

Технологии программирования

Часть 3

Направление:
Информационные системы и технологии

Ст. преподаватель
кафедры ПИВТ
Петрова О.Б.

2018 год

Наследование

Произ-
вольный
класс

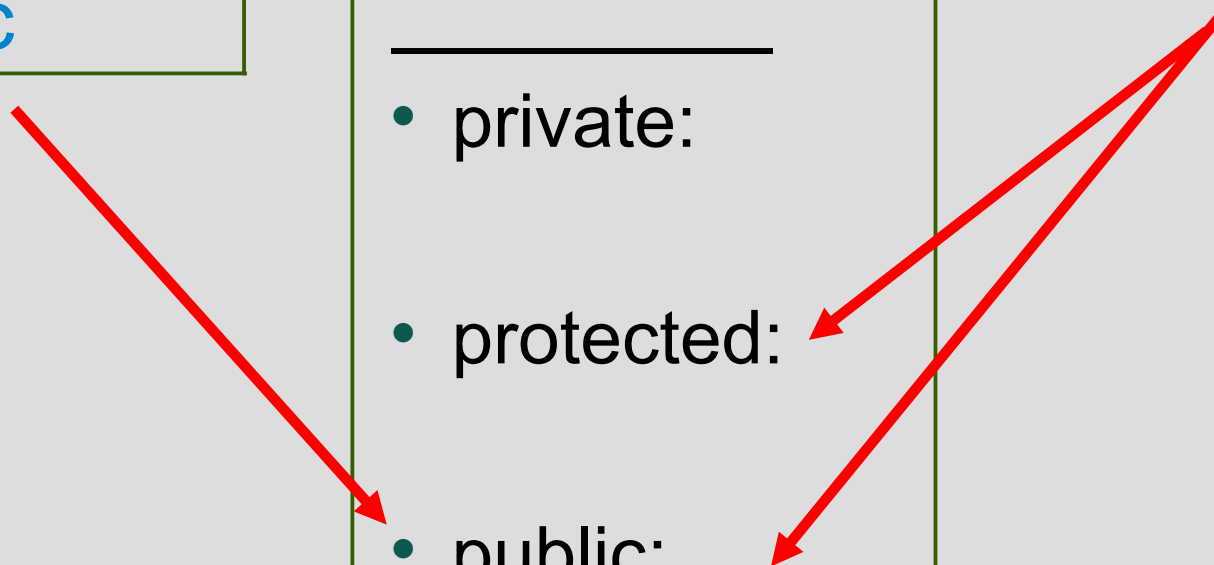
- Базовый
класс

- private:

- protected:

- public:

Класс-
наследник



Создание класса-наследника в C++

// базовый класс

```
class A
```

```
{
```

```
...
```

```
};
```

//класс-наследник

```
class B : public A
```

```
{
```

```
...
```

```
};
```

Пример базового класса

```
class Person
{
protected:
    char name[30];
public:
    Person ();
    void setName (char* );
    void show ();
};
```

Класс-наследник

```
class Student : public Person
{
    char dept[30];
public:
    Student (char*, char* );
    void setDept (char* );
    void show ();
};
```

Методы класса Person

```
Person::Person()
{
    strcpy (name, "Noname");
}

void Person::setName (char* n)
{
    strcpy (name, n);
}

void Person::show ()
{
    std::cout << "My name is " << name << std::endl;
}
```

Собственные методы класса Student

```
Student::Student (char* n, char* d)
{
    strcpy (name, n);
    strcpy (dept, d);
}
void Student::setDept (char* d)
{
    strcpy (dept, d);
}
void Student::show ()
{
    std::cout << name << " " << dept << std::endl;
}
```

Текст программы

```
int main ()  
{  
    Person a;  
    a.setName ("Tom");  
    Student b("Ann", "MTS");  
    a.show ();  
    b.show ();  
    Person* pperson = &b;  
    Student* pstudent = &b;  
    pperson->show ();  
    pstudent->show ();  
    b.setName ("Kate");  
    b.setDept ("GF");  
    pstudent->show ();  
    return 0;  
}
```

На экране:

My name is Tom

Ann MTS

My name is Ann

Ann MTS

Kate GF

Виртуальная функция

```
class Person
{
    char name[30];
public:
    Person (char* );
    void setName (char* );
    virtual void show ();
};
```

Текст программы

```
int main ()
{
    Person a;
    a.setName ("Tom");
    Student b("Ann", "MTS");
    a.show ();
    b.show ();
    Person* pperson = &b;
    Student* pstudent = &b;
    pperson->show ();
    pstudent->show ();
    b.setName ("Kate");
    b.setDept ("GF");
    pstudent->show ();
    return 0;
}
```

На экране:

My name is Tom

Ann MTS

Ann MTS

Ann MTS

Kate GF

Полиморфизм

Статический полиморфизм —
поддерживается посредством перегрузки
функций и операторов во время
компиляции.

Динамический полиморфизм —
поддерживается посредством виртуальных
функций во время выполнения программы.

Конструкторы и наследование

```
class Person
{
    protected:
        char name[30];
    public:
        Person (char*); //конструктор с параметром
        void setName (char* );
        void show ();
};
```

Реализация конструкторов

```
Person::Person (char* n)
{
    strcpy (name, n);
}
```

```
Student::Student (char* n, char* d) : Person (n)
{
    strcpy (dept, d);
}
```

Деструкторы и наследование

```
class A {  
protected:  
    int a;  
public:  
    A() { a=5; cout<<"Create A"<<endl;}  
    virtual ~A() { cout<<"Delete A"<<endl;}  
    void setA(int pa) { a=pa; }  
    int getA() { return a; }  
};
```

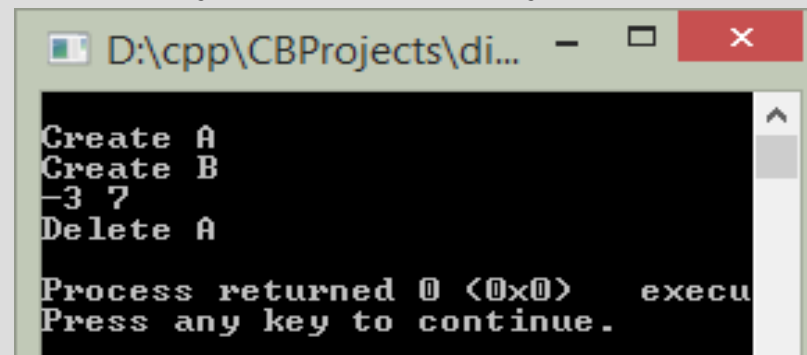
Деструкторы и наследование (2)

```
class B:public A {  
protected:  
    int b;  
public:  
    B():b(7) { cout<<"Create B"<<endl; }  
    virtual ~B() { cout<<"Delete B"<<endl; }  
    void setB(int pb) { b=pb; }  
    int getB() { return b; }  
    void print() { cout<<a<<" "<<b<<endl; }  
};
```

