В. С. Романчик, А. Е. Люлькин

C++
ЛАБОРАТОРНЫЕ РАБОТЫ
по курсу «МЕТОДЫ ПРОГРАММИРОВАНИЯ»

Учебно-методическое пособие
для студентов механико-математического факультета

МИНСК
БГУ
2005

УДК 681.142.2(072)
ББК 32.973.26-018.1я73
Р69

А в т о р ы :

В. С. Романчик, А. Е. Люлькин

Р е ц е н з е н т ы:
кандидат физико-математических наук, доцент *Галкин И. М.,*
кандидат физико-математических наук, доцент, *Суздаль С. В.*

Рекомендовано Ученым советом механико-математического факультета БГУ
29 марта 2005 года, протокол № ___

В пособии рассматриваются шесть лабораторных работ, выполняемых студентами 2-го курса при изучении вопросов, связанных с программированием на языке C++ . Эти вопросы являются составной частью курса “Методы программирования”, изучаемого студентами 1-го – 2-го курсов механико-математического факультета. Для выполнения каждого задания отводится 2-3 недели. Для выполнения всех заданий отводится 14 недель.

УДК 681.142.2(072)
ББК 32.973.26-018.1я73

Лабораторная работа № 1
Тема. Простейшие классы и объекты

Теоретическое введение. Классы представляют абстрактные типы данных с открытым интерфейсом и скрытой внутренней реализацией. В классах реализованы базовые принципы *объектно-ориентированного программирования* (ООП):

1) абстракция данных;

2) инкапсуляция – в классах объединяются данные и методы (функции) для работы с ними, так как лишь через методы возможен доступ к сокрытым данным класса;

3) наследование – в производных классах наследуются члены базового класса;

4) полиморфизм – возможность использования одних и тех же методов для работы с различными объектами базового и порожденных им классов.

Определение простейшего класса без наследования имеет вид:

```
class имя_класса {
   // по умолчанию раздел private – частные члены класса
   public:  // открытые функции и переменные класса
    };
```

Имя класса является новым типом данных. Понятию переменной данного типа соответствует понятие объекта класса. Список членов класса включает описание данных и функций. Функции, описания которых находятся в определении класса, называются функциями-членами класса.

Переменные и функции, объявленные в разделе класса по умолчанию или явно как ***private,*** имеют область видимости в пределах класса. Их можно сделать видимыми вне класса, если объявить в разделе ***public:***. Обычно переменные объявляются в разделе ***private***, а функции в разделе ***public***.

Классами в C++ являются также структуры ***(struct*)** и объединения ***(union*)**. В отличие от класса члены структуры по умолчанию являются открытыми, а не закрытыми. Кроме того, объединения не могут наследоваться и наследовать.

При реализации функциональной части класса могут быть использованы функции-члены класса, конструкторы, деструкторы, функции-операторы. Функции класса всегда объявляются внутри класса. Определение функции может находиться и внутри класса. Такие функции называются ***inline***-функциями. Обычно определения

функций-членов класса помещаются вне класса. При этом перед именем функции указывается ***имя_класса::*** .

тип имя_класса:: имя_функции (описание аргументов)

{ /*тело функции*/ }

Вызов функций осуществляется одним из двух способов:

имя_объекта.имя_функции(аргументы);

указатель_на_объект -> имя_функции(аргументы);

Обращение к данным объекта класса осуществляется с помощью функций, вызываемых из объектов. При этом функции-члену класса передается скрытый указатель ***this*** на объект, вызывающий функцию.

Функции-«друзья» класса, объявляемые в классе со спецификатором ***friend***, указатель ***this*** не содержат. Объекты, с которыми работают такие функции, должны передаваться в качестве их аргументов. Это обычные функции языка C++, которым разрешен доступ к закрытым членам класса.

Пример.

```
/* Создается класс Student. Формируется динамический массив объектов. При тестировании  выводится: сформированный список студентов, список студентов заданного факультета, список студентов для заданных факультета и курса.*/
#include <conio.h>
#include <string.h>
#include <iostream.h>
struct date // дата рождения
  {char daymon[6];
   int year;  };
//=======     class Student    ==================
   class Student{
   char name[30]; //private
   date t;
   char adr[30], fac[20];
   int kurs;
     public:
       Student();
       char *getfac();
       int getkurs();
       void show();
        };
Student::Student()
       {cout<<"Input name:"; cin>>name;
       cout<<"Input date of born\n";
       cout<<"Day.mon:";    cin>>t.daymon;
       cout<<"Year:";       cin>>t.year;
       cout<<"Input adr:";  cin>>adr;
       cout<<"Input fac:";  cin>>fac;
```

```
      cout<<"Input kurs:"; cin>>kurs;
}
void Student::show()
   {
    cout<<"Name      :"<<name<<endl;
    cout<<"Was born  :"<<t.daymon<<'.'<<t.year<<endl;
    cout<<"Address    :"<<adr<<endl;
    cout<<"Fac       :"<<fac<<endl;
    cout<<"Kurs      :"<<kurs<<endl;
    }
char *Student::getfac() { return fac; }
int Student::getkurs() { return kurs;  }
void spisfac(Student spis[],int n)//список студентов заданного факультетата
    {char fac[20];
     cout<<"Input faculty:";  cin>>fac;
     for(int i=0;i<n;i++)
         if(strcmp(spis[i].getfac(),fac)==0)spis[i].show();
    }
void spisfackurs(Student spis[],int n)
//список студентов заданных факультета и курса
    {int i,k;
     char fac[20];
     cout<<"Input faculty:";     cin>>fac;
     cout<<"Input the course:";  cin>>k;
     for(i=0;i<n;i++)
     if ((strcmp(spis[i].getfac(),fac)==0)&&(spis[i].getkurs()==k))
            spis[i].show();
    }
//=========    main   ================
 void main()
{    Student *spis;
    int n;
    cout<<"Input a number of students: "; cin>>n;
    spis=new Student [n];
    for(int i=0;i<n;i++) {
      cout<<"============================="<<endl;
      spis[i].show();
      }
spisfac(spis,n);
spisfackurs(spis,n);
delete [] spis;
cout<<"press any key!"
while(!kbhit());
}
```

Задания для самостоятельного решения

Разработать классы для описанных ниже объектов. Включить в класс методы set (…), get (…), show (…). Определить другие методы.

1. **Student**: Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс. Создать массив объектов. Вывести:

а) список студентов заданного факультета;

б) списки студентов для каждого факультета и курса;

в) список студентов, родившихся после заданного года.

2. **Abiturient**: Фамилия, Имя, Отчество, Адрес, Оценки. Создать массив объектов. Вывести:

а) список абитуриентов, имеющих неудовлетворительные оценки;

б) список абитуриентов, сумма баллов у которых не меньше заданной;

в) выбрать N абитуриентов, имеющих самую высокую сумму баллов, и список абитуриентов, имеющих полупроходной балл.

3. **Aeroflot**: Пункт назначения, Номер рейса, Тип самолета, Время вылета, Дни недели. Создать массив объектов. Вывести:

а) список рейсов для заданного пункта назначения;

б) список рейсов для заданного дня недели;

в) список рейсов для заданного дня недели, время вылета для которых больше заданного.

4. **Book**: Автор, Название, Издательство, Год, Количество страниц. Создать массив объектов. Вывести:

а) список книг заданного автора;

б) список книг, выпущенных заданным издательством;

в) список книг, выпущенных после заданного года.

5. **Worker**: Фамилия и инициалы, Должность, Год поступления на работу, Зарплата. Создать массив объектов. Вывести:

а) список работников, стаж работы которых на данном предприятии превышает заданное число лет;

б) список работников, зарплата которых больше заданной;

в) список работников, занимающих заданную должность.

6. **Train**: Пункт назначения, Номер поезда, Время отправления, Число общих мест, Купейных, Плацкартных. Создать массив объектов. Вывести:

а) список поездов, следующих до заданного пункта назначения;

б) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;

в) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

7. **Product**: Наименование, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести:

а) список товаров для заданного наименования;

б) список товаров для заданного наименования, цена которых не превышает указанной;

в) список товаров, срок хранения которых больше заданного.

8. **Patient**: Фамилия, Имя, Отчество, Адрес, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести:

а) список пациентов, имеющих данный диагноз;

б) список пациентов, номер медицинской карты которых находится в заданном интервале.

9. **Bus**: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег. Создать массив объектов. Вывести:

а) список автобусов для заданного номера маршрута;

б) список автобусов, которые эксплуатируются больше 10 лет;

в) список автобусов, пробег у которых больше 10 000 км.

10. **Customer**: Фамилия, Имя, Отчество, Адрес, Телефон, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести:

а) список покупателей в алфавитном порядке;

б) список покупателей, номер кредитной карточки которых находится в заданном интервале.

11. **File**: Имя файла, Размер, Дата создания, Количество обращений. Создать массив объектов. Вывести:

а) список файлов, упорядоченный в алфавитном порядке;

б) список файлов, размер которых превышает заданный;

в) список файлов, число обращений к которым превышает заданное.

12. **Word**: Слово, Номера страниц, на которых слово встречается (от 1 до 10), Число страниц. Создать массив объектов. Вывести:

а) слова, которые встречаются более чем на N страницах;

б) слова в алфавитном порядке;

в) для заданного слова номера страниц, на которых оно встречается.

13. **House**: Адрес, Этаж, Количество комнат, Площадь. Создать массив объектов. Вывести:

а) список квартир, имеющих заданное число комнат;

б) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в определенном промежутке;

в) список квартир, имеющих площадь, превосходящую заданную.

14. **Phone**: Фамилия, Имя, Отчество, Адрес, Номер, Время внутригородских разговоров, Время междугородних разговоров. Создать массив объектов. Вывести:

а) сведения об абонентах, время внутригородских разговоров которых превышает заданное;

б) сведения об абонентах, воспользовавшихся междугородней связью;

в) сведения об абонентах, выведенные в алфавитном порядке.

15. **Person**: Фамилия, Имя, Отчество, Адрес, Пол, Образование, Год рождения. Создать массив объектов. Вывести:

а) список граждан, возраст которых превышает заданный;

б) список граждан с высшим образованием;

в) список граждан мужского пола.

Тесты

1. Для чего нужны классы?

Варианты ответа:*

*1) для определения новых типов в программе; 2) для упрощения работы с функциями; *3) для соединения данных и операций над ними; 4) для упрощения работы с указателями.

2. Методы класса определяют:

*1) какие операции можно выполнять с объектами данного класса; 2) какие типы значений могут принимать данные-члены класса; 3) каким образом реализуется защита данных-членов класса.

3. Атрибуты (данные-члены) класса могут быть:

1) только целыми числами; 2) любыми встроенными типами; *3) любого определенного в программе типа.

4. Какая из записей соответствует обращению к атрибуту arg класса A в определении метода этого же класса?

Варианты ответа:

*1) this –> arg; *2) arg; 3) A –> arg; 4) A –> this –> arg.

5. Если имеется код

```
class A {public: int a; }; A *obj;
```

то как обратиться к переменной a?

Варианты ответа:

1) obj.a; 2) (*obj) –> a; *3) obj –> a; 5) obj :: a.

6. Если имеется код

```
class A {public: int a, b, c;}; A *obj;
```

то как обратиться к переменной b?

