МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра защиты информации и систем связи

УТВЕРЖДАЮ
проректор по учебной работе
_____ О. Г. Локтионова
«___» _____ 2014 г.

РЕАЛИЗАЦИЯ СТРУКТУРНОГО И ОБЪЕКТНОГО ПОДХОДОВ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ С++

Методические указания по выполнению
лабораторных работ для студентов направления
подготовки бакалавров 090900.62
и специальности 090303.65.

Курск 2014

УДК 004.43

Составитель К.А. Тезик

Рецензент

Кандидат технических наук, доцент О. Ф. Корольков

Реализация структурного и объектного подходов на языке программирования С++ : методические указания по выполнению лабораторных работ по дисциплине «Технологии и методы программирования» / Юго-Зап. гос. ун-т; сост.: К. А. Тезик. Курск, 2014. 63 с.: ил. 2, табл. 1, Библиогр.: с. 63.

Содержат краткие теоретические положения, указания к решению задач и задания для самостоятельной работы по языку программирования С++.

Методические указания соответствуют требованиям программы по направлению подготовки бакалавров: информационная безопасность и специальности: информационная безопасность автоматизированных систем.

Предназначены для студентов направления подготовки бакалавров 090900.62 и специальности 090303.65 дневной и заочной форм обучения.

Текст печатается в авторской редакции

Подписано в печать 6.06.14 Формат 60x84 1/16.
Усл. печ. л. 3,7 . Уч. – изд. л. 3,5 .Тираж 100 экз. Заказ. 297 Бесплатно.
Юго - Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Введение

Язык программирования C++ создан Бьерном Страуструпом в 1979 году в компании Bell Laboratories (г. Муррей-Хилл, шт. Нью-Джерси). Сначала новый язык получил название “С с классами” (C witch Classes), но в 1983 году он стал называться C++. C++ полностью включает язык С. Однако язык С является процедурным языком и реализует структурный подход к программированию. Данный подход успешно позволял разрабатывать программы средней сложности. Но если программный проект достигал определенного размера, его сложность резко возрастала и оказывалась непреодолимой для возможностей программиста. Для решения данной проблемы был создан язык C++, который поддерживает как структурный так и объектный подходы к разработке программ. По сути C++ стал объектно-ориентированной версией языка С.

В свою очередь C++ является родительским языком для Java и C#. И хотя разработчики Java и C# добавили к первоисточнику, удалили из него или модифицировали различные средства, в целом синтаксис этих трех языков практически идентичен. Объектная модель, используемая C++, подобна объектным моделям языков Java и C#. Изучив основы C++, студенты в дальнейшем могут легко приступить к самостоятельному изучению языков Java и C#. Основное различие между C++, Java и C# заключается в типе вычислительной среды, для которой разрабатывался каждый из этих языков. C++ создавался с целью разработки высокоэффективных программ, предназначенных для выполнения под управлением определенной операционной системы. Java позволяет создавать межплатформенный (совместимый с несколькими операционными средами) переносимый программный код для Internet. C# разработан для среды .NET Framework (Microsoft), которая поддерживает многоязычное программирование (mixed-language programmirung) и компонентно- ориентированный код, выполняемый в сетевой среде. Итак, Java и C# позволяют создавать переносимый программный код, но программы на языке C++ более эффективны (скорость их выполнения выше).

1. Структура программы в языке программирования С++. Ввод и вывод данных в языке программирования С++.

Краткие теоретические положения

В общем случае программа С++ состоит из четырех блоков или частей.

1. **Блок заголовков программы.** В этом блоке с помощью инструкции # include <имя файла> подключаются внешние файлы. Например, в задании 1 файл iostream подключается для поддержки системы ввода-вывода, в задании 2 файл cmatch для поддержки математических функций. Команда using namespace std является инструкцией для компилятора использовать стандартную область имен.
2. **Блок с объявлением классов (базовых и производных), прототипами и объявлениями функций.**
3. **Главный метод программы:** каждая программа имеет такой метод. У метода стандартное название main ( ). Метод – это фактически синоним слова функция (или процедура). Запуск программы означает выполнение ее главного метода main ( ). У программы может быть один и только один метод main( ).
4. **Блок с описанием функций** (прототип которых указан во втором блоке)

При этом обязательными являются только первый и третий блоки: программа содержит блок подключения файлов и главный метод main ( ). В инструкции int main(void) идентификатор int означает тип результата, возвращаемого функцией, идентификатор void означает, что у функции нет параметров. В общем случае после инструкции return указывается значение, возвращаемое функцией в качестве результата. Команда return 0 является формальным подтверждением того, что работа программы завершена корректно.

Для вывода данных на экран в С++ используется команда:

cout << “Текст\n”; . Она состоит из идентификатора cout (сокращение от console output, означает устройство вывода), оператора вывода << и текста вывода в двойных кавычках (например в

задании 1 – это фраза “Hello, World\n”). Необязательная инструкция \n служит для перехода к новой строке.

Для ввода данных с клавиатуры в С++ используется команда: cin >> переменная; . Она состоит из идентификатора cin (сокращение от console input,означает устройство ввода), оператора ввода >> и имени переменной. Переменной присваивается значение, которое пользователь вводит в командной строке. В задании 3 для считывания строки вместо оператора cin>> s используется функция getline(cin,s). Дело в том, что если строка состоит из нескольких слов, то первый пробел указывает объекту cin об окончании ввода. Поэтому при использовании оператора cin>> s программа считывает только первое слово вводимой строки. При использовании функции getline(cin,s) программа корректно работает в случае ввода строки, состоящей из нескольких слов.

Задание 1

Цель работы: изучить структуру программы на языке СИ++

Назначение программы: программа выводит на экран сообщение “Привет мир”

```
#include <iostream>
  using namespace std;
  int main(void)
   {
   // cout << "Привет мир\n";
    cout << "Hello world\n";
  return 0;
   }
```

Задание 2

Цель работы: изучить методику ввода и вывода числовых данных в языке СИ++

Назначение программы: программа рассчитывает площадь круга по введенному значению радиуса круга.

```
#include <iostream>
#include <cmath>
  using namespace std;
  int main(void)
   {
         double rad, pl;
        //cout<<" Введи радиус   ";
        cout<<"  Enter radius  ";
        cin>>rad;
        pl=3.14*pow(rad,2);
        //cout<<"Площадь =  "<<pl<<endl;
        cout<<" area =  "<<pl<<endl;
  return 0;
   }
```

Задание 3

Цель работы: изучить методику ввода и вывода строк на языке СИ++

Назначение программы: программа считывает строку символов с клавиатуры и затем выводит ее на экран. Строка символов может содержать пробелы. После окончания ввода строки нажмите ENTER два раза.

```
#include <iostream>
#include <string>
  using namespace std;
  int main(void)
   {
         string s;
        //cout<<"Введите строку  ";
        cout<<" enter a line    ";
        getline(cin,s);
        cout<<"Строка = "<<s<<"\n";
        cout<<" line  "<<s<<"\n";
  return 0;
   }
```

2. Условные операторы в языке программирования С++

Краткие теоретические положения

В С++ два условных оператора: if( ) и switch( ). Оператор if( ) позволяет выполнить разные блоки операторов в зависимости от того, выполняется ли определенное условие. Общий синтаксис оператора следующий:

```
If (условие )  {операторы 1}
else {операторы 2}
```

Если условие, указанное после ключевого слова if , верно выполняется блок операторов **операторы 1**. В противном случае выполняется блок операторов **операторы 2,** указанных после ключевого слова else.

Допускается использование упрощенного варианта условного оператора, в котором отсутствует ветвь else:

```
If (условие )  {операторы 1}
```

Нередко на практике используют комбинацию из нескольких вложенных условных операторов. Синтаксис вызова вложенных условных операторов имеет вид:

```
If (условие 1) {операторы 1}
      else if (условие 2) { операторы 2}
…….
            else if (условие N) { операторы N}
                  else {операторы N+1}
```

В тех случаях, когда проверяется больше одного условия, вместо нескольких вложенных условных операторов if( ) можно использовать оператор switch( ). Синтаксис вызова оператора switch( ) следующий:

```
switch( ) (выражение) {
```

```
case значение 1:
        операторы 1
                break;
case значение 2:
        операторы 2
                break;
……..
case значение N:
        операторы N
                break;

default :
                операторы N+1
        }
```

В круглых скобках после ключевого слова switch указывается выражение, значение которого проверяется. Результатом выражения может быть целое число или символ. Значение, возвращаемое выражением, сравнивается со значениями, указанными после ключевых слов case. Если имеет место совпадение, выполняется соответствующий блок операторов. Операторы выполняются до конца оператора switch или пока не встретится инструкция break. Если совпадения нет, выполняются операторы после инструкции default. Инструкции break и default не являются обязательными. В общем случае инструкция break используется для выхода из оператора цикла и перехода к следующему оператору.

