Язык программирования Си++

Иванов А.П., Князева О.С.

Семинар 3. Наследование и полиморфизм

Для эффективной разработки программ удобно использовать иерархическое упорядочение понятий и объектов. Такое упорядочение позволяет легче справляться со сложностью разрабатываемых программ, сделать логику их работы более простой и понятной. Иерархия понятий реализуется в виде древовидной структуры, в основе которой лежит наиболее общее понятие. В языке Си++ подобная структура реализуется через механизм производных классов, наследующих свойства базовых классов.

Производные классы – развитие классов, определенных ранее, имеющие доступ к `protected` и `public` части базового класса. Производные классы наследуют все свойства базовых классов, при этом некоторые свойства могут быть запрещены к наследованию, другие – можно изменить при наследовании, а третьи – могут быть добавлены к свойствам базового класса.

1. Права доступа к членам базового класса

Права доступа к членам базового класса из производного класса определяются модификатором доступа, задаваемом при описании производного класса.

```
class Base {
  private:
  ..............
  protected:
  ..............
  public:
  ..............

}  ;
class Derived : public Base { // здесь может применяться private и protected
  public:
  ..............
};
```

Естественно, нет никаких ограничений на состав функций и полей данных, определяемых в производном классе дополнительно по отношению к базовому.

Таблица прав доступа к членам базового класса в производном классе

Модификатор доступа, указанный при наследовании	Право доступа в базовом классе	Наследуемое право доступа в производном классе
private	**private** **protected** **public**	**не доступен** **private** **private**
protected	**private** **protected** **public**	**не доступен** **protected** **protected**
public	**private** **protected** **public**	**не доступен** **protected** **public**

2. Дружественные функции и классы

В ряде случаев бывает удобно получить доступ к приватным и защищенным (`private`, `protected`) членам класса из функций, не являющихся членами этого класса. Для того в тело класса нужно вставить прототип такой функции, и перед ним поставить ключевое слово `friend`.

В тех случаях, когда классы тесно «взаимодействуют» друг с другом (то есть, когда объекты одного класса являются аргументами членов функции другого класса) бывает удобно разрешить доступ таким классам к приватным и защищенным (`private`, `protected`) членам этих классов. Для этого в теле класса, к членам которого открывается доступ, нужно описать класс, получающий доступ, с ключевым словом `friend`.

Отметим, что помимо воли автора этого класса получить доступ к его защищенным полям и методам – нельзя, то есть, чтобы этот доступ получить – в нем самом нужно сделать дополнительное объявление дружественных классов или функций.

```
#include <ostream>

class CVector  {
  private:
     int     dim;  // размерность вектора
     float  *ptr;     //  указатель  на  область  памяти,  содержащую  элементы
вектора.
  private:
     ...
     // дружественный оператор вывода:
     friend ostream& operator<< (ostream& os, const CVector &v);
     friend class CMatrix; // дружественный класс
     ...
};

ostream& operator<< (ostream& os, const CVector &v)
{
   for (int n=0; n < v.dim; n++) {
       os << "[" << n << "] " << v.ptr[n] << endl;
    }
    return os;
}

class CMatrix  {
  private:
     int   cx, cy;  //размерности матрицы
     float *ptr;    // указатель на память, содержащую элементы матрицы

  public:
     ...
     // оператор умножения матрицы на вектор:
     CVector& operator * (CVector &v);
     ...
};
```

3. Перегрузка при наследовании

В производных классах могут существовать функции-члены базового класса с именем, совпадающим с именем какой-либо функции базового класса.

```
class Base {
  ...
  public:
     void mf(void) { cout << 1; }
}  ;
class Derived : public Base {
  ...
  public:
     void mf(void) { cout << 2; }
};

Base    x;
Derived y;
...
x.mf();  // 1, вызывается функция базового класса Base
y.mf();  // 2, вызывается функция производного класса Derived;
```

4. Полиморфизм: виртуальные методы класса

Позволяют выбирать методы с одним и тем же именем через указатель функции в зависимости от типа реального объекта, на который указывает указатель, *а не в зависимости от типа указателя*.

```
class Base {
  ...
  public:
     virtual void mf(void) { cout << 1; }
     void mfstd(void) { cout << 10; }
}  ;
class Derived : public Base {
  ...
  public:
     void mf(void) { cout << 2; }
     void mfstd(void) { cout << 20; }
}  ;

Base  x;
Derived y;

Base *px = &x;
Derived *py = &y;
Base *pxy = &y;

px->mf();  // 1, вызывается функция базового класса Base

py->mf();  // 2, вызывается функция производного класса Derived;

pxy->mf(); // 2, полиморфный вызов: мы не знаем, что работаем с Derived,
           // так как располагаем указателем на Base, но метод вызывается
           // из Derived

pxy->mfstd(); // 10, так как вызов не полиморфный – в базовом классе функция
              // не виртуальная.
```