Варианты ответа:

1) (*obj) –> b; 2) A :: b; *3) (*obj).b; 4) obj –> a.b.

---

* Звездочкой в тестах отмечены правильные ответы.

7. Каков будет результат выполнения следующего кода:

```
class A {
public:
int inc (int x) {return x++;}
int inc (short x) (return x+2;}
};
A obj; int y=5;
cout << obj.inc(y); ?
```

Варианты ответа:

*1) 5; 2) 6; 3) 7; 4) ошибка при компиляции.

8. Каков будет результат выполнения следующего кода:

```
class A {
public:
int y;
int inc (int x) {return y++;}
int inc (short x) {return x+y;}
};
A obj; int y=5; obj.y= 6;
cout << obj.inc(y); ?
```

Варианты ответа:

1) 5; *2) 6; 3) 11; 4) 7; 5) ошибка при компиляции.

9. Какими по умолчанию объявляются элементы структуры?

Варианты ответа:

1) private; *2) public; 3) protected; 4) по умолчанию не объявляются.

Лабораторная работа № 2
Тема. Разработка классов

Теоретическое введение. При разработке класса необходимо определить данные класса и его методы, конструкторы и деструкторы. Конструктор – это функция-член класса, которая вызывается автоматически при создании статического или динамического объекта класса. Он инициализирует объект и переменные класса. У конструктора нет возвращаемого значения, но он может иметь аргументы и быть перегружаемым.

Противоположные конструктору действия выполняет деструктор, который вызывается автоматически при уничтожении объекта. Деструктор имеет то же имя, что и класс, но перед ним стоит ‘~’. Деструктор можно вызывать явно в отличие от конструктора. Конструкторы и деструкторы не наследуются, хотя производный класс может вызывать конструктор базового класса.

Операторы-функции. Используются для введения операций над объектами, связываемых с символами:

+ , - , * , / , % , ^ , & , | , ~ , ! , = , < , > ,+= , [] , -> , ( ) , new, delete.

Оператор-функция является членом класса или дружественной (***friend***) классу. Общая форма оператор-функции-члена класса:

***возвращаемый_тип имя_класса*::operator#(список_аргум)**
{/*тело функции*/}

После этого вместо ***operator#(a,b)*** можно писать ***a#b***. Здесь # представляет один из введенных выше символов. Примерами являются операторы >> и << – перегружаемые операторы ввода-вывода. Отметим, что при перегрузке нельзя менять приоритет операторов и число операндов. Если оператор-функция-член класса перегружает бинарный оператор, то у функции будет только один параметр-объект, стоящий справа от знака оператора. Объект слева вызывает оператор-функцию и передается неявно с помощью указателя ***this***. В дружественную функцию указатель ***this*** не передается, поэтому унарный оператор имеет один параметр, а бинарный – два.

Оператор присваивания не может быть дружественной функцией, а только членом класса.

Пример. Создается класс Polynom. В головной программе выполняется тестирование класса.

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
class Polynom {
 int n;
 double *koef;
 public:
 Polynom(); //конструкторы
 Polynom(int k);
 Polynom(int k,double *mas);
 Polynom(const Polynom&ob); //конструктор копирования
 ~Polynom(){delete[]koef;}
 void GiveMemory(int k);
 void SetPolynom(int k,double *mas);
 void SetDegree(int k){n=k;}; //установить степень
 void CalculateValue(double x);  //вычислить значение
 int GetDegree(){return n;};   //получить степень
 double GetOneCoefficient(int i){return(koef[i]);};
 Polynom operator+(Polynom ob); //перегрузка операторов
 Polynom operator*(Polynom ob);
 double& operator[](int i){return(koef[i]);}//перегрузка []
 Polynom& operator = (const Polynom p) {
   if(&p==this) return *this;
   if(koef) delete [] koef;
```

```
   n=p.n;
   koef=new double [p.n+1];
   for(int i=0;i<=p.n;i++)
      koef[i]=p.koef[i];
   return *this;
}
 friend ostream& operator<<(ostream& mystream,Polynom &ob);
 friend istream& operator>>(istream& mystream,Polynom &ob);
int min(int n,int m)
{return (n<m)? n:m; }
 int max(int n,int m)
 {return (n>m)? n:m; }
};
//********** Polynom() *********************************
Polynom::Polynom()
{  randomize();
 n=random(5);
 koef=new double[n+1];
 if(!koef){cout<<"Error";getch();return;}
 for(int i=n;i>=0;i--)
    koef[i]=random(10)-5;
}
//************* Polynom(int k) ******************************
Polynom::Polynom(int k)
{   n=k;
 koef=new double[n+1];
 if(!koef){cout<<"Error";getch();return;}
 for(int i=n;i>=0;i--)
    koef[i]=random(10)-5;
}
//****************** Polynom(int k,double mas[]) ******************
Polynom::Polynom(int k,double mas[])
{n=k;
 koef=new double[n+1];
 if(!koef){cout<<"Error";getch();return;}
 for(int i=n;i>=0;i--)
    koef[i]=mas[i];
}
//*************** Polynom(const Polynom&ob) *********************
Polynom::Polynom(const Polynom&ob)
{n=ob.n;
 koef=new double[n+1];
 if(!koef){cout<<"Error";getch();return;}
 for(int i=0;i<=n;i++)
   koef[i]=ob.koef[i];
}
//*************** void GiveMemory(int k) *********************
```

```
void Polynom::GiveMemory(int k)
{
 if(koef) delete [] koef;
 koef=new double[k+1];
 if(!koef){cout<<"Error";getch();return;}
}
//******************** SetPolynom *************************
void Polynom::SetPolynom(int k,double *mas)
{ n=k;
 if(koef) delete [] koef;
 koef = new double [n+1];
 for(int i=n;i>=0;i--)
     koef[i]=mas[i];
}
//*************** CalculateValue ****************************
void Polynom::CalculateValue(double x=1.0)
{ double s;
  int i;
 for(s=koef[0],i=1;i<=n;i++)
    s=s+koef[i]*pow(x,i);
 cout<<"f("<<x<<")="; cout<<s<<endl;
}
//***************** Polynom operator+(Polynom ob) ***************
Polynom Polynom::operator+(Polynom ob)
{ int i;
 Polynom rab;
 rab.GiveMemory(max(n,ob.GetDegree()));
 for(i=0;i<=min(n,ob.GetDegree());i++)
    rab.koef[i]=koef[i]+ob.GetOneCoefficient(i);
 if(n<ob.GetDegree())
   {
    for(i=min(n,ob.GetDegree())+1;i<=ob.GetDegree();i++)
       rab.koef[i]=ob.GetOneCoefficient(i);
    rab.n=ob.GetDegree();
   }
 else
   {
    for(i=min(n,ob.GetDegree())+1;i<=n;i++) rab.koef[i]=koef[i];
  rab.n=n;
   }
 return rab;
}
//*************** Polynom operator*(Polynom ob) ***************
Polynom Polynom::operator*(Polynom ob)
{
 int i,j,k;
 double s;
```

```
 Polynom rab;
 rab.GiveMemory(n+ob.GetDegree());
 for(i=0;i<=n+ob.GetDegree();i++)
    { s=0;
     for(j=0;j<=n;j++)
        for(k=0;k<=ob.GetDegree();k++)
          if(j+k==i)s=s+koef[j]*ob.GetOneCoefficient(k);
     rab.koef[i]=s;
    }
 rab.n=n+ob.GetDegree();
 return rab;
}
//********** ostream& operator<<(ostream& mystream,Polynom &ob) ******
ostream& operator<<(ostream& mystream,Polynom &ob)
{ char c=' '; //пропустим "+" перед первым коэффициентом
   for(int i=ob.n;i>=0;i--)
    { double ai=ob.koef[i];
        if(ai==0) continue;
          else {if(ai>0) mystream<<c; mystream<<ai;}
         if(i==0) continue; else mystream<<"x";
         if(i==1) continue; else mystream<<"^"<<i;
         if(ai!=0)c='+';
    }
 if(c==' ')mystream<<0;
 mystream<<endl;
 return mystream;
}
//********* istream& operator>>(istream& mystream,Polynom &ob) *
istream& operator>>(istream& mystream,Polynom &ob)
{
 int i;
 cout<<"Enter Degree:";  mystream>>ob.n; cout<<endl;
 for(i=ob.n;i>=0;i--)
    {
     cout<<"Enter koeff "<<i<<":";  mystream>>ob.koef[i];
    }
 return mystream;
}
//******************** MAIN ****************************
int main(int argc, char* argv[])
{ const int m=3;
   Polynom f,g,masp[m],*p1,s;
   int n=5,i;
   double K[6]={1.0,3.2,0.0,4.1,0.0,1.1};
    p1=new Polynom(n,K);
    cout<<*p1;
    p1->CalculateValue(2.0);
```

