ОБЪЕКТНО-ОРИЕНТРОВАННОЕ ПРОГРАММИРОВАНИЕ

Классы

План

1. Понятие класса
2. Конструктор и деструктор
3. Пример

Класс - это производный структурированный тип, введенный программистом на основе уже существующих типов.

Класс задает некоторую совокупность типизированных данных и позволяет определить набор операций над этими данными.


Компоненты класса
Данные
Функции
Дружественные классы
Дружественные функции
Имена типов

Описание класса

class <имя класса>

{<список элементов>}

<имя класса> - правильный идентификатор

{<список элементов>} – тело класса. Содержит определения и описания типи-зированных данных и принадлежащих классу функций.

Объект класса

Для описания объекта класса используется конструкция:

<имя класса> <имя объекта>

В объекты класса входят данные (элементы), соответствующие компонентным данным класса. Компонентные функции позволяют обрабатывать данные конкретных объектов класса.

Определение объекта класса предусматривает выделение участка памяти и деление этого участка на фрагменты, соответствующие отдельным элементам объекта.

Обращение к объектам класса

2 способа

С помощью квалифицированных имен:

С использованием уточненного имени

<имя объекта>.<имя класса>::<имя компонента>

<имя объекта>.<имя элемента>

<имя объекта>.<имя функции>

Для инициализации объектов класса в его определение можно явно включать специальную функцию – к*онструктор.*

<имя конструктора> ([список параметров])

{ <операторы тела конструктора> };

<имя конструктора> – должно совпадать с именем класса.

Такая функция автоматически вызывается при определении каждого объекта класса. Основное назначение конструктора - инициализация объектов (выделение памяти и разбиение ее на блоки).

Конструктор не ничего возвращает. Даже тип void недопустим. С помощью параметров можно передать любые данные, необходимые для создания и инициализации объектов класса.

Конструктор может быть любой сложности.

Деструктор обеспечивает высвобождение памяти при уничтожении объекта класса.

~<имя деструктора>( )

{<операторы тела деструктора> };

Имя деструктора всегда начинается с символа '~'(тильда), за которым без пробелов или других разделителей помещается имя класса.

У деструктора не может быть параметров (даже типа void). Деструктор не имеет возвращаемого значения (даже типа void). Вызов деструктора выполняется неявно, автоматически, как только объект класса уничтожается.

```
struct Man {
    char name[iName+1];
    int birth_year;
    float pay;
    };
class Man {
    char name[iName+1];
    int bith_year;
    float pay;
    };
```

```
class Man {
        public:
        Man(int iName = 30) // конструктор
              { pName = new char[iName + 1]; }
       ~Man() { delete [] pName; } // деструктор
private:
      char  * pName;
      int  birth_year;
     float  pay;
};
```

```
const  int i_name = 30;
const  int i_year =5;
const  int i_pay =10;
const  int i_buf = i_name + i_year + i_pay;
class Man { public:
        Man(int  iName = 30);
        ~Man();
       int GetBirthYear( ) { return birth_year; }
       float GetPay( ){ return pay; }
       char* GetName( ){ return pName; }
```

```
    void Print( );
    void SetBirthYear(const char*);
    void SetName(const char*);
    void SetPay(const char*);
   int CompareName(const char* name);
   private:
   char* pName;
   int birth__year;
   float pay;
};
```

```
Man::Man(int  iName) {
    cout « “Работает конструктор" « endl;
    pName = new char[iName + 1];
   }

Man::~Man() {
    cout « “Работает деструктор" « endl;
    delete [ ] pName;
}
```

```
void Man::SetName (const char  * fromBuf) {
        strncpy(pName, fromBuf, i_name);
        pName[i_name] = 0;
}

void Man::SetBirthYear(const char  * fromBuf) {
        birth_year = atoi (fromBuf + i_name);
}
```

```
void Man :: SetPay(const char * fromBuf) {
      pay = atof (fromBuf + i_name + i_year);
}

int Man::CompareName(const char * name) {
   if ((strstr (pName,name)) && (pName[strlen(name)] == ' '))
      return 1;
   else
       return 0;}

void Man::Print( ){
cout « pName « birth_year;}
```

```
int main( ) {
    const int maxn_record = 10;
    Man man[maxn_record];
    char buf [i_buf +1] ;
    char name[i_name + 1];
    int i = 0;
    FILE *f;
```

```
f=fopen(“data.txt”,”r”);
for (i=0;i<100;i++)
     fscanf(f,”%s”,&buf);
for (i=0;i<100;i++)
{   man[i].SetName(buf);
    man[i].SetBi rthYear(buf);
    man[i].SetPay(buf);
}
int n_record = i, n_man =0:
float mean_pay =0;
```

```
while (true) {
  cout « "Введите фамилию или слово end: ";
  cin » name;
  if (0 ==strcmp(name, "end")) break;
  int not_found = 1;
  for (i= 0;  i< n_record;  ++i ) {
      if (man[i].CompareName (name) ) {
         man[i].Print();
         n_man++;
         mean_pay += man[i].GetPay();
         not_found = 0;
         break;
    } // if
  } //for
```

```
if (not_found)
    cout « "Такого сотрудника нет" « endl;
}  //while
if (n_man)
     cout « "Ср. оклад: "« mean_pay / n_man « endl;
return 0;
} //main
```