Задание 4

Цель работы: изучить методику применения условного оператора в языке СИ++

Назначение программы: программа по значению введенного балла выдает сообщение о положительном результате тестирования или о необходимости пересдать тест

```
#include <iostream>
  using namespace std;
  int main(void)
```

```
  {
      int s;
      //cout<<"Введите балл за тест ";
      cout<<" Enter point for the test ";
      cin>>s;
      if (s>=60)
      //cout<<"Вам по тесту зачет" << endl;
      cout<<" To you according to dough offset   " << endl;
      else
   // cout<<"Вам по тесту не зачет, надо пересдать тест"<<endl;
cout<<"  To you according to dough not offset, it is necessary to
repeat the test "<<endl;
     return 0;
  }
```

Задание 5

Цель работы: изучить методику применения оператора выбора в языке СИ++

Назначение программы: программа по введенной оценке за тест определяет диапазон баллов

```
#include <iostream>
  using namespace std;
int main(void)
  {
      char n;
      //cout<<"Введите  оценку   ";
      cout<<" Enter an assessment ";
      cin>>n;
      switch (n)
      {
      case '5':
      //cout<<"Ваш балл > 80" << endl;
      cout<<" Your point >80" << endl;
      break;
       case '4':
```

```
        //cout<<"Ваш балл от 70 до 80" << endl;
        cout<<"  Your point from 70 to 80 " << endl;
        break;
        case '3':
        //cout<<"Ваш балл от 60 до 70" << endl;
        cout<<" Your point from 60 to 70 " << endl;
        break;
        case '2':
        //cout<<"Ваш балл <60" << endl;
        cout<<" Your point <60 " << endl;
        break;
    default:
        //cout<<"Неверный ввод" << endl;
        cout<<" Incorrect input" << endl;
        }
  return 0;
   }
```

3. Операторы цикла в языке программирования C++

Краткие теоретические положения

Операторы цикла позволяют многократно выполнять серии однотипных действий. Действия выполняются до тех пор, пока остается справедливым (или пока не будет выполнено) некоторое условие. В C++ используются три оператора цикла: for(), while( ) и do-while( ).

Общий синтаксис вызова оператора for ( ) следующий:

for (инициализация; условие; изменение переменных) {команды}

В круглых скобках после ключевого слова for указывается программный код из трех блоков (при этом каждый из блоков может быть пустым). Блоки разделяются точкой с запятой. Первый блок является блоком инициализации. В нем присваивается на-

чальное значение для переменной цикла. Второй блок – условие выполнения оператора цикла. Пока справедливо условие, оператор цикла будет выполняться. Третий блок – это блок изменения индексных переменных.

Отметим общий принцип выполнения оператора for( ). Сначала выполняются команды, указанные в первом блоке оператора. После этого проверяется условие, указанное во втором блоке оператора. Если условие справедливо, выполняются команды после инструкции for (если команд несколько, они заключаются в фигурные скобки). Затем выполняются команды третьего блока. Далее снова проверяется условие (второй блок). При справедливости условия снова выполняются команды в фигурных скобках и команды третьего блока и т. д.

Синтаксис вызова оператора while ( ) следующий:

```
while (условие) {
команды
}
```

Сначала проверяется условие, указанное в круглых скобках после ключевого слова while. Если условие справедливо, поочередно выполняются операторы, указанные в фигурных скобках после инструкции while. Если инструкция одна, фигурные скобки можно не указывать.

Синтаксис вызова оператора do – while ( ) имеет вид:

```
do {
     команды
    }
while (условие);
```

В операторе цикла do – while ( ) выполняемые команды (заключенные в фигурные скобки) указываются после ключевого слова do. Далее проверяется условие, указанное в круглых скобках после ключевого слова while. Если условие выполнено, снова выполняются команды после ключевого слова do и т. д.

Принципиальная разница между операторами while( ) и do-while ( ) состоит в том, что в первом случае сначала проверяется условие, а затем (если верно условие) выполняются команды. Во втором случае сначала, по крайней мере, один раз выполняются команды, а затем проверяется условие.

Задание 6

Цель работы: изучить методику применения оператора цикла for в языке СИ++

Назначение программы: программа рассчитывает значение функции y=f(x), значение аргумента x изменяется в цикле

```
#include <iostream>
 using namespace std;
 int main(void)
  {
        int x,y;
      for (x=1; x<=20; x+=2)
      {
       y=2*x*x+5*x+3;
       cout<<"at  x="<<x;
       cout<<" y=  "<<y<<"\n";

      }
      return 0;
  }
```

Задание 7

Цель работы: изучить методику применения оператора do while в языке СИ++

Назначение программы: в данной программе цикл выполняется до тех пор, пока пользователь не введет символ “Y”.

```
#include <iostream>
using namespace std;
int main()
{
```

```
char ch;
do {
        //cout<<"Введите  Y  для выхода из программы";
        cout<<" Enter Y for an exit from the program ";
        cin>>ch;
}
while((ch !='Y') && (ch !='y'));
        return 0;
}
```

Задание 8

Цель работы: изучить методику программирования структуры алгоритма “цикл в цикле”

Назначение программы: программа печатает матрицу символов “z”. Во внутреннем цикле происходит изменение номера столбца, а во внешнем –номера строки элемента матрицы.

```
#include <iostream>
 using namespace std;
 int main(void)
  {
        int x;
        int y;
        for(x=1; x<=5; x++)
        {
              for(y=1; y<=10; y++)
              {
       cout<<'z';
              }
    cout<<'\n';
       }
       return 0;
  }
```

Задание 9

Цель работы: Изучить методику программирования структуры алгоритма “разветвление внутри цикла”

Назначение программы: Программа реализует калькулятор. Пользователь поочередно вводит числа и символы операций с этими числами (сложение, вычитание, умножение, деление). Процесс ввода пользователем реализован посредством формально бесконечного оператора цикла. Для выхода из оператора цикла необходимо, чтобы пользователь ввел знак равенства.

```
#include <iostream>
using namespace std;
int main()
{
	double x,s;
	char op;
	cout<<">>";
	cin>>x;
	s=x;
	while(true) {
		cout<<">>";
		cin>>op;
		if(op=='='){
		cout<<"--->"<<s<<endl;
		exit(0);
		}
		cout<<">>";
		cin>>x;
	switch(op)
	{
	case '+':
		s+=x;
		break;
	case '-':
		s-=x;
		break;
	case '*':
```

```
            s*=x;
            break;
        case '/':
            s/=x;}
        }
        return 0;
}
```

4. Статические массивы в языке программирования C++.

Краткие теоретические положения

Массив – это совокупность переменных одного типа, объединенных общим именем. Доступ к элементам массива осуществляется путем индексирования. Размерность массива определяется количеством индексов, необходимых для однозначного определения элемента массива. Статическими называются массивы, размер которых известен при компиляции программы. Объявление массива выполняется следующим образом: указывается тип данных, к которому принадлежат элементы массива, имя массива, а также его размер (количество элементов массива). Например, командой int m[10] объявляется целочисленный массив с именем m, который состоит из 10 элементов. Обращение к элементу массива выполняется через имя массива с индексом элемента в квадратных скобках. При этом индексация элементов массива в C++ начинается с нуля. Первым элементом указанного выше массива является m[0], а последним, десятым – элемент m[9].

Размерность массива может быть больше единицы. Например, инструкцией double n[4] [5] объявляется двумерный массив действительных чисел двойной точности размером 4x5. Чтобы обратиться к элементу массива, необходимо после имени массива указать индексы этого элемента (каждый индекс в отдельных квадратных скобках). Первый индекс определяет строку, второй индекс опре-

деляет столбец в этой строке. Например, элемент n[1] [3] находится на пересечении второй строки и четвертого столбца.

В C++ существует возможность инициализации массивов при их объявлении. Для инициализации одномерного массива после имени и размера массива через оператор присваивания указывается список со значениями элементов. Список заключается в фигурные скобки, сами значения разделяются запятыми. Например, командой int n[4]= {2, 4, 6, 3} инициализируется массив из четырех целочисленных элементов. При инициализации двумерного массива присваивание значений элемента массива выполняется построчно. Например, инициализация двумерного массива может выглядеть так:

```
Double numbers [3] [2] {1.1, 3.2,
                        8.3, 5.4,
                        9.5, 2.6};
```

Текстовые строки в C++ реализуются в виде массивов символов, либо в виде объектов класса string. Пример объявления символьного массива

```
char str[80];
```

Чтобы вписать в массив строку, необходимо, чтобы размер массива по крайней мере на единицу превышал количество символов в строке. Этот дополнительный элемент необходим для записи нуль - символа '\0' окончания строки. Данный символ показывает, где в массиве записана полезная информация. Поэтому в указанный выше массив можно записать строку с максимальной длиной 79 символов.

Инициализироваться символьные массивы, могут также как и числовые, смотрите пример, представленный в задании 11. Однако существует еще один способ инициализации символьного массива: вместо списка символов указывается в двойных кавычках текстовая строка. Например, char str1[ ]=”hello”; .

При этом нуль-символ окончания строки добавляется в конец массива автоматически. Размер одномерного массива при его инициализации можно не указывать. Он определяется автоматически по числу значений в списке.