```
  cin>>f;
     cout<<" f(x)= ";  cout<<f;
    cout<<" g(x)= "; cout<<g;
  s=f+g;
  cout<<"f(x)+g(x) = "; cout<<s;
  s=f*g;
  cout<<" f(x)*g(x) = ";  cout<<s;
  s=masp[0]; cout<<masp[0];
  for(i=1;i<m;i++)
    { s=s+masp[i]; cout<<masp[i];}
  cout<<"Summa: ";  cout<< s;
  while(!kbhit());
  delete p1;
  return 0;
}
```

Задания для самостоятельного решения

Разработать перечисленные ниже классы. При разработке каждого класса возможны два варианта решения: а) данные-члены класса представляют собой переменные и массивы фиксированной размерности; б) память для данных-членов класса выделяется динамически.

1. «**Комплексное число**» – **Complex**. Класс должен содержать несколько конструкторов и операции для сложения, вычитания, умножения, деления, присваивания. Создать два вектора размерности n из комплексных координат. Передать их в функцию, которая выполняет сложение комплексных векторов.

2. Определить класс «**Дробь**» – **Fraction** в виде пары (m,n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Перегрузить операции сложения, вычитания, умножения, деления, присваивания и операции отношения. Создать массив объектов и передать его в функцию, которая изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.

3. Разработать класс «**Вектор**» – **Vector** размерности n. Определить несколько конструкторов, в том числе конструктор копирования. Реализовать методы для вычисления модуля вектора, скалярного произведения, сложения, вычитания, умножения на константу. Перегрузить операции сложения, вычитания, умножения, инкремента, декремента, индексирования, присваивания для данного класса. Создать массив объектов. Написать функцию, которая для заданной пары векторов будет определять, являются ли они коллинеарными или ортогональными.

4. Определить класс «**Квадратная матрица**» – **Matrix**. Класс должен содержать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для сложения, вычитания, умножения матриц; вычисления нормы матрицы. Перегрузить операции сложения, вычитания, умножения и присваивания для данного класса. Создать массив объектов класса **Matrix** и передать его в функцию, которая изменяет i-ю матрицу путем возведения ее в квадрат. В головной программе вывести результат.

5. Разработать класс **«Многочлен» – Polynom** степени n. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для вычисления значения полинома; сложения, вычитания и умножения полиномов. Перегрузить операции сложения, вычитания, умножения, инкремента, декремента, индексирования, присваивания. Создать массив объектов класса. Передать его в функцию, вычисляющую сумму полиномов массива и возвращающую полином-результат, который выводится на экран в головной программе.

6. Определить класс **«Стек» – Stack**. Элементы стека хранятся в массиве. Если массив имеет фиксированную размерность, то предусмотреть контроль выхода за пределы массива. Если память выделяется динамически и ее не хватает, то увеличить размер выделенной памяти. Включение элементов в стек и их извлечение реализовать как в виде методов, так и с помощью перегруженных операций. Создать массив объектов класса **Stack**. Передавать объекты в функцию, которая удаляет из стека первый (сверху), третий, пятый и т. д. элементы.

7. Построить классы для описания плоских фигур: круг, квадрат, прямоугольник. Включить методы для изменения объектов, перемещения на плоскости, вращения. Перегрузить операции, реализующие те же действия. Выполнить тестирование класса, создав массив объектов.

8. Определить класс «**Строка**» – **String** длины n. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для выполнения конкатенации строк, извлечения символа из заданной позиции, сравнения строк. Перегрузить операции сложения, индексирования, отношения, добавления $(+=)$, присваивания для данного класса. Создать массив объектов и передать его в функцию, которая выполняет сортировку строк.

9. Разработать класс «**Множество (целых чисел, символов, строк** и т. д.)» – **Set** мощности n. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для определения принадлежности заданного элемента множеству, пересечения, объе-

динения, разности двух множеств. Перегрузить операции сложения, вычитания, умножения (пересечения), индексирования, присваивания. Создать массив объектов и передавать пары объектов в функцию, которая строит множество, состоящее из элементов, входящих только в одно из заданных множеств, т. е. $(A \cup B) \setminus (A \cap B)$, и возвращает его в головную программу.

10. Разработать класс для массива строк. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для поэлементной конкатенации двух массивов, упорядочения строк в лексикографическом порядке, слияния двух массивов с удалением повторяющихся строк, а также для вывода на экран всего массива и заданной строки. Перегрузить операции сложения, умножения, индексирования, присваивания для данного класса. Создать массив объектов и передавать объекты в функцию, которая выполняет слияние объектов и для полученного объекта-результата производит лексикографическое упорядочения строк.

11. Составить описания класса, обеспечивающего представление матрицы заданного размера $n \times m$ и любого минора в ней. Память для матрицы выделять динамически. Написать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для отображения на экране как матрицы в целом, так и заданного минора, а также для изменения минора; сложения, вычитания, умножения миноров. Перегрузить операции сложения, вычитания, умножения и присваивания для данного класса. Создать массив объектов данного класса и передать его в функцию, которая изменяет для i-й матрицы ее минор путем умножения на константу.

12. Построить класс «**Булев вектор**» – **BoolVector** размерности n. Определить несколько конструкторов, в том числе конструктор копирования. Реализовать методы для выполнения поразрядных конъюнкции, дизъюнкции и отрицания векторов, а также подсчета числа единиц и нулей в векторе. Реализовать те же действия над векторами с помощью перегруженных операций. Перегрузить операции отношения и присваивания для данного класса. Создать массив объектов. Передавать объекты в функцию, которая будет их изменять по формуле $A = A \vee \overline{B}$.

13. Реализовать класс «**Троичный вектор**» – **Tvector** размерности n. Компоненты вектора принимают значения из множества $\{0, 1, X\}$. Два троичных вектора t_k=(t_1^k,…,t_n^k) и t_l=(t_1^l,…,t_n^l) называются ортогональными, если существует такое i, что t_i^k, $t_i^l \in \{0,1\}$ и $t_i^k \neq t_i^l$. Операция

пересечения не ортогональных векторов выполняется покомпонентно по следующим правилам: $1\cap1=1\cap X=X\cap1=1$, $0\cap0=0\cap X=X\cap0=0$, $X\cap X=X$. Реализовать методы для проверки векторов на ортогональность, для пересечения не ортогональных векторов, сравнения векторов, подсчета числа компонент, равных X. Осуществить те же действия над векторами с помощью перегруженных операций. Перегрузить операцию присваивания для данного класса. Выполнить тестирование класса, создав массив объектов.

14. Определить класс «**Булева матрица**» – **BoolMatrix** размерности $n\times m$. Класс должен содержать несколько конструкторов, в том числе конструктор копирования. Реализовать методы для логического сложения (дизъюнкции), умножения и инверсии матриц. Реализовать методы для подсчета числа единиц в матрице и лексикографического упорядочения строк. Перегрузить операции для логического сложения, умножения и инверсии матриц, а также операцию присваивания. Создать массив объектов класса **BoolMatrix.** Передавать объекты в функцию, которая их изменяет по формуле $A=\overline{A}\vee\overline{B}$.

Тесты

1. В какой строке допущена ошибка?

```
class X {
   int a;
   int f() const;          //1
   int g() {return a++;}       //2
   int h() const {return a++;}    //3
};
```

Варианты ответа:

1) здесь нет ошибок; (*) 2) //3; 3) //2; 4) //1.

2. Выберите три верных утверждения:

[1] статическая функция-член класса, не обладает указателем this;
[2] статическая функция-член класса, может быть виртуальной;
[3] в классе не может быть двух функций: статической и нестатической – с одним и тем же именем и списком параметров;
[4] статическая функция в классе не может быть объявлена const.

Варианты ответа:

1) все утверждения не верны; (*) 2) [4]; (*) 3) [3]; 4) [2]; (*) 5) [1].

3 Правильно ли перегружены функции?

```
class X {
   static void f();
   void f() const;
};
```

Варианты ответа:

*1) нет; 2) да; 3) да, если убрать const.

4. Какие из операторов не могут быть перегружены?

Варианты ответа:

1) new; *2) . ; 3) * ; 4) []; 5) ().

5. Что будет выведено?

```
#include < iostream.h >
class X {
   int a;
public:
   X() : a(1) {}
   X& operator++() {a++; return *this;};
   X& operator++(int) {a--; return *this;};
   friend ostream& operator<<(ostream&, X&);
};
ostream& operator<<(ostream& _stream, X& _x) {
   _stream << _x.a;
   return _stream;
}
void main() {
   X x;
   cout << ++x;
   cout << x++;
}
```

Варианты ответа:

1) ничего; *2) 21; 3) 12; 4) 22; 5) 11.

6. Даны следующие три свойства, которыми обладает функция-член класса:

[1] функция имеет право доступа к закрытой части объявления класса;
[2] функция находится в области видимости класса;
[3] функция должна вызываться для объекта класса (имеется указатель this).

Какими из этих свойств обладает дружественная функция?

Варианты ответа:

1) [3]; 2) ни одним; 3) [2]; *4) [1].

7. Какие функции являются неявно inline? Выберите три варианта ответа.

[1] все дружественные функции;
[2] дружественные функции, определенные в классе;
[3] неявный конструктор по умолчанию;
[4] неявный деструктор по умолчанию;
[5] виртуальные функции.

Варианты ответа:

1) [5]; *2) [4]; *3) [3]; *4) [2]; 5) [1].

8. Дан следующий код:

```
class X {
   friend void f(X&);
};
```

С помощью какого синтаксиса может быть вызвана функция f?

Варианты ответа:

1) f не может быть вызвана, т. к. она объявлена как private; 2) X x; X::f(x); 3) X x; f(&x); 4) X x; x.f(x); *5) X x; f(x).

Лабораторная работа № 3
Тема. Классы для работы с динамическими структурами данных

Теоретическое введение. Объекты классов могут храниться в виде массивов или динамических связных списков. Если класс содержит конструктор, массив может быть инициализирован, причем конструктор вызывается столько раз, сколько задается элементов массива: **samp ob[4]={1,2,3,4};**

Список инициализации – это сокращение общей конструкции:

samp ob[4]={samp(1,2),samp(3,4),samp(5,6),samp(7,8)};

При создании динамических объектов используется оператор ***new***, который вызывает конструктор и производит инициализацию. Для разрушения динамического объекта используется оператор ***delete***, который может помещаться в деструкторе.

Кроме указателей на классы используются ссылки. Ссылка является скрытым указателем и работает как другое имя переменной. При передаче объекта через ссылку в функцию передается адрес объекта и не делается его копия. Это уменьшает вероятность ошибок, связанных с выделением динамической памяти и вызовом деструктора. Так, при передаче в функцию параметра-объекта может возникнуть ошибка из-за разрушения деструктором на выходе копии объекта, которая должна быть исправлена созданием конструктора копирования. В такой ситуации лучше передать в функцию ссылку на объект и возвратить ссылку на объект.

Пример. Создается класс **Tree** (бинарное дерево). Информационная часть узла дерева содержит целое число. Тестирование класса выполняется с помощью меню, которое позволяет сформировать дерево, вывести содержимое его узлов в порядке возрастания, найти узел по ключу, вывести содержимое листьев дерева (вершин, не имеющих потомков).

```
#include"vip\menu.cpp" //реализация работы с меню
#include <conio.h>
#include <string.h>
#include <iostream.h>
char bufRus[256];
char*Rus(const char*text){
 CharToOem(text,bufRus);
 return bufRus;}
 struct node
     {
      int n; //информационное поле узла дерева
```

```
        int count;
        node*left,*right;
       };
    class Tree
    {
     public:
     node*root;
     Tree(){root=0;}
     Tree(int t); // Формирование дерева из t случайных чисел
     void CopyTree(node*&rootnew,node*rootold);
      /* Копирует дерево с корнем rootold в дерево с  корнем rootnew. В результате
деревья находятся в различных динамических участках памяти.*/
     Tree(const Tree&ob); //конструктор копирования
   // Рекурсивная функция, используемая в деструкторе (освобождение памяти)
     void DelTree(node *wer);
     ~Tree(){DelTree(root);}
     void Push(node*&wer,int data);// Вставка элемента в дерево
     void Look(node*wer);          //- Вывод дерева на экран
     node*Find(node*wer,int key);  // Поиск по ключу
     void PrintLeaves(node *wer);  // Вывод листьев дерева на экран
    };
   //********************** Tree::Tree(int t) ******************
   Tree::Tree(int t)
   {
    root=0;
    for(int i=0;i<t;i++)
      Push(root,random(10)-5);
   }
   void Tree::CopyTree(node*&rootnew,node*rootold)
   {
    if(rootold->left!=0)
     {Push(rootnew,(rootold->left)->n);CopyTree(rootnew,rootold->left);}
    if(rootold->right!=0)
     {Push(rootnew,(rootold->right)->n);CopyTree(rootnew,rootold->right);}
   }
   Tree::Tree(const Tree&ob)
   {
    if(ob.root==0)root=0;
    else {
        root=new node;
        root->n=ob.root->n;
        root->count=1;
        root->left=0;
        root->right=0;
        CopyTree(root,ob.root);
       }
   }
```

