Класове в C++

Любомир Чорбаджиев
Технологическо училище "Електронни системи"
Технически университет, София
lchorbadjiev@elsys-bg.org
Revision : 1.4

4 октомври 2004 г.

Класове: въведение

- Концепцията на класовете е въведена в C++ за да предостави на програмистите механизъм за създаване на нови типове, които да не се различават от базовите (вградените) типове.
- Класът е тип, дефиниран от потребителя. Добре подбраният набор от типове, дефинирани от потребителя, прави програмата по-кратка и по-изразителна.
- Основният смисъл на дефинирането на нови типове се състои в разделянето на несъществените детайли от свойствата, които имат определящо значение за правилното функциониране на програмата.

Дефиниция на клас

- Дефиницията на класа се състои от две части:
  - ◇ **Заглавна част**, която се състои от ключовата дума **class** и идентификатор, обозначаващ името на класа.
  - ◇ **Тяло на класа**, което е затворено във фигурни скоби.
- Дефиницията на класа трябва да бъде последвана от точка и запетая или от списък от дефиниции на променливи.

```
class Point {/*...*/};
class Rectangle {/*...*/} r1,r2;
```


Тяло на класа

- В тялото на класа се дефинира списъкът от членове на класа и нивото на достъп до тях.
- Тялото на класа дефинира област на действие на класа. Декларирането на членове на класа в тялото на класа въвежда имената им в областта на действие на класа.
- Напълно възможно е два класа имат членове с еднакви имена, тъй като те се отнасят до различни области на действие.

Обекти

- Дефиницията на класа може да се разглежда като шаблон, по който се създават обекти.
- Дефинирането на клас създава нов тип в областта на видимост, в която е направена дефиницията. За да се дефинира обект от даден клас, трябва да се дефинира променлива от съответния тип.

```
Point p;

Point* ptr=&p;
Point& ref=p;
```


Член-променливи

- Дефинирането на член-променливите на класа е аналогично на дефинирането на променливи. Класовете могат да имат **нестатични** и **статични** член-променливи.

```
class Point {
  double x_;
  double y_;
};
```

```
class Rectangle {
  double x_, y_, widht_, height_;
};
```


Член-променливи

- Нестатичните член-променливи на класа не мога да се инициализират при дефиницията си в тялото на класа.

```
1 class Foo {
2   int bar_=42; // error!
3 };
```

- Член-променливите на класа се инициализират от **конструктора** на класа.

Член-Функции

- **Член-функциите** реализират множеството от операции, които се извършват върху обектите от даден клас.
- Член-функциите трябва да бъдат декларирани в тялото на класа.

```
1 class Point {
2 public:
3   void set_x(double x);
4 };
```

- Член-функциите могат да се дефинират в тялото на класа.

```
1 class Point {
2 public:
3   double get_x(void) {return x_;}
4 };
```

Член-Функции

- Член-функциите са дефинирани в областта на действие на класа. Тяхното име не се вижда извън областта на действие на класа.
- Извикването на член функция става като се използва оператора . (точка) или оператора -> (стрелка).

```
1 Point p1;
2 p1.set_x(4.2);
3
4 Point* ptr=&p1;
5 ptr->get_x();
```

- Член-функциите имат пълен достъп до всички член-променливи и член-функции на класа.

Достъп до членовете на класа

- Капсулирането (скриването на информацията) е формален механизъм който предпазва вътрешното представяне на даден клас от директни манипулации.
- За ограничаването на достъпа до членове на класа в C++ се използват спецификаторите за достъп — **public**, **private** и **protected**. Чрез спецификаторите за достъп тялото на класа се разделя на секции — публична (public), скрита (private) и защитена (protected).

Достъп до членовете на класа

- **Публичните членове** на класа са достъпни от всички точки на програмата.
- **Скритите членове** на класа са достъпни само в член-функциите на класа и в **приятелите** на класа.
- **Защитените членове** се държат като публични за членовете на производните класове и като скрити за всички останали точки на програмата.

Достъп до членовете на класа

```
1 class Point {
2   double x_;
3   double y_;
4 public:
5   void set_x(double x);
6   void set_y(double y);
7   double get_x(void) {return x_;}
8   double get_y(void) {return y_;}
9 };
```

Достъп до членовете на класа

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

class Point {
  double x_;
  double y_;
};

void dump(const Point& p) {
  cout << p.x_ << "␣" << p.y_ << endl; // error
}
```

```
lubo@dobby:~/school/cpp/notes> g++ -c code/lecture03-point08.cpp
code/lecture03-point08.cpp: In function ‘void dump()’:
code/lecture03-point08.cpp:5: error: ‘double Point::x_’ is private
code/lecture03-point08.cpp:10: error: within this context
...
```


Приятели на клас

- Механизмът на **приятелите** позволява да се даде достъп на определени функции до скритата част на класа.
- Приятелските декларации съдържат ключовата дума **friend**. Няма значение в коя секция на тялото на класа е направена приятелската декларация.

```
class Point {
  friend void dump(const Point& p);
public:
  //...
};
void dump(const Point& p) {
  cout << p.x_ << "␣" << p.y_ << endl;
}
```


Декларация на клас и дефиниция на клас

- Класът са смята за дефиниран когато достигнем края на тялото на класа. Когато класът е дефиниран всички негови членове са известни.
- Възможно е да се декларира клас без да се дефинира. В следният пример е деклариран класът `Point`, без той да е дефиниран.

```
class Point;
```

- Когато класът е деклариран, но не е дефиниран, възможната му употреба е силно ограничена.

Обекти

- При дефиниране на променлива от типа на даден клас се дефинира обект (инстанция) от класа.
- Достъпът до членовете на класа се извършва чрез операторите за достъп — операторът `.` (точка) и операторът `->` (стрелка).

```
Point p;
Point* ptr=&p;
Point& ref=p;

p.set_x(42.0);
ptr->get_x();
ref.get_x();
```

Обекти

- Всеки обект притежава собствено копие на нестатичните член-променливи на класа.
- Всички обекти от даден клас си поделят само едно копие на член-функциите на класа. С други думи в програмата има само едно копие на член-функциите на класа.

```
Point p1,p2;
p1.get_x();
p2.get_x();
```

Когато методът `get_x()` се извиква чрез обекта `p1`, то използваната в метода член-променливата `x_` принадлежи на обекта `p1`. Аналогично за `p2`.

Указател this

- Всяка член-функция притежава указател, който е насочен към обекта, за който тази член-функция е извикана. Това е указателят **this**.
- Ако член-функцията `get_x()` е извикана за обекта `p1`, то указателят **this** ще бъде насочен към обекта `p1`. Член-променливата `x_`, която се обработва в този случай от член-функцията `get_x()`, е свързана с обекта `p1`.
- Дефиницията на член-функцията `get_x()` е еквивалентна на следната:

```
class Point {
public:
  double get_x() {return this->x_;}
  //...
};
```


Указател this

- Програмистът може да използва указателят **this** в член-функциите, за да се обръща към член-променливите (както в предходния пример), но това е излишно.
- Има случаи, в които използването на указателя **this** е задължително.

```
Point p;
p.set_x(1.0).set_y(1.0);
```

- За да е възможно подобно поведение е необходимо метода `set_x()` да връща препратка към обекта, чрез който е извикан.

Указател this

```
class Point {
  double x_;
  double y_;
public:
  Point& set_x(double x) {
    x_=x;
    return *this;
  }
  Point& set_y(double y) {
    y_=y;
    return *this;
  }
};
int main() {
  Point p;
  p.set_x(1.0).set_y(1.0);
  return 0;
}
```