Класове. Конструктори. Указатели към обекти на класове. Масиви и обекти.

гл.ас. д-р. Нора Ангелова

Член-функции за достъп

?

Член-функции за достъп

- Тези функции не променят член-данните на класа, а само извличат информация за тях.
- Последното е указано чрез записването на запазената дума const след затварящата скоба на формалните параметри и пред знака ";" на декларацията им и в края на заглавието в дефиницията им.
- Трябва да извиква в себе си само const функции.

```
struct rat {
  ...
  int getNumer() const;
};

int rat::getNumer() const {
  return numer;
}
```

Спецификатори за достъп

- Областта на един спецификатор за достъп започва от спецификатора и продължава до следващия спецификатор или до края на декларацията на класа.

```
class rat {
    int numer; // private компонента
    int denom; // private компонента
  public:
    rat(); // public компонента
    rat(int, int); // public компонента
};
```

Достъп до компонентите на клас

- Вътрешен достъп – достъп на компонентите на класа до други компоненти на същия клас

Член-функциите на клас имат пряк достъп до всички компоненти на класа в т.ч. до себе си без значение на секцията, в която се намират компонентите

Не е необходимо член-данните на класа да се предават явно като параметри. Това се осъществява чрез указателя this

Достъп до компонентите на клас

- Външен достъп – достъп до компоненти на клас чрез обекти на класа, дефинирани във външни за класа функции.

- Външни функции - функциите, които не са член-функции на класа

Външен достъп е възможен единствено до public компонентите на класа

Конструктори

- Името на конструктора съвпада с името на класа.
- Типът на резултата е указателят this и не се указва явно.
- Изпълнява се автоматично при създаване на обекти.
- Не може да се извиква явно (обръщение от вида r.rat(1,4) е недопустимо).

КОНСТРУКТОРИ

- Ако в клас явно не е дефиниран никакъв конструктор, дефиницията на обект от този клас не трябва да съдържа инициализация с явно обръщение към конструктор

  <име_на_клас> <обект>;

В този случай автоматично в класа се създава *подразбиращ се конструктор* и инициализацията на обекта се осъществява чрез него.

Този конструктор изпълнява редица действия, като заделяне на памет за обектите, инициализиране на някои системни променливи и др.

КОНСТРУКТОРИ

Пример:

```
class example {
  public:
    void read();
    void print() const;
  private:
    int a;
    int b;
};

int main() {
   example x; // инициализация чрез подразбиращия се
                // конструктор, създаден от компилатора
   x.read();
   x.print();
   return 0;
}
```

a	b
-	-

КОНСТРУКТОРИ

- Ако в клас явно са дефинирани конструктори, дефиницията на обект от този клас трябва да е в съответствие със съдържащите се в класа конструктори.

```
class example {
  public:
    example(int x, int y) {
      a = x;
      b = y;
    }
    void read();
    void print() const;
  private:
    int a;
    int b;
};
int main() {
   example x; // предизвиква грешка. Няма конструктор по подразбиране.
   return 0;
}
```

Конструктори

Инициализация на член-данните

- в тялото на конструктора

```
class example {
  public:
    example(int x, int y);
    …
  private:
    int a;
    int b;
};
example::example(int x, int y) {
    a = x;
    b = y;
}
```

Конструктори

Инициализация на член-данните

- пред тялото на конструктора (в заглавието) – обобщена синтактична конструкция

```
class example {
  public:
    example(int x, int y);
    …
  private:
    int a;
    int b;
};
example::example(int x, int y) : a(x), b(y) {
}
```

КОНСТРУКТОРИ

Инициализация на член-данните

- в тялото и пред тялото на конструктора

```
class example {
  public:
    example(int x, int y);
    …
  private:
    int a;
    int b;
};
example::example(int x, int y) : a(x) {
    b = y;
}
```

КОНСТРУКТОРИ

Инициализация на член-данните

- Обобщената синтактична конструкция инициализира член-данните в заглавието преди изпълнението на тялото на конструктора.

