Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Чувашский государственный университет имени И.Н.Ульянова»

Лекция
Множественное наследование

Ильина Лариса Алексеевна

Иерархия классов

Производный класс может использоваться в качестве базового для другого производного класса, создавая многоуровневую иерархию классов. В этом случае исходный базовый класс является **косвенным** базовым классом для второго производного класса.

При таком наследовании конструкторы классов вызываются в порядке наследования, деструкторы в обратном порядке.

Если производный класс наследует иерархию классов, каждый производный класс должен передавать предшествующему в цепочке базовому классу все необходимые аргументы

Прямое наследование

Производный класс может прямо наследовать более одного базового класса, тогда используется расширенное объявление:

class *имя_производного_класса:*
сп_доступа имя_базового_1,
*сп_*доступа *имя_базового_2,*
...
сп*_доступа имя_базового_N*
{
//. . . тело производного класса
}

Спецификаторы доступа могут отличаться у разных базовых классов. Конструкторы выполняются слева направо в порядке, заданном в объявлении производного класса, а деструкторы – в обратном.

Передача аргументов конструкторам

Если конструкторам базовых классов необходимы аргументы, то производный класс передает их, используя расширенную форму:

```
констр_произв_класса(список_арг):
имя_базового1(список_арг),
имя_базового2(список_арг),
        ..., имя_базового_N(список_арг)
{/* тело конструктора производного
  класса*/}
```

Прямое наследование производного класса и косвенное наследование базового

```
#include <iostream>
using namespace std;
class B {
int a;
public:
B(int x) { a = x; }
int geta() { return a; }
};
class D1: public B {
int b;
public:
// передача переменной y классу B
D1(int x, int y): B(y)
{b = x;}
int getb(){ return b; }
};
```

Окончание программы

```
/*Прямое наследование производного класса и косвенное
наследование базового*/
class D2: public D1{
int c;
public:
D2(int x, int y, int z): D1(y, z)//передача аргументов классу Dl
{c = x;}
/* Поскольку базовые классы наследуются как открытые, класс D2
имеет доступ к открытым элементам классов B1 и D1 */
void show() {
cout << geta()<<" "<<getb()<< " ";
cout <<c << " \n " ;}
};
int main()
{D2 ob(1, 2, 3);
ob.show();
// функции geta() и getb() здесь тоже открыты
cout << ob.geta()<< " " << ob.getb() << endl;
return 0;
}
```

Производный класс прямо наследует два базовых класса

```
#include <iostream>
using namespace std;
class B1 {
int a;
public:
B1 (int x) { a = x; }
int geta() { return a; }
};
class B2 {
int b;
public:
B2(int x)
{b = x;}
int getb() { return b; }
};
```

Окончание программы

```
// Прямое наследование двух базовых классов
class D: public B1, public B2 {
int c;
public:
D(int x, int y, int z) : B1(z), B2(y)
{c = x;}
void show() {
cout << geta() << " " << getb()<<" ";
cout << c << "\n";}
};
int main()
{D ob(13, 24, 31);
ob.show();
return 0;
}
```