```
void Tree::DelTree(node *wer)
{
 if(wer->left!=0)DelTree(wer->left);
 if(wer->right!=0)DelTree(wer->right);
 delete wer;
}
void Tree::Push(node*&wer,int data)
    {
     if(wer==0)
       {
        wer=new node;
        wer->n=data;
        wer->left=0;wer->right=0;
        wer->count=1;
       }
     else if(data<wer->n)Push(wer->left,data);
          else if(data>wer->n)Push(wer->right,data);
               else wer->count++;
    }
void Tree::Look(node*wer)
{
 if(wer!=0)
   {
    Look(wer->left);
    cout<<Rus("Число: ")<<wer->n<<" - "<<wer->count;
    cout<<Rus(" штук")<<endl;
    Look(wer->right);
   }
}
node* Tree::Find(node*wer,int key)
{
 if(wer==0) return 0;
 else if(key<wer->n) return Find(wer->left,key);
      else if(key>wer->n) return Find(wer->right,key);
           else return wer;
}
void Tree::PrintLeaves(node *wer)
{
 if(wer==0)return;
 else if( (wer->left==0)&&(wer->right==0) ) {
      cout<<Rus("Число: ")<<wer->n<<"-"<<wer->count;
      cout<<Rus("штук")<<endl;
      }
      else
          {
           PrintLeaves(wer->left);
           PrintLeaves(wer->right);
```

```
        }
}
//------------------------------ MAIN -------------------------------------
int main(int argc, char* argv[])
{
 Tree tr;
 node *u;
 int k=0,max,kol;
 char menu[][100]={ {"   PushElement "}, {"   ShowTree "}, {"   FindElement   "},
                    {"   PrintLeaves   "}, {"   EXIT "},  };
 kol=5;//КОЛИЧЕСТВО СТРОК МЕНЮ. Используется в выравнивании строк
        // меню по центру.
//------------------ВЫРАВНИВАНИЕ СТРОК МЕНЮ ПО ЦЕНТРУ----------------
max=viravnivaniestrok(menu,kol);
//----------------------- МЕНЮ НА ЭКРАНЕ-------------------------------------
 textmode(C80);
   while(1){
   switch(mmm(kol,menu,max,k))
       {    case 0: {
                    int data;
                    cout<<Rus("Введите число:");
                    cin>>data;
                    tr.Push(tr.root,data);
                    k=0;break;
                   }
            case 1: {
                    if(tr.root==0)cout<<Rus("Дерево пустое");
                    else
                        {
                         cout<<Rus("Наше дерево:")<<endl;
                         tr.Look(tr.root);
                        }
                    while(!kbhit());
                    k=1;break;
                   }
            case 2: {
                    if(tr.root==0)cout<<Rus("Дерево пустое");
                    else
                       {
                        int key;
                        cout<<Rus("Введите искомое число:");
                        cin>>key;
                        if((u=tr.Find(tr.root,key))!=0){
                          cout<<Rus("Элементов: ");
                          cout<<key;
                          cout<<Rus(" найдено ");
                          cout<<u->count<<Rus(" штук");
```

```
                    }
                   else cout<<Rus("Таких элементов нет!");
                  }
                 while(!kbhit());
                 k=2;break;
                }
          case 3: {
                 if(tr.root==0)cout<<Rus("Дерево пустое");
                 else{
                     cout<<Rus("Листья:")<<endl;
                     tr.PrintLeaves(tr.root);
                    }
                 while(!kbhit());
                 k=3;break;
                }
           case 4:{
                 exit(0);
                }
       } }
    return 0;
}
```

Задания для самостоятельного решения

При решении задач необходимо описать класс, который используется для представления элементов динамической структуры данных. Затем разрабатывается класс для работы с используемой динамической структурой данных, которая при тестировании класса может быть построена путем ввода данных: а) с клавиатуры; б) из файла. Возможны два варианта решения:

а) динамическая структура данных постоянно хранится в памяти;

б) динамическая структура данных хранится в файле.

1. Создать класс для работы со стеком. Элемент стека – действительное число. Применить класс для вывода возрастающих серий последовательности действительных чисел: а) в обратном порядке; б) в том же порядке (серия – упорядоченная последовательность максимальной длины).

2. Построить класс для работы со стеком. Элемент стека – целое число. Ввести две неубывающие последовательности чисел в два стека. Использовать третий стек для слияния двух последовательностей в одну неубывающую.

3. Создать класс для работы со стеком. Элемент стека – символ. Сформировать два стека, содержащие последовательности символов. Подсчитать общее число элементов в стеках, предусмотреть восстановление их исходного расположения.

4. Создать класс для работы со стеком. Элемент стека – символ. Использовать стек для проверки правильности расстановки скобок трех типов (круглых, квадратных и фигурных) в выражении.

5. Построить класс для работы с односвязным списком. Элемент списка – действительное число. Сформировать список, содержащий неубывающую последовательность чисел, и преобразовать его так, чтобы последовательность была невозрастающей. Для этого необходимо совершить переворот списка, т. е. такую переустановку указателей в списке, при которой элементы его следуют друг за другом в обратном порядке.

6. Построить класс для работы с односвязным списком. Элементы списка – целые числа. Сформировать список, упорядочить элементы списка по возрастанию, используя сортировку: а) методом выбора; б) методом пузырька; в) методом вставки.

7. Построить класс для работы с односвязным списком. Элементы списка – действительные числа. Создать два упорядоченных по невозрастанию списка, слить их в один (также упорядоченный по невозрастанию), построив новый список.

8. Построить класс для работы с односвязным списком. Элементы списка – слова. Создать список, содержащий некоторую последовательность слов. Заменить в списке каждое вхождение заданного слова другим (также заданным).

9. Построить класс для работы с односвязным списком. Создать два списка: List1 и List2. Проверить, содержатся ли элементы списка List1 в списке List2 в указанном списком List1 порядке.

10. Построить класс для работы с односвязным списком. Элементы списка – целые числа. Создать список List1. Построить список List2, содержащий порядковые номера максимальных элементов списка List1.

11. Построить класс для работы с двусвязным списком. Элементы списка – действительные числа. Создать список List1, содержащий последовательность $x_1, x_2, \ldots, x_n$. Построить список List2, содержащий последовательность $x_1, x_n, x_2, x_{n-1}, \ldots, x_n, x_1$.

12. Создать класс для работы с бинарным деревом, узлы которого содержат целые числа. Построить дерево, затем копию дерева. Подсчитать число листьев в нем (листьями называются узлы, не содержащие поддеревьев).

13. Построить класс для работы с бинарным деревом, узлы которого содержат действительные числа. Создать дерево. Определить высоту дерева (максимальное число узлов, принадлежащих пути от корня

дерева до любого из его листьев). Подсчитать число элементов, равных максимальному.

14. Построить класс для работы с бинарным деревом, узлы которого содержат действительные числа. Создать дерево для заданной последовательности чисел. Используя его, упорядочить последовательность по возрастанию, убыванию.

15. Построить класс для работы со списком. Элемент списка содержит информацию о заявке на авиабилет: пункт назначения, номер рейса, фамилию и инициалы пассажира, желаемую дату вылета.

Программа должна обеспечивать: хранение всех заявок в виде списка, добавление заявок в список, удаление заявок, вывод заявок по заданному номеру рейса и дате вылета, вывод всех заявок.

16. Построить класс для работы с бинарным деревом, узел которого содержит информацию о заявках на авиабилеты (в последовательности, используемой для упорядочения заявок): желаемую дату вылета, номер рейса, фамилию и инициалы пассажира, пункт назначения.

Программа должна обеспечивать: хранение всех заявок в виде бинарного дерева, добавление заявок, удаление заявок, вывод заявок по заданному номеру рейса и дате вылета, вывод всех заявок.

17. Построить класс для работы со списком, который содержит динамическую информацию о наличии автобусов в парке: номер автобуса, фамилию и инициалы водителя, номер маршрута, признак местонахождения автобуса – на маршруте или в парке.

Программа должна обеспечивать: начальное формирование списка, введение номера автобуса при выезде и установление программой значения признака «автобус на маршруте». Аналогичным образом изменяется информация об автобусе при его возвращении в парк. По запросу выдаются сведения об автобусах, находящихся в парке, или об автобусах, находящихся на маршруте.

18. Решить предыдущую задачу, используя не список, а бинарное дерево.

19. Построить класс для работы с бинарным деревом, содержащим англо-русский словарь.

20. Построить класс для работы со списком, содержащим информацию о поездах дальнего следования. Элемент списка содержит следующую информацию о поезде: номер поезда, станция назначения, время отправления.

Составить программу, которая обеспечивает первоначальное формирование списка, производит вывод списка, вводит номер поезда и

выводит информацию о нем, вводит название станции назначения и выводит данные о всех поездах, следующих до этой станции.

Тесты

1. На каком этапе (компиляция, выполнение) происходит выделение памяти под динамические структуры данных (ДСД)?
2. Какие основные операции выполняются над следующими ДСД:
   1) стек;
   2) односвязный список;
   3) односвязная очередь;
   4) двусвязная очередь;
   5) бинарное дерево?
3. Есть ли ошибки в деструкторе дерева:

```
~Tree(){del_tree(root);}
void del_tree(Node *root){
if(!root){
delete root;
del_tree(root->left);
del_tree(root->right);}} ?
```

4. Есть ли ошибки в деструкторе списка:

```
~List(){
Node *V;
if(begin){
V=begin;
while(V){
delete V;
V=V->next;}}} ?
```

Здесь begin – указатель на начало списка.

Лабораторная работа № 4
Тема. Шаблоны классов

Теоретическое введение. С помощью шаблонов можно создавать родовые ***(generic)*** функции и классы. Родовая функция определяет базовый набор операций, которые будут применяться к разным типам данных, получаемых функцией в качестве параметра.

Определение функции с ключевым словом ***template*** (шаблон) имеет вид:

template<class *Ttype*>тип имя_функции(список аргументов)
{//тело функции}

Здесь ***Ttype*** – фиктивный тип, который используется при объявлении аргументов и локальных переменных функции. Компилятор заменит этот фиктивный тип на один из реальных и создаст соответственно несколько перегружаемых функций. При этом перегружаемые функции являются ограниченными, поскольку выполняют одни и те же действия над данными различных типов.

С помощью шаблона класса можно создать класс, реализующий стек, очередь, дерево и т. д. для любых типов данных. Компилятор будет генерировать правильный тип объекта на основе типа, задаваемого при создании объекта. Общая форма объявления шаблона класса:

template <class *Ttype*> class имя_класса {
//имена класса, поля класса
}

Здесь ***Ttype*** – фиктивное имя типа, которое заменится компилятором на фактическое. Шаблон класса может иметь больше одного родового типа данных:

template<class Type1,class Type2> class m {Type1 a;Type2 b;}

Пример. Создается шаблон класса для работы с односвязным списком. Тестирование класса выполняется с использованием меню, которое позволяет создать список из чисел типов integer, float, double и выполнить типичные операции с ним.