КОНСТРУКТОРИ

Инициализация на член-данните

- Когато член-данни на клас са обекти, в дефиницията на конструктора на класа се използват конструкторите на класовете, от които са обектите (член-данни).

```
class prat {
  public:
    prat(int, int, int);
    ...
  private:
    int a;
    rat r;
};
```

```
prat::prat(int x, int y, int z) {
    a = x;
    r = rat(y, z);
}
```

КОНСТРУКТОРИ

Инициализация на член-данните

- Преди да започне изпълнението на конструктора, автоматично се извикват конструкторите по подразбиране на всички член-данни, които са обекти.
- Веднага след това тези член-данни се инициализират с обектите, резултат от изпълнението на извикания конструктор.
- Това двойно извикване на конструктори намалява ефективността на програмата.

```
class prat {
  public:
    prat(int, int, int);
    ...
  private:
    int a;
    rat r;
};
```

```
prat::prat(int x, int y, int z) {
    a = x;
    r = rat(y, z);
}
```

Конструктори

Инициализация на член-данните

- Решение

```
class prat {
   public:
     prat(int, int, int);
     ...
   private:
     int a;
     rat r;
};

prat::prat(int x, int y, int z) : r(y, z) {
    a = x;
}
```

Функции и конструктори

Подразбиращи се параметри

- Задаването на подразбираща се стойност се извършва чрез задаване на конкретна стойност в прототипа на функцията или в заглавието на нейната дефиниция

```
class prat {
   public:
     prat(int, int=10, int=5);
     ...
   private:
     int a;
     rat r;
};

prat::prat(int x, int y, int z) : r(y, z) {
    a = x;
}
```

Функции и конструктори

Подразбиращи се параметри

- Ако параметър на функция е подразбиращ се, всички параметри след него също са подразбиращи се
- Подразбиращите се параметри могат да се използват за функции и конструктори

КОНСТРУКТОР ЗА ПРИСВОЯВАНЕ/КОПИРАНЕ

- Инициализацията на новосъздаден обект на даден клас може да зависи от друг обект на същия клас.

Пример:

rat p(1,3);

rat q = p; OR rat q(p);

Тази инициализация се създава от специален конструктор, наречен **конструктор за присвояване (копи конструктор).**

КОПИ КОНСТРУКТОР

- Конструкторът за присвояване е конструктор, поддържащ формален параметър от тип <име_на_клас> const &

Пример:

```
rat::rat(rat const & r) {
    numer = r.numer;
    denom = r.denom;
}
```

- Ако в един клас явно не е дефиниран конструктор за присвояване, компилаторът автоматично създава такъв, в момента когато новосъздаден обект се инициализира с обекта, намиращ се от дясната страна на знака за присвояване или в кръглите скоби.

КОПИ КОНСТРУКТОР

Конструктор за присвояване се използва и при:

- предаване на обект като аргумент на функция
- връщане на обект като резултат от изпълнение на функция

(изключение правят параметрите псевдоними)

КОПИ КОНСТРУКТОР

```
rat::rat(rat const & r) {
  numer = r.numer + 1;
  denom = r.denom + 1;
}
```

...

```
rat p; // p се инициализира с 0/1
rat q = p; // q се инициализира с 1/2
```

Указатели към обекти

- Дефинират се по същия начин както указател към основен тип данни

```
rat p;
rat * ptr = &p;
```

МАСИВИ И ОБЕКТИ

- Елементите на масив могат да са обекти, но от един и същ клас

<дефиниция_на_променлива_от_тип_масив> ::=
T <променлива>[size]; |
T <променлива>[size] = {<инициализиращ_списък>};
където
T е име на клас;
<променлива> ::= <идентификатор>

size е константен израз от интегрален или изброен тип със положителна стойност;
<инициализиращ_списък> ::= <стойност>{, <стойност>}опц
{, <име_на_конструктор>(<фактически_параметри>)}опц
{, <обект_от_тип_T>(<фактически_параметри>)}

КОНСТРУКТОРИ

rat numbers[60];

Какво ще се случи ?

