство C++, называемое значениями параметров по умолчанию?
57. Каким образом происходит передача в функцию имени функции?
58. Каким образом происходит передача одномерных массивов в функцию?
59. Каким образом происходит передача строк в функцию?
60. Каким образом происходит передача двумерных массивов в функцию?
61. Каким образом происходит передача структур в функцию?
62. Какая функция называется рекурсивной? Преимущества и недостатки рекурсии.
63. В чем смысл использования шаблонов?
64. Как записать параметр шаблона?
65. Перечислите основные свойства параметризованных классов?
66. Что такое ассоциация?
67. Что такое наследование?
68. Что такое агрегация?
69. Что такое зависимость?
70. Назовите и охарактеризуйте три основных принципа в использовании классов.

1. Критерии оценивания компетенций (*в соответствии с результатами освоения дисциплины***)**

Оценка «**отлично**» выставляется студенту, если студент дал полный, развернутый ответ на поставленные вопросы, при этом показана совокупность осознанных знаний по дисциплине, доказательно раскрыты основные положения вопросов; в ответе прослеживается четкая структура, логическая последовательность, отражающая сущность раскрываемых понятий. Знание демонстрируется на фоне понимания его в системе дисциплины. Ответ изложен литературным языком с использованием технической терминологии. Могут быть допущены недочеты в определении понятий, исправленные студентом самостоятельно в процессе ответа, продемонстрировал высокое умение применять полученные знания на практике через решение конкретной задачи, свободно без затруднений справился с поставленной задачей, показав владение разносторонними приемами и навыками ее выполнения, не допустил ошибок и неточностей.

Оценка «**хорошо**» выставляется студенту, если студент дал полный, развернутый ответ на поставленные вопросы, при этом показано умение выделить существенные и несущественные признаки, причинно-следственные связи. Ответ четко структурирован, логичен, изложен литературным языком с использованием технической терминологии. Могут быть допущены 2-3 неточности или незначительные ошибки, исправленные студентом с помощью преподавателя, продемонстрировал умение применять полученные знания на практике через решение конкретной задачи, справился

с поставленной задачей, показав владение необходимыми приемами и навыками ее выполнения, при этом допустил не более одной ошибки.

Оценка «**удовлетворительно**» выставляется студенту, если студент дал недостаточно полные и недостаточно развернутые ответы по вопросам билета. Логика и последовательность изложения имеют нарушения. Допущены ошибки в раскрытии понятий, употреблении терминов. Студент не способен самостоятельно выделить существенные и несущественные признаки и причинно-следственные связи. В ответе отсутствуют выводы. Умение раскрыть значение обобщенных знаний не показано. Речевое оформление требует поправок, коррекции, продемонстрировал посредственное умение применять полученные знания на практике через решение конкретной задачи, с трудом справился с поставленной задачей, при этом допустил не более двух ошибок.

Оценка «**неудовлетворительно**» выставляется студенту, если ответ представляет собой разрозненные знания с существенными ошибками по вопросу. Присутствуют фрагментарность, нелогичность изложения. Студент не осознает связь обсуждаемых вопросов с другими объектами дисциплины. Отсутствуют выводы, конкретизация и доказательность изложения. Речь неграмотная, гистологическая терминология не используется. Дополнительные и уточняющие вопросы преподавателя не приводят к коррекции ответа студента, ответ на вопросы полностью отсутствует или студент отказался от ответа, не продемонстрировал умение применять полученные знания на практике через решение конкретной задачи, не справился с поставленной задачей или допустил при ее решении три и более серьезные ошибки.

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения данного оценочного мероприятия включает в себя:

- отдельное собеседование по каждой из указанных тем;
- составление конспекта.

Предлагаемые задания позволяют проверить компетенции **ОПК-1**

Для подготовки к каждой теме следует изучить и законспектировать рекомендованные источники и научную литературу и на их основании подготовиться к указанным в методических материалах вопросам для обсуждения.

При проверке задания, оцениваются

- увереное владение материалом;
- умение четко и логично излагать свои мысли:
- умение творчески подходить к решению основных вопросов темы;
- самостоятельность мышления,
- полное соответствие конспекта установленным требованиям;
- самостоятельная устная речь, полностью раскрывающая суть суть

проблематики собседования.

7. ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ ПРИ ПРОВЕРКЕ ГОТОВНОСТИ К ПРАКТИЧЕСКИМ РАБОТАМ

Вопросы для защиты практических работ

Тема: Форматы представления констант и данных, виды операций, ввод-вывод в языке C++

1. Форматы представления констант и данных, виды операций, ввод-вывод в языке C++

Тема: Исследование программирования алгоритмов разветвляющейся структуры

1. Операторы if и switch языков Си и C++.