Задание 10

Цель работы: изучить методику обработки числовых массивов на языке С++

Назначение программы: программа рассчитывает сумму элементов массива и максимальный элемент массива. Элементы массива вводятся пользователем с клавиатуры.

```
#include <iostream>
 using namespace std;
 int main(void)
  {
        int s=0;
        int max;
        int test[5];
        max = test[1];
      for (int i=0; i<5; i++)
      {
            //cout<<"Введите элемент массива  ";
            cout<<" Enter a massif element  ";
            cin>>test[i];
            if (test[i]>max)
                  max=test[i];
            cout<<'\n';
            s+=test[i];
      }
            //cout<<"Сумма элементов массива =    " <<s;
            cout<<" Sum of elements of the massif    =    " <<s;
            cout<<'\n';
      //cout<<"Максимальный элемент массива =  "<<max;
      cout<<"  Maximum element of the massif   =  "<<max;
            cout<<'\n';
        return 0;
  }
```

Задание 11

Цель работы: изучить методику инициализации элементов символьного массива на языке С++.

Назначение программы: программа выводит на экран массив символов. Нулевой символ '\0' сигнализирует объекту cout, когда заканчивать вывод массива.

```
#include <iostream>
 using namespace std;
 int main(void)
  {
        char test[6]={'1', '2', 'r', 'h', 't', '\0'};
        // cout<<"выводим на экран массив символов"<<"\n";
        cout<<" we display the massif of symbols "<<"\n";
 cout<<test;
 cout<<'\n';
  return 0;
   }
```

Задание 12

Цель работы: изучить методику программирования на языке C++ задач сортировки массивов.

Назначение программы: программа генерирует исходный массив чисел, выводит его на экран, затем сортирует массив по возрастанию элементов и выводит отсортированный массив на экран. Алгоритм сортировки следующий. Каждый элемент массива сравнивается с соседним и если первый элемент больше второго, эти элементы меняются местами. После однократного перебора всех элементов самый большой элемент оказывается последним в массиве. Еще раз перебрав элементы массива, на предпоследнее место перемещаем второй по величине элемент и так далее. Продолжая эту процедуру необходимое количество раз, добиваемся ситуации, когда элементы массива размещены в порядке возрастания.

```
#include<iostream>
using namespace std;
int main() {
       const int m=10;
```

```
        int MyArray[m];
        int i, j, s;
    cout<<"Before:\n";
    // исходный массив
    for (i=0; i<m; i++) {
        MyArray[i]=rand() % 20;
        cout<<MyArray[i]<<" ";}
    // сортировка массива
    for(j=1; j<=(m-1);j++)
    for (i=0; i<m-j; i++)
    if (MyArray[i]>MyArray[i+1]) {
    s=MyArray[i+1];
    MyArray[i+1]=MyArray[i];
    MyArray[i]=s;}
    cout<<"\nAfter:\n";
    //массив после сортировки
    for (i=0;i<m;i++)
    cout<<MyArray[i]<<" ";
    cout<<"\n";
    return 0;
    }
```

Задача 13

Цель работы: изучить методику обработки матриц на языке C++.

Назначение программы: в данной программе решена задача умножения двух матриц размером 3x3. Результат (матрица) выводится на экран.

```
#include <iostream>
using namespace std;
int main () {
//размер матриц:
const int N=3;
//индексные переменные:
int i,j,k;
//первая матрица:
```

```
double A[N] [N];
// Вторая матрица:
double B[N] [N];
// Третья матрица(результат):
double C[N] [N];
// Ввод элементов первой матрицы:
cout<<"Matrix A:\n";
for (i=0;i<N;i++)
for (j=0; j<N; j++)
cin>>A[i] [j];

// Ввод элементов второй матрицы:
cout<<"Matrix B:\n";
for (i=0;i<N;i++)
for (j=0; j<N; j++)
cin>>B[i] [j];

// Вычисление произведения матриц:
cout<<"Matrix C=AB:\n";
for (i=0; i<N;i++){
        for (j=0; j<N; j++){
        C[i] [j]=0;
        for (k=0; k<N; k++)
        C[i] [j]+=A[i] [k]*B[k] [j];
//Вывод значения элемента на экран:
        cout<<C[i] [j]<<" ";}
        cout<<endl;}
return 0;
}
```

5. Указатели в языке программирования С++. Динамические массивы в языке программирования С++.

Краткие теоретические положения

Указатель – это переменная, которая содержит адрес другой переменной. Переменные –указатели должны быть объявлены. Формат объявления переменной – указателя таков:

тип_переменной *указатель;

Чтобы объявить переменную p указателем на int-значение следует использовать следующую инструкцию:

int *p;

С указателями используются два унарных оператора: “*” и “&”. Оператор “&” возвращает адрес памяти, по которому расположен операнд. Второй оператор “*” обращается к значению переменной, расположенной по адресу, заданному его операндом. То есть он ссылается на значение переменной, адресуемой заданным указателем. Пример работы с переменными-указателями смотрите в задании 14.

Особенность С++ связана с тем, что имя массива (без индексов) является указателем на первый элемент массива. Например, имя массива n является указателем (адресом) на первый элемент массива n [0].

Динамический массив отличается от статического тем, что на момент компиляции размер динамического массива не известен. Размер динамического массива определяется в процессе выполнения программы. Динамические массивы реализуются посредством операторов динамического распределения памяти. В С++ для выделения области памяти используется оператор new, а для освобождения выделенной ранее памяти используют оператор delete.

Общий синтаксис для команды выделения памяти выглядит так:

тип_переменной *указатель;

указатель = new тип_переменной;

Например, чтобы выделить память под целочисленную переменную p типа int, достаточно воспользоваться последовательностью команд:

```
int  *p;
p=new int;
```

Очистка памяти, выделенной под переменную, осуществляется с помощью оператора delete. Синтаксис вызова оператора следующий:

```
delete указатель;
```

Например, чтобы освободить память для ячейки с адресом p, используем команду delete p;

При выделении памяти можно сразу выполнять инициализацию. Например, корректной является такая последовательность команд:

```
int  *p;
p=new int(25);
```

В этом случае выделяется память для целочисленной переменной и в соответствующую ячейку заносится значение 25.

Аналогично обычным переменным выделяется память для массивов.

Главное отличие в синтаксисе вызова оператора new состоит в том, что после имени базового типа переменной указывается размер массива. При освобождении памяти, выделенной под массив, после оператора delete перед именем указателя на массив указывается оператор [ ]. Таким образом, для выделения памяти под массив используют синтаксис вида:

```
тип_массива  *указатель;
указатель = new тип_массива [размер];
```

Для освобождения памяти, выделенной под массив, используют команду вида:

delete [ ] указатель;

Примеры динамического выделения памяти под массивы представлены в заданиях 15, 16.

Задание 14

Цель работы: изучить применение указателей при программировании на языке C++

Назначение программы: программа выводит на экран адрес переменной num. Затем данный адрес изменяется и полученный адрес ставится в соответствие переменной iptr. Далее происходит разыменование указателя и присвоение переменной iptr конкретного значения: *iptr=10, после чего происходит сложение переменных num и iptr.

```
#include <iostream>
using namespace std;
int main()
{
int num=5;
//cout<<"Адрес переменной num ="<< &num<<endl;
cout<<" Variable address  num ="<< &num<<endl;
int* iptr=&num+3;
//cout<<"Адрес переменной  iptr= "<< iptr<<endl;
cout<<" Variable address  iptr= "<< iptr<<endl;
*iptr=10;
cout<<"num+iptr="<<num+*iptr<<endl;
return 0;
}
```

Задание 15

Цель работы: изучить методику применения одномерных динамических массивов при программировании на языке с++.

Назначение программы: программа рассчитывает значения n элементов массива и выводит их на экран. Число n вводится пользователем с клавиатуры.

```
#include <iostream>
using namespace std;
int main() {
int *p,n;
cout<<"enter n=";
cin>>n;
p=new int[n];
for(int i=0; i<n; i++){
        p[i]=2*i+1;
        cout<<p[i]<<" ";}
delete [] p;
cout<<endl;
return 0;
}
```

Задание 16

Цель работы: изучить методику применения двумерных динамических массивов при программировании на языке С++.

Назначение программы: в программе по указанным пользователем параметрам (количество строк и столбцов) создается двумерный динамический массив и заполняется целыми числами от 0 до 9.

```
#include <iostream>
using namespace std;
int main()
{
int **p;
int n,m,i,j;
//cout<<"Введите число строк   ";
```

```
cout<<" Enter number of lines   ";
cin>>n;
//cout<<"Введите число столбцов  ";
cout<<" Enter number of columns ";
cin>>m;
p=new int*[n];
for (i=0; i<n; i++){
        p[i]=new int[m];
        for(j=0; j<m; j++)
        {
                p[i] [j]=(i*m+j)%10;
                cout<<p[i] [j]<<" ";}
        cout<<endl;}
for(i=0; i<n; i++)
delete [] p[i];
delete [] p;
return 0;
}
```

6. Функции в языке программирования C++

Краткие теоретические положения

Под функцией понимают именованный программный код, который может многократно вызываться в программе. Функции реализуются отдельными программными блоками, могут иметь аргументы и возвращать значения в качестве результата (а могут и не возвращать).

