Конструктори

Трифон Трифонов

Обектно-ориентирано програмиране,
спец. Компютърни науки, 1 поток,
2018/19 г.

13 март 2019 г.

Жизнен цикъл на обект

- За обекта се заделя памет и се свързва с неговото име
- Извиква се подходящ конструктор на обекта
- Работа с обекта (достъп до компоненти на обект, изпълняване на операции)
- Достига се края на областта на действие на обекта
- Извиква се деструкторът на обекта
- Заделената за обекта памет се освобождава

Ролята на конструкторите

- Инициализират паметта за обекта
- Осигуряват, че преди да почне да се работи с обекта, той е във валидно състояние
- Позволяват предварително задаване на стойности на полетата

Видове конструктори

- Обикновен конструктор с параметри
- Конструктор по подразбиране
- Конструктор с параметри по подразбиране
- Конструктор за копиране
- Системно генерирани конструктори
  - по подразбиране
  - за копиране
- Конструктор за преобразуване на тип

Дефиниция на конструктор

<конструктор> ::=
  <име-на-клас>::<име-на-клас>(<параметри>)
  [: <член-данна>(<израз>) {, <член-данна>(<израз>) }]
  { <тяло> }

Пример:

```
Rational::Rational(int n, int d) : numer(n), denom(d) {
  if (denom == 0)
    cerr << "Нулев знаменател!";
}
```

Инициализиращият списък се изпълнява преди тялото на конструктора!

Извикване на конструктори

<описание на обект> ::=
<име-на-обект> [= <израз>] |
<име-на-обект>(<параметри>) |
<име-на-обект> = <име-на-клас>(<параметри>)

Примери:

```
Rational r1, r2 = Rational(), r3(1, 2), r4 = Rational(3,4);
Rational r5 = r1, r6(r2), r7 = Rational(r3);
```

Конструктор по подразбиране

- Конструктор без параметри: <име-на-клас>()
- Извиква се при дефиниция на обект без параметри
  - `Rational r1;`
  - ~~`Rational r2();`~~
  - `Rational r3 = Rational();`
- Инициализира обекта с "празни", но валидни стойности
- **Пример:** `Rational::Rational() : numer(0), denom(1) {}`
- Ако в един клас не се дефинира нито един конструктор, системно се създава конструктор по подразбиране с празно тяло

Подразбиращи се параметри

- В C++ е позволено да се задават стойности по подразбиране на някои или всички параметри на функции
- <функция-с-подразбиращи-се-параметри> ::=
<тип> <име> (<параметри> <подразбиращи-се-параметри>)
- <параметри> ::= void | <празно> | <параметър> { , <параметър> }
- <подразбиращи-се-параметри> ::= <празно> |
<параметър> = <израз> { , <параметър> = <израз> }
- **Пример:**

```
int f(int x, double y, int z = 1, char t = 'x')
void g(int *p = nullptr, double x = 2.3)
int h(int a = 0, double b)
```

Конструктор с подразбиращи се параметри

- Конструкторите могат да бъдат с подразбиращи се параметри като всички останали функции
- **Пример:** `Rational(int n = 0, int d = 1)`
- Дефинираме три конструктора наведнъж!
  - `Rational()` $\Longleftrightarrow$ `Rational(0,1)` (конструктор по подразбиране)
  - `Rational(n)` $\Longleftrightarrow$ `Rational(n,1)`
  - `Rational(n, d)`
- Подразбиращите параметри се задават в декларацията на конструктора, ако има такава

Конструктор за копиране

- Конструкторът за копиране служи за инициализиране на обект като се ползва като образец друг обект
- <име-на-клас>(<име-на-клас> const&)
- Образецът не трябва да може да се променя!
- Пример:

```
Rational(Rational const& r) :
        numer(r.numer), denom(r.denom) {}
```

- Ако не напишете конструктор за копиране се създава системен такъв, който копира дословно полетата на образеца
- Конструкторът за копиране обикновено се пише, ако при копирането на обекта е нужно да се случи нещо допълнително