Деструкторы и наследование (3)

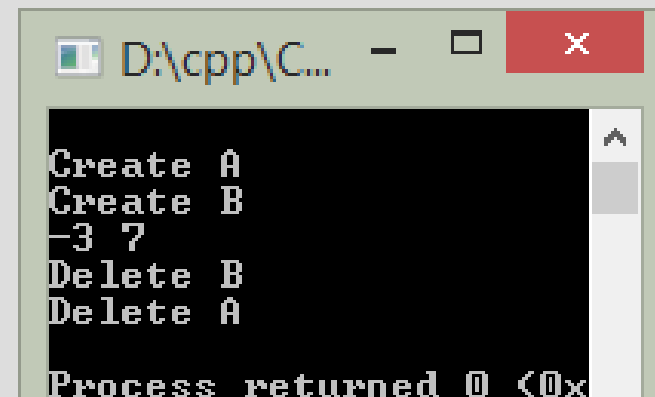
```
int main()
{
    A* fobj = new B;
    fobj->setA(-3);
    ((B*)fobj)->print();
    delete fobj;
    return 0;
}
```

Невиртуальный деструктор



```
D:\cpp\CBProjects\di...
Create A
Create B
-3 7
Delete A
Process returned 0 (0x0)
Press any key to continue.
```

Виртуальный деструктор



```
D:\cpp\C...
Create A
Create B
-3 7
Delete B
Delete A
Process returned 0 (0x0)
```


Множественное (прямое) наследование

```
#include <iostream>
using namespace std;
```

```
class A {
protected:
    int a;
public:
    A() { a=5; }
    virtual ~A() {}
    void setA(int pa){ a=pa;}
    int getA(){ return a;}
};
```

```
class B {
protected:
    int b;
public:
    B():b(7){}
    virtual ~B() {}
    void setB(int pb){ b=pb;}
    int getB(){ return b;}
};
```

Множественное наследование (продолжение)

```
class C:public A, public B {  
public:  
    virtual ~C() {}  
    void printAll() { cout<<a<<" "<<b<<endl; }  
};  
int main()  
{  
    C object;  
    object.printAll();  
    object.setA(-3);  
    object.setB(-18);  
    object.printAll();  
    return 0;  
}
```

На экране:
5 7
-3 -18

Иерархия классов

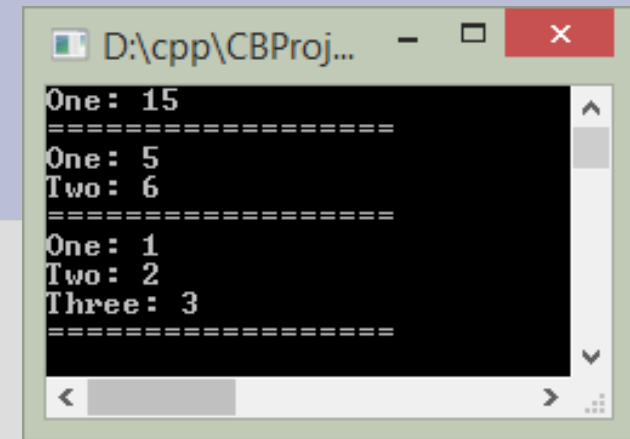
```
class One {  
protected:  
    int a;  
public:  
    One(int pa){ a=pa; }  
    virtual ~One() {}  
    virtual void print() { cout <<"One: " <<a<<endl; }  
};  
class Two: public One {  
protected:  
    int b;  
public:  
    Two(int pa,int pb) : One(pa) { b=pb; }  
    virtual ~Two() {}  
    virtual void print() { One::print(); cout <<"Two: " <<b<<endl; }  
};
```

Иерархия классов

(продолжение)

```
class Three: public Two {
protected:
    int c;
public:
    Three(int pa,int pb,int pc): Two(pa,pb) { c=pc; }
    virtual ~Three() {}
    virtual void print() { Two::print(); cout <<"Three: " <<c <<endl; }
};

int main() {
    One obj1(15), *ptr=&obj1;
    ptr->print(); cout <<"=====\n";
    Two obj2(5,6); ptr=&obj2;
    ptr->print(); cout <<"=====\n";
    Three obj3(1,2,3); ptr=&obj3;
    ptr->print(); cout <<"=====\n";
    return 0;
}
```



The screenshot shows a console window titled "D:\cpp\CBProj..." with the following output:

```
One: 15
====
One: 5
Two: 6
====
One: 1
Two: 2
Three: 3
=====
```

