# Информатика. Спецглавы

Лекция 3

Направление: Инфокоммуникационные технологии и системы связи
2012 год

# Наследование

# 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()
{
}

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);
}
```

# 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:

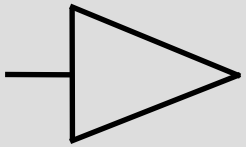
Полное обозначение

Упрощенное обозначение

# Взаимодействие классов

- Виды взаимодействия:
  - Наследование
  - Агрегация (включение)
  - Использование (клиент-серверное взаимодействие)
  - Ассоциация

# Наследование



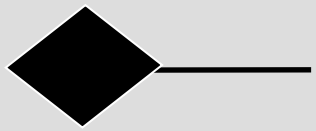
```
class A
{
// . . .
} ;

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

Класс-наследник

Базовый класс

# Агрегация



Внешний класс

Внутренний класс

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

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

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

# Использование

Клиент

Сервер

```
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

1 1



Пример связи один-к-одному

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

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

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

# Пример диаграммы классов

# Друзья класса

```
class First
{
    int x;
    public:
    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);
     first.Show ();
     Second second;
     second.SetX (first, 20);
     first.Show ();
     Set_X (first, 30);
     first.Show ();
}
```

- На экране:
- 10
- 20
- 30

# Указатель this

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

```
void A::Prim ()
{
    Greet ();
    this->Greet ();
    (*this).Greet ();
//. . .
} ;
```

# 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&);
```

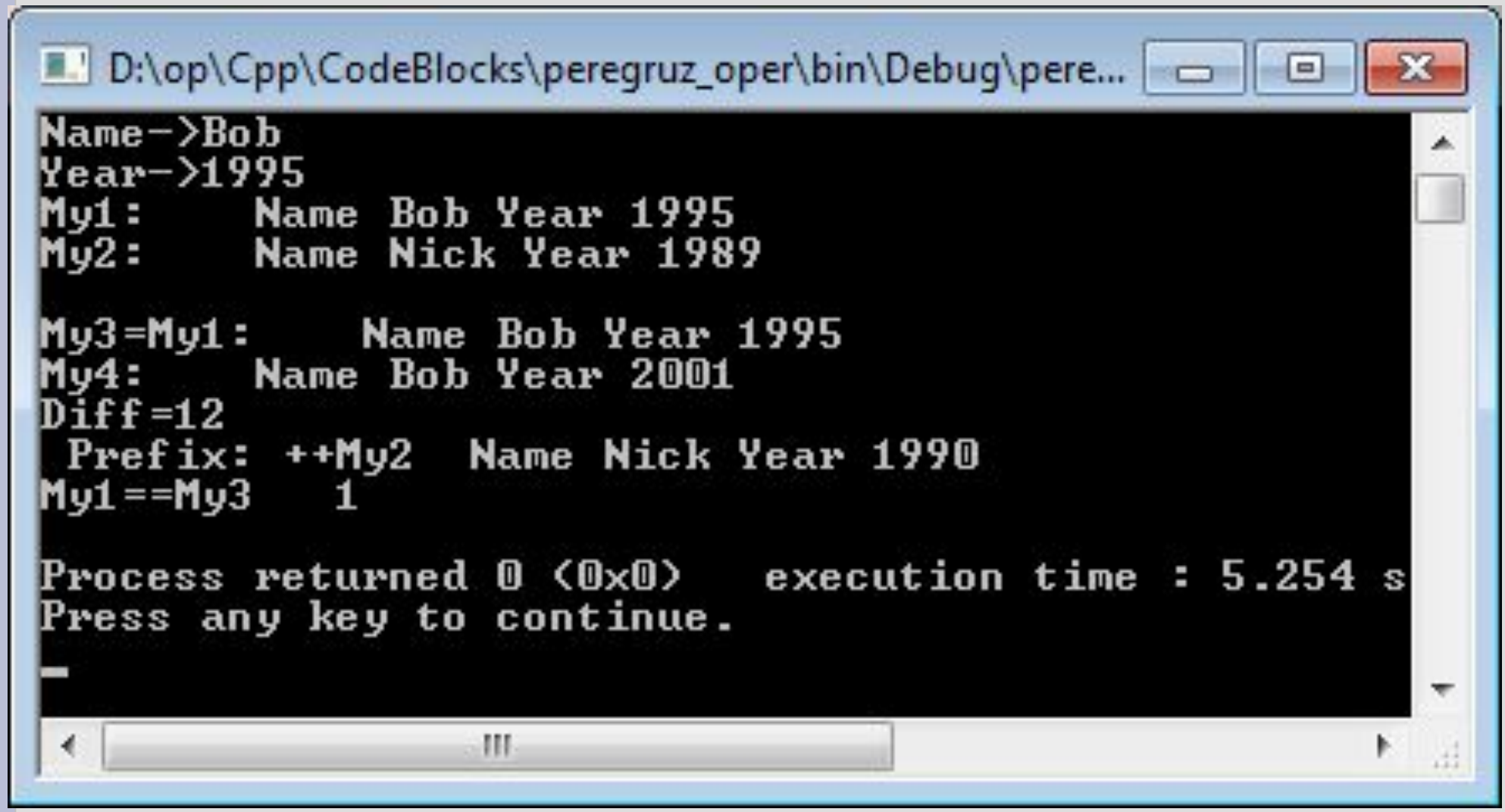
# Функции-члены класса

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

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

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

# Результат работы программы



# Person.h

```
#pragma once
#include <iostream>

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 в виде
                               //целого числа
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;
}
```

# Person.cpp (1)

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

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;
}
```