

Обектно ориентирано програмиране

КЛАСОВЕ. МАСИВИ И ОБЕКТИ

Масиви и обекти

Елементите на масив могат да са обекти, но разбира се от един и същ клас. Дефинират се по общоприетия начин.

Дефиниция на масив от обекти

<дефиниция_на_променлива_от_тип_масив_от_обекти> ::=

T <променлива>[size] [= {<инициализиращ_списък>}];

където

- T е име или декларация на клас;
- <променлива> е идентификатор;
- size е константен израз от интегрален или изброим тип с *положителна* стойност;
- <инициализиращ_списък> се дефинира по следния начин:
 <инициализиращ_списък> ::= <стойност>{, <стойност>}
 {, <име_на_конструктор>(<фактически_параметри>)}

Масиви и обекти ...

Пример:

```
Rational table[10];
```

определя масив от 10 обекта от клас rat.

Достъпът до елементите на масива е пряк и се осъществява по стандартния начин – чрез индексирани променливи.

Пример: Чрез индексираните променливи

```
table[0], table[1], ..., table[9]
```

се осъществява достъп до първия, втория и т.н. до десетия елемент на table.

Тъй като table[i] (i = 0, 1, ..., 9) са обекти, възможни са следните обръщания към техни компоненти:

Масиви и обекти ...

```
table[i].read();           // въвежда стойност на table[i]
table[i].print();          // извежда стойността на table[i]
table[i].getNumerator();    // намира числителя на table[i]
table[i].getDenominator(); // намира знаменателя на table[i].
```

Връзката между масиви и указатели е в сила и в случая когато елементите на масива са обекти. Името на масива е указател към първия му елемент, т.е. ако

```
Rational *p = table; // p сочи към table[0]
                    // т.е. p==&table[0]
                    // *(p+i) == table[i], i = 0, 1,...,9
```

Тогава

```
(* (p + i)).print(); // е еквивалентно на table[i].print();
```

Масиви и обекти ...

Масивът може да е член-данна на клас.

Пример:

```
class Example
{
    int a;
    int table[10];
public:
    int array[10];
} x[5];
```

дефинира масив с 5 компоненти, които са от тип Example. Достъпът до компонентите на масива array ще се осъществи по следния начин:

$x[i].array[j]$, $i = 0, 1, \dots, 4$; $j = 0, 1, \dots, 9$.

Масиви и обекти ...

Конструкторите (в частност конструкторът по подразбиране) играят важна роля при дефинирането и инициализирането на масиви от обекти. Масив от обекти, дефиниран в програма, се инициализира по два начина:

- *неявно* (чрез извикване на системния конструктор по подразбиране за всеки обект – елемент на масива);
- *явно* (чрез инициализиращ списък).

Масиви и обекти ...

Примери:

а) Класът

```
const NUM = 25;

class Student
{
public:
    void readStudent();
    void printStudent() const;
    double average() const;
```

Масиви и обекти ...

```
private:
```

```
    int facnom;
```

```
    char name[26];
```

```
    double marks[NUM];
```

```
};
```

няма явно дефиниран конструктор. Дефиницията

```
Student table[30];
```

на масива table от 30 обекта от клас Student е правилна.

Инициализацията се осъществява чрез извикване на “системния” конструктор по подразбиране за всеки обект – елемент на масива.

Масиви и обекти ...

б) Класът Rational, дефиниран по-долу

```
class Rational {  
private:  
    int numer, denom;  
    int gcd(int, int);  
public:  
    Rational(int = 0, int = 1);  
    int getNumerator() const;  
    int getDenominator() const;  
    void print() const;  
    void read();  
};
```

Масиви и обекти ...

притежава явно дефиниран конструктор с два подразбиращи се параметъра. В този случай са допустими дефиниции от вида:

```
Rational x[10]; // x[i] се инициализира с 0/1, за всяко i=0,1,...9.
```

```
Rational x[10] = { 1,2,3,4,5,6,7,8,9,10 }; //x[i] == i+1/1
```

```
Rational x[10] = { Rational(1,21),Rational(2),
```

```
                Rational(3, 5),4,5,6,7,8,9,10 };
```

```
                // x[0] == 1/21; x[1] == 2/1; x[2] == 3/5, x[3]== 4/1, ...
```

Ако променим конструктора на класа Rational от

```
Rational(int = 0, int = 1);
```

в

```
Rational(int, int);
```

т.е. без подразбиращи се параметри и трите дефиниции от по-горе ще съобщят за грешка. Единствено допустима дефиниция на x[10] е с инициализация с 10 обръщения към двуаргументния конструктор Rational с явно указани два аргумента.