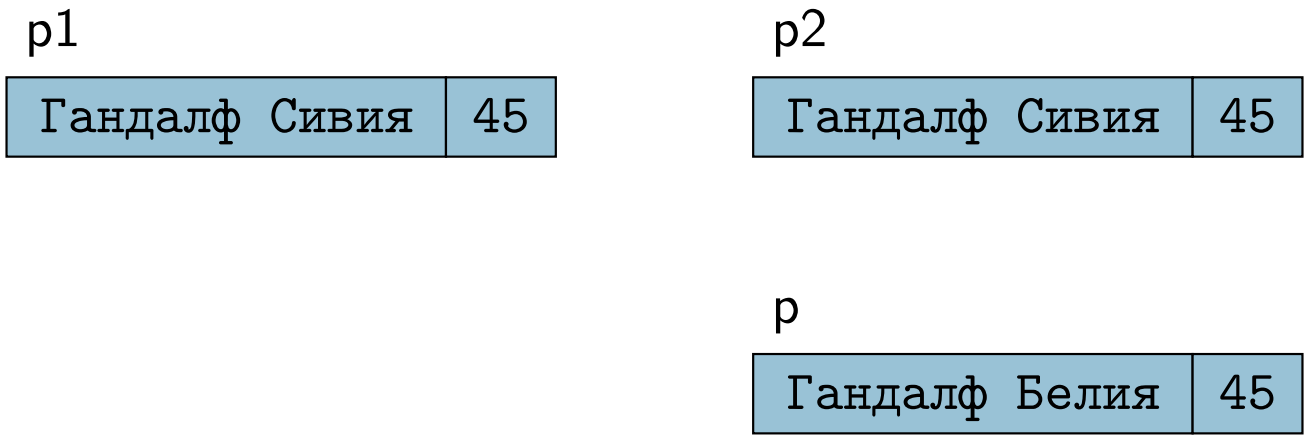
Извикване на конструктор за копиране

- <име-на-клас> <обект>(<образец>)
- <име-на-клас> <обект> = <образец>
- <име-на-клас> <обект> = <име-на-клас>(<образец>)
- Конструктор за копиране се извиква автоматично и при:
  - предаване на обекти като параметри на функции
  - връщане на обекти като резултат от функции
- Конструктор за копиране не се извиква при:
  - предаване и връщане на обекти по указател
  - предаване и връщане на обекти по препратка

Копиране на обекти със статични полета

```
Player p1("Гандалф Сивия", 45);
Player p2 = p1;
p2.setName("Гандалф Белия");
anonymousPrint(p2);
```

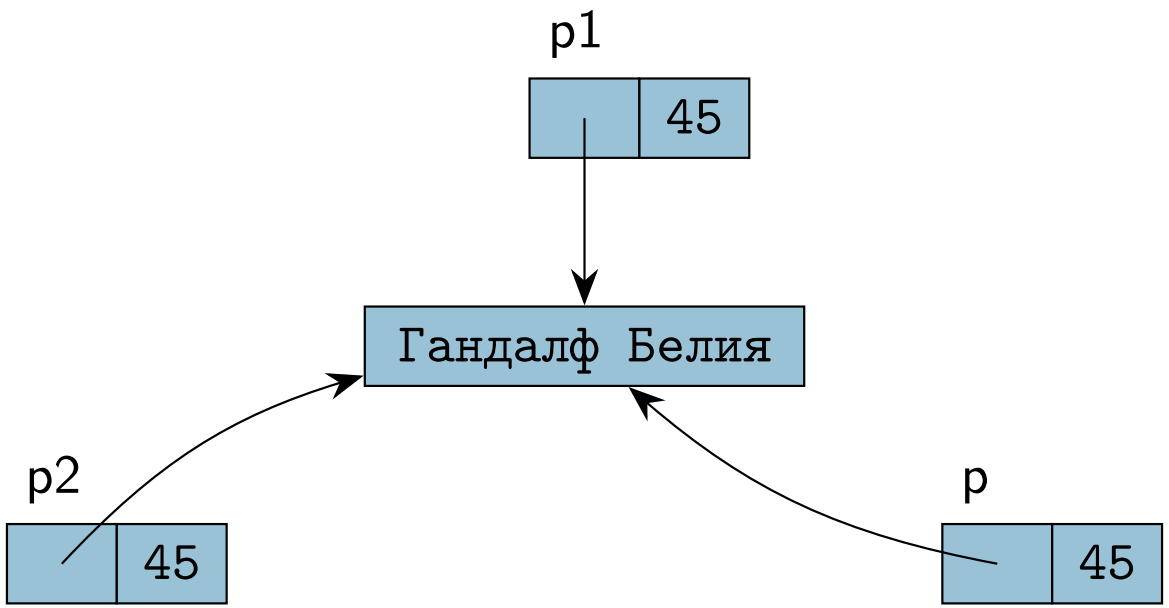
```
void anonymousPrint(Player p) {
  p.setName("Анонимен");
  cout << "Играч:";
  p.print();
}
```



Копиране на обекти с динамични полета

```
Player p1("Гандалф Сивия", 45);
Player p2 = p1;
p2.setName("Гандалф Белия");
anonymousPrint(p2);
```

```
void anonymousPrint(Player p) {
  p.setName("Анонимен");
  cout << "Играч:";
  p.print();
}
```



Конструктор за копиране на динамични полета

- Системният конструктор сляпо копира полетата
- При работа с динамична памет трябва да напишем собствен конструктор за копиране
- Трябва да се погрижим да заделим нова динамична памет и да копираме съдържанието на оригинала
- **Пример:**

```
Player(Player const& p) : score(p.score) {
  name = new char[strlen(p.name)+1];
  strcpy(name, p.name);
}
```