

КОНСТРУКТОРИ.

гл.ас. д-р. Нора Ангелова

КОНСТРУКТОРИ

- Конструкторите са член-функции, чрез които се инициализират член-данните на класа (структурата).

КОНСТРУКТОРИ

- Името на конструктора съвпада с името на класа (структурата).
- Изпълнява се **автоматично** при създаването на обекти.
- Не може да се извика явно.
- Не се указва тип на връщания резултат.
- Връща референция към създадения обект - `this`.

КОНСТРУКТОРИ

Специфики

- ⦿ В клас не е дефиниран конструктор

<име_на_клас> <обект>;

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

- В клас не е дефиниран конструктор

```
class point2 {  
    public:  
        void print() const;  
    private:  
        double x;  
        double y;  
};  
...  
int main() {  
    point2 myFirstPoint; // инициализира се чрез  
                        // подразбиращия се конструктор,  
                        // създаден от компилатора  
    myFirstPoint.print();  
    return 0;  
}
```

x	y
-	-

КОНСТРУКТОРИ

Специфики

- ⦿ В клас явно е дефиниран конструктор/и

Дефиницията на обект от този клас трябва **задължително** да е в съответствие с един от съдържащите се в класа конструктори.

* В класа вече има дефинирани конструктори.
Подразбиращ се конструктор **НЕ** се създава автоматично.

КОНСТРУКТОРИ

Специфики

- В клас явно е дефиниран конструктор/и

```
class point2 {  
    public:  
        point2(double xValue, double yValue) {  
            x = xValue;  
            y = yValue;  
        }  
        void print() const;  
    private:  
        double x;  
        double y;  
};  
  
int main() {  
    point2 myFirstPoint; // Предизвиква грешка  
                        // Няма конструктор по подразбиране  
    return 0;  
}
```

КОНСТРУКТОРИ

- Инициализация на член-данните

КОНСТРУКТОРИ

- Инициализация на член-данните

В тялото на конструктора

```
class point2 {  
    public:  
        point2(double xValue, double yValue);  
        void print() const;  
    private:  
        double x;  
        double y;  
};  
  
point2::point2(double xValue, double yValue) {  
    x = xValue; // Инициализира член-данната x  
    y = yValue; // Инициализира член-данната y  
}
```

КОНСТРУКТОРИ

- ⦿ Инициализация на член-данните

В заглавието на конструктора - обобщена синтактична конструкция

```
class point2 {  
    public:  
        point2(double xValue, double yValue);  
        void print() const;  
    private:  
        double x;  
        double y;  
};  
  
// Инициализира член-данните x и y  
point2::point2(double xValue, double yValue) : x(xValue), y(yValue)  
{}
```

КОНСТРУКТОРИ

- Инициализация на член-данните

Комбинация на двата подхода

```
class point2 {  
    public:  
        point2(double xValue, double yValue);  
        void print() const;  
    private:  
        double x;  
        double y;  
};  
  
// Инициализира член-данната x  
point2::point2(double xValue, double yValue) : x(xValue) {  
    y = yValue; // Инициализира член-данната 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

Член-данни на клас, които са обекти

В дефиницията на конструктора на класа се използват конструкторите на класовете, от които са обектите.

```
class rect {  
    public:  
        rect (double, double, double, double);  
        ...  
    private:  
        point2 topLeft;  
        point2 bottomRight;  
};  
  
rect::rect(double x1, double y1, double x2, double y2) {  
    topLeft = point2(x1, y1);  
    bottomRight = point2 (x2, y2);  
}
```

КОНСТРУКТОРИ

- Инициализация на член-данните


Член-данни на клас, които са обекти

Преди да започне изпълнението на конструктора, **автоматично** се извикват **конструкторите по подразбиране** на всички член-данни, които са обекти (АКО ТОВА Е ВЪЗМОЖНО).

След това тези член-данни се инициализират в тялото на конструктора.

Това двойно извикване на конструктори намалява ефективността на програмата.