5. Абстрактные классы

Абстрактные классы содержат, по крайней мере, одну чистую виртуальную функцию. В программе не могут быть определены объекты абстрактных классов или ссылки на них, но можно определить и использовать указатели на объекты абстрактных классов.

```
class CPoint {
  public:
     int x,y;
  public:
     CPoint (int nx=0, int ny=0) : x(nx), y(ny) {} // конструктор
     CPoint (const CPoint& src) : x(src.x), y(src.y) {} // конструктор копии
};

class CShape { // абстрактный класс, так как в нем есть чистая функция
   protected:
      CPoint center; // центр объекта
    public:
      CShape (const CPoint &nc): center(nc) {} // конструктор
      // перемещение фигуры, невиртуальная функция:
      void MoveTo( int nx=0, int ny=0 ) { center.x = nx; center.y = ny; }

      // чистая функция (pure function) для подсчета площади:
      virtual double Square() = 0;
};

class CCircle : public CShape {
   public:
      int radius; // радиус круга
      CCircle(int nx=0, int ny=0, int rad=0) :
              CShape(CPoint(nx,ny)), radius(rad) {}

   public:
      double Square() { return double(3.14159) * radius * radius; }
};

class CQuadrat : public CShape {
   public:
      int side; // сторона квадрата
      CQuadrat (int nx=0, int ny=0, int ns=0) :
              CShape(CPoint(nx,ny)), side(ns) {}
   public:
      double Square() { return side * side; }
};

CCircle c1(1,2,1), c2(2,3,8);
CQuadrat q1(-2,0,3), q2(-2,0,5);
CShape* shapes[4] = { &c1, &q1, &c2, &q2 };

// подсчет площади всех фигур:

double s = 0;
for ( int i=0; i < 4; i++ ) {
  s += shapes[i]->Square();
}
```

6. Множественное наследование

Множественное наследование используется тогда, когда необходимо наделить производный класс свойствами более чем одного класса. В тех случаях, когда в базовых

классах содержатся методы с одинаковыми именами, доступ к ним из производного класса осуществляется с помощью явного указания имени класса, членами которого они являются.

```
class A {
  public:
    ...
    void  mf(void);
};

class B {
  ...
  public:
   ...
    void  mf(void);
};

class C  : public A, public B {
  public:
  ...
};

C c;
c.A::mf(); // вызывается функция класса A
c.B::mf(); // вызывается функция класса B
```

7. Виртуальные базовые классы

Для классов, порожденных от производных классов, с общим виртуальным классом, существует только один экземпляр объекта общего базового класса.

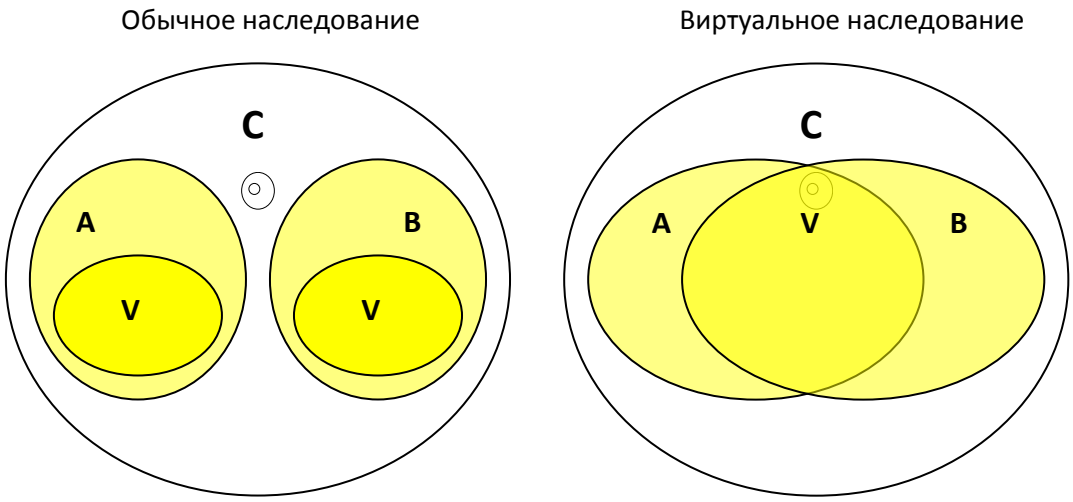
```
class V {
  ...
  public:
    ...
    void  Vmf(void);
};

class A: virtual public V, public S {
  ...
  public:
    ...
    void  Amf(void);
};

class B : virtual public V, public  T {
  ...
  public:
   ...
    void  Bmf(void);
};

class C  : public A, public B {
  ...
  public:
};
```

Класс `V` является единственным общим объектом внутри класса `C`, то есть - общим для классов `A` и `B`, входящих в состав класса `C`.