Тема: Исследование программирования алгоритмов циклической структуры

1. Операторы while, do … while и for языков Си и C++.

Тема: Исследование программирования алгоритмов с использованием статических массивов

1. Определение и использование статических массивов в языках программирования Си и C++.

Тема: Исследование программирования алгоритмов с использованием динамических массивов

1. Связь между массивами и указателями в языках Си и C++.
2. Выделение и освобождение динамической памяти в языках Си и C++.
3. Обращение к элементам динамического массива.

Тема: Исследование программирования алгоритмов с использованием многомерных массивов

1. Определение и использование статических и динамических многомерных массивов в языках Си и C++.

Тема: Исследование средств обработки строк в C++

1. Определение и операции со строками стандартной библиотеки C++.

Тема: Исследование программирования алгоритмов обработки ASCIIZ строк

1. Определение и использование ASCIIZстрок в языках Си и C++.
2. Исследование средств стандартной библиотеки для обработки ASCIIZстрок.

Тема: Исследование возможностей повторного использования кода в структурном программировании

1. Определение и использование обычных и встраиваемых функций в языках Си и C++.

Тема: Исследование средств препроцессора

1. Средства препроцессора в языках Си и C++.
2. Определение и использование макроопределений в языках Си и C++.
3. Сравнение макроопределений и функций.

Тема: Исследование средств для создания многомодульных программ

1. Директивы условной компиляции.
2. Создание подключаемых файлов.
3. Статических и динамические библиотеки.
4. Компиляция многомодульных программ с использованием компиляторов GCCи Clang.

Тема: Исследования средств создания классов и объектов в языке C++

1. Определение класса.
2. Использование класса.
3. Определение методов класса.
4. Вложенные классы.

Тема: Исследование средств инициализации объектов и перегрузка операций в языке C++

1. Перегрузка операций.
2. Конструкторы и деструктор.
3. Константы в классе.
4. Поля-массивы в классе.
5. Статические элементы класса.

Тема: Наследование классов в языке C++

1. Простое открытое наследование.
2. Конструкторы и деструкторы при наследовании.
3. Поля и методы при наследовании.
4. Статические элементы класса при наследовании.
5. Вложенные классы и наследование.
6. Операторы присваивания и принцип подстановки.
7. Функции-операции преобразования.
8. Закрытое и защищенное наследование.
9. Виртуальные и абстрактные методы.
10. Абстрактные классы.

Тема: Исследование средств организации ввода-вывода в языке C++

1. Классификация потоков.
2. Подключение потоков.
3. Операции ввода-вывода.
4. Состояние потока.
5. Форматирование ввода-вывода.
6. Файловые потоки.
7. Буферизация.
8. Строковые потоки.
9. Позиционирование в потоке.
10. Широкие потоки.

1. Критерии оценивания компетенций

оценка «отлично» выставляется, если студент продемонстрировал высокое умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, свободно без затруднений справился с поставленной задачей, показав владение разносторонними приемами и навыками ее выполнения, не допустил ошибок и неточностей;

оценка «хорошо» выставляется, если студент продемонстрировал умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, справился с поставленной задачей, показав владение необходимыми приемами и навыками ее выполнения, при этом допустил не более одной ошибки;

оценка «удовлетворительно» выставляется, если студент продемонстрировал посредственное умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, с трудом справился с поставленной задачей, при этом допустил не более двух ошибок;

оценка «неудовлетворительно» выставляется, если студент не продемонстрировал умение применять полученные знания на практике через решение конкретной задачи с применением алгоритмов и языков программирования, не справился с поставленной задачей или допустил при ее решении три и более серьезные ошибки.

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура проведения данного оценочного мероприятия включает в себя:

- собеседование;
- защита практических работ.

Предлагаемые задания позволяют проверить компетенции **ОПК-1**.

К практическому занятию студент должен подготовить ответы на вопросы, проработать дополнительные научные источники, сделать конспект теоретического материала (если это предусмотрено).

Максимальное количество баллов студент получает, если он выполнит лабораторное задание, владеет материалом, умеет логично и четко излагать мысли, показвает самостоятельность выполнения задания.

Основанием для снижением оценки являются:

- слабое знание темы и неполностью выполненное задание;
- невыполненное задание к лабораторной работе;
- отсутствие умения применить теоретические знания для решения задач.

При проверке задания, оцениваются

- активное участие в работе;
- увереное владение материалом;
- умение четко и логично излагать свои мысли:
- умение творчески подходить к решению заданий;
- самостоятельность мышления,
- самостоятельная устная речь, полностью раскрывающая суть суть проблематики собседования.