# Objekty a třídy

Program v jazyce C: samostatné proměnné + samostatné funkce.

Objektově orientovaný program: proměnné i funkce jsou spolu v objektech.

Třída – popisuje strukturu objektů.

Objekt – je instance třídy.

Třída je strukturovaný datový typ obsahující nejen proměnné, ale i funkce.

```
float a1=3,a2=5;
float obsahCtverce(float a) { return a*a; }
obsahCtverce(a1)     // obsah prvního čtverce
obsahCtverce(a2)     // obsah druhého čtverce
```

---

```
struct Ctverec { float a;
                 float obsah() { return a*a; }
};
Ctverec c1,c2;
c1.a=3; c2.a=5;
c1.obsah()     // obsah prvního čtverce
c2.obsah()     // obsah druhého čtverce
```

---

## Specifické funkce třídy - konstruktor a destruktor

Vlastnosti konstruktoru:

- Má stejné jméno jako třída.
- Nevrací hodnotu a ani se před jeho jménem neuvádí klíčové slovo *void* .
- Je polymorfní funkce – můžeme mít více konstruktorů lišící se typem nebo počtem parametrů.
- Uvedení konstruktoru není ve třídě povinné.
- Konstruktor explicitně nevoláme, je volán automaticky při vzniku objektu.

Typické použití konstruktoru je inicializace členských proměnných objektu, přidělení paměti pro dynamické části objektu, otevření souboru atd.

```
struct Ctverec { float a;
                 Ctverec() { }                    // 1. konstruktor
                 Ctverec(float aa) { a=aa; }      // 2. konstruktor
                 float obsah() { return a*a; }
};
```

Použijeme-li konstruktor s parametry, jejich argumenty uvádíme v deklaraci objektu.

```
Ctverec c1,c2(),c3(5);  // c1,c2 - 1. konstruktor, c3 - 2. konstruktor
```

Konstruktor s jedním parametrem je tzv. konvertující konstruktor. Deklaraci objektu můžeme zapsat i pomocí operátoru přiřazení:

```
Ctverec c4=2;   // 2. konstruktor z předchozího příkladu
```

Pro inicializaci proměnných lze v jejich deklaraci vedle přiřazení počáteční hodnoty operátorem přiřazení rovněž použít *inicializátor*. Výraz, kterým je proměnná inicializována, je uzavřen v závorkách za proměnnou.

```
int i(3);    // int i=3;
```

Inicializátor lze použít v konstruktoru třídy pro přiřazení počáteční hodnoty proměnným třídy, kde za hlavičkou konstruktoru uvedeme znak : a za ním seznam inicializátorů (jednotlivé inicializátory jsou vzájemně oddělené čárkami).

```
struct Obdelnik { float a,b;
                  Obdelnik(float a,float b): a(a),b(b) { }
};
```

Členské proměnné, která je referencí, nelze přiřadit počáteční hodnotu jinak než inicializátorem.

```
struct Bod { float x,y; };

struct Usecka { Bod &b1,&b2;
                Usecka(Bod &b1,Bod &b2): b1(b1),b2(b2) { }
};
```

Vlastnosti funkce destruktoru:

- Jeho jméno začíná znakem ~ a za ním je jméno třídy.
- Nemá žádné parametry.
- Nevrací hodnotu a ani se před jeho jménem neuvádí klíčové slovo *void* .
- Uvedení destruktoru ve třídě není povinné.
- Destruktor explicitně nelze zavolat, je volán automaticky při zániku objektu.

Typické použití destruktoru je pro vykonání činností souvisících s ukončením existence objektu (uvolnění dynamicky alokované paměti v objektu, zavření souborů používaných v objektu a další).

```
struct Pole { float *p; int n;
              Pole(int m) { p=new float [n=m]; }
              ~Pole() { delete [] p; }
};
```

Inicializaci proměnných lze udělat i bez konstruktoru. Počáteční hodnoty můžeme přiřadit členským proměnným přímo v jejich deklaraci.

```
struct Ucet { int stav=0;
              void vlozit(int v) { stav+=v; }
              void vybrat(int v) { stav-=v; }
};
```

Pro definici členské funkce platí:

- Může být uvnitř definice třídy – taková funkce má charakter *inline* funkce.
- Může být vně definice třídy a uvnitř třídy je jen její deklarace (prototyp). Definice uvedená vně třídy má před jménem funkce jméno třídy oddělené operátorem rozlišení **::** .

```
struct Pole { float *p; int n;
              Pole(int m) { p=new float [n=m]; }
              void tridit();
              ~Pole() { delete [] p; }
};

void Pole::tridit() { … }
```

Můžeme mít definici všech členských funkcí vně třídy. U těch, které mají mít charakter *inline* funkcí, uvedeme u jejich definice klíčové slovo *inline*.

```
struct Pole { float *p; int n;
              Pole(int);
              void tridit();
              ~Pole();
};

inline Pole::Pole(int m) { p=new float [n=m]; }

void Pole::tridit() { … }

inline Pole::~Pole() { delete [] p; }
```

---

## Zapouzdření

Je základní rys objektově orientovaného programování. Má dva aspekty:

- Proměnné a funkce, které s nimi pracují, jsou spojeny v jeden celek – třídu.
- Členské proměnné třídy jsou chráněny proti tomu, aby jejich hodnoty nemohly změnit (a případně i používat) jiné funkce, než jsou funkce dané třídy. Rovněž i členské funkce třídy mohou být chráněny proti tomu, aby je nemohly volat jiné funkce než členské funkce stejné třídy.

Omezení přístupu k členům třídy:

| | | |
|---|---|---|
| **private** | soukromý | přístupný jen pro funkce dané třídy |
| **protected** | chráněný | přístupný i pro funkce dědících tříd |
| **public** | veřejný | veřejně přístupný |

Implicitní omezení přístupu k členům třídy:

| | |
|---|---|
| **struct** | **public** |
| **class** | **private** |

Konstruktory a destruktor musí být veřejné funkce (chceme-li vytvořit nějaký objekt třídy).

```
class Ctverec { float a;
        public: Ctverec(float a): a(a) { }
                float obsah() { return a*a; }
};
Ctverec c(4);
c.a=3;                  // chyba !!
cout << c.a;            // chyba !!
cout << c.obsah();      // lze
```

---

```
class Ctverec { float a;
        public: Ctverec(float a): a(a) { }
                float strana() { return a; }
                float obsah()  { return a*a; }
};
Ctverec c(4);
cout << c.strana();
cout << c.obsah();
```

---

```
class Ctverec { float a;
        public: Ctverec(float a): a(a) { }
                void zmenit(float aa) { a=aa; }
                float strana() { return a; }
                float obsah()  { return a*a; }
};
```

```
Ctverec c(4);
c.zmenit(3);
cout << c.strana();
cout << c.obsah();
```

---

## Kopírující konstruktor

Vytváří nový objekt jako kopii již existujícího objektu. Je implicitně definován v každé třídě.

Zápis deklarace objektu s použitím kopírujícího konstruktoru:

*třída objekt₂(objekt₁);*

*třída objekt₂=objekt₁;*

```
class Bod { float x,y;
    public: Bod(float x,float y):x(x),y(y) { }
};
Bod b1(1,2);
Bod b2(b1);
Bod b3=b1;
```