2. Třídy v C++

Třídy v C++

Třídy definujeme pomocí klíčového slova **`class`** za kterým uvádíme název třídy, potom následují složené závorky mezi kterými jsou uvedeny členy třídy.

Definice třídy je ukončena středníkem.

Metody se definují podobně jako funkce, jejich příslušnost ke třídě se vyjádří uvedením jména třídy před jménem funkce oddělené dvěma dvojtečkama **::**

Třída je datový typ a proto se jménem třídy pracujeme podobně jako se jmény základních datových typů (`int`, `float`, `double`, `char` ...).

Jmého třídy lze požít k definici proměnné, taková proměnná se nazývá instance třídy nebo objekt nebo objektová proměnná.

K datovým položkám a k metodám přistupujeme pomocí operátoru tečka.

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

class Circle  // Trida pro kruznici
{
public:
  int x, y;  // Souradnice stredu kruznice
  int radius;  // Polomer kruznice
  void draw();   // Deklarace metody
};     // Na konci musi byt strednik

// Definice metody; prislusnost ke tride
// vyjadrime pomoci jmena tridy a ::
void Circle::draw()
{
  cout << "Volana metoda draw()." << endl;
}

int main()
{
  // Definice promenne, instance tridy
  // (též nazývaná objekt)
  Circle circ;

  circ.x = 120;
  circ.y = 150;
  circ.radius = 80;

  circ.draw();  // Volame metodu draw()

  cout << "Souradnice stredu: "
       << circ.x << ", " << circ.y << endl;

  return 0;
}
```

Uvnitř metod lze ke členům dané třídy přistupovat přímo.

Metody mohou přijímat argumenty (parametry) podobně jako ostatní funkce.

Názvy argumentů volíme odlišné od členů třídy.

```
class Circle
{
public:
  int x, y;
  int radius;
  void setCentre(int ax, int ay);
  void draw();
  void printValues();
};

void Circle::setCentre(int ax, int ay)
{
  x = ax;
  y = ay;
}

void Circle::draw()
{
  // Volame metodu dane tridy:
  printValues();
  // Zde by byl kod pro kresleni kruznice
}

void Circle::printValues()
{
  cout << "Stred kruznice: "
       << x << ", " << y << endl;
  cout << "Polomer kruznice: "
       << radius << endl;
}

int main()
{
  // Pro kazdou ze dvou objektovych
  // promennych se v pameti alokuje misto
  // pro datove cleny x, y a radius
  Circle circ1, circ2;

  // Nasledujici dve metody budou operovat
  // nad daty objektu circ1
  circ1.setCentre(100, 90);
  circ1.draw();

  // Nasledujici dve metody budou operovat
  // nad daty objektu circ2
  circ2.setCentre(200, 120);
  circ2.draw();

  return 0;
}
```

Veřejné a soukromé členy třídy

Členy třídy rozdělujeme na veřejné a soukromé; v definici třídy je rozlišujeme pomocí klíčových slov **`public`** a **`private`**. K soukromým členům lze přistupovat pouze z metod dané třídy. K