```
#include <conio.h>
#include <string.h>
#include <iostream.h>
#include"vip\menu.cpp"
void ListForInt();
void ListForFloat();
void ListForDouble();
char bufRus[256];
char*Rus(const char*text){
 CharToOem(text,bufRus);
 return bufRus;
}
//шаблон класса для работы с односвязным списком
template<class mytype>class List {
//внутренний класс, для представления элементов списка
  class Node{
       public:
         mytype d;
         Node* next;
         Node(mytype dat=0){d=dat; next=0;}
       };
  Node* pbeg; //указатель на начало списка
  public:
  List(){pbeg=0;} //конструктор
  ~List(); //деструктор
  Node * Add(mytype d); //добавление в конец списка
  Node * Find(mytype key); //поиск по ключу
  Node * Insert(mytype key,mytype d); //вставка узла d после узла с
```

```
                                        // ключом key
  bool Remove(mytype key); //удаление узла
  void Print(); //печать списка
 };
//**********************~List() ************************
//ДЕСТРУКТОР. Освобождает память для всех узлов списка
template<class mytype> List<mytype>::~List(){
 if(pbeg!=0){
    Node* pv=pbeg;
    while(pv){
         pv=pv->next;
         delete pbeg;
         pbeg=pv;
        }
   }
}
//*************************** void Add(mytype d) **********
//Добавляет узел в конец списка и возвращает указатель
// на вставленный узел.
template<class mytype> List<mytype>::Node*
List<mytype>::Add(mytype d){
 Node* pv=new Node(d); //Создание нового узла
 if(pbeg==0)pbeg=pv; //первый узел списка
 else {
       Node* rab=pbeg;
       while(rab!=0){
            if((rab->next)==0){rab->next=pv;return pv;}
            rab=rab->next;
           }
      }
}
//*************************** Node* Find(mytype key)
//Выполняет поиск узла с заданным ключом и возвращает указатель
// на него в случае успешного поиска и 0 в случае отсутствия узла в списке
template<class mytype> List<mytype>::Node*
List<mytype>::Find(mytype key){
 Node* pv=pbeg;
 while(pv){
      if((pv->d)==key)break;
      pv=pv->next;
     }
 return pv;
}
//************* Node* Insert(mytype key,mytype d)
//Вставляет в список узел после узла с ключом key и возвращает
// указатель на вставленный узел. Если такого узла в списке нет,
// вставка не выполняется и возвращается значение 0
```

```
template<class mytype> List<mytype>::Node*
List<mytype>::Insert(mytype key,mytype d){
 if(Node* pkey=Find(key)) //поиск узла с ключом key
   {
    Node* pv=new Node(d);
//выделение памяти под новый узел и его инициализация
    pv->next=pkey->next;
//установление связи нового узла с последующим
    pkey->next=pv; //установление связи предыдущего узла с новым
    return pv;
   }
 return 0;
}
//******************* bool Remove(mytype key)
//Удаляет узел с заданным ключом из списка и возвращает значение true при
//успешном удалении и false, если узел с таким ключом не найден
template<class mytype> bool List<mytype>::Remove(mytype key){
 if(Node* pkey=Find(key)){
    if(pkey==pbeg)pbeg=pbeg->next; //удаление из начала списка
    else{                          //Находим указатель на узел,
        Node*rab=pbeg;             //стоящий в списке перед
        while(rab)                 //удаляемым узлом.
            {                      //rab-этот указатель.
             if((rab->next)==pkey)break;
             rab=rab->next;
            }
        rab->next=pkey->next;
       }
    delete pkey;
    return true;
   }
 return false;
}
//******************* void Print() -Печать списка
template<class mytype> void List<mytype>::Print(){
 Node*pv=pbeg;
 cout<<Rus("Наш список:");cout<<endl;
 while(pv){
     cout<<pv->d<<' ';
     pv=pv->next;
    }
 cout<<endl;}
//-------------------------- MAIN -----------------------------------------
int main(int argc, char* argv[]){
 int k=0,max,kol;
 char menu[][100]= {{"   ListForInt "},  {"   ListForFloat "},
                   {"   ListForDouble "}, {"   EXIT "},   };
```

```
 kol=4;  //КОЛИЧЕСТВО СТРОК МЕНЮ. Это используется в выравнивании
          //строк меню по центру.
//----ВЫРАВНИВАНИЕ СТРОК МЕНЮ ПО ЦЕНТРУ------------------
max=viravnivaniestrok(menu,kol);
//----------------- МЕНЮ НА ЭКРАНЕ-------------------------------------
 textmode(C80);
while(1){
   switch(mmm(kol,menu,max,k))
         {    case 0: {
                    ListForInt();
                    k=0;break;
                    }
              case 1: {
                    ListForFloat();
                    k=1;break;
                    }
              case 2: {
                    ListForDouble();
                    k=2;break;
                    }
              case 3:{
                    exit(0);
                    }
               }
         }
      return 0;
}
//*************************** void ListForInt()
//Эта функция вызывается из главного меню.
void ListForInt(){
 List<int>l1;
 int k=0,max,kol;
 char menu[][100]=
                { {"   PrintList "}, {"   Add "}, {"   Find "}, {"   Insert   "},
                  {"   Remove   "}, {"   EXIT "},  {"   Back "}  };
 kol=7;  //КОЛИЧЕСТВО СТРОК МЕНЮ.
max=viravnivaniestrok(menu,kol);
//----------------------- МЕНЮ НА ЭКРАНЕ----------------------------
 textmode(C80);
 while(1){
   switch(mmm(kol,menu,max,k))
         {    case 0: {
                    l1.Print();
                    while(!kbhit())
                    k=0;break;}
              case 1: {
                    cout<<Rus("введите число, которое надо вставить:");
```

```
        int t;cin>>t;
        if( (l1.Add(t)) )cout<<Rus("вставка осуществлена");
        else cout<<Rus("вставка не осуществлена");
        while(!kbhit());
        k=1;break;}
    case 2: {
        cout<<Rus("введите искомое число:");
        int t;
        cin>>t;
        if(l1.Find(t))cout<<Rus("искомое число есть в списке.");
        else cout<<Rus("искомого числа нет в списке.");
        while(!kbhit());
        k=2;break;}
    case 3: {
        cout<<Rus("введите число, которое надо вставить:");
        int t;cin>>t;
        cout<<Rus("введите число, после которого надо вставить:");
        int key;cin>>key;
        if( (l1.Insert(key,t)) )cout<<Rus("вставка осуществлена");
        else cout<<Rus("вставка не осуществлена");
        while(!kbhit());
        k=3;break;}
    case 4: {
        cout<<Rus("введите число, которое надо удалить:");
        int t;cin>>t;
        if( (l1.Remove(t)) )cout<<Rus("удаление осуществлено");
        else cout<<Rus("такого числа нет в списке.");
        while(!kbhit());
        k=4;break;}
    case 5:{exit(0);}
     } } }
```

Таким же образом, как и функция ListForInt(), реализуются функции ListForFloat() и ListForDouble(), предназначенные для тестирования списков из чисел типа float и double, соответственно.

Задания для самостоятельного решения

Для разработки шаблонов классов можно использовать результаты выполнения лабораторных работ № 2 и № 3. При тестировании созданных шаблонов классов необходимо создавать объекты с различными допустимыми значениями параметров шаблона (например, компоненты вектора могут быть целыми, действительными или комплексными числами).

1. Создать шаблон класса для работы со стеком. Применить его для решения задач № 1 – 4 (лаб. работа № 3).

2. Создать шаблон класса для работы с одномерным массивом. Выполнить тестирование путем создания и обработки массивов, содержащих элементы различных типов (например, сортировка элементов массивов различными методами).

3. Создать шаблон класса **Vector** размерности *n* (см. задачу № 3, лаб. работа № 2).

4. Создать шаблон класса «Квадратная матрица» – **Matrix** размерности $n \times n$ (см. задачу № 4, лаб. работа № 2).

5. Создать шаблон класса **Polynom** степени *n* (см. задачу № 5, лаб. работа № 2) или создать шаблон класса для работы с односвязным списком. Применить его для решения задачи № 5 (лаб. работа № 3).

6. Создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 6 (лаб. работа № 3).

7. Создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 7 (лаб. работа № 3).

8. Создать шаблон класса для работы с бинарным деревом. Применить его для сортировки действительных чисел и строк, вводимых с клавиатуры или из файла.

9. Создать шаблон класса **Set** (множество) мощности *n* (см. задачу № 9, лаб. работа № 2) или создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 9 (лаб. работа № 3).

10. Создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 10 (лаб. работа № 3).

11. Создать шаблон класса, обеспечивающего описание матрицы заданного размера $n \times m$ и любого минора в ней (см. задачу № 11, лаб. работа № 2) или создать шаблон класса для работы с двусвязным списком. Применить шаблон класса для решения задачи № 11 (лаб. работа № 3).

12. Создать шаблон класса для работы с бинарным деревом. Применить шаблон класса для решения задачи № 12 (лаб. работа № 3).

13. Создать шаблон класса для работы с бинарным деревом. Применить шаблон класса для решения задачи № 13 (лаб. работа № 3).

14. Создать шаблон класса для работы с двусвязным списком. Применить шаблон класса для решения задачи № 15 (лаб. работа № 3).

15. Создать шаблон класса для работы с односвязным списком. Применить шаблон класса для решения задачи № 16 (лаб. работа № 3).

Тесты

1. Отметьте все утверждения, которые считаете верными:

1) нельзя с помощью шаблона создать функцию с таким же именем, как у явно определенной функции;

2) цель введения шаблонов – создание функций, которые могут обрабатывать однотипные данные;

*3) при использовании шаблонов функций возможна перегрузка как с помощью шаблонов, так и функций;

*4) в качестве описания шаблона функции используется прототип шаблона:

$$template\langle список\_параметров\_шаблона\rangle$$

2. Можно ли задать шаблон со значением параметра по умолчанию?

Варианты ответа:

*1) да; 2) нет.

3. Каков правильный заголовок шаблона?

Варианты ответа:

*1) template<class t1,class t2>; 2) template <class t1,t2>; 3) template <class t,class t>;

4. Сколько параметров может быть у шаблона при определении шаблона функции?

Варианты ответа:

1) 1; 2) столько, сколько аргументов у функции; *3) столько, сколько типов используется для параметризации.

5. Отметьте правильный вариант описания шаблона семейства функций:

1) template(class T)
   void func(T* p1,T* p2){…}
2) template <class T>;
   void func(T* p1,T* p2){…}
*3) template<class T>
   void func(T* p1,T* p2){…}
4) template<class T>
   void func(T* p1,T* p2){…}

6. Можно ли использовать класс-шаблон в качестве базового класса?

Варианты ответа:

*1) да; 2) нет.

Лабораторная работа № 5
Тема. Наследование

Теоретическое введение. Важнейшим принципом ООП, реализованным в C++, является *наследование.* Класс, который наследуется, называется базовым классом, а наследующий класс –

производным классом. Указателю на объект базового класса можно присвоить значение указателя на объект производного класса.

Все члены класса из разделов ***public*** и ***protected*** наследуются, а члены из раздела ***private*** – нет**.** Члены раздела ***protected*** являются частными для базового и производного классов. При наследовании базового класса к его членам также может объявляться спецификатор ***доступ***.

class имя_класса: ***доступ*** **имя_базового_класса**
{/*члены класса*/}

Спецификатор ***доступ*** базового класса при наследовании может принимать одно из трех значений: ***public*** (по умолчанию), ***private, protected.*** Если спецификатор принимает значение ***public,*** то все члены разделов ***public*** и ***protected*** базового класса становятся членами разделов ***public*** и ***protected*** производного класса. Если доступ имеет значение ***private***, то все члены разделов ***public*** и ***protected*** становятся членами раздела ***private*** производного класса. Если доступ имеет значение ***protected***, то все члены разделов ***public, protected*** становятся членами раздела ***protected*** производного класса.

Конструкторы и деструкторы базовых классов не наследуются**,** однако при создании объектов производных классов конструкторы базовых классов выполняются в порядке наследования, а деструкторы в обратном порядке. При необходимости передачи аргументов конструктору базового класса из производного класса используется следующий синтаксис:

конструктор_производного_класса(арг):base(арг)
{/*тело конструктора производного класса*/}

Множественное наследование. Один класс может наследовать атрибуты двух и более базовых классов, которые перечисляются после двоеточия через запятую. Если базовые классы содержат конструкторы, то они вызываются поочередно в порядке перечисления.

Виртуальные функции и полиморфизм. Механизм виртуальных функций в ООП используется для реализации полиморфизма: создания метода, предназначенного для работы с различными объектами за счет механизма позднего связывания *(late binding)*. Виртуальные функции объявляются в базовом и производных классах с ключевым словом ***virtual***. При этом каждый объект класса, управляемого из базового класса с виртуальными функциями, имеет указатель на *vmtbl (virtual method table)*, содержащую адреса виртуальных функций. Эти

адреса устанавливаются в адреса нужных для данного объекта функций во время выполнения.

В отличие от перегружаемых функций виртуальные объявляются в порожденных классах с тем же именем, возвращаемым значением и типом аргументов. Если различны типы аргументов, виртуальный механизм игнорируется. Тип возвращаемого значения переопределить нельзя.

Основная идея в использовании виртуальной функции состоит в следующем: она может быть объявлена в базовом классе, а затем переопределена в каждом производном классе. При этом доступ через указатель на объект базового класса осуществляется к этой функции из базового класса, а доступ через указатель на объект производного класса – из производного класса. То же происходит при передаче функции объекта производного класса, если аргумент объявлен как базовый класс.

Абстрактные классы – это классы, содержащие *чисто абстрактные виртуальные функции*. Чисто абстрактные виртуальные функции при объявлении в классе приравниваются к нулю. Абстрактные классы используются только для наследования, так как объекты таких классов не могут быть созданы.

Пример. Создаются три класса: MyPoint (базовый класс – «Точка»), MyEllipse (производный класс от класса MyPoint – «Окружность») и MyCylinder (производный класс от класса MyEllipse – «Цилиндр»). Классы объявлены в файле MyClass.h. Используются кнопки: Button1 – очистка области для рисования (компонент Image1), Button2 – создание объекта класса MyPoint и рисование точки, Button3 – создание объекта класса MyEllipce и рисование окружности, Button4 – создание объекта класса MyCylinder и рисование цилиндра, компоненты CSpinEdit1 – CSpinEdit4 для задания координат точки и размеров фигур, а также перемещения объектов в области рисования.

Заголовочный файл Myclass.h

