Объектно-ориентированное программирование на алгоритмическом языке C++

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 = new int;          // выделение памяти для целого
        if ( ! p ) {
                cout << "Ошибка выделения памяти\n";
                return 1;
        }
        * p = 1000;
        cout << "Это целое, на которое указывает p: " << * p << "\n";
        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; }
};
int main() {
      samp * p;
      p = new samp; // выделение памяти объекту
      if( ! p ) {
            cout << "Ошибка выделения памяти\n";
            return 1;
      }
      p -> 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»;

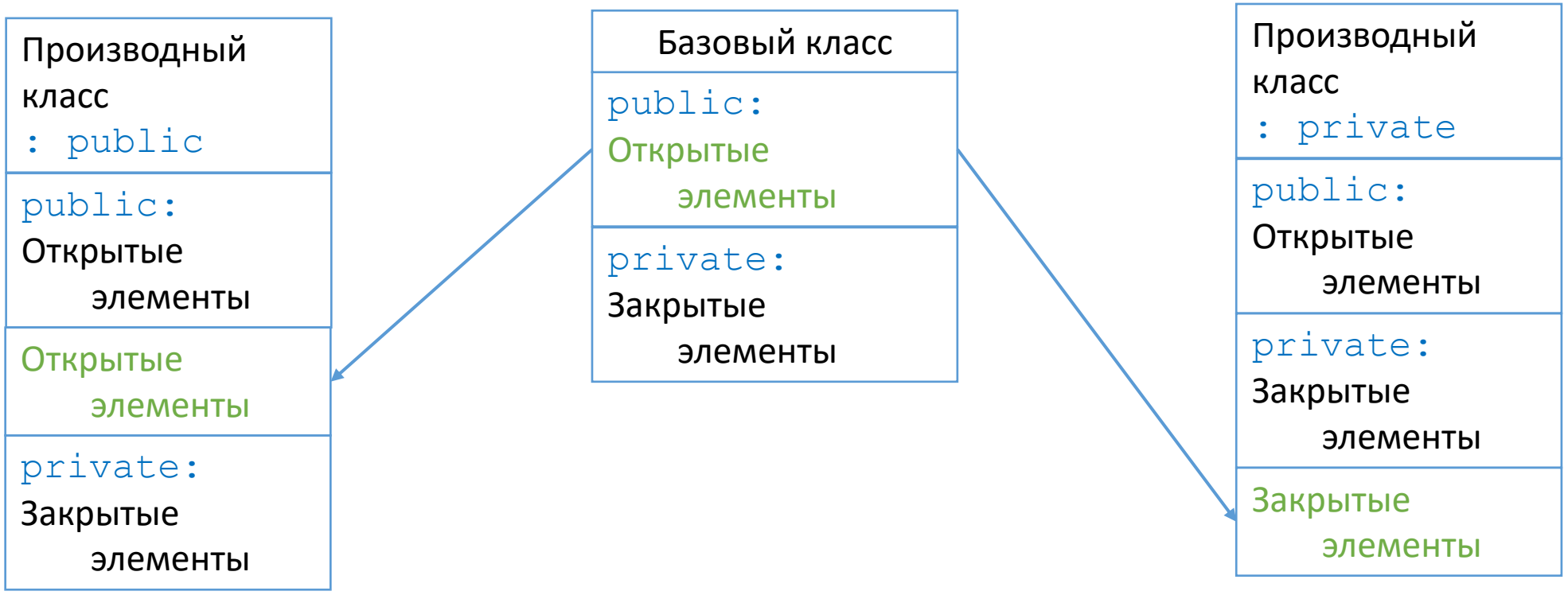
Наследование

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

«спецификатор доступа» — public | private | protected

Рассмотрим спецификаторы public и private

Наследование

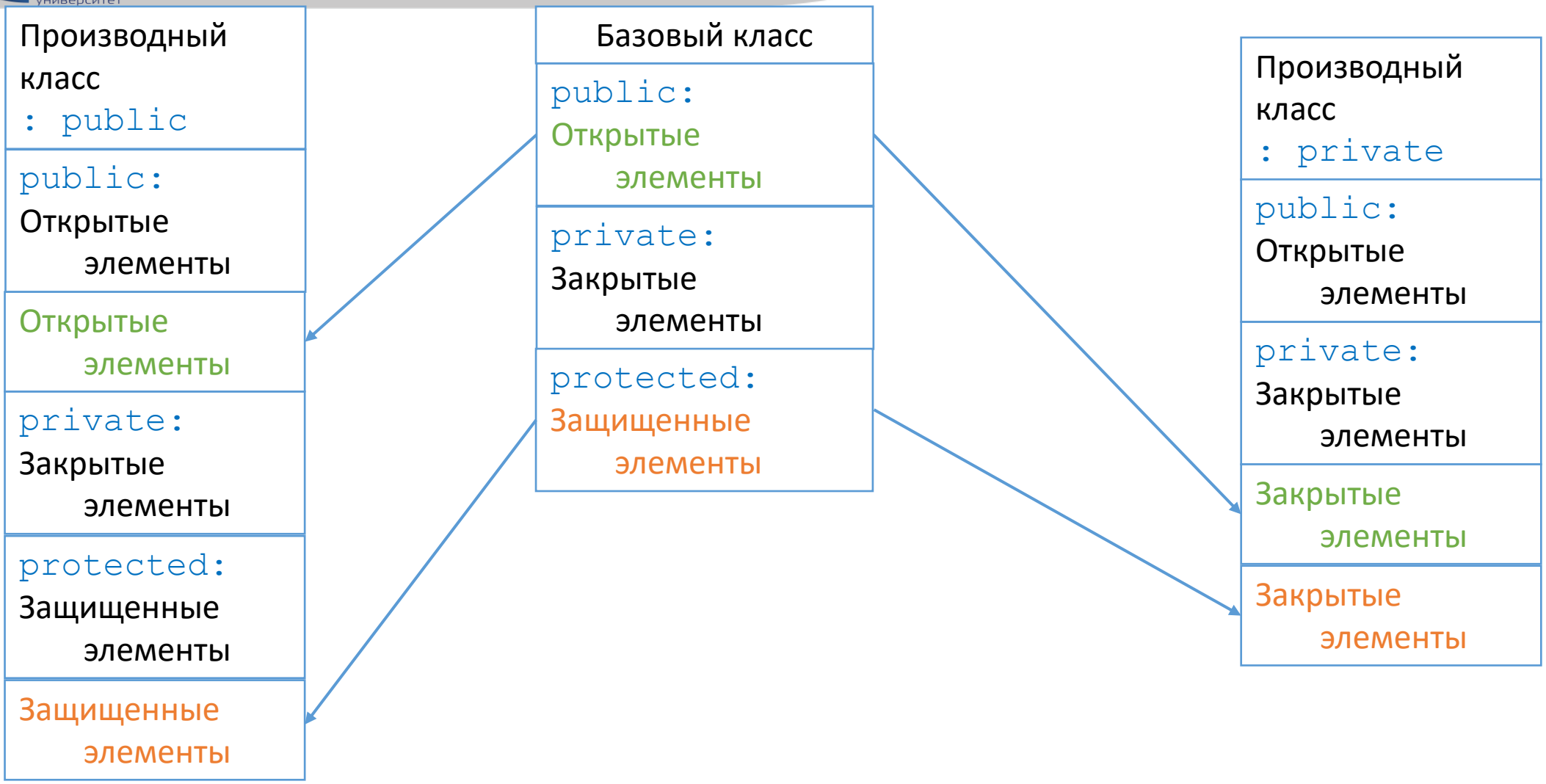


Наследование

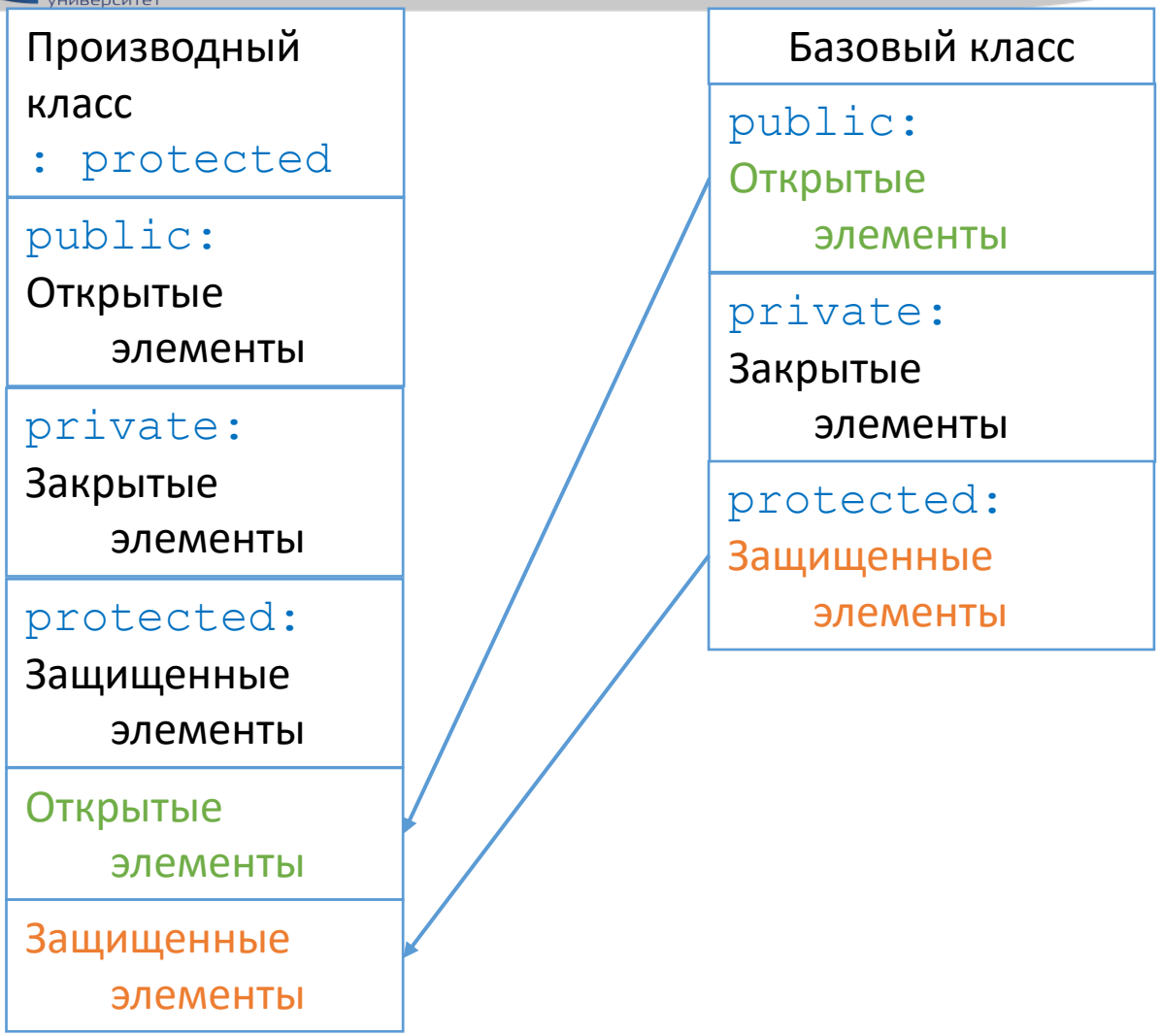
```
#include <iostream>
using namespace std;
class base {
      int x;
public:
      void setx  ( int n ) { x = n; }
      void showx ( )       { cout << x << "\n"; }

class derived: public base // Класс наследуется как открытый
{
      int y;
public:
      void sety  ( int n ) { y = n; }
      void showy ( )       { cout << y << "\n"; }
};
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 0;
}
```

Защищенные члены класса



Защищенные члены класса



Множественное наследование

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

«спецификатор доступа» — public | private | protected

Неоднозначность 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 классы

```
1. Определение области видимости
ob.derived1 :: i = 10;
2.
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 классы

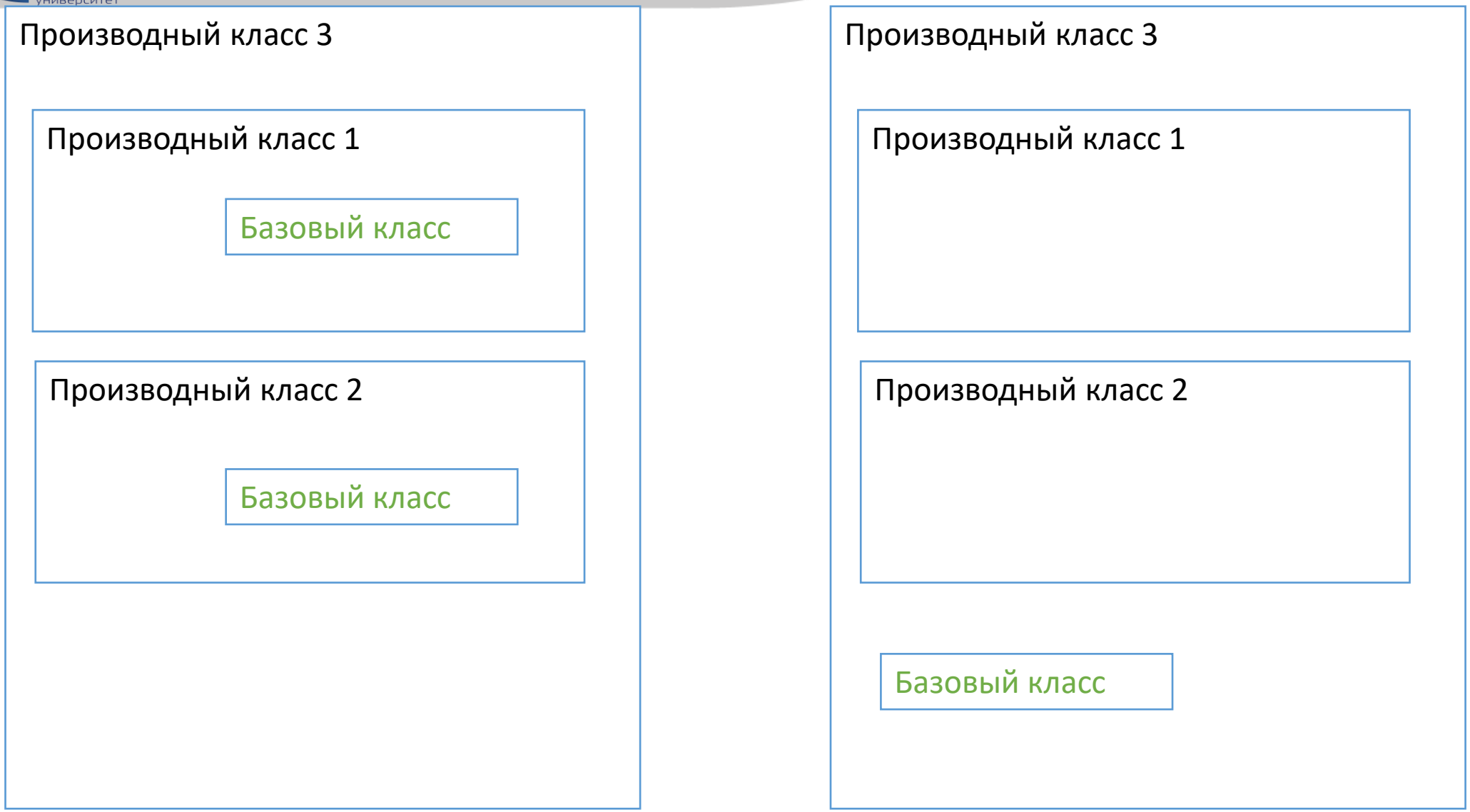
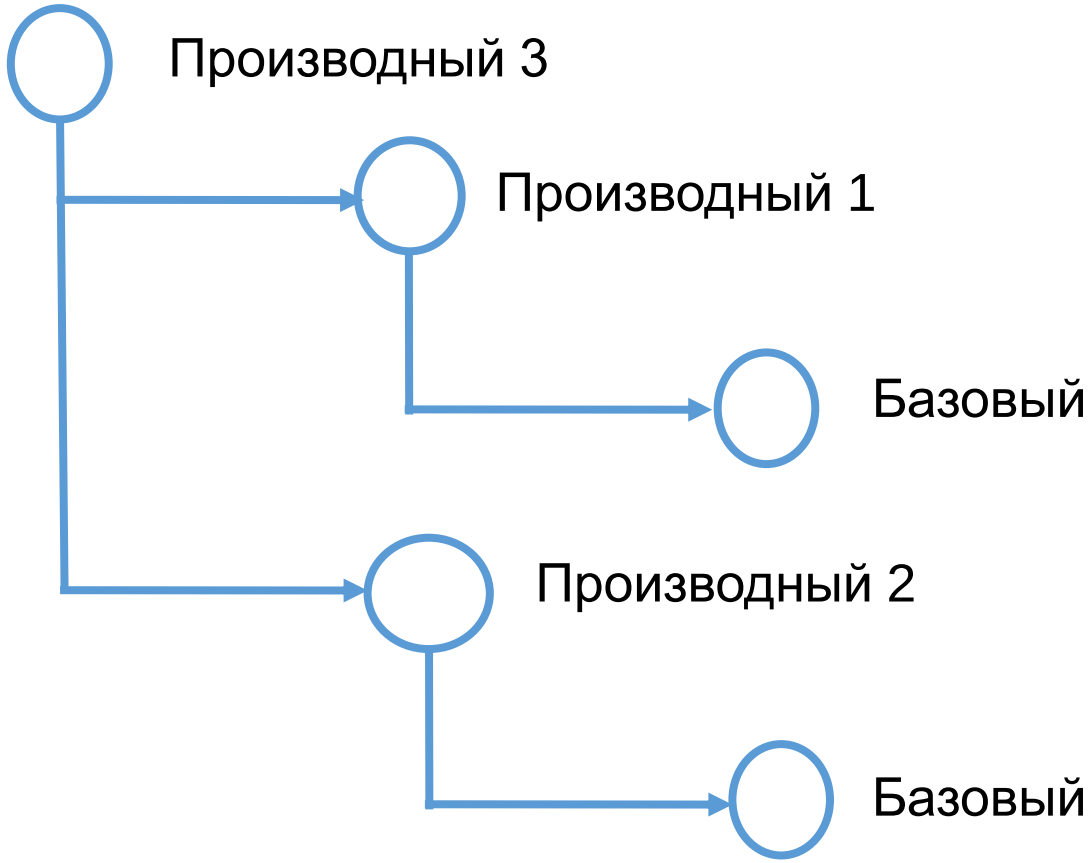


Схема множественного наследования



Указатели 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
```