Лекция 7. Производные классы. Часть 2.

Артур Садеков

НИЖЕГОРОДСКИЙ ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
НИИТ

Виртуальные функции

- Интерфейсные функции (public)
- Их можно заместить в каждом производном классе

```
class Employee {
public:
  Worker(string _name,
         string _surname);

  // ...
  virtual void print() const;

private:
  string name, surname;
  Date hire_date, fire_date;
};
```

```
class Programmer: public Employee
{
public:
  Programmer(string _name,
             string _surname,
             string _team);

  virtual void print() const;

private:
  string team;
};
```

Пример

```
void Employee::print() const
{
 cout << first_name << " " << surname    << endl;
}
```

```
void Programmer::print() const
{
  Employee::print();
  cout << "team: " << team << endl;
}
```

Пример (продолжение)

```
void print_emp(const Employee* pEmp)
{
  cout << "Employee info:" << endl;
  pEmp->print();
}
```

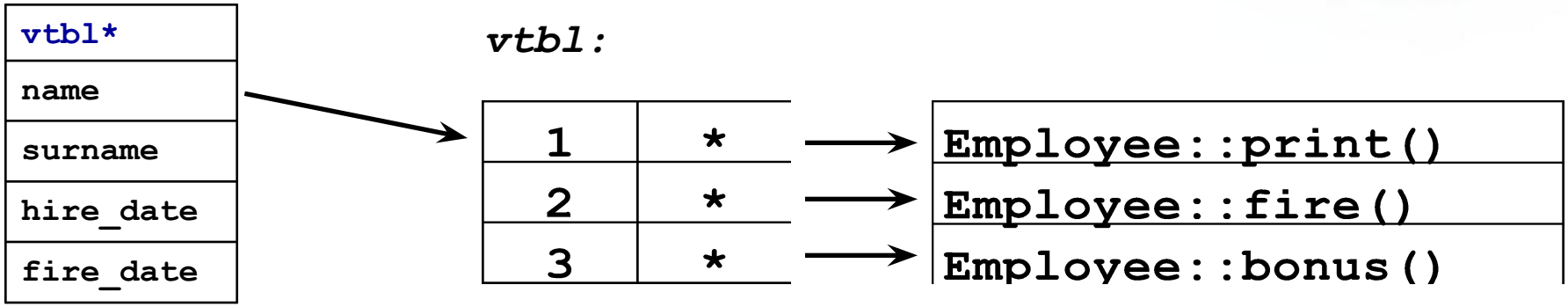
```
int main()
{
  Employee emp("Vassya", "Pupkin");
  Programmer prog("Ivan", "Sidorov", "GM12");
  print_emp ( &emp );
  print_emp ( &prog );
  return 0;
}
```

```
Employee info:
Vassya Pupkin
```

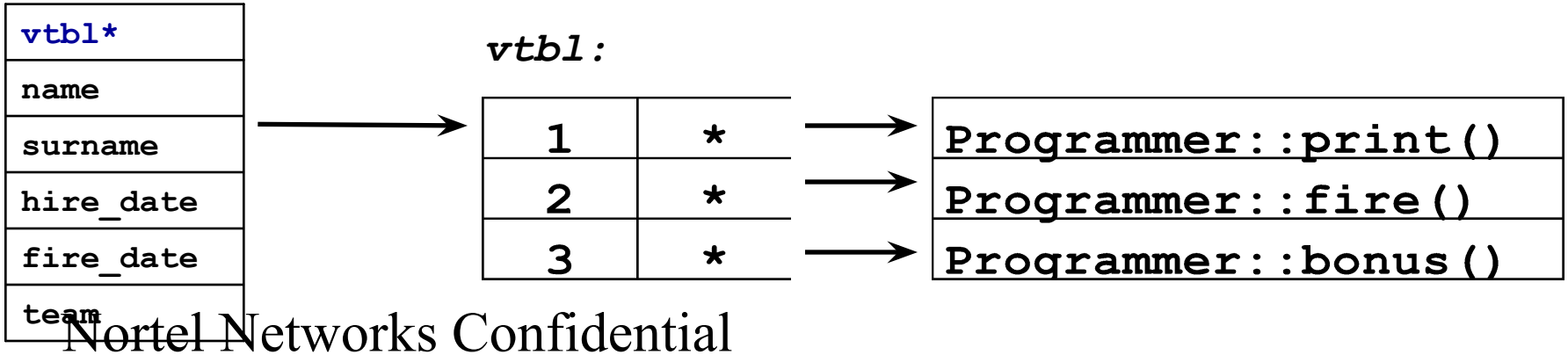
```
Employee info:
Ivan Sidorov
team: GM12
```