Формат объявления функции в C++ имеет следующую структуру: указывается тип значения, которое возвращает функция, название функции, в круглых скобках список аргументов (с указанием типа аргументов). Программный код функции указывается в блоке из фигурных скобок:

```
тип_результата имя_функции ( тип аргумент1, тип аргумент2, …)
        {
        код функции
        }
```

Если функция не возвращает результат, в качестве типа функции указывается void. Однако данная функция может содержать инструкцию return, которая является командой завершения функции. Если у функции нет аргументов, после названия функции все равно указываются круглые скобки при объявлении функции и при ее вызове. Вызывать функцию до момента ее объявления можно только в том случае, если в начале программы указан прототип функции. Прототипом функции называется “шапка” с типом возвращаемого результата, именем функции и списком аргументов.

В C++ существует два механизма передачи аргументов функциям: по значению и через ссылку. Для того, чтобы аргумент передавался не по значению, а по ссылке, перед именем соответствующего аргумента необходимо указать оператор &, смотрите задание 21. При передаче аргумента функции по значению при вызове функции для переменных, которые указаны ее аргументами, создаются копии, которые фактически и передаются функции. После завершения выполнения кода функции эти копии уничтожаются (выгружаются из памяти). При передаче аргументов функции по ссылке функция получает непосредственный доступ (через ссылку) к переменным, указанным аргументами функции. С практической точки зрения разница между этими механизмами заключается в том, что при передаче аргументов по значению изменить передаваемые функции аргументы в теле самой функции нельзя, а при передаче аргументов по ссылке – можно.

Иногда возникает необходимость передать информацию программе при ее запуске. Это реализуется путем передачи аргументов командной строки функции main(). Аргумент командной строки –это информация, указываемая в командной строке после имени программы. Чтобы принять аргументы командной строки используются два специальных аргумента функции main( ) – argc и argv. Параметр argc содержит количество аргументов в командной строке. Его значение всегда не меньше единицы, потому что первым аргументом считается имя программы. Параметр argv представляет собой указатель на массив символьных указателей. Каждый указатель в массиве argv ссылается на строку, содержащую аргумент командной строки. Элемент argv[0] указывает на имя программы; элемент argv[1] – на первый аргумент, элемент argv[2] – на второй

аргумент и т. д. Все аргументы командной строки передаются программе как строки, поэтому числовые аргументы необходимо преобразовать в программе в соответствующий внутренний формат. Правильное объявление аргумента argv имеет вид: char *argv[ ] . Пустые квадратные скобки указывают на то, что у массива неопределенная длина. Пример передачи аргументов командной строки функции main ( ) представлен в задании 22.

Задание 17

Цель работы: изучить определение и вызов функции, создание прототипа функции при программировании на языке C++.

Назначение программы: данная программа, как и программа в задании 1, выводит сообщение на экран “Привет мир”. Однако здесь код разделен на две функции main и printMessage, которая выводит на экран: Привет мир

```
#include <iostream>
 using namespace std;
 void printMessage(void); // это прототип
 int main()

  {
      printMessage();
      return 0;
 }
 void printMessage(void)
 {
  cout << "Hello World";
  cout<<'\n';
 }
```

Задание 18

Цель работы: изучить методику передачи аргументов по значению при использовании функций в языке C++.

Назначение программы: данная программа, как и программа в задании 3, считывает строку символов с клавиатуры и выводит ее на экран. Сообщение выводится с помощью функции printMessage. Содержимое выводимого сообщения передается функции printMessage как параметр.

```
 #include <iostream>
#include <string>
 using namespace std;
 void printMessage(string); // это прототип
 int main()
  {
        string str;
        cout<<"Enter a string :      ";
        getline(cin, str);
      printMessage(str);
      return 0;
 }
 void printMessage(string s)

 {
  cout << "You inputted       "<<s;
  cout<<'\n';
 }
```

Задание 19

Цель работы: изучить методику применения функций, возвращающих значение с помощью команды return.

Назначение программы: программа рассчитывает сумму натуральных чисел от 1 до n, значение n вводится пользователем с клавиатуры.

```
#include<iostream>
using namespace std;
int msum(int n)
{
      int s=0;
```

```
        for(int i=1; i<=n; i++) s+=i;
        return s;
}
int main()
{
        int n;
        cout<<"enter n=";
        cin>>n;

        cout<<"Sum =  "<<msum(n)<<"\n";
        return 0;
}
```

Задание 20

Цель работы: изучить использование рекурсии при программировании на языке C++

Назначение программы: программа рассчитывает факториал числа. При этом в теле функции factorial() вызывается та же самая функция.

```
#include <iostream>
using namespace std;
int  factorial (int n) {
        if(n==1) return 1;
        else return n*factorial(n-1);
}

int main() {
int n;
cout<<"Enter n=";
cin>>n;
cout<<"n!="<<factorial(n)<<endl;
return 0;
}
```

Задание 21

Цель работы: изучить методику передачи аргументов по ссылке при использовании функций в языке С++

Назначение программы: функция в качестве значения возвращает целочисленную величину, на единицу превышающую аргумент функции. В первом варианте программы аргумент передается по значению. Во втором варианте аргумент передается по ссылке. Сравните результаты выполнения двух вариантов программы.

Первый вариант программы:

```
#include <iostream>
  using namespace std;
//Аргумент передается по значению
Int incr(int m) {
m=m+1;
return m;
}
Int main ( ) {
int  n=5;
cout<<”n=”<<incr(n)<<endl;
cout<<”n=”<<n<<endl;
return 0;
}
```

Результат выполнения программы :

```
n=6
n=5
```

Второй вариант программы:

```
#include <iostream>
  using namespace std;
//Аргумент передается по ссылке
Int incr(int &m) {
m=m+1;
return m;
}
Int main ( ) {
```

```
int  n=5;
cout<<”n=”<<incr(n)<<endl;
cout<<”n=”<<n<<endl;
return 0;
}
```

Результат выполнения программы:

```
n=6
n=6
```

Задание 22

Цель работы: изучить методику передачи информации функции main( ) с помощью аргументов командной строки.

Назначение программы: Программа выводит на экран слово Hello и имя пользователя, которое надо указать в виде аргумента командной строки(через пробел).

C:\Program Files\Microsoft Visual Studio \MyProjects\primer55\Debug Александр.

```
#include <iostream>
  using namespace std;
  int main(int argc, char *argv[])
   {
   cout << "Hello";
   cout<<argv[1];
   cin.get();
  return 0;
   }
```

7. Файлы в языке программирования С++

Краткие теоретические положения

В С++ файл открывается путем связывания его с потоком. Существуют потоки трех видов: ввода, вывода и ввода-вывода. Чтобы открыть входной поток, необходимо объявить потоковый объект типа ifstream. Для открытия выходного потока нужно объя-

вить поток класса ofstream. Поток, который предполагается использовать для операций как ввода, так и вывода, должен быть объявлен как объект класса fstream. Например, при выполнении следующего фрагмента кода будет создан входной поток, выходной поток и поток, позволяющий выполнение операций в обоих направлениях.

```
Ifstream infile; // входной поток
ofstream outfile; // выходной поток
fstream bothfile; // поток ввода вывода
```

Чтобы открыть файл, необходимо использовать функцию open().

Например, следующий фрагмент кода открывает файл students.dat для записи.

```
ofstream outfile;
outfile.open("students.dat");
```

При этом содержание файла переписывается.

Для того, чтобы информация в файле сохранялась и весь вывод записывался в конец файла, необходимо использовать флаг режима файла ios::app, смотрите текст программы в задании 24.

Следующий фрагмент кода открывает файл для чтения:

```
Ifstream infile;
Infile.open(“students.dat”);
```

Чтобы закрыть файл, необходимо вызвать функцию close(). Например, инструкция infile.close() закрывает файл, связанный с потоковым объектом Infile. Проще всего записывать информацию в файл с помощью оператора <<, а считывать информацию из файла с помощью оператора >>. Например, для того чтобы записать в файл строку data, необходимо использовать оператор outfile <<data. Для того, чтобы при чтении из файла в строку информация считывалась после пробела необходимо использовать функцию getline, смотрите оператор getline(infile,data) в заданиях 23, 24.

Чтобы проверить, был ли достигнут конец файла при чтении всех строк можно воспользоваться функцией fail(), смотрите зада-

ние 24. Альтернативный вариант обнаружения конца файла – использование функции eof().

Язык программирования C++ поддерживает работу не только с форматированными текстовыми файлами, но и с неформатированными двоичными файлами. Для выполнения двоичных операций файлового ввода-вывода необходимо открыть файл с использованием спецификатора режима ios::binary. Функция get( ) считывает символ из файла, а функция put( ) записывает символ в файл. Функции get( ) и put( ) имеют множество форматов, но чаще всего используются их версии:

```
istream &get(char &ch);
ostream &put(char ch);
```

Функции get считывает один символ из соответствующего потока и помещает его значение в переменную ch. Она возвращает ссылку на поток, связанный с предварительно открытым файлом. При достижении конца этого файла значение ссылки станет равным нулю. Функция put( ) записывает символ ch в поток и возвращает ссылку на этот поток. Пример посимвольной обработки файлов представлен в задании 25.