8. Пример. Наследование вектора.

Унаследуем динамический массив из предыдущего задания таким образом, чтобы в элементах базового вектора хранились рациональные числа. Размерность производного вектора при этом будем считать вдвое меньшей, по сравнению с базовым классом.

```
class Vector;
class Ratio;


class ratiovector: public Vector
{
public:
//========== Конструкторы ======//
// Конструктор – указываем количество пар, элементов в базовом классе
// вдвое больше:
      ratiovector(int N=0) : Vector(N*2) {}

// Конструктор копирования – не наследуется:
      ratiovector(const ratiovector& src) : Vector((const vector&)src) {}


//============= Размерность ==========//
int size() const { return Vector::size()/2; }

//============= Перегрузка [] ==========//
Ratio operator [] (int index)
{
// индекс – в производном классе, в базовом – удвоение координаты для v:
  return Ratio(v[2*index], v[2*index+1]);
}

//============= Присваивание ==========//
void Set(int index, const Ratio& src)
{
  v[2*index] = src.a; v[2*index+1] = src.b;
}

//============= Сложение ==========//
ratiovector& operator+=(ratiovector& src)
```

```
{
  if ( len != src.len ) return *this;

  for ( int k=0; k < src.len/2; k++ ) {
      int ch = v[2*k] * src.v[2*k+1] + v[2*k+1] * src.v[2*k];
      int zn = v[2*k+1] * src.v[2*k+1];
      v[2*k] = ch;
      v[2*k+1] = zn;
  }

  return *this;
}

// Деструктор - наследуется
};
```

Типовое задание 1: реализовать класс – наследник динамического массива из Задания № 2 с переопределением его функциональности: например, производный класс в элементах базового вектора должен хранить объекты из Задания № 1, размерность производного класса будет меньше размерности базового, оператор [ ] должен возвращать не числа, а соответствующие объекты.

Типовое задание 2: реализовать класс – наследник динамического массива из Задания № 2 с переопределением его функциональности: например, производный класс сам должен являться матрицей некоторой размерности.

1. Вариант

Базовый класс – массив вещественных чисел, производный класс – массив вещественных чисел, представленных в виде пар чисел x=M*10^e, где M – мантисса числа, e – экспонента (например, число 15.432=1.5432*10^(1), то есть: M=1.5432 e=1).

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания.

Переопределите операции вставки, удаления элемента, а также операторы: [ ], +=, -=, *=, /=. Арифметические операторы применяются поэлементно.

2. Вариант

Базовый класс – массив целых чисел, производный класс – массив рациональных чисел чисел, представленных по модулю числа N (то есть в виде пар: целая часть при делении на N, остаток от деления на N).

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания.

Переопределите операции вставки, удаления элемента, а также операторы: [ ], +=, -=, *=, /=. Арифметические операторы применяются поэлементно.

3. Вариант

Базовый класс – массив вещественных чисел, производный класс – массив двумерных векторов.

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания. Переопределите операции вставки, удаления элемента, а также операторы: [ ], +=, -=, *=. Умножение – произведение каждого вектора на матрицу 2x2, представленную отдельным классом.

4. Вариант

Базовый класс – массив вещественных чисел, производный класс – массив трехмерных векторов.

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания. Переопределите операции вставки, удаления элемента, а также операторы: [ ], +=, -=, *=. Умножение – векторное произведение векторов.

5. Вариант

Базовый класс – массив вещественных чисел, производный класс – массив пар чисел (x,y) с операциями:

`(x,y)+(z,k)=(xz,yk), (x,y)-(z,k)=(x/z,y/k), a(x,y)=(xa,ya)`

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания. Переопределите операции [ ], +=, -=, *= (число).

6. Вариант

Базовый класс – массив чисел, производный класс – массив комплексных чисел.

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания. Переопределите операции вставки, удаления элемента, а также операторы: [ ], +=, -=, *=. Определите метод класса, возвращающий минимальный по модулю вектор в массиве.

7. Вариант

Базовый класс – массив чисел, производный класс – массив рациональных чисел.

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания. переопределите операции вставки, удаления элемента, переопределите операции [ ], +=, -=, *=.

8. Вариант

Базовый класс – массив чисел, производный класс – массив дат, представленных тройками чисел: день, месяц, год.

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания.

Переопределите операции вставки, удаления элемента, переопределите операции [ ], +=, -=.

9. Вариант

Базовый класс – массив чисел, производный класс – массив времен, представленных тройками чисел: часы, минуты, секунды.

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания. Переопределите операции вставки, удаления элемента, переопределите операции [ ], +=, -=.

10. Вариант

Базовый класс – массив чисел, производный класс – массив квадратных матриц 2x2.

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания. Переопределите операции [ ], +=, -=, *=.

11. Вариант

Базовый класс – массив чисел, производный класс – массив квадратных трехчленов.

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания. Переопределите операции [ ], +=, -=, *= (число).

12. Вариант

Базовый класс – массив чисел, производный класс – массив полиномов 4 порядка с условием P(0)=0.

Определите в нем конструктор, деструктор, конструктор копирования, оператор присваивания. Переопределите операции [ ], +=, -=, *= (число).

13. Вариант

Базовый класс – массив чисел, производный класс – квадратная матрица 2x2 со скалярным произведением (A,B)=tr(AB) (tr – «след» матрицы, сумма ее диагональных элементов).

Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции доступа к элементу матрицы, +=, -=, *=, где умножение - скалярное произведение по приведенной выше формуле.

14. Вариант

Базовый класс – массив чисел, производный класс – массив полиномов 3 порядка.

Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции [ ], +=, -=, *= (число).

15. Вариант

Базовый класс – массив чисел, производный класс – квадратный трехчлен со скалярным произведением:

`(P,Q)=P(-1)Q(-1)+P(0)Q(0)+P(1)Q(1)`

Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции [ ], +=, -=, *=. Умножение – скалярное произведение по приведенной выше формуле.

16. Вариант

Базовый класс – массив чисел, производный класс – квадратный трехчлен со скалярным произведением:

`(P,Q)=P'(-1)Q'(-1)+P'(0)Q'(0)+P'(1)Q'(1)`

Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции [ ], +=, -=, *=. Умножение – скалярное произведение по приведенной выше формуле.

17. Вариант

Базовый класс – массив чисел, производный класс – квадратный трехчлен со скалярным произведением:

`(P,Q)=P (0)Q(0)+P'(0)Q'(0)+P''(0)Q''(0)`

Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции [ ], +=, -=, *=. Умножение – скалярное произведение по приведенной выше формуле.

18. Вариант

Базовый класс – массив чисел, производный класс – квадратная симметричная матрица 2x2, поддерживающая дополнительные операции умножения.
Определите в нем конструктор, деструктор, конструктор копирования.
Переопределите операции доступа к элементу матрицы, +=, -=,*= (число) и *= (матрица).

19. Вариант

Базовый класс – массив чисел, производный класс – матрица произвольной размерности, поддерживающая операции сложения и умножения на другие матрицы по правилам линейной алгебры.
Определите в нем конструктор, деструктор, конструктор копирования.
Переопределите операции доступа к элементу матрицы, +=, -= и *=.

20. Вариант

Базовый класс – массив чисел, производный класс – матрица произвольной размерности, поддерживающая операции вставки дополнительного ряда и столбца (оба – представлены векторами).
Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции доступа к элементу матрицы, операции вставки дополнительного ряда и столбца.

21. Вариант

Базовый класс – массив чисел, производный класс – матрица произвольной размерности, поддерживающая операции удаления любого ряда и столбца.
Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции доступа к элементу матрицы, операции удаления дополнительного ряда и столбца.

22. Вариант

Базовый класс – вектор чисел, производный класс – матрица произвольной размерности, поддерживающая операции сложения, вычитания, умножения на число и умножения на вектор.
Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции доступа к элементу матрицы, операции +=, -=, *= (число), *=(вектор).

23. Вариант

Базовый класс – массив чисел, производный класс – матрица размерности 3x3, сумма всех элементов которой равна нулю. Должны поддерживаться операции сложения и умножения на другие такие матрицы по правилам линейной алгебры.
Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции доступа к элементу матрицы, +=, -= и *=.

24. Вариант

Базовый класс – массив чисел, производный класс – матрица размерности 2x2 со скалярным произведением:
$(A,B)=a_{11}b_{11}+a_{12}b_{12}+a_{22}b_{22}$
Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции доступа к элементу матрицы, +=, -= и *=.

25. Вариант

Базовый класс – массив чисел, производный класс – евклидово пространство вектор-столбцов 3x1, ортогональных заданному вектору a.
Определите в нем конструктор, деструктор, конструктор копирования. Переопределите операции [ ], +=, -= и *= (число).