Virtual function table (vtbl)

Объект класса Employee



Объект класса Programmer



Когда используется виртуальность

```
Employee emp("Vasya", "Pupkin");
Programmer prog("Ivan", "Sidorov", "GM12");
emp.print();  // нет, Employee::print()
prog.print(); // нет, Programmer::print()
```

```
void fn1(Employee *p)
{
  p->print(); // да
  p->hire();  // да
}
```

```
void fn2(Employee &r)
{
  r.print();  // да
  r.fire();   // да
}
```

```
{
  fn1(&emp);
  fn1(&prog);

  fn2(emp);
  fn2(prog);
}
```

- **Механизм виртуальности используется, только когда вирт. функция вызывается через указатель или ссылку на базовый класс.**

Более сложный пример

```
// массив указателей на Employee, размер
void give_a_bonus(Employee *list[], int size)
{
  for(int i=0; (i<size && list[i]); ++i)
    list[i]->bonus();
}
```

```
void create_lucky_list_and_give_bonus()
{
   Employee **list = new (Employee*) [10];
   for(int i=0; i<10; ++i)
    list[i] = next_lucky_man();
   give_a_bonus(list);
}
```

Более сложный пример - 2

```
class Unit {
public:
   virtual bool action()
              {return false};
};
class Soldier: public Unit {/*...*/};
class Tank : public Unit {/*...*/};
class Mine : public Unit {/*...*/};
```

```
void Field::turn()
{
  for(int i=0; i<unit_number; ++i)
      if ( units[i]->action() != true)
         move_to_end(units[i]);
}
```

```
class Field
{
public:
  Field();
  ~Field();
  void refresh_field();
  void turn();
  void move_to_end(Unit*);

private:
  int unit_number;
  Unit **units;
  //...
}
```

Абстрактные классы

```
class Cosmetics {
public:
  virtual void make_up() = 0;
  virtual void touch_up() = 0;
  virtual void remove() = 0;
};
```

```
class Lipstick :
 public Cosmetics {
public:
  virtual void make_up();
  virtual void touch_up();
  virtual void remove();
};
```

```
class Mascara :
  public Cosmetics {
public:
  virtual void make_up();
  virtual void touch_up();
};
```

```
class WaterRes_Mascara :
  public Mascara {
public:
  virtual void remove();
};
```

```
class Plain_Mascara :
  public Mascara {
public:
  virtual void remove();
};
```

Абстрактные классы (продолжение)

```
Cosmetics cosmo;    // ошибка !!!
Lipstick lips;
Mascara masc;       // ошибка !!!
WaterRes_Mascara wr_masc;
Plain_Mascara plain_masc;
```

```
void complete_touch_up(Cosmetics* todo[], int n)
{
  for (int i=0; (i<n && todo[i]); ++i)
  {
      todo[i]->touch_up();
  }
}
```

```
class Shaver {
public:
  virtual void shave() = 0;
  virtual void reload() = 0;
  virtual void clean() = 0;
};
```

```
class Electric_Shaver :
 public Shaver {
public:
  virtual void shave();
  virtual void reload() {}
  virtual void clean();
};
```

```
class Razor :
  public Shaver {
public:
  virtual void clean();
};
```

```
class Safe_Razor :
  public Razor {
public:
  virtual void shave();
  virtual void reload();
};
```

```
class Blade :
  public Razor {
public:
  void shave();
  virtual void reload() {...}
};
```

Абстрактные классы (продолжение)

- Абстрактный класс – это чистый интерфейс
- Класс асбтрактный, если есть хотя бы одна чисто виртуальная функция (=0)
- Нельзя создать экземпляр абстрактного класса
- Чисто виртуальная фнкция, которая не определена в производном классе, остается чисто виртуальной

Конец