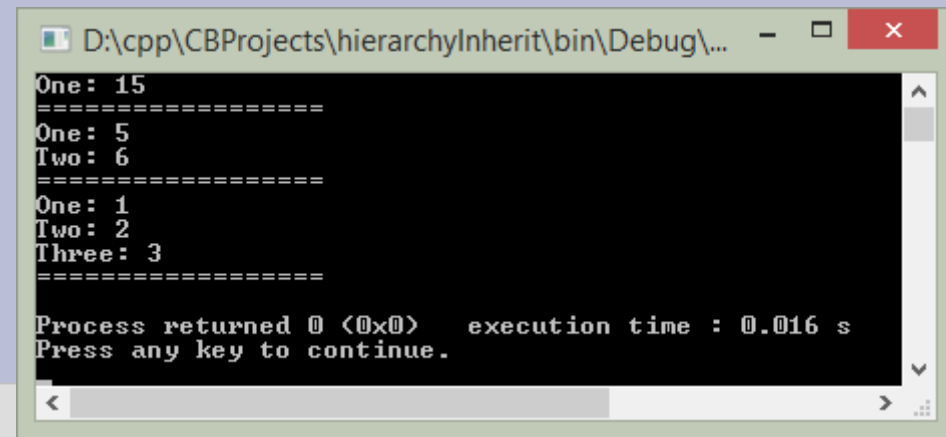
Иерархия классов

```
class One {
protected:
    int a;
public:
    One(int pa){ a=pa;}
    virtual ~One() {}
    virtual void print() {
        cout <<"One: " <<a<<endl;
    }
};

class Two: public One {
protected:
    int b;
public:
    Two(int pa,int pb) : One(pa) {b=pb;}
    virtual ~Two() {}
    virtual void print() {
        One::print();
        cout <<"Two: " <<b<<endl;
    }
};
```

```
class Three: public Two {
protected:
    int c;
public:
    Three(int pa,int pb,int pc):Two(pa,pb) {c=pc;}
    virtual ~Three() {}
    virtual void print() { Two::print();
        cout <<"Three: " <<c<<endl;
    }
};

int main() {
    One obj1(15), *ptr=&obj1;
    ptr->print(); cout<<"=====\n";
    Two obj2(5,6); ptr=&obj2;
    ptr->print(); cout<<"=====\n";
    Three obj3(1,2,3); ptr=&obj3;
    ptr->print(); cout<<"=====\n";
    return 0;
}
```



```
D:\cpp\CBProjects\hierarchy\inherit\bin\Debug\...
One: 15
=====
One: 5
Two: 6
=====
One: 1
Two: 2
Three: 3
=====

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

Абстрактный класс

```
#ifndef SHAPE_H
#define SHAPE_H

class Shape // объявление абстрактного класса
{
public:
    // Shape();
    virtual ~Shape() {}
    virtual float getSquare()=0;    //функция
    virtual void getInfo()=0;       // не реализована
protected:
private:
};

#endif // SHAPE_H
```

Абстрактный класс (2)

```
#ifndef CIRCLE_H
#define CIRCLE_H
#include "Shape.h"

class Circle:public Shape //создание класса-наследника
{ // базовый класс - абстрактный
public:
    Circle();
    virtual ~Circle() { };
    float getSquare();
    void getInfo();
private:
    int r;
    int xc;
    int yc;
};
#endif // CIRCLE_H
```

Абстрактный класс (3)

```
#include "..\include\Circle.h"
#include <cmath>
#include <iostream>
using namespace std;
```

```
Circle::Circle():r(10),xc(15),yc(15) { }
```

```
float Circle::getSquare()
{
    return M_PI*r*r;
}
```

```
void Circle::getInfo()
{
    cout<<"Circle:" << r << " " << xc << " " << yc <<endl;
}
```


Абстрактный класс (4)

```
#ifndef RECT_H
#define RECT_H
#include "Shape.h"

class Rect:public Shape //создание класса-наследника
{ // базовый класс - абстрактный
public:
    Rect();
    virtual ~Rect() { };
    float getSquare();
    void getInfo();
private:
    int x1;
    int y1;
    int x2;
    int y2;
};
#endif // RECT_H
```

Абстрактный класс (5)

```
#include "..\include\Rect.h"
#include <iostream>
using namespace std;

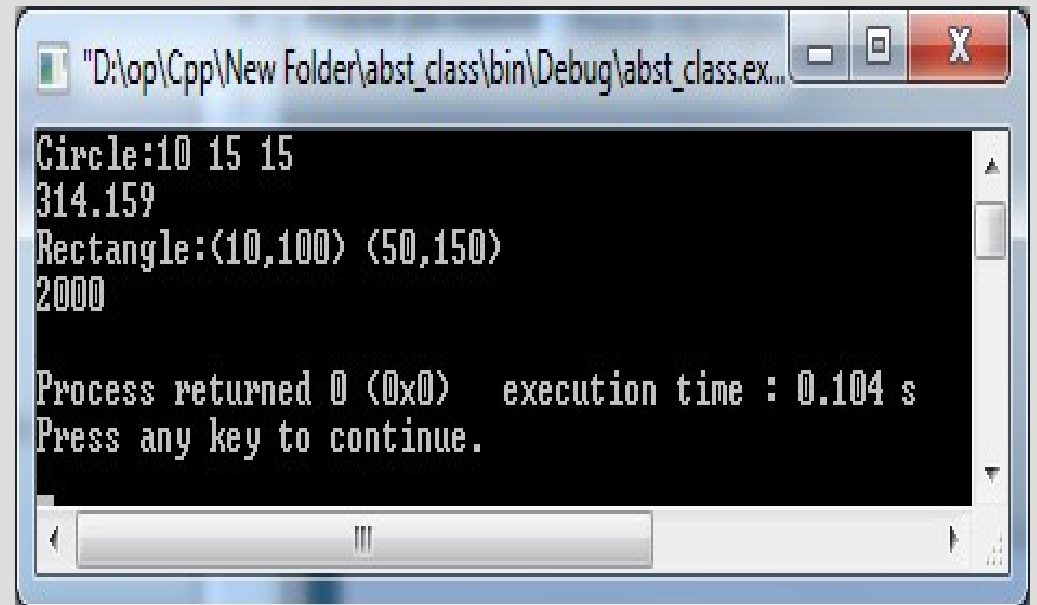
Rect::Rect() { x1=10; y1=100; x2=50; y2=150; }

float Rect::getSquare()
{
    return (x2-x1)*(y2-y1);
}

void Rect::getInfo()
{
    cout<<"Rectangle:(" << x1<< ","<< y1<<") (" ;
    cout << x2 << "," << y2 << ")"<<endl;
}
```

Абстрактный класс (6)

```
#include <iostream>
#include "include/Shape.h"
#include "include/Rect.h"
#include "include/Circle.h"
using namespace std;
int main()
{
    Circle s1;
    Rect s2;
    Shape* mas[2];
    mas[0]=&s1;
    mas[1]=&s2;
    for(int i=0;i<2;i++)
    {
        mas[i]->getInfo();
        cout<< mas[i]->getSquare()<<endl;
    }
    return 0;
}
```



```
"D:\op\Cpp\New Folder\abst_class\bin\Debug\abst_class.ex...
Circle:10 15 15
314.159
Rectangle:(10,100) (50,150)
2000