```
//***************** class MyPoint ************************
class MyPoint {
  int x,y;
 public:
  int GetX(){return x;}
  int GetY(){return y;}
  void SetX(int x1){x=x1;}
  void SetY(int y1){y=y1;}
  MyPoint(int xx=0,int yy=0){x=xx;y=yy;}
  virtual void Show();
  virtual bool Checking();
```

```
 // Проверка, не выходит ли точка за пределы области рисования ?
  virtual void MoveTo(int x1,int y1);
};
//**************classs MyEllipse ************************
class MyEllipse:public MyPoint {
  int xa,yb;
 public:
  int GetXA(){return xa;}
  int GetYB(){return yb;}
  void SetXA(int xa1){xa=xa1;}
  void SetYB(int yb1){yb=yb1;}
  MyEllipse(int a=0,int b=0,int c=0,int d=0):MyPoint(a,b)
    {xa=c;yb=d;}
  virtual void Show();
  virtual bool Checking();
 // Проверка, не выходит ли окружность за пределы области рисования ?
};
//***************** class MyCylinder ***********************
class MyCylinder:public MyEllipse {
  int h;
 public:
  int GetH(){return h;}
  void SetH(int h1){h=h1;}
  MyCylinder(int X=0,int Y=0,int R=0,int H=0):MyEllipse(X,Y,R,R/2)
    {h=H;}
  void Show();
  bool Checking();
// Проверка, не выходит ли цилиндр за пределы области рисования ?
};
//****************** class MyCylinder ********************
bool ExistFigure=false;//Если ExistFigure==false,  объект не изображался
MyPoint *Figure;//- Указатель на базовый класс MyPoint.
```

Файл Unit1.Cpp

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Myclass.h"
#pragma package(smart_init)
#pragma link "CSPIN"
#pragma resource "*.dfm"
TForm1 *Form1;
__fastcall TForm1::TForm1(TComponent* Owner): TForm(Owner)
{ }
//********** Реализация класса MyPoint ***********************
void MyPoint::Show()
   ( Form1->Image1->Canvas->Pixels[x][y]=clRed;}
 bool MyPoint::Checking(){
```

```
//- Проверка, не выходит ли точка за пределы области рисования ?
if((x>=0)&&(x<=Form1->Image1->Width)&&(y>=0)&&
(y<=Form1->Image1->Height))
   return true; else return false;
   }
 void MyPoint::MoveTo(int x1,int y1)
   {    // если выходим за область рисования, то точку не сдвигаем.
    int a=x,b=y;
    x=x1;y=y1;
if(!Checking()){x=a;y=b;}
   }
//***********Реализация класса MyEllipse *********************
void MyEllipse::Show()
   {
Form1->Image1->Canvas->Ellipse(GetX()-xa,GetY()-yb,GetX()+xa,GetY()+yb);
   }
// Проверка, не выходит ли окружность за пределы области рисования ?
 bool MyEllipse::Checking()
{
    int rabx=GetX(),raby=GetY();
    SetX(GetX()-xa);SetY(GetY()-yb);
    if(MyPoint::Checking())
      {
       SetX(GetX()+2*xa);SetY(GetY()+2*yb);
       if(MyPoint::Checking())
         {
          SetX(rabx);SetY(raby); // восстанавливаем координаты x,y.
          return true;
         }
      }
    SetX(rabx);SetY(raby); // восстанавливаем координаты x,y.
    return false;
   }
//**********Реализация класса MyCylinder ***********************
void MyCylinder::Show()
   {
    MyEllipse::Show();
    Form1->Image1->Canvas->MoveTo(GetX()-GetXA(),GetY());
    Form1->Image1->Canvas->LineTo(GetX()-GetXA(),GetY()-h);
    Form1->Image1->Canvas->MoveTo(GetX()+GetXA(),GetY());
    Form1->Image1->Canvas->LineTo(GetX()+GetXA(),GetY()-h);
    SetY(GetY()-h);
    MyEllipse::Show();
    SetY(GetY()+h);
   }
bool MyCylinder::Checking()
{
```

```
    if(MyEllipse::Checking())
      {
       SetY(GetY()-h);
       if(MyEllipse::Checking()){SetY(GetY()+h);return true;}
      }
    return false;
   }
void __fastcall TForm1::Button1Click(TObject *Sender)
{ // Чистим "графическое" окно
 Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));
 ExistFigure=false;
}
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 if(ExistFigure==true)delete Figure;
 Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));
  // Рисуем точку, проверив её попадание в область рисования
 Figure=new MyPoint(CSpinEdit1->Value,CSpinEdit2->Value);
 ExistFigure=true;
 if(Figure->Checking()) Figure->Show();
 else ShowMessage("Точка не попадает  в область рисования!");
}
void __fastcall TForm1::Button3Click(TObject *Sender)
{
 if(ExistFigure==true) delete Figure;
 Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));
  // Рисуем окружность, проверив её попадание в область рисования
 Figure=new MyEllipse(CSpinEdit1->Value,CSpinEdit2->Value,
CSpinEdit3->Value,CSpinEdit3->Value);
 ExistFigure=true;
 if(Figure->Checking())Figure->Show();
 else ShowMessage("Окружность не помещается в область рисования!");
}
void __fastcall TForm1::Button4Click(TObject *Sender)
{
 if(ExistFigure==true) delete Figure;
 Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));
 // Рисуем цилиндр, проверив его попадание в область рисования
 Figure=new MyCylinder(CSpinEdit1->Value,CSpinEdit2->Value,
CSpinEdit3->Value,CSpinEdit4->Value);
 ExistFigure=true;
 if(Figure->Checking())Figure->Show();
 else ShowMessage("Цилиндр не помещается в область рисования!");
}
void __fastcall TForm1::CSpinEdit1Change(TObject *Sender)
{
 if(ExistFigure==true){
```

```
    Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));
    Figure->MoveTo(CSpinEdit1->Value,CSpinEdit2->Value);
    Figure->Show();
   }
}
void __fastcall TForm1::CSpinEdit2Change(TObject *Sender)
{
 if(ExistFigure==true){
    Image1->Canvas->FillRect(Rect(0,0,Image1->Width,Image1->Height));
    Figure->MoveTo(CSpinEdit1->Value,CSpinEdit2->Value);
    Figure->Show();
   }
}
```

Задания для самостоятельного решения

При выполнении данной работы необходимо определить базовый класс и производные от него классы. Предусмотреть передачу аргументов конструкторам базового класса; использование виртуальных и перегруженных функций; обработку исключительных ситуаций.

Вариант А

В следующих заданиях требуется создать базовый класс (как вариант абстрактный базовый класс) и определить общие методы show ( ), get ( ), set ( ) и другие, специфические для данного класса. Создать производные классы, в которые добавить свойства и методы.

Часть методов переопределить. Создать массив объектов базового класса и заполнить объектами производных классов. Объекты производных классов идентифицировать конструктором по имени или идентификационному номеру.

Вызвать метод show ( ) базового класса и просмотреть массив объектов.

Использовать объекты для моделирования реальных ситуаций.

1. Создать базовый класс «Транспортное средство» и производные классы «Автомобиль», «Велосипед», «Повозка». Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

2. Создать базовый класс «Грузоперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость перевозки для указанных городов и расстояний.

3. Создать аналогичный базовый класс «Пассажироперевозчик» и производные классы «Самолет», «Поезд», «Автомобиль». Определить время и стоимость передвижения.

4. Изменить задания 1–3, чтобы базовый класс стал абстрактным. Сделать некоторые методы абстрактными.

5. Создать базовый класс «Учащийся» и производные классы «Школьник» и «Студент». Создать массив объектов базового класса и заполнить этот массив объектами. Показать отдельно студентов и школьников.

6. Создать базовый класс «Музыкальный инструмент» и производные классы «Ударный», «Струнный», «Духовой». Создать массив объектов «Оркестр». Выдать состав оркестра, переопределив метод.

7. Определить базовый класс «Множество» и производный класс «Кольцо» (операции сложения и умножения обе коммутативные и ассоциативные, связанные законом дистрибутивности; сложение обладает обратной операцией – вычитанием). Ввести кольца целых чисел, многочленов, систему классов целых чисел, сравнимых по модулю. Кольцо является полем, если в нем определена операция деления, кроме деления на нуль. Рациональные числа, дробно рациональные функции.

8. Создать абстрактный класс «Работник фирмы» и производные классы «Менеджер», «Администратор», «Программист».

9. Создать базовый класс «Домашнее животное» и производные классы «Собака», «Кошка», «Попугай» и др. С помощью конструктора установить имя каждого животного и его характеристики.

10. Создать базовый класс «Садовое дерево» и производные классы «Яблоня», «Вишня», «Груша» и др. С помощью конструктора автоматически установить номер каждого дерева. Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.