```
class rect {  
    public:  
        rect (double, double, double, double);  
        ...  
    private:  
        point2 topLeft;  
        point2 bottomRight;  
};  
rect::rect(double x1, double y1, double x2, double y2) {  
    topLeft = point2(x1, y1);  
    bottomRight = point2 (x2, y2);  
}
```



КОНСТРУКТОРИ

- Инициализация на член-данните

Член-данни на клас, които са обекти

Решение

```
class rect {  
    public:  
        rect (double, double, double, double);  
        ...  
    private:  
        point2 topLeft;  
        point2 bottomRight;  
};  
  
rect::rect(double x1, double y1, double x2, double y2) :  
topLeft(x1, y1), bottomRight(x2, y2)  
{}
```

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 point2 {  
    public:  
        point2(double xValue = 0, double yValue = 0);  
        void print() const;  
    private:  
        double x;  
        double y;  
};
```


ФУНКЦИИ И КОНСТРУКТОРИ

- Подразбиращи се параметри

Ако параметър на функция е подразбиращ се, всички параметри след него също трябва да са подразбиращи се.

Подразбиращите се параметри могат да се използват за функции и конструктори.

ФУНКЦИИ И КОНСТРУКТОРИ

- Подразбиращи се параметри

Колко конструктора са дефинирани?

```
class point2 {  
    public:  
        point2(double xValue = 0, double yValue = 0);  
        void print() const;  
    private:  
        double x;  
        double y;  
};
```

(КОПИ) КОНСТРУКТОР ЗА КОПИРАНЕ

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

Пример:

```
point2 p(1,3);
```

```
point2 secondPoint = p; ИЛИ point2 secondPoint(p);
```

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

(КОПИ) КОНСТРУКТОР ЗА КОПИРАНЕ

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

Пример:

```
point2::point2(point2 const & p) {  
    x = p.x;  
    y = p.y;  
}
```

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

(КОПИ) КОНСТРУКТОР ЗА КОПИРАНЕ

Конструктор за копиране се използва при:

- създаване на обект на даден клас като копие на друг обект на същия клас.
- предаване на обект (по стойност) като аргумент на функция;
- връщане на обект (по стойност) като резултат от изпълнение на функция;

Изключение правят параметрите, които се подават по референция.

(КОПИ) КОНСТРУКТОР ЗА КОПИРАНЕ

```
class point2 {  
    public:  
        point2(double xValue = 0, double yValue = 0);  
        point2(point2 const & p);  
        void print() const;  
    private:  
        double x;  
        double y;  
};
```

```
point2::point2(point2 const & p) {  
    x = p.x;  
    y = p.y;  
}
```

...

```
point2 p;                // p се инициализира с (0,0)  
point2 secondPoint = p;  // secondPoint се инициализира с (0,0)
```

(КОПИ) КОНСТРУКТОР ЗА КОПИРАНЕ

```
class point2 {  
    public:  
        point2(double xValue = 0, double yValue = 0);  
        point2(point2 const & p);  
        void print() const;  
    private:  
        double x;  
        double y;  
};
```


```
point2::point2(point2 const & p) {  
    x = p.x + 1;  
    y = p.y + 1;  
}
```

...

```
point2 p; // p се инициализира с (0,0)  
point2 secondPoint = p; // secondPoint се инициализира с (1,1)
```

(КОПИ) КОНСТРУКТОР ЗА КОПИРАНЕ

```
class rect {  
    public:  
        rect(double x1 = 0, double y1 = 0, double x2 = 0, double y2 = 0);  
        rect(rect const & r);  
    private:  
        point2 topLeft;  
        point2 bottomRight;  
};  
  
rect::rect(rect const & r) {  
    // ...  
}  
  
...  
  
rect r;                // r се инициализира с (0,0), (0,0)  
rect secondRect = r;
```



// член-данни от тип point2 – извиква
// се конструктор по подразбиране за
// двете точки.

(КОПИ) КОНСТРУКТОР ЗА КОПИРАНЕ

Какво се случва ако не дефинираме конструктор за копиране?

- Извършва се автоматично чрез директно присвояване.

Кога да дефинираме конструктор за копиране?

- Когато обект не може да се създаде чрез директно присвояване на член-данните.

Пример - динамично заделяне на памет.

КРАЙ