Чтобы считывать и записывать в файл блоки двоичных данных, необходимо использовать функции read( ) и write ( ). Их прототипы имеют вид.

```
istream &read(char *buf, streamsize num);
ostream &write(const  char *buf, int streamsize num);
```

Функция read( ) считывает num байт данных из связанного с файлом потока и помещает их в буфер, адресуемый параметром buf. Функция write ( ) записывает num байт данных в связанный с файлом поток из буфера, адресуемого параметром buf. Тип streamsize определен как разновидность целочисленного типа. Он позволяет хранить самое большое количество байтов, которое может быть передано в процессе любой операции ввода-вывода.

Функция gcount() возвращает количество символов, считанных при выполнении последней операции ввода данных. Пример использования функций write( ) и read( ) представлен в задании 26. Операция sizeof используется в программе для определения размера памяти, соответствующей массиву n.

Задание 23

Цель работы: изучить методику записи информации в файл и чтения строки из файла в языке C++

Назначение программы: программа предназначена для записи информации в файл и чтения информации из файла в виде двух строк. В первой строке пользователь вводит название факультета, а во второй количество студентов

```
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
int main()
{
	string data;
	ofstream outfile;
	outfile.open("students.dat");
	cout<<"Writing to the file    "<<endl;
	cout<<"==================="<<endl;
	//cout<<"Факультет ";
	cout<<" faculty   ";
	getline(cin, data);
	outfile<<data<<endl;
	//cout<<"Количество студентов   ";
	cout<<"  Number of students ";
	cin>>data;
	cin.ignore();
	outfile<<data<<endl;
	outfile.close();
	ifstream infile;
	cout<<'\n';
```

```
        cout<<"Reading from the file    "<<endl;
        cout<<"================"<<endl;
        infile.open("students.dat");
        getline(infile,data);
        cout<<data<<endl;
        getline(infile,data);
        cout<<data<<endl;
        infile.close();
        return 0;
}
```

Задание 24

Цель работы: изучить методику обнаружения конца файла при программировании на языке C++.

Назначение программы: программа аналогична программе в задании 23. Но здесь из файла считываются не только последние две строки, а вся информация, содержащаяся в текстовом файле.

```
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
int main()
{
        string data;
        ofstream outfile;
        outfile.open("students.dat", ios::app);
        cout<<"Writing to the file    "<<endl;
        cout<<"==================="<<endl;
        //cout<<"Факультет    ";
        cout<<"  faculty  ";
        getline(cin, data);
        outfile<<data<<endl;
        //cout<<"Количество студентов    ";
        cout<<"  Number of students  ";
        cin>>data;
        cin.ignore();
        outfile<<data<<endl;
```

```
	outfile.close();
	ifstream infile;
	cout<<'\n';
	cout<<"Reading from file    "<<endl;

	cout<<"================"<<endl;
	infile.open("students.dat");
	getline(infile, data);

	while(!infile.fail())
		{
	cout<<data<<endl;
	getline(infile, data);
	}
	infile.close();
	return 0;
}
```

Задание 25

Цель работы: изучить методику записи в файл и чтения информации из файла по отдельным символам с помощью функции put() и get().

Назначение программы: Программа записывает в файл строку символов.

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
	char *p = "Привет мир";
	ofstream out("test.txt", ios::out | ios::binary);
	if (!out) {
		//cout << "Не удается открыть файл\n";
		cout << " It isn't possible to open the file \n";
```

```
        cout << "  \n";
        return 1;
    }

    while(*p) out.put(*p++);
    out.close();
    return 0;
}
```

Задание 26

Цель работы: изучить методику записи в файл и считывания из файла блоков двоичных данных

Назначение программы: при выполнении следующей программы в файл записывается массив целых чисел, а затем он же считывается из файла.

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    int n[5]={1, 2, 3, 4, 5};
    register int i;
    ofstream out("test", ios::out | ios::binary);
    if(!out) {
        //cout<<"не открыт файл\n";
        cout<<" the file isn't open  \n";
        return 1;
    }

    out.write((char *) &n, sizeof n);
    out.close();
    for(i=0; i<5; i++)
        n[i]=0;
    ifstream in("test", ios::in | ios:: binary);
    if(!in) {
```

```
        //cout<<"не открыт файл\n";
        cout<<" the file isn't open \n";
        return 1;
    }

    in.read((char *) &n, sizeof n);
    for(i=0; i<5; i++)
        cout<<n[i]<<" ";
    in.close();
    return 0;
}
```

8. Структуры в языке программирования С++

Краткие теоретические положения

Под структурами понимают группу переменных, объединенных общим именем. Удобство структуры состоит в первую очередь в том, что она позволяет группировать данные разных типов, что бывает полезно при работе с базами данных. Объявление структуры начинается с ключевого слова struct, после которого следует имя структуры и в фигурных скобках перечисляются поля структуры (типы и имена переменных, входящих в структуру). Общий синтаксис объявления структуры следующий:

```
struct  имя {
тип1 поле1;
тип2 поле 2;
…….
типN полеN;
} список_переменных;
```

Например, следующая структура служит для описания оценок по трем дисциплинам: физика, химия, математика для трех студентов: Иванова, Петрова, Сидорова.

```
struct  Marks {
char name[80];
int phys;
int chem;
int maths;
} Ivanov, petrov, sidorov;
```

Чтобы обратиться к члену структуры, нужно перед его именем поставить имя структурной переменной и оператор “точка”.

Например, Ivanov.chem=4;

Структуры могут быть элементами массивов. Чтобы объявить массив структур, необходимо сначала объявить структуру, а затем объявить массив элементов этого структурного типа. Чтобы получить доступ к конкретной структуре в массиве структур, необходимо индексировать имя структуры. Подобно всем переменным массивов, у массивов структур индексирование начинается с нуля. Пример использования массива структур представлен в задании 27. Объявлен массив Person p[MAX], где MAX=3. То есть, объявлен массив структурного типа, состоящий из трех структур. Например, для того, чтобы вывести на экран значение поля name третьей структуры, необходимо использовать инструкцию:
cout<<p[2].name.

Задание 27

Цель работы: изучить объявление и использование структуры при программировании на языке C++

Назначение программы: программа записывает в структуру данные об имени и росте трех человек, затем эти данные выводятся на экран. В программе одновременно для описания объекта (человека) используются данные разных типов: имя - символьный тип данных, рост - числовой тип данных. Поэтому в данном случае целесообразно использовать структуру.

```
#include <iostream>
#include <string>
using namespace std;
const int MAX=3;
```

```
struct Person {
string name;
int height;
};
int main()
{
Person p[MAX];
for (int x=0; x<MAX; x++)
{
        cout<<"\n";
cout<<"Enter person name    ";
getline(cin, p[x].name);
cout<<"Enter height in centimetres     ";
cin>>p[x].height;
cin.ignore();
}
cout<<"\n";
// cout<<"Вывод данных о человеке \n";
cout<<"Outputing person data\n";
cout<<"====================\n";
for ( x=0; x<MAX; x++)
cout<<"Person #"<<x+1<<"s name   "<<p[x].name
<<"  and height <<p[x].height<<endl;
return 0;
}
```

9. Классы в языке программирования C++

Краткие теоретические положения

Классом называется специальный тип данных, обладающий набором переменных – полей и функций, имеющих доступ к полям – методов. Процесс объединения переменных и методов называется инкапсуляцией. Экземпляр класса называется объектом.

Объявление класса начинается с ключевого слова class, далее указывается имя класса. Поля и методы класса описываются в блоке в фигурных скобках. Поля и методы класса могут быть открытыми и закрытыми. К закрытым членам класса можно обращаться

только внутри класса, открытые члены доступны и за его пределами. Сначала перечисляются и определяются закрытые члены класса (поля и методы). Они объявляются с идентификатором доступа private (или без идентификатора доступа). Перед объявлением открытых членов указывается ключевое слово public. После закрывающих фигурных скобок объявления класса можно указать список объектов созданного класса.

Общий синтаксис объявления класса имеет вид:

```
class имя_класса {
закрытые поля и методы класса
public:
открытые поля и методы класса
} список_объектов.
```

Хотя каждый объект класса имеет одинаковый набор полей и методов, значения полей у каждого объекта свои, а результат вызова методов в общем случае зависит от того, из какого объекта метод вызывается. Однако существуют особые методы класса, которые называются статическими. Статический член является общим для всех объектов данного класса. Статические члены объявляются, как и обычные, но перед статическим членом указывается ключевое слово static.

Методы классов можно перегружать. В этом случае создается несколько вариантов одного и того же метода, но с различными прототипами. Хорошим стилем программирования считается, если разные варианты перегруженного метода объединены общей идеей. Такой подход позволяет использовать единый интерфейс и при этом учесть особенности вызова метода с разным набором аргументов. Такая концепция в объектно-ориентированном программировании получила названия полиморфизма.