Вариант Б

1. Создать класс Item (единица хранения в библиотеке), содержащий данные-члены: invNumber – инвентарный номер и taken – взято на руки или имеется в наличии, а также методы:

```
virtual void Show(); //показать  информацию о единице хранения
bool isAvailable(); // есть ли единица хранения в наличии ?
int GetinvNumber(); //возвращает инвентарный номер
void Take(); //  операция «взять»
void Return(); //  операция «вернуть»
```

Построить производные классы Book и Magazin. Класс Book содержит данные-члены: author, title, publisher, year и методы: Author(); Title(); Publisher(); YearOf Publishing(); Show().

Класс Magazin включает данные-члены: volume; number; year; title и методы: Volume(); Title(); Number(); Year(); Show().

2. Создать базовый класс Polygon (многоугольник). Класс должен содержать методы для рисования многоугольника, вычисления периметра, нахождения площади и др. Построить производный класс Triangle (треугольник), содержащий также методы для нахождения точки пересечения медиан, длин медиан, длин биссектрис, координат точек пересечения биссектрис, высот треугольника.

3. Создать абстрактный класс Shape для рисования плоских фигур. Построить производные классы Square (квадрат, который характеризуется координатами левого верхнего угла и длиной стороны), Circle (окружность с заданными координатами центра и радиусом), Ellipse (эллипс с заданными координатами вершин описанного вокруг него прямоугольника), позволяющие рисовать указанные фигуры, а также передвигать их на плоскости.

4. Создать класс CPoint – точка и производные от него классы CcoloredPoint и CLine. На основе классов CcoloredPoint и CLine создать класс CcoloredLine. Все классы должны иметь методы для установки и получения значений всех координат, а также изменения цвета и получения текущего цвета.

5. Описать базовый класс Stroka. Обязательные данные-члены класса: указатель типа char – для хранения строки; значение типа int – длина строки.

Методы: конструктор без параметров; конструктор, принимающий в качестве параметра С-строку (заканчивается нулевым байтом); конструктор копирования; получение длины строки; очистка строки (сделать строку пустой); деструктор.

Описать производный класс «БИТОВАЯ_СТРОКА» (строки данного класса могут содержать только символы ‘0’ и ‘1’). Если в основе инициализирующей строки встретятся любые символы, отличные от допустимых, то БИТОВАЯ_СТРОКА становится пустой. Содержимое строки рассматривается как двоичное представление целого числа со знаковым разрядом. Отрицательные числа хранятся в дополнительном коде.

Обязательные методы: конструктор без параметров; конструктор, принимающий в качестве параметра С-строку; конструктор копирования; деструктор; изменение знака числа (перевод числа в дополнительный код).

Переопределить следующие операции (длина строки результата в случае необходимости расширяется влево знаковым разрядом): присваивание; сложение (+); проверка на равенство (==).

6. Создать производный класс «СТРОКА10» (целое неотрицательное десятичное число) от класса «СТРОКА» (описание приведено выше).

Методы: конструктор без параметров; конструктор, принимающий в качестве параметра С-строку; конструктор копирования; деструктор; метод, определяющий, можно ли представить данное число в формате int; метод, определяющий, равно ли число нулю; метод, возвращающий представление числа в виде целого (int); метод, удаляющий незначащие нули.

Переопределить операции: сложение (+); проверка на больше (по значению) (>); проверка на меньше (<); присваивание (=).

7. Создать производный класс «БУЛЕВ ВЕКТОР» (**BoolVector**) от класса **Vector**. Компоненты принимают значения из множества {0,1}.

Методы: конструктор без параметров; конструктор, принимающий в качестве параметров указатель на массив целого типа (если элементы массива содержат числа, отличные от 0 и 1, то создается пустой вектор) и размер вектора; конструктор копирования; деструктор; метод, возвращающий число единиц в векторе; метод, возвращающий позицию самой левой единицы в векторе.

Переопределить операции: поразрядная конъюнкция (&); поразрядная дизъюнкция (|); поразрядная инверсия (~); поразрядная операция ИСКЛЮЧАЮЩЕЕ ИЛИ (^); присваивание (=).

8. Создать производный класс «ТРОИЧНЫЙ ВЕКТОР» от класса **Vector**. Компоненты принимают значения из множества {0,1,2}.

Методы: конструктор без параметров; конструктор, принимающий в качестве параметров указатель на массив целого типа и размер вектора; конструктор копирования; деструктор; проверка двух векторов на ортогональность (два троичных вектора называются ортогональными, если в них существует пара одноименных компонент, имеющих в одном из векторов значение 0, а в другом – 1); метод, возвращающий число компонент в векторе, принимающих значение 2.

Переопределить операции: присваивание (=); поразрядная конъюнкция (пересечение) двух не ортогональных векторов (&): 0&0=0, 1&1=1, 2&2=2, 0&2=0, 2&0=0, 1&2=1, 2&1=1; индексирование ([]).

9. Создать производный класс «БУЛЕВА МАТРИЦА» от класса «ЦЕЛОЧИСЛЕННАЯ МАТРИЦА».

Методы: конструктор без параметров; конструктор, принимающий в качестве параметров целочисленный двумерный массив, содержащий матрицу, и ее размеры *n* и *m*; конструктор копирования; деструктор; метод возвращает число единиц в матрице; метод, возвращающий

α-каноническую матрицу (в исходной матрице удалены повторяющиеся строки; строки составлены в порядке возрастания неотрицательных чисел, в качестве двоичных кодов которых рассматриваются данные строки).

Переопределить операции: поэлементная конъюнкция двух матриц (&); поэлементная дизъюнкция двух матриц (|); поэлементная операция ИСКЛЮЧАЮЩЕЕ ИЛИ двух матриц (^); произведение двух матриц A=[a_{ij}] и B=[b_{jk}], где i=$\overline{1,n}$, j=$\overline{1,m}$, k=$\overline{1,l}$ (*). При вычислениях операция целочисленного умножения заменяется конъюнкцией, а сложение – дизъюнкцией; присваивание (=).

10. Создать производный класс «МИНОР» для базового класса «МАТРИЦА» размерности $n \times m$. Переопределить для производного класса операции и методы (см. лаб. работу № 2, задание № 4).

11. Расширить возможности стандартного класса Time, чтобы можно было выводить время дня: утро, вечер и т. д.

12. Расширить возможности стандартного класса Date, чтобы можно было выводить время года: зима, лето и т. д.

13. Расширить возможности класса Annotation, чтобы можно было выводить время и дату изменения аннотации.

14. Расширить возможности класса Dictionary, чтобы можно было выводить дату последнего изменения в словаре.

15. Расширить возможности класса File, чтобы можно было выводить время и дату создания файла.

16. Расширить возможности класса Stack, чтобы можно было выводить время последнего сеанса работы со стеком.

17. Определить базовый класс для работы с прямоугольными матрицами, предусмотрев ввод-вывод матриц и выполнение следующих операций: сложение матриц; умножение матрицы на скаляр; перестановка строк матрицы по заданному вектору транспозиции; перестановка столбцов матрицы по заданному вектору транспозиции. В производном классе реализовать указанные операции для квадратных матриц, добавив выполнение следующих операций: транспонирование матрицы; умножение матриц.

Тесты

1. Какая из записей является правильной записью абстрактного класса?

Варианты ответа:

1) abstract class A{virtual f()=0;};
*2) class A{virtual f()=0;};
3) class A{virtual f();};

2. Абстрактный класс – это класс, в котором:
1) есть хотя бы один виртуальный метод;
2) есть виртуальный конструктор;
3) есть виртуальный деструктор;
*4) есть чисто виртуальный метод.

3. Основная проблема множественного наследования состоит в:
1) замедлении выполнения программ;
2) необходимости явно указывать, к какому базовому классу принадлежит метод;
*3) возможности существования двух экземпляров объекта базового класса;
4) неэкономном расходовании памяти.

4. Если записано
```
class A{public:virtual void f(){cout<<1;}};
class B:public A {public:virtual void f(){cout<<2;}};
```
то что будет напечатано, если
```
B b;A &a=b; a.f(); ?
```
Варианты ответа:
*1) 2; 2) 21; 3) 12; 4) 1; 5) ошибка.

5. Если записано
```
class A {public:void f(){cout<<1;}};
class B:public A{public: void f(){cout<<2;}};
```
то что будет напечатано, если
```
B b; b.f(); ?
```
Варианты ответа:
*1) 2; 2) 21; 3) 12; 4) 1.

6. Если записано
```
class A{public:virtual void f(){cout<<1;}};
class B:public A{public:virtual void f(){cout<<2;}};
```
то что будет напечатано, если
```
B b; b.f(); ?
```
Варианты ответа:
*1) 2; 2) 21; 3) 12; 4)1; 5)ошибка.

Лабораторная работа № 6
Тема. Потоки, обработка исключительных ситуаций в C++

Теоретическое введение. В C++ ввод-вывод осуществляется через потоки. Потоки являются объектами соответствующих классов. При запуске программы автоматически открываются стандартные потоки ***cin, cout, cerr, clog.*** Последние два потока используются для вывода сообщений об ошибках. В файле ***iostream.h*** определены классы: ввода – ***istream***, вывода – ***ostream***, ввода-вывода – ***iostream***.

Для реализации файлового ввода-вывода небходимо включить файл ***fstream.h***, содержащий производные от ***istream*** и ***ostream*** классы ***ifstream, ofstream*** и ***fstream***, и объявить соответствующие объекты. Например:

ifstream in;//ввод
ofstream out;//вывод
fstream io;//ввод-вывод

После объявления потоков производится открытие файла, связывающее его с потоком с помощью функции ***open()*** или с помощью конструктора. Прототип функции ***open():***

void open (char *filename,int mode,int access);

Здесь **filename** – имя файла, включающее путь; ***mode*** – режим открытия файла (**ios::in** – открытие файла для чтения, **ios::out** – открытие для записи, **ios::binary** – открытие файла в двоичном режиме, по умолчанию в текстовом); **access:** 0 – файл со свободным доступом, 1 – только для чтения, 8 – архивный файл. Файл закрывается с помощью функции ***close().***

Для чтения-записи здесь можно использовать перегружаемые оператор-функции >> и << или использовать методы классов. Для ввода-вывода одного символа используются функции:

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

Для записи и считывания блоков двоичных данных используются функции считывания-записи *n* байт в буфер или из буфера:

istream &read(unsigned char *buf, int n);
ostream &write(const unsigned char *buf, int n);

Обработка исключительных ситуаций. В программах на C++ следует использовать механизм обработки исключительных ситуаций. Операторы программы при обработке исключительных ситуаций располагаются в блоке ***try***. Перехватывается и обрабатывается исключительная ситуация в блоке ***catch***. Форма операторов ***try-catch*** следующая:

try {/*блок try*/ }
catch(type1 arg){/*блок catch*/}

С блоком ***try*** может связываться несколько блоков ***catch***. Выполняется тот блок ***catch***, для которого тип аргумента соответствует типу возникшей исключительной ситуации. При этом ее значение присваивается аргументу ***catch***. Если ошибка имеет место внутри блока ***try***, она может генерироваться с помощью ***throw***, после чего управление передано блоку ***catch.*** В случае необходимости перехвата всех исключительных ситуаций независимо от типа используется

catch(…)
{/*тело*/}

Для функций, вызываемых из блока ***try***, можно указать число типов исключительных ситуаций, которые будет генерировать функция:

тип имя (список аргументов) throw(список типов)
{/*тело*/}

Пример. Из текстового потока ввести информацию о студентах и записать ее в виде дерева. Отсортировать или по фамилии, или по баллу (вывести тех студентов, у которых балл выше среднего). Использовать потоки ввода-вывода и шаблоны.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
struct Student{
	int num;
	char surname[10];
	int group;
	int balls;
	friend ostream &operator<< (ostream &stream, Student stud){
		stream << " " << stud.num << " " << stud.surname << " " << stud.group
		<< " " << stud.balls;
		return stream;
	}
	friend istream &operator>> (istream &stream, Student &stud){
		stream >> stud.num >> stud.surname >> stud.group >> stud.balls;
		return stream;
	}
};
struct node{
	Student info;
	node *nextl, *nextr;
	node (){
		info.num = info.group = info.balls=0;
		nextl = nextr = 0;
	}
	node (Student newinfo){
		info = newinfo;
		nextl = nextr = 0;
	}
};
template <class T, class T1> class tree{
	public:
	T *root;
	tree() { root = 0; }
	void push (T *& wer, T1 dat, int n){
		if (wer == 0){
			try{
			 wer = new T;
			  if(!wer) throw 1;
			 wer->nextl = 0; wer->nextr = 0; wer->info = dat;
```

```
                    }
                    catch (int mthrow) {cout<<"No memory!"<<endl;return;}
            }
        else if (n == 1)
            if (strcmp(dat.surname,wer->info.surname) < 0) push (wer->nextl, dat, 1);
                else push (wer->nextr, dat, 1);
            else
                if (dat.balls > wer->info.balls) push (wer->nextl, dat, 2);
                else push (wer->nextr, dat, 2);
    }
    void insert (T1 dat, int n){
        if (root == 0) root = new T(dat); else push (root, dat, n);
    }
    void look (ostream &stream, T *&wer){
        if (wer != 0){
            look (stream, wer->nextl);
            stream << " " << wer->info << endl;
            look (stream, wer->nextr);
        }
    }
    friend ostream &operator<< (ostream &stream, tree ob)
        { ob.look (stream, ob.root); return stream; }
};
void main(){
  int m;
  do{
  cout << "1. Sort with names\n";
  cout << "2. Sort with balls\n";
  cout << "3. Exit\n";
  cin >> m;
  switch (m){
      case 1: {
            tree<node, Student> q;
            node *n;
            ifstream infile("stud.txt");
            while(!infile.eof()){
                    Student c;
                    infile >> c;
                    q.insert(c, 1);
                }
             infile.close();
             cout<<q;
             break;
            }
      case 2: {
            tree<node, Student> q;
            node *n;
```

```
            ifstream infile("stud.txt");
            Student *c;
            c = new Student;
            int i = 1;
            float s = 0;
            while(!infile.eof()){
                        infile >> c[i];
                        s+=c[i].balls;
                        i++;
                  }
             for (int j=1; j<=i; j++)
                  if (c[j].balls > s/i)
                        q.insert(c[j], 2);
             infile.close();
             cprintf("  Miide ball is %1.3f",s/i);
             cout<<'\n' << q;
             break;
            }
      case 3: {return;}
      default: {cout<<"Error! Try again\n"; break;}
 }
      getch();
      clrscr();  }
 while(m != 3);
 return;}
```

Задания для самостоятельного решения

При выполнении приводимых ниже заданий можно использовать классы, разработанные в лабораторных работах № 1–3. Осуществлять контроль состояния потоков. В случае возникновения ошибок потоков генерировать и обрабатывать исключительные ситуации. Для соответствующих классов перегрузить операции вставки в поток и извлечения из потока. При динамическом выделении памяти предусмотреть обработку исключения, возникающего при нехватке памяти.

I

а) Для класса **Student** (лаб. работа № 1) предусмотреть ввод данных из файла. Полученные при выполнении лаб. работы № 1 списки студентов вывести в файл.

То же задание для классов:

б) **Abiturient** (лаб. работа №1);

в) **Aeroflot** (лаб. работа № 1);

г) **Worker** (лаб. работа № 1);

д) **Train** (лаб. работа № 1);

е) **Product** (лаб. работа № 1);

ж) **Patient** (лаб. работа № 1);

з) **Bus** (лаб. работа № 1);

и) **Customer** (лаб. работа № 1);

к) **File** (лаб. работа № 1);

л) **Word** (лаб. работа № 1);

м) **House** (лаб. работа № 1);

н) **Phone** (лаб. работа № 1);

о) **Person** (лаб. работа № 1).

II

а) При выполнении задания № 1 лаб. работы № 2 (класс **Complex**) предусмотреть формирование массива объектов путем считывания комплексных чисел из файла. Результат также вывести в файл.

То же задание для классов:

б) **Fraction** (лаб. работа № 2);

в) **Vector** (лаб. работа № 2). Предусмотреть обработку исключения при динамическом выделении памяти;

г) **Matrix** (лаб. работа № 2);

д) **Polynom** (лаб. работа № 2);

е) **Stack** (лаб. работа № 2);

ж) **Строка** (лаб. работа № 2);

з) **Set** (лаб. работа № 2);

и) **«Массив строк»** (зад. № 10 лаб. работы № 2);

к) **«Булев вектор»** (лаб. работа № 2);

л) **«Троичный вектор»** (лаб. работа № 2);

м) **«Булева матрица»** (лаб. работа № 2).

III

Те же задания, что и в разделах I и II, но для классов, реализующих работу с динамическими структурами данных (см. лаб. работу № 3).

Тесты

1. Если имеется код char a[8]; cin>>a; и вводится текст “Hello World”, то что будет в массиве a?

Варианты ответа:

1) “Hello W”; 2) “Hello Wo”; *3) “Hello”; 4) “Hello World”; 5) “lo World”.

2. Что будет выведено в результате

double x=12.4;

cout<<setw(5)<<x<<setw(3)<<setfill(‘*’)<<””<<endl; ?

Варианты ответа:

1) 12.40***; *2) 12.4***; 3) 12.4 ***; 4) 12.40; 5) .124e2***.

3. Если имеется код int x; cin>>x; и вводится “1.2”, что будет в переменной x?

Варианты ответа:

*1) 1; 2) 2; 3) 1.2; 3) другое; 4) произойдет ошибка.

4. Какой из классов используется для вывода строк на экран?

Варианты ответа:

1) strstream; *2) ostream; 3) ofstream; 4) istream; 5) ifstream.

5. Каким будет результат работы программы:

```
#include <iostream.h>
void main (){
char A[]=”ABC”;
char *U=&A[2];
cout<<”\n”<<*U--<<*U--<<*U<<endl; } ?
```

Варианты ответа:

1) BAA; *2) CBA.

6. Если имеется код double x; cin>>x; и вводится “12-3”, то что будет в переменной x?

Варианты ответа:

1) 9.0; *2) 12.0; 3) другое; 4) произойдет ошибка.

7. Для того, чтобы выполнить чтение из файла с произвольной позиции, надо использовать объект класса:

1) strstream; 2) ostream; 3) ofstream; 4) istream; *5) ifstream.

8. Что будет выведено при выполнении оператора throw C, если заданы классы

```
class A {…};
class B: public A {…};
class C: public A {…};
```

а обработка исключительной ситуации записана

```
catch (B&b) {cout<<1;}
catch (C&c) {cout<<2;}
catch (A&a) {cout<<3;}
catch (…) {cout<<4;} ?
```

Варианты ответа:

1) 1; *2) 2; 3) 3; 4) 4; 5) 34; 6) 234.

9. Если в конструкторе класса

```
class A {
char *ptr;
public:
A () {ptr=new char [size]; Init ();}
~A () { if (ptr) delete [] ptr;}
};
```

произойдет исключительная ситуация, будет ли потеряна память при откате по стеку?

Варианты ответа:

1) да, будет во всех случаях; 2) будет, только если объект класса создавался с помощью new; *3) будет, если создавалась автоматическая переменная класса A; 4) нет, не будет.

10. Об ошибке в конструкторе класса может сигнализировать:

1) возвращаемое значение; *2) исключительная ситуация; 3) вызов деструктора сразу в конструкторе.

11. Что будет выведено, если заданы классы

```
class A {…};
class B: public A {…};
class C: public A {…};
```

а операторы throw и catch записаны так:

```
throw A;
catch (B&b) {cout<<1;}
catch (C&c) {cout<<2;}
catch (A&a) {cout<<3;}
catch (…) {cout<<4;} ?
```

Варианты ответа:

1) 1; 2) 2; *3) 3; 4) 4; 5) 34; 6) 234.

12. Оператор throw без аргументов

*1) повторно вызывает обрабатываемую исключительную ситуацию;

2) вызывает исключительную ситуацию типа Exception.

13. Что будет выведено, если заданы классы

```
class A {…};
class B: public A {…};
class C: public B {…};
```

а операторы throw и catch записаны так:

```
throw C;
catch (B&b) {cout<<1;}
catch (C&c) {cout<<2;}
catch (A&a) {cout<<3;}
catch (…) {cout<<4;} ?
```

Варианты ответа:

*1) 1; 2) 2; 3) 3; 4) 4; 5) 1234; 6) 234.

ЛИТЕРАТУРА

1. *Страуструп, Б.* Язык программирования С++/*Б. Страуструп.* СПб.:БИНОМ, 1999.
2. *Шилдт, Г.* Самоучитель С++/*Г. Шилдт*. 3-е изд. СПб.:BXV-Петербург, 2002.
3. *Эккель, Б.* Философия С++. Введение в стандартный С++/*Б. Эккель.* 2-е изд. СПб.:Питер, 2004.
4. *Эккель, Б.* Философия С++. Практическое программирование/*Б. Эккель, Ч. Эллисон* . СПб.:Питер, 2004.
5. *Павловская, Т.А.* С++. Объектно-ориентированное программирование: Практикум/ *Т. А. Павловская, Ю. А. Щупак.* СПб.:Питер, 2004.
6. *Глушаков, С.В.*Язык программирования С++/*С. В. Глушаков, А. В. Коваль, С. В. Смирнов.* Харьков:Фолио, 2002.
7. *Фридман, А. Л.* Язык программирования С++. Курс лекций/*А. Л. Фридман.* М.:ИНТУИТ, 2003.

СОДЕРЖАНИЕ

Учебное издание

Романчик Валерий Станиславович
Люлькин Аркадий Ефимович

С++. ЛАБОРАТОРНЫЕ РАБОТЫ
по курсу «Методы программирования»

Учебно-методическое пособие
для студентов механико-математического
факультета

Технический редактор __
Корректор

Ответственный за выпуск *В. В. Власова*

Подписано в печать __.__.2005. Формат 60x84/16. Бумага офсетная. Печать офсетная.
Усл.печ.л. ____. Уч.-изд.л. . Тираж 100 экз. Зак.

Белорусский государственный университет.
Лицензия ЛВ №315 от 14.07.98.
220050, Минск, пр. Ф. Скорины,4.

Отпечатано в издательском центре БГУ.
220030, Минск, ул. Красноармейская, 6.