Инициализацията се осъществява чрез извикване на конструктора по подразбиране за всеки обект – елемент на масива (60 пъти).

Всички елементи на масива се инициализират със стойности 0/1.

КОНСТРУКТОРИ

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

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

rat x[10] = {rat(1,2),rat(2),rat(3, 5),4,5,6,7,8,9,10};
// x[0] == ½; x[1] == 2/1; x[2] == 3/5, x[3]== 4/1 ...
```

КОНТРОЛНА ЛЕКЦИИ 2

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

class A {
public:
    void Hello();
};

void A::Hello() {
    cout << "Hello world!\n";
}

int main() {
    A e;
    e.Hello();
    return 0;
}
```

Резултат:
Hello world!

КОНТРОЛНА ЛЕКЦИИ 2

Задача 2

КОНТРОЛНА ЛЕКЦИИ 2

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

class B {
public:
    int a, b;
    void init(int, int);
};

void B::init(int x, int y) {
    a = x;
    b = y;
}

int main() {
    B c;
    c.init(5, 8);
    cout << c.a << " " << c.b << endl;
    return 0;
}
```

Резултат:
5 8

КОНТРОЛНА ЛЕКЦИИ 2

Задача 3

КОНТРОЛНА ЛЕКЦИИ 2

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

class C {
private:
    int a, b;

public:
    void init(int, int);
    void display() const;
};

void C::init(int x, int y) {
    a = x;
    b = y;
}

void C::display() const {
    cout << a << " " << b << endl;
}

int main() {
    C c;
    c.init(5, 8);
    c.display();
    return 0;
}
```

Резултат:

5 8

КОНТРОЛНА ЛЕКЦИИ 2

Задача 4

КОНТРОЛНА ЛЕКЦИИ 2

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

class ExampleClass {
   public:
       ExampleClass();
       ExampleClass(int);
       void f();
       void g();
       int h() const;
       int k() const;
       void pr() const;
   private:
       int n;
};

ExampleClass::ExampleClass() {
    cout << "ExampleClass\n";
    n = 5;
}

ExampleClass::ExampleClass(int a) {
    cout << "ExampleClass(" << a << ")\n";
    n = 2*a-3;
}
```

```
void ExampleClass::f() {
     n++;
}

void ExampleClass::g() {
     f();
     n = 3*n+5;
     cout << k() << endl;
}

int ExampleClass::h() const {
     g();
     return n;
}

int ExampleClass::k() const {
    return n+2;
}

void ExampleClass::pr() const {
    cout << "n= " << n << endl;
}
```

КОНТРОЛНА ЛЕКЦИИ 2

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

class ExampleClass {
   public:
       ExampleClass();
       ExampleClass(int);
       void f();
       void g();
       int h() const;
       int k() const;
       void pr() const;
   private:
       int n;
};

ExampleClass::ExampleClass() {
    cout << "ExampleClass\n";
    n = 5;
}

ExampleClass::ExampleClass(int a) {
    cout << "ExampleClass(" << a << ")\n";
    n = 2*a-3;
}
```

```
void ExampleClass::f() {
     n++;
}

void ExampleClass::g() {
     f();
     n = 3*n+5;
     cout << k() << endl;
}

int ExampleClass::h() const {
     g();
     return n;
}

int ExampleClass::k() const {
    return n+2;
}

void ExampleClass::pr() const {
    cout << "n= " << n << endl;
}
```

КОНТРОЛНА ЛЕКЦИИ 2

```
int main() {
   ExampleClass a;
   ExampleClass b(3);
   a.f(); b.g();
   cout << a.n << endl << b.n << endl;
   ExampleClass c(a.h()+b.h()+b.k()-
a.k());
   a.pr();
   b.pr();
   c.pr();

   return 0;
}
```

```
ExampleClass
ExampleClass(3)
19
ExampleClass(34)
n= 6
n= 17
n= 65
```

КРАЙ