В C++ классы могут наследовать друг друга. Это означает, что класс-наследник (в C++ он называется производным классом) получает свойства родительского (базового) класса. Кроме того, производный класс может добавлять к свойствам, полученным от базового класса свои собственные. Производный класс может быть базовым по отношению к другому классу. Итак, при наследовании

классов производный класс фактически создается на основе базового класса, получая в наследство от родителя поля и методы. Ранее было отмечено, что члены класса могут быть открытыми и закрытыми. Кроме того существуют защищенные члены класса, которые объявляются с идентификатором доступа protected.

Существуют определенные правила, которые определяют, какие поля и методы базового класса наследуются, а какие нет.

1) private – члены базового класса не наследуются;
2) при public - наследовании уровень доступа члена класса не меняется (если этот член наследуется)
3) при private –наследовании наследуемые члены становятся private-членами производного класса.
4) При protected – наследовании наследуемые члены становятся protected – членами производного класса.

Представим в виде таблицы результаты 9 возможных вариантов наследования.

Важной задачей объектно – ориентированного программирования является определение базового набора операций, которые необходимо выполнить при создании объектов. Такая задача реализуется через механизм конструкторов и деструкторов. Конструктор – метод, который автоматически вызывается при создании объекта. Деструктор – метод, вызываемый автоматически при выгрузке объекта из памяти.

Таблица1. Варианты наследования классов

Механизм наследования / Тип члена	Public-наследование	Private-наследование	Protected-наследование
Public -член	public	private	protected
Private-член	Не наследуется	Не наследуется	Не наследуется
Protected-член	protected	private	protected

Существуют конструкторы и деструкторы по умолчанию, однако их можно определять и в явном виде.

Конструктор в классе создается как обычный метод с учетом двух принципиальных особенностей:

1. Имя конструктора совпадает с именем класса.
2. Конструктор не возвращает результат и для него тип результата не указывается.

Обычный метод необходимо вызывать в явном виде, а конструктор вызывается автоматически и только при создании объекта. Конструктор может иметь аргументы, а может и не иметь. Конструктор можно перегружать.

Деструкторы, как и конструкторы, можно явно описывать при создании класса. Правила создания деструкторов следующие:

1) Имя деструктора совпадает с именем класса, но перед именем деструктора указывается символ “тильда” ~ .
2) Тип результата для деструктора не указывается (как и для конструктора)
3) У деструктора нет аргументов.

Деструктор не может быть перегружен.

Виртуальная функция – это функция, которая объявляется в базовом классе с использованием ключевого слова virtual и переопределяется в одном или нескольких производных классах. При переопределении виртуальной функции в производном классе ключевое слово virtual повторять не нужно. Каждый производный класс может иметь собственную версию виртуальной функции. Класс, который включает виртуальную функцию, называется полиморфным классом. Переопределение виртуальной функции в производном классе отличается от перегрузки функций тем, что тип и количество параметров для версий переопределенной функции, в отличии от версий перегруженной функции должны в точности совпадать. Атрибут virtual передается по наследству. Если функция объявляется как виртуальная, она остается такой независимо от того, через сколько уровней производных классов она может пройти. Если производный класс не переопределяет виртуальную функцию, то используется функция в ближайшем по иерархии базовом классе. Виртуальные функции позволяют производным

классам использовать множество методов, но в то же время использовать единый интерфейс, определяемый базовым методом. Такой интерфейс позволяет программисту справляться с более высокой сложностью программ. Ко всем объектам, выведенным из базового класса можно получать доступ единым способом, несмотря на то, что действия производных классов могут различаться между собой.

Чисто виртуальная функция – это виртуальная функция, объявленная в базовом классе, но не имеющая в нем никакого определения. Поэтому любой производный тип должен определить собственную версию этой функции, потому что у него нет возможности использовать версию из базового класса (по причине ее отсутствия). Чтобы объявить чисто виртуальную функцию, необходимо использовать следующий общий формат:

Virtual тип имя_функции (список_параметров)=0;

Здесь под элементом тип подразумевается тип значения, возвращаемого функцией, а элемент имя_функции – ее имя. Обозначение =0 является признаком того, функция объявляется как чисто виртуальная.

Класс, который содержит хотя бы одну чисто виртуальную функцию, называется абстрактным классом. У абстрактного класса не может быть объектов, потому что его функция не имеет определения. Абстрактный класс можно использовать только в качестве базового, из которого будут выводиться другие классы.

Задание 28

Цель работы: изучить методику объявления и применения классов при программировании на языке C++

Назначение программы: в классе SimpleClass имеется 2 целочисленных поля n и m, а также три метода : summa( ) для вычисления суммы полей объекта, show( ) для отображения значений полей объекта и mult( ) для умножения полей объекта на число.

```
#include<iostream>
using namespace std;
```

```
class SimpleClass{
public:
	int m;
	int n;
	int summa() {
		int k=n+m;
		return k;
	}
	void show() {
		cout<<"m="<<m<<endl;
		cout<<"n="<<n<<endl;
	}
	void mult(int k){
		n*=k;
		m*=k;
	}
};
int main() {
	SimpleClass MyObj1, MyObj2;
	MyObj1.m=1;
	MyObj1.n=2;
	MyObj2.m=8;
	MyObj2.n=9;
	cout<<"Total value for MyObj1
is"<<MyObj1.summa()<<endl;
	cout<<"Total value for MyObj2
is"<<MyObj2.summa()<<endl;
	MyObj1.mult(3);
	MyObj2.mult(2);
	MyObj1.show();
	MyObj2.show();
	return 0;
}
```

Задача 29

Цель работы: изучить методику использования закрытых членов класса при программировании на языке C++

Назначение программы: полям n и m класса SimpleClass присваиваются значения n=2 и m=1 и затем данная информация выводится на экран. Однако поля n и m объявлены как закрытые, так как они объявлены до блока public. Поэтому для заполнения полей используется не главный метод main( ), а метод setnm( ), который специально объявлен, как открытый.

```
#include<iostream>
using namespace std;
class SimpleClass{
        int m;
        int n;
public:
        void show();
        void setnm(int i,int j);
};
int main() {
        SimpleClass obj;
        obj.setnm(1,2);
        obj.show();
        return 0;}
void SimpleClass::show(){
        cout<<"m="<<m<<endl;
        cout<<"n="<<n<<endl;}
void SimpleClass::setnm(int i, int j) {
        m=i;
        n=j;
}
```

Задание 30

Цель работы: изучить методику перегрузки методов (полиморфизма) при программировании на языке C++.

Назначение программы: в программе создан класс MyClass, в котором 2 закрытых целочисленных поля a и b , метод getab( ) для

отображения значений закрытых полей и перегруженный метод setab( ) для записи значений в закрытые поля. У метода setab ( ) два варианта: с одним целочисленным аргументом и с двумя аргументами. Если методу передано два аргумента, то эти аргументы присваиваются в качестве значений полям a и b. Если у метода один аргумент, то соответствующее значение присваивается каждому полю a и b.

```
class MyClass{
        int a,b;
public:
        void setab(int i,int j)
        {
        a=i;
        b=j;
        }
        void setab(int i)
        {
        a=i;
        b=i;
        }
        void getab()
        {
        cout <<"a=  "<<a<<endl;
        cout<<"b=  "<<b<<endl;

        }
        }
        obj1,obj2;
        int main()
        {
        obj1.setab(1,2);
        obj2.setab(3);
        obj1.getab();
        obj2.getab();
        return 0;
        }
```

Результат выполнения программы имеет вид:
a=1
b=2
a=3
b=3

Задание 31

Цель работы: изучить методику применения статических методов класса при программировании на языке C++ .

Назначение программы: В программе используется статический метод msum(), которым к статическому полю m класса прибавляется число, указанное аргументом функции. С помощью команды Simple Class::msum(10), которой статическое поле m получает значение 10 (начальное значение по умолчанию равно 0, и оно увеличивается на 10). В этой команде статический метод вызывается через ссылку на класс SimpleClass. Статический метод можно вызывать и через объект класса, например, в команде bj1.msum(90). В результате выполнения программы m=100.

```
#include<iostream>
using namespace std;
class SimpleClass{
public:
static int m;
int n;
void show();
static void msum(int k);
}obj1, obj2;
int SimpleClass::m;
int main(){
        SimpleClass::msum(10);
        obj1.n=1;
        obj2.n=2;
        obj1.show();
        obj2.show();
    obj1.msum(90);
```

```
obj2.show();
return 0; }
void SimpleClass::show()
{
        cout<<"Static field m="<<m<<endl;
        cout<<"Nonstatic field n="<<n<<endl;}
void SimpleClass::msum(int k) {
        m+=k;}
```

Задание 32

Цель работы: изучить методику наследования классов при программировании на языке С++.

Назначение программы: в базовом классе А объявлено 2 поля: поле x закрытое и поле y открытое. Класс В является производным от класса А и содержит описание поля z и открытого метода show ( ). Но объекты класса В получают в наследство от класса А поле y. Поэтому при описании метода show( ) допустимой является ссылка на поле y. Поле x не может наследоваться, потому что оно объявлено как закрытое. В результате выполнения программы получим

```
y=1
z=2
```

```
#include <iostream>
using namespace std;
class A
{
private:
int x;
public:
int y;
};
class B: public A{
public:
        int  z;
```

```
        void show()
        {
                cout<<"y=  "<<y<<endl;
                cout<<"z=  "<<z<<endl;
        }
};

int main()
{
        B obj;
        obj.y=1;
        obj.z=2;
        obj.show();
        return 0;
}
```

Задание 33

Цель работы: Изучить методику использования конструкторов и деструкторов при программировании на языке C++

Назначение программы: в классе MyClass предусмотрен конструктор, которым присваиваются нулевые значения полям объекта и выводится сообщение о создании объекта, а также деструктор, которым выводится сообщение об удалении объекта.

```
#include <iostream>
using namespace std;
class MyClass{
public:
        int m,n;
        MyClass(){
        m=0;
        n=0;
        cout<<"Object has been created"<<endl;
        }
        ~MyClass(){
                cout<<"Object has been deleted"<<endl;}
```

```
};
int main() {
	MyClass obj;
	return 0;
}
```

Задание 34

Цель работы: изучить методику применения виртуальных методов при программировании на языке с++.

Назначение программы: в классе А используются виртуальные методы set( ) и get( ). Методом set () задаются значения полей, а методом get() значения полей отображаются. В классе В, производном от класса А, методы set( ) и get( ) переопределяются. Командой a.get() вызывается метод, описанный в классе А, командой b.get() вызывается метод, описанный в классе В. Метод set() не только переопределяется, но и перегружается. Класс В содержит описание двух вариантов метода set() – с одним аргументом и двумя аргументами. Когда методу передается 2 аргумента, они определяют значения полей x и y. При передаче методу одного аргумента поля x и y получают одинаковые значения.

```
#include <iostream>
using namespace std;
class A{
protected:
int x;
public:
	virtual void set(int i) {x=i;}
	virtual void get() {
		cout<<"x="<<x<<endl;}
};
class B: public A{
private:
int y;
public:
	void set (int i) {
```

```
            x=i; y=i;}
        void set(int i, int j) {
            x=i;
            y=j;}
        void get() {
        cout<<"x="<<x<<endl;
        cout<<"y="<<y<<endl;}
};

int main() {
        A a;
        B b;
        a.set(1);
        a.get();
        b.set(2);
        b.get();
        b.set(3,4);
        b.get();
        return 0;}
```

Задание 35

Цель работы: изучить методику применения абстрактных классов при программировании на языке C++.

Назначение программы: в программе создается абстрактный класс Figure с числовым полем R (линейный размер геометрической фигуры), конструктором, которым полю R присваивается единичное значение и чисто виртуальным методом area( ) для вычисления площади фигуры. На основе этого абстрактного класса создается 3 производных класса, в каждом из которых переопределяется метод area( ). В классе Circle метод area( ) переопределяется для вычисления площади круга, в классе Square – квадрата, в классе Triangle – равностороннего треугольника. Результат выполнения программы:

Circle: 3.1415
Square: 1
Triangle: 0.433013

```
include <iostream>
#include <cmath>
using namespace std;
const double pi=3.1415;
class Figure{
public:
	double R;
	Figure () {R=1;}
	virtual double area()=0;
};

class Circle:public Figure{
public:
	double area()
	{return pi*R*R;}
};


class Sguare:public Figure{
public:
	double area()
	{return R*R;}
};
class Triangle:public Figure{
public:
	double area()
	{return sqrt(3)*R*R/4;}
};
int main()
{
	Circle A;
	Sguare B;
	Triangle C;
	cout<<"Circle: "<<A.area()<<endl;
cout<<"Sguare: "<<B.area()<<endl;
cout<<"Triangle: "<<C.area()<<endl;
return 0;
```

10. Задания для самостоятельной работы

Задача №1

Расчитать сопротивление R участка A B электрической цепи, см. рис. 1

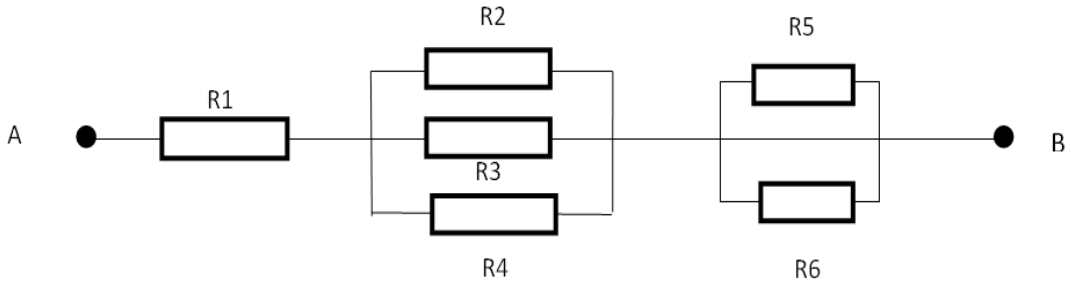

Рис. 1 Схема электрической цепи

Задача №2

Определить, принадлежит ли точка B с координатами (x;y) заштрихованной фигуре, см. рис. 2. Значения координат x и y вводятся с клавиатуры.

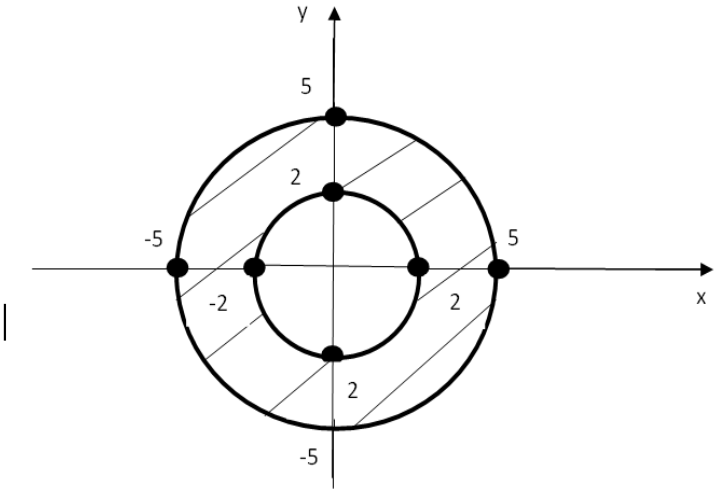

Рис 2. Геометрическая фигура

Задание №3

Составить программу с помощью оператора switch, которая выводит информацию об оценках на экзамене по информатике и математике для одного из четырех студентов.

1) №1 – Иванов
2) №2- Петров
3) №3- Сидоров
4) №4 –Кольцов

Программа должна выдавать пользователю сообщение “Введи номер студента” и по введенному номеру выдавать сообщение об оценках студента, полученных на экзамене.

Задание №4

Составить программу с помощью оператора While, которая находит предельное максимальное значение x, при котором выполняется неравенство:

$x^3+5x^2+10x+50 < n$,

число n – вводится с клавиатуры.

Задание №5

Рассчитать значение Y при x=1..10,1 в цикле, используя операторы for и if. Число a вводится с клавиатуры.

$$Y=\begin{cases} a\cdot x^2+5/x+\cos(a\cdot x) & \text{при } a>10 \\ a^2+2\cdot x+5, & \text{при } a<10 \\ 5\cdot a\cdot x^3+4\cdot x/a+7, & \text{при } a=10 \end{cases}$$

Задача №6

Найти сумму n членов ряда, общий член которого определяется формулой:

$$a_n = n^2 \cdot e^{-\sqrt{n}}$$

Задача №7

Найти сумму ряда с точностью $\varepsilon=10^{-2}$, общий член которого определяется формулой: $a_n = \frac{(2 \cdot n - 1)}{2^n}$.

Указания к решению:

Для решения задачи надо взять отношение $\frac{a_{n+1}}{a_n}$ и отсюда вывести рекуррентную функцию $a_{n+1} = f(a_n)$. Сумму ряда считать до тех пор, пока выполняется неравенство: $a_n > \varepsilon$.

Задача №8.

Сгенерировать массив длиной 10 элементов с значениями элементов от 15 до 30 и отсортировать его по убыванию.

Задача 9

Организовать ввод с клавиатуры одномерного динамического массива произвольной длины. Найти минимальный элемент массива и количество нулевых элементов массива.

Задача №10

Составить программу, которая рассчитывает определитель матрицы рамером 3 х 3.

Задача 11.

Даны две матрицы A и B размером 3 х 2. Рассчитать и вывести на экран матрицу C = 2 · A - B

Задача 12

Составить программу, которая рассчитывает корни двух квадратных уравнений: $x^2 + 6 \cdot x + 5 = 0$; $6x^2 - 10 \cdot x + 4 = 0$.

При этом решение квадратного уравнения должно быть реализовано в виде отдельной функции kwadrat . Данной функции должны передаваться в качестве значений аргументов коэффициенты квадратных уравнений.