Process returned 0 (0x0)   execution time : 0.104 s
Press any key to continue.
```

Язык UML

UML: Unified Modeling Language.

Стандарты: 1.0, 2.0

Назначение: графический язык для визуализации, конструирования и документирования систем, в т.ч. программного обеспечения.

Разработчики: Грейди Буч, Джеймс Рамбо, Айвар Джекобсон.

Элементы языка: диаграммы и их компоненты (предметы, отношения).

Диаграммы UML

Структурные диаграммы:

- диаграмма компонентов,
- диаграмма классов,
- диаграмма объектов,
- диаграмма развертывания.

Диаграммы поведения:

- диаграмма прецедентов,
- диаграмма последовательности,
- диаграмма состояний.

Диаграмма классов

Обозначение класса на UML-диаграммах:



Полное обозначение



Упрощенное обозначение

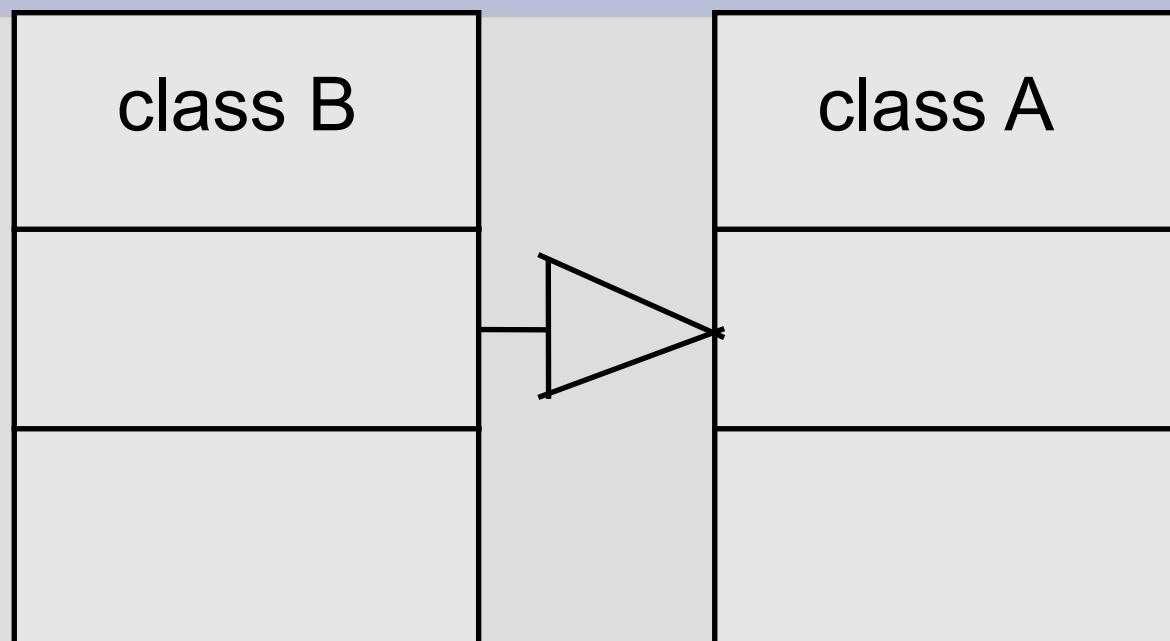
Обозначение доступности члена класса:

- private
- + public
- # protected

Взаимодействие классов

- ♦ Виды взаимодействия:
 - Наследование
 - Композиция
 - Агрегация
 - Зависимость (клиент-серверное взаимодействие)
 - Ассоциация

Наследование



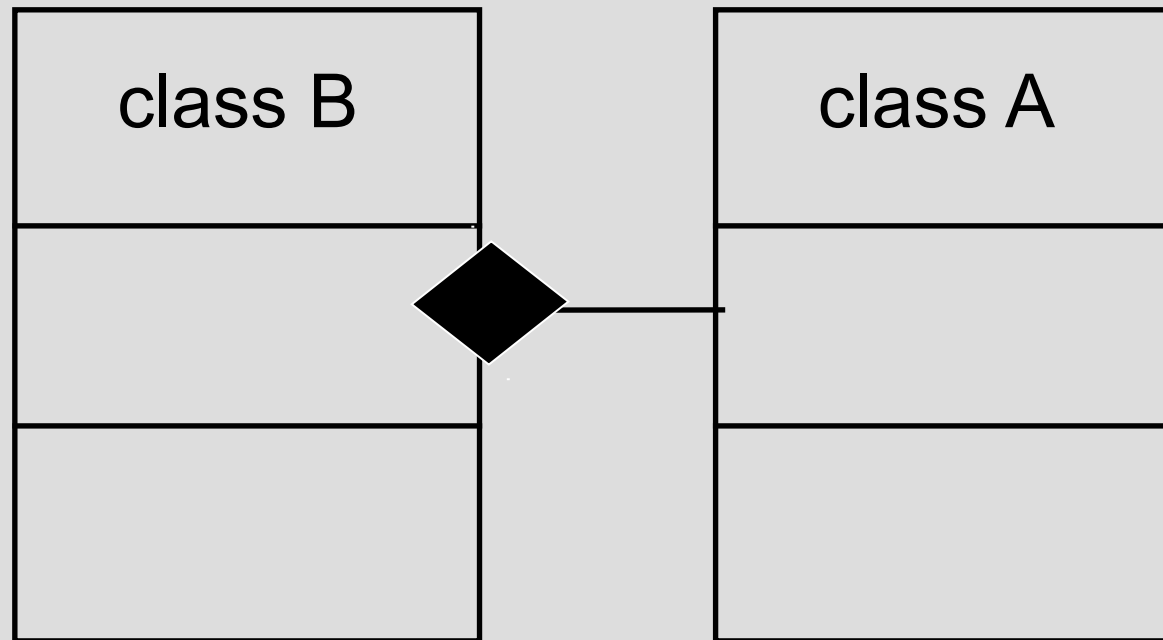
```
class A  
{  
  // ...  
};
```

