

Объектно-ориентированное программирование на алгоритмическом языке С++

МИРЭА, Институт Информационных технологий, кафедра Вычислительной техники

Автор: доцент, канд. физ.-мат. наук,

Путуридзе Зураб Шотаевич



Операторы new и delete

«указатель на объект класса» = new «имя класса» [(«аргументы»)]; delete «указатель на объект класса»;

```
#tinclude <iostream>
using namespace std;
int main() {
       int * p;
       p = \text{new int}; // выделение памяти для целого
       if (!p) {
               cout << "Ошибка выделения памяти\п";
               return 1;
       * p = 1000;
       cout << "Это целое, на которое указывает р: " << * р << "\п";
       delete p; // освобождение памяти
       return 0;
```



Операторы new и delete

```
#include <iostream>
using namespace std;
class samp {
      int i, j;
public:
      void set_ij ( int a, int b ) { i = a; j = b; }
int get_product() { return i * j; }
    main()
      samp * p;
      p = new samp; // выделение памяти объекту if (! p ) {
             cout << "Ошибка выделения памяти\n";
             return 1;
      p \rightarrow set ij (4, 5);
      cout << "Итог равен:" << p -> get product() << "\n";
      delete p;
      return 0;
```



Операторы new и delete

Для динамически размещаемого одномерного массива используется такая форма оператора new:

«указатель на массив» = new «имя типа» [«размер»];

Для удаления динамически размещенного одномерного массива используется следующая форма оператора delete:

delete [] «указатель на массив»;



Наследование

```
class «имя производного класса» : «спецификатор доступа» «имя базового класса»
```

«спецификатор доступа» — public | private | protected

Рассмотрим спецификаторы public и private



элементы

Наследование

Производный Базовый класс Производный класс класс public: : private : public Открытые public: элементы public: Открытые Открытые private: элементы элементы Закрытые private: элементы Открытые Закрытые элементы элементы private: Закрытые Закрытые элементы



Наследование

```
#Imperedude <iostream>
using namespace std;
class base {
         int x;
public:
        void setx ( int n ) { x = n; }
void showx ( ) { cout << x << "\n"; }
class derived: public base // Класс наследуется как открытый
         int y;
public:
        void sety ( int n ) { y = n; }
void showy ( ) { cout << y << "\n"; }</pre>
int main ( ) {
         derived ob;
        ob.setx (10); // доступ к члену базового класса ob.sety (20); // доступ к члену производного класса ob.showx (); // доступ к члену базового класса ob.showy (); // доступ к члену производного класса
         return \bar{0};
```



Защищенные члены классаразование в стиле hi tech

Производный класс

: public

public:

Открытые элементы

Открытые элементы

private:

Закрытые элементы

protected:

Защищенные элементы

Защищенные элементы

Базовый класс Производный public: класс Открытые : private элементы public: private: Открытые Закрытые элементы элементы private: protected: Закрытые Защищенные элементы элементы Закрытые элементы Закрытые элементы

8



Защищенные члены класса́разование в стиле hi tech

Производный класс

: protected

public:

Открытые элементы

private:

Закрытые элементы

protected:

Защищенные элементы

Открытые элементы

Защищенные элементы

Базовый класс

public:

Открытые

элементы

private:

Закрытые

элементы

protected:

Защищенные

элементы



Множественное наследование в стиле hi tech

```
class «имя производного класса» :
         «спецификатор доступа» «имя базового класса 1»
         [, «спецификатор доступа» «имя базового класса 2»]...
{
            . . . . .
};
```

«спецификатор доступа» — public | private | protected

еоднозначность при множественном наследовании

```
class base {
public:
  int i;
class derived1 : public base { .... };
class derived2 : public base { .... };
class derived3 : public derived1, public derived2 {
};
int main()
  derived3 ob;
  ob.i = 10;
                          // Неоднозначность!!!
  return 0;
```



Виртуальные базовые классызование в стиле hi tech

```
Определение области видимости
ob.derived1 :: i = 10;
class derived1 : virtual public base { .... };
class derived2 : virtual public base { .... };
class derived3 : public derived1, public derived2 {
int main()
 derived3 ob;
  ob.i = 10;
  return 0;
```



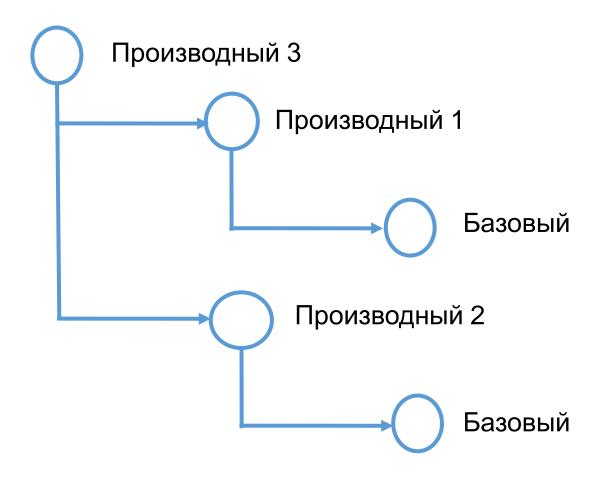
Виртуальные базовые классызование в стиле hi tech

Производный класс 3 Производный класс 1 Базовый класс Производный класс 2 Базовый класс

Производный класс 3 Производный класс 1 Производный класс 2 Базовый класс



Схема множественного наследования стиле hi tech



россий казатели на объекты производного класса

```
base * p;
                    // указатель базового класса
base base ob; // объект базового класса
derived derived ob; // объект производного класса
p = \& base ob;
             // для объекта базового класса
p = \& derived ob; // для объекта производного класса
Приведение типа указателя базового класса к производному
( ( derived * ) p ) -> show title ();
```



Исключение повторного добавления описания класса

#ifndef наименование класса_H #define наименование класса_H

Описание заголовочной части класса

#endif // наименование класса_H