Задача 13

Написать программу с функцией для вычисления числа из последовательности Фибоначчи. Аргументом функции является порядковый номер числа в последовательности.

Задача 14

Составить программу, которая записывает в файл 3 строки: автора книги, название книги, шифр книги. Информация должна добавляться в конец файла. Затем информация выводится из файла на экран.

Задача 15

Составить программу, которая сравнивает информацию, содержащуюся в двух файлах и выдает сообщение о сходстве или различии файлов.

Задача 16

Составить программу, которая рассчитывает средний балл для каждого из трех студентов по трем дисциплинам: информатика, математика, электротехника. Исходные данные представить в виде структуры student.

Переменные экземпляра структуры – 3 студента: iwanow, petrow, sidorow. Поля структуры соответствуют трем дисциплинам: inform, matem, electr. Значения полей, то есть оценки по данным дисциплинам вводятся с клавиатуры.

Задача 17

Написать программу для вычисления суммы векторов в декартовом пространстве. Векторы реализовать в виде структуры.

Задача 18

Составить путем создания специального класса программу для определения модуля комплексного числа x+yi, где модуль z рассчитывается по формуле $Z = \sqrt{x^2 + y^2}$.

Поля класса x и y, специальный метод рассчитывает модуль z. В качестве объектов класса взять 3 комплексных числа: 5+7i, 12+3i, 16+9i и рассчитать для них значение модуля z.

Задача 19

Составить программу для вычисления двойного факториала числа n!!=n · (n-2) · (n-4) … Использовать специальный класс, полем которого является число n, а двойной факториал вычисляется методом класса.

Задача 20

Написать, путем создания специального класса, программу для решения уравнения вида: a · sin(x) + b · cos(x) = c

Список контрольных вопросов

1) Чем принципиально отличаются процедурное и объектно-ориентированное программирование?
2) Из каких блоков состоит программа, написанная в C++.
3) Какие основные типы данных используются в C++.
4) Чем константа отличается от переменной? Как объявляется константа?
5) Какие арифметические операторы используются в C++.
6) Что такое оператор инкремента и декремента?
7) Какие логические операторы и операторы сравнения используются в C++?
8) Что такое автоматическое приведение типов? Как и когда оно выполняется? Как в C++ выполняется явное приведение типов?
9) Когда и как используются условный оператор if( )? Каковы его особенности?
10) В чем особенности условного оператора switch( )? Когда и как он используется?
11) Когда и как используется оператор цикла for( )? В каких случаях он используется и каковы его особенности?
12) Когда и как используется оператор цикла while( )? В каких случаях он используется и каковы его особенности?

13) В чем особенности оператора цикла do-while( ) по сравнению с оператором цикла while ( )?

14) Что такое инструкция безусловного перехода? Каков принцип её использования?

15) Что такое указатель? Как указатель объявляется и как используется? Какие операции допустимы с указателями?

16) Что такое ссылка? Чем она отличается от указателя?

17) Что такое массив? Как индексируются элементы массива? Выполняется ли в C++ проверка выхода за пределы массива?

18) Как объявляется одномерный массив? Как выполняется обращение к элементам массива?

19) Как объявляются двумерные массивы? Как выполняется обращение к элементам двумерного массива?

20) Как выполняется инициализация массивов?

21) Как имя массива связано с указателем на первый его элемент?

22) В чем особенность массивов, состоящих из символов?

23) Что такое функция? Как она объявляется?

24) Каким образом функция может возвращать результат?

25) Как функции передаются аргументы? Что такое передача аргумента по значению и по ссылке? Чем эти способы отличаются и как реализуются?

26) Какие аргументы у главного метода программы–функции main ( )?

27) Что такое рекурсия и в каких случаях она используется?

28) Каким образом функции в качестве аргумента передаются массивы?

29) Что такое аргументы по умолчанию и как они определяются?

30) Каким образом в C++ реализуются текстовые строки?

31) Что такое динамическое выделение памяти и в каких случаях к нему прибегают? Как динамически выделяется память под переменную?

32) Зачем нужно освобождать неиспользуемую динамически выделенную память и как это делается?

33) Как динамически выделяется память для массива?

34) Каким образом выполняется чтение и запись текстовых файлов?

35) Каким образом выполняются двоичные операции файлового ввода-вывода?

36) Каким образом осуществляется считывание и запись в файл блоков данных?

37) Каким образом в программе можно обнаружить конец файла?

38) Что такое структура и как она определяется?

39) Что такое поля структуры и как к ним выполняется обращение?

40) Чем переменная (экземпляр) структуры отличается от непосредственно структуры?

41) Каким образом создаются массивы структур?

42) Что такое класс и объект? Чем класс отличается от объекта?

43) Как описывается класс и как создаются объекты?

44) Что такое поля и методы класса? Как они объявляются?

45) Чем закрытые члены класса отличаются от открытых?

46) Что такое статическое поле? В чем его особенности?

47) Что такое перегрузка методов и как она выполняется?

48) Что такое конструктор? Когда он вызывается?

49) Что такое деструктор? Как и когда он вызывается?

50) Как создаются конструкторы?

51) Как создаются деструкторы?

52) Что такое наследование классов и в чем оно состоит?

53) Какие существуют типы наследования?

54) Что такое виртуальные методы? Как происходит переопределение виртуальных методов?

55) Что такое многоуровневое наследование?

56) Что такое многократное наследование?

57) Что такое чисто виртуальный метод?

58) Что такое абстрактный класс?

Список литературы

1. Шилдт, Герберт. C++: базовый курс, 3-е издание.: Пер. с англ. – М.: Издательский дом “ Вильямс”, 2012. – 624 с.: ил.

2. Васильев А. Н. Самоучитель C++ с примерами и задачами. Спб.: Наука и Техника, 2010. – 480 с.: ил.

3. Кент, Д. C++. Основы программирования / Джефф Кент; пер. с англ. Ю. В. Кирпичев. – М.: НТ Пресс, 2008. -366 с.: ил.

4. Павловская Т. А. C++. Объектно-ориентированное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. - СПб. : Питер, 2004. - 265 с. - ISBN 5-94723-842-X :
Гриф: Допущено Министерством образования РФ.

5. C++ / Б. Страуструн. - М.: Бином-Пресс, 2004. - 1104 с.

Павловская Т. А. C/C++. Программирование на языке высокого уровня [Текст] : учебник / Т. А. Павловская. - СПб. : Питер, 2006. - 461 с. : ил. - (Учебник для вузов). - ISBN 5-94723-568-4 :
Гриф: Допущено Министерством образования РФ

6. Павловская Т. А. C/C ++. Структурное программирование [Текст] : практикум: учебное пособие / Т. А. Павловская, Ю. А. Щюпак. - СПб. : Питер, 2007. - 239 с. : ил. - ISBN 5-94723-967-1 :
Гриф: Допущено Министерством образования РФ.

Приложение 1

Порядок работы с программой на языке программирования С++ в пакете Microsoft Visual Studio 6.0.

1 . Выберите команду File – New.

2. Укажите имя проекта, например Primer 30.

3. Выберете команды Win 32 Console Application, ok, Finish, ok

4. Добавьте к проекту файл с расширением cpp, для этого указываем имя файла: например primer 30, далее выбираем С++ Source File, затем ok.

5. Наберите текст программы, затем компилируйте программу:

Compile primer 30

Команда Build компилирует изменения в программе с момента последней компиляции.

Программа должна выдать сообщение об отсутствии ошибок

0 error(s), 0 warning(s)

6. Затем запустите программу на исполнение:

! Execute primer 30.exe

Программа автоматически записывается по следующему пути:

C:\ Programm Files\ Microsoft Visual Studio\ MyProjects.

Приложение 2

Порядок работы с программой на языке программирования С++ в пакете Embarcadero RAD Studio 2010

1. Запустите Embarcadero RAD Studio.
2. Выберите команду New Project –Console Application

3. Далее выбираем переключатель C++ , устанавливаем флажок на Console Application, снимаем флажок с Multi Threaded. Откроется окно редактирования программы нового проекта.

4. Имя проекта указывается автоматически в начале создания проекта, например Project1, номер после Project указывается автоматически в соответствии с последовательностью создания проектов, однако при сохранении проекта имя проекта можно откорректировать.

5. Удалите шаблон:

```
//---------------------------------------------------------------------

#pragma hdrstop

#include <tchar.h>
//---------------------------------------------------------------------

#pragma argsused
int _tmain(int argc, _TCHAR* argv[])
{
        return 0;
}
//---------------------------------------------------------------------
```

6. Наберите текст программы. При необходимости в конце текста программы перед командой return 0 добавьте команду cin.get(), предназначенную для задержки экрана командной строки.

7. Выполните компиляцию программы.
Зеленая стрелочка на панели, означающая команду Run, автоматически компилирует изменения в программе с момента последней компиляции и запускает проект. Программа должна выдать сообщение об отсутствии ошибок

0 error(s), 0 warning(s)

Затем программа автоматически запускается на исполнение.

8. Программа автоматически записывается по следующему пути:

C:\Users\.....\Documents\RAD Studio\Projects