```
class B : public A  
{  
  // ...  
};
```

Класс-наследник

Базовый класс

Композиция



Внешний
класс

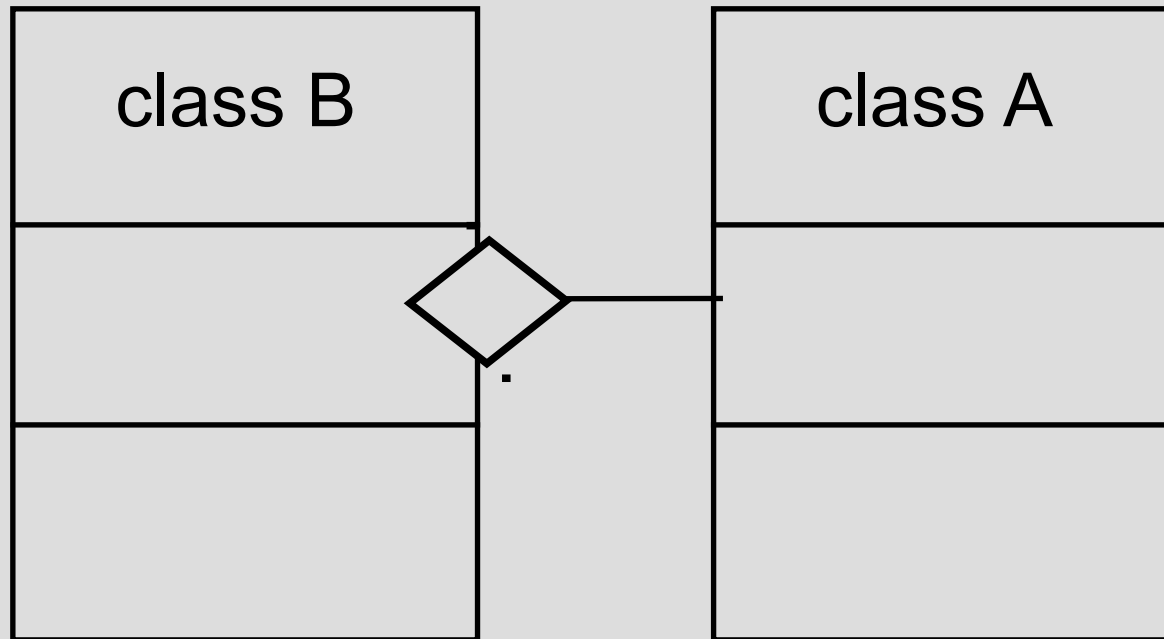
Внутренний
класс

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

class B
{
    // ...
    A a;
};

int main ( )
{
    B b ;
    // ...
    b . a . Message ( );
    // ...
}
```

Агрегация



Внешний
класс

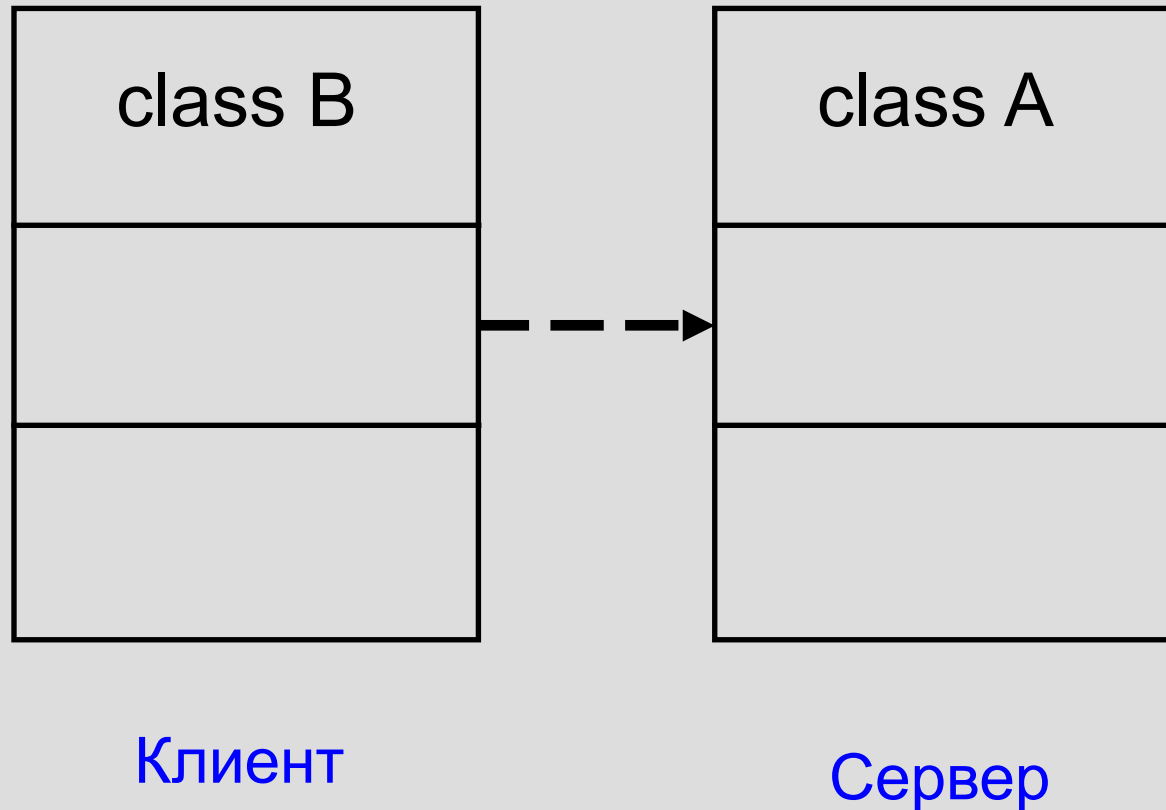
Внутренний
класс

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

class B
{
    // ...
    A* pa;
};

int main ( )
{
    A a;
    B b ;
    b.pa = &a;
    // ...
    b . pa-> Message ( );
    // ...
}
```

Зависимость



```
class A  
{  
    // ...  
};
```

```
class B  
{  
    // ...  
public:  
    void Fn ( A* );  
};
```

```
int main ( )  
{  
    A a;  
    B b;  
    // ...  
    b . Fn ( &a );  
    // ...  
}
```

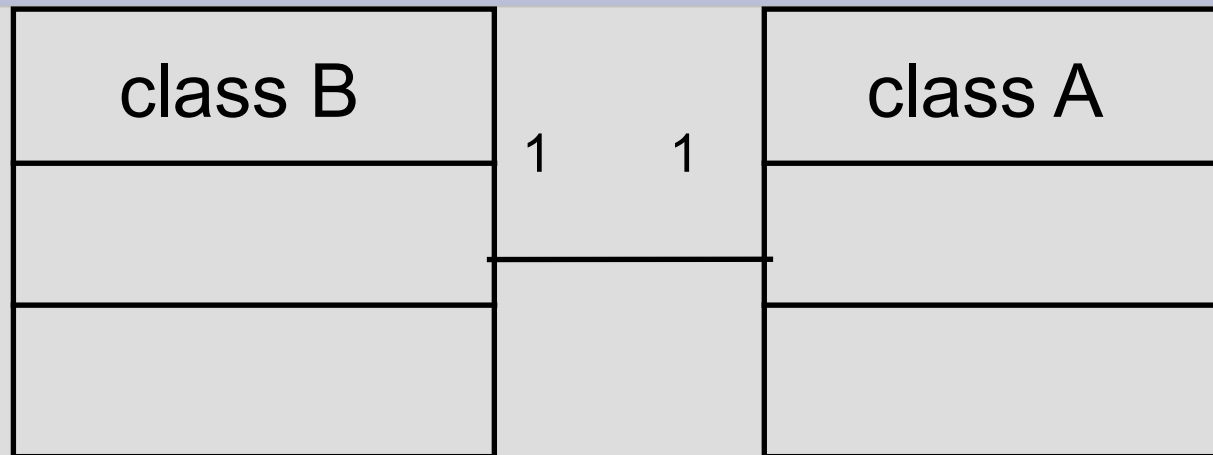
Ассоциация

Мощность связи:

Один-к-одному

Один-ко-много

Много-ко-много



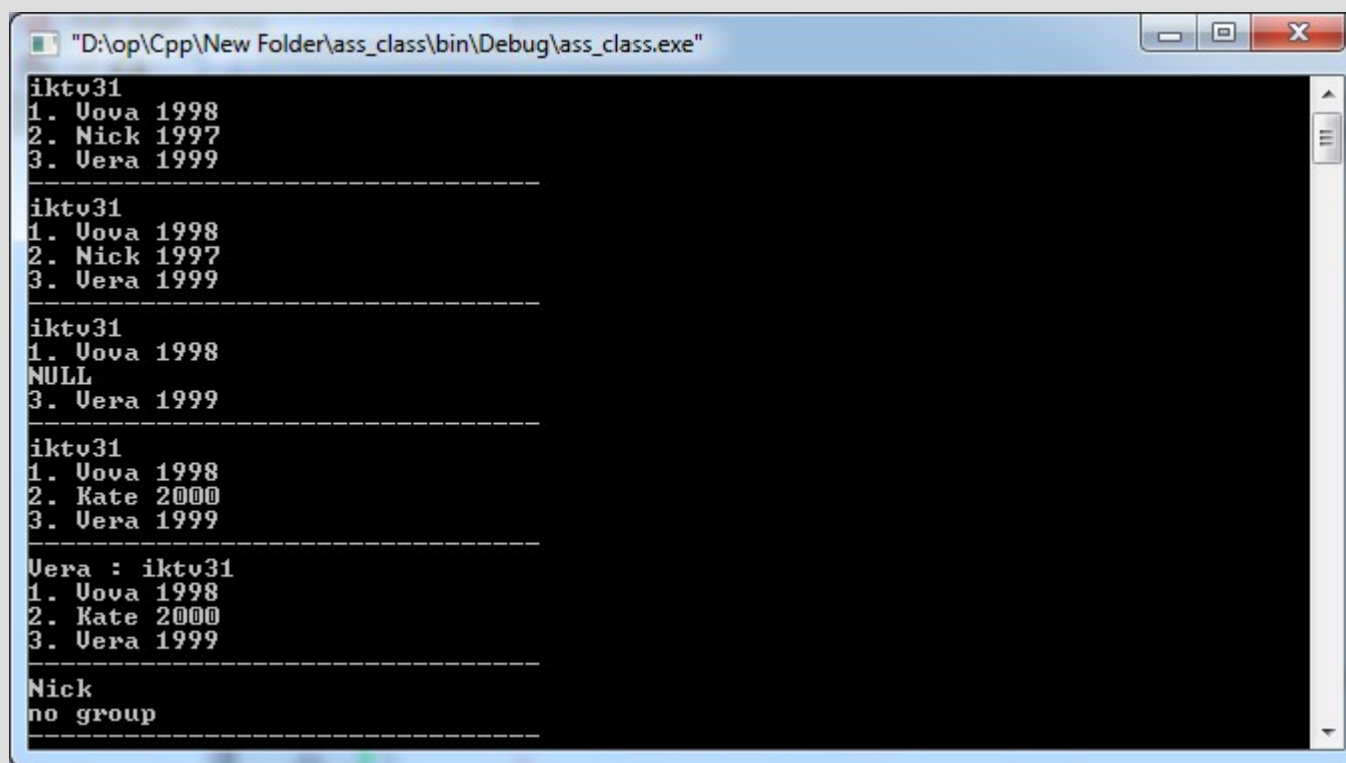
Пример связи один-к-одному

```
class A;
class B
{
// ...
public:
A* pa;
};
```

```
class A
{
// ...
public:
B* pb;
};
```

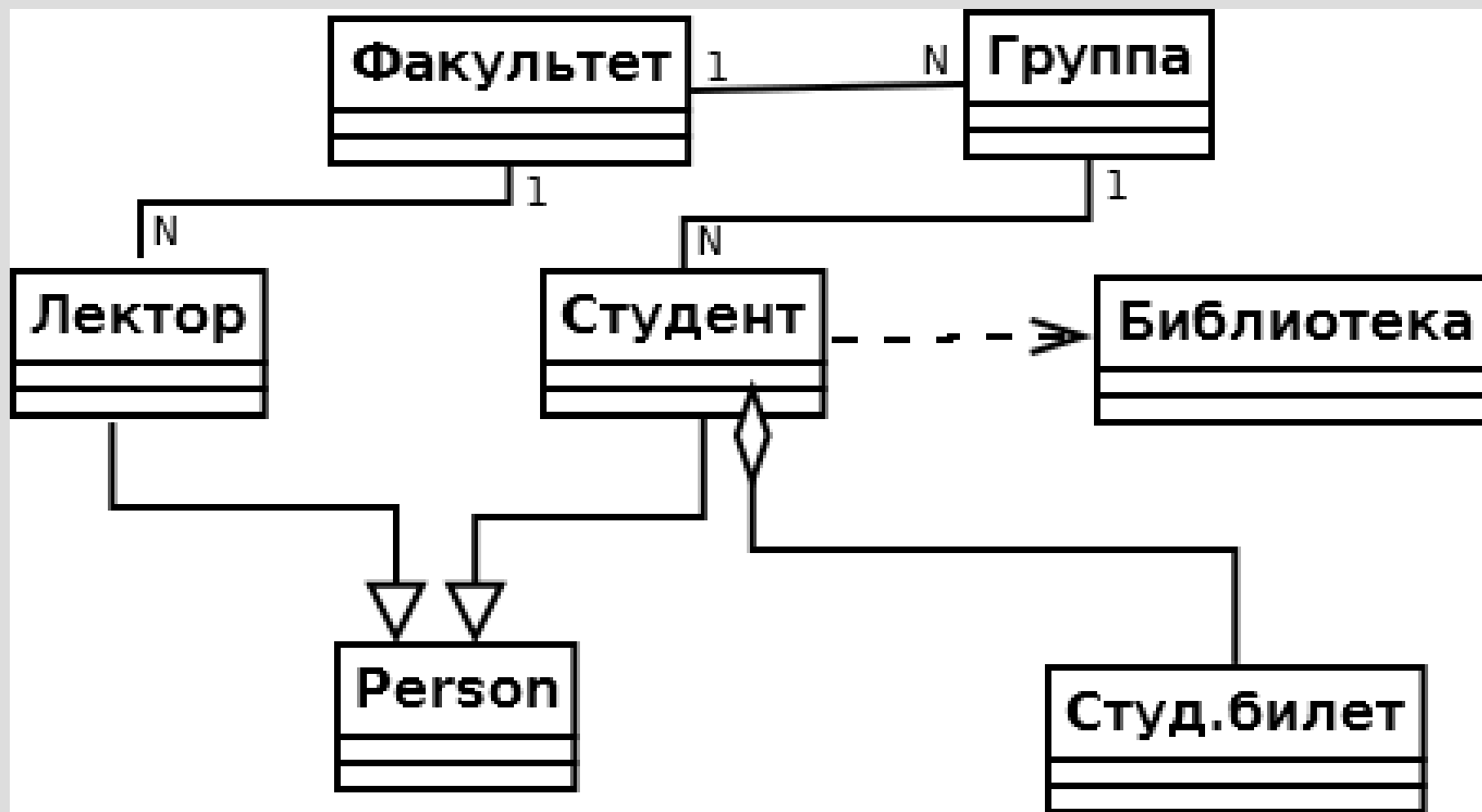
```
int main ( )
{
    A a;
    B b ;
    // ...
    b.pa = &a;
    a.pb = &b;
    // ...
}
```

Результат работы программы ass_class



```
"D:\op\Cpp\New Folder\ass_class\bin\Debug\ass_class.exe"
iktu31
1. Uova 1998
2. Nick 1997
3. Vera 1999
-----
iktu31
1. Uova 1998
2. Nick 1997
3. Vera 1999
-----
iktu31
1. Uova 1998
NULL
3. Vera 1999
-----
iktu31
1. Uova 1998
2. Kate 2000
3. Vera 1999
-----
Uera : iktu31
1. Uova 1998
2. Kate 2000
3. Vera 1999
-----
Nick
no group
-----
```

Пример диаграммы классов



Друзья класса

```
class First
{
    int x;
    public:
        int getX() { return x; }
        void setX (int);
        friend class Second;
        friend void set_X (First&, int);
};
void First::setX (int xx)
{
    x = xx;
}
```

Друзья : глобальная функция или класс

```
class Second
{
// ...
public:
    void setX (First&, int);
};
void Second::setX (First& f,
    int xx)
{
    f.x = xx;
}
void set_X (First& f, int xx)
{
    f.x = xx;
}
```

```
int main ()
{
    First first;
    first.setX (10);
    cout << first.getX () << endl;
    Second second;
    second.setX (first, 20);
    cout << first.getX () << endl;
    set_X (first, 30);
    cout << first.getX () << endl;
}
```

На экране:

 10

 20

 30

Перегрузка операторов

Встроенные перегружаемые операторы:

`= == != < > <= >= + - * / ++(инкремент) --
(декремент) <<(вывод) >>(ввод) += -=` и т.д.

Встроенные неперегружаемые операторы:

`.` `::` `?:` (оператор условия)

Операторные функции:

`<возвр_тип> operatorX(<тип_парам>);`

Глобальная функция:

`MyClass operator+(MyClass&, MyClass&);`

Функция-член класса:

`MyClass MyClass::operator+(MyClass&);`

Глобальные операторные функции

`bool operator==(MyClass, MyClass); // != < >`

`MyClass operator+(MyClass&, AnyType); // - * /`

`istream& operator>> (istream&, MyClass&);`

`ostream& operator<< (ostream&, const MyClass&);`

Функции-члены класса

`bool operator==(MyClass); // != < >`

`MyClass operator+(AnyType); // - * /`

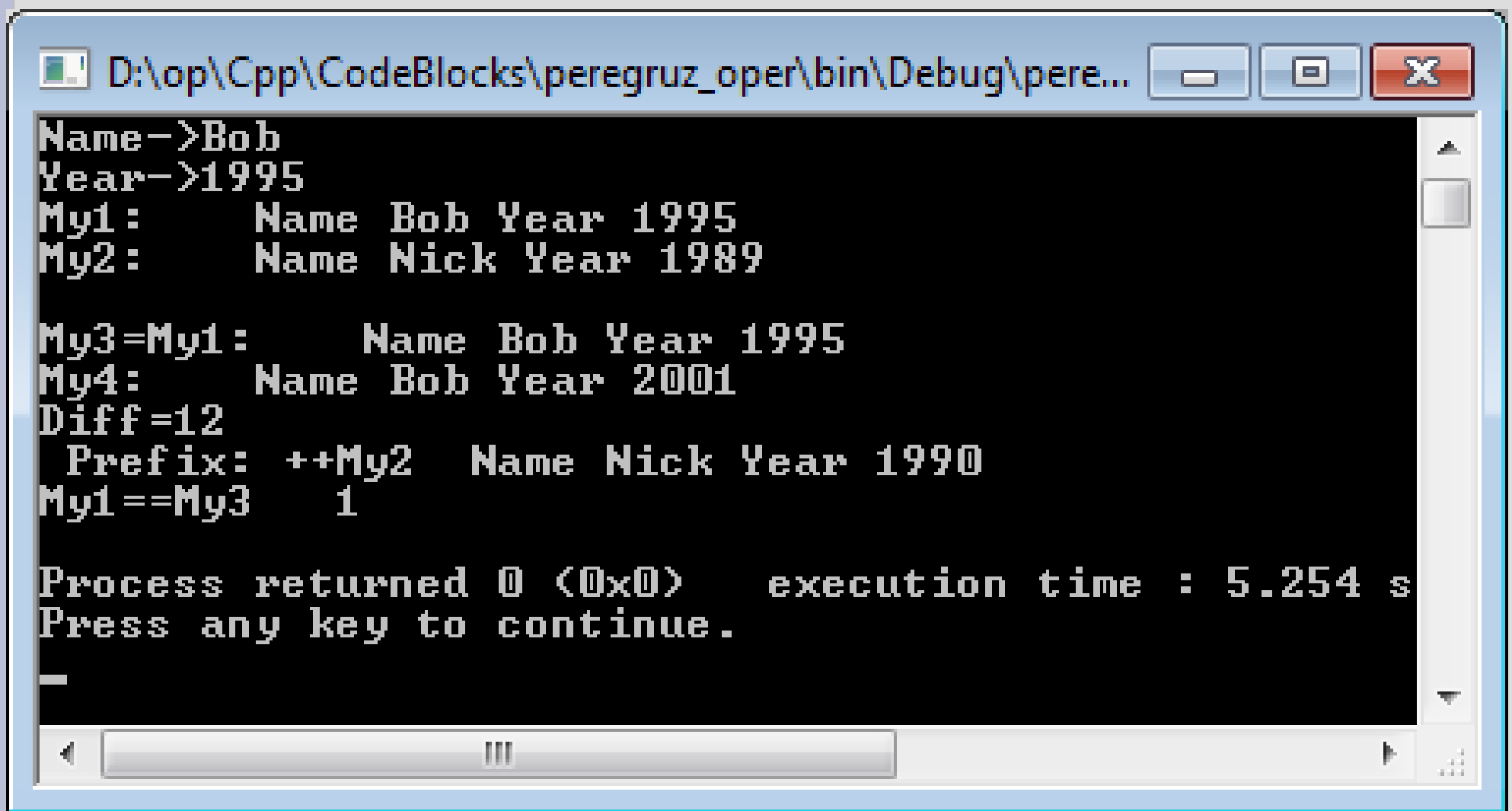
`MyClass& operator++(); // - - (++a)`

`MyClass operator++(int); // - - (a++)`

`MyClass& operator=(MyClass&);`

`MyClass& operator+=(AnyType); // -=`

Результат работы программы



A screenshot of a Windows command prompt window. The title bar shows the file path: D:\op\Cpp\CodeBlocks\peregruz_oper\bin\Debug\pere... The window contains the following text output from a program:

```
Name->Bob
Year->1995
My1:      Name Bob Year 1995
My2:      Name Nick Year 1989

My3=My1:   Name Bob Year 1995
My4:      Name Bob Year 2001
Diff=12
Prefix: ++My2   Name Nick Year 1990
My1==My3    1

Process returned 0 (0x0)    execution time : 5.254 s
Press any key to continue.
```

The output shows the creation and manipulation of objects with 'Name' and 'Year' attributes. It includes a comparison (My1==My3) resulting in 1, and a prefix increment operation (++My2) that updates the year of 'My2' from 1989 to 1990. The program ends with a return code of 0 and an execution time of 5.254 seconds.

```
#pragma once
#include <iostream>
```

Person.h

```
class person
{
    char name[20];
    int year;
public:
    person(void);
    person(char* n, int y);
    bool operator==(person&);           // оператор отношения
    person operator+(int);               // сложение
    int operator-(person&);             //вычитание (результат - целое число)
    person& operator++();                // префиксный инкремент
    person& operator=(person&);         //присваивание
    //Перегрузка ввода через объект класса istream (cin)
    friend std::istream& operator>>(std::istream&,person&);
    //Перегрузка вывода через объект класса ostream (cout)
    friend std::ostream& operator<<(std::ostream&,const person&);
};
```

main.cpp

```
#include "person.h"
```

```
int main(int argc, char* argv[])
```

```
{
```

```
    person my1, my2("Nick",1989);
```

```
    std::cin >> my1;
```

```
    std::cout << "My1:  " << my1;
```

```
    std::cout << "My2:  " << my2 <<std::endl;
```

```
    person my3;
```

```
    my3=my1;
```

```
    std::cout << "My3=My1:  " << my3;
```

```
    person my4=my3+6;
```

```
    std::cout << "My4:  " << my4;
```

```
    int diff=my4-my2;
```

```
    std::cout << "Diff=" << diff << std::endl;
```

```
    std::cout << " Prefix: ++My2 " << ++my2;
```

```
    std::cout << "My1==My3  " << (my1==my3) <<std::endl;
```

```
    return 0;
```

```
}
```

```
#include "..\person.h"  
#include <string.h>
```

Person.cpp (1)

```
person::person(void)  
{  
    strcpy(name, "Noname");  
    year=0;  
}  
person::person(char* n, int y)  
{  
    strcpy(name, n);  
    year=y;  
}  
bool person::operator==(person& p)  
{  
    bool cmp =!strcmp(name,p.name)&&year==p.year;  
    return cmp;  
}
```

Person.cpp (2)

```
person person::operator+(int y)
{
    person temp;
    temp.year= year+y;
    strcpy(temp.name,name);
    return temp;
}
```

```
int person::operator-(person& p)
{
    int temp;
    temp=year-p.year;
    return temp;
}
```


Person.cpp (3)

```
person& person::operator++()  
{  
    year++;  
    return *this;  
}
```

```
person& person::operator=(person& p)  
{  
    if (this==&p)  
        return *this;  
    strcpy(name, p.name);  
    year=p.year;  
    return *this;  
}
```

Person.cpp (4)

```
std::istream& operator>>(std::istream& is, person& p)
{
    char buf[20]; int y;
    std::cout << "Name->";
    is>>buf;    strcpy(p.name,buf);
    std::cout << "Year->";
    is >> y;    p.year = y;
    return is;
}
```

```
std::ostream& operator<<(std::ostream& os,const person& p)
{
    os << "Name " << p.name << " Year " << p.year << std::endl;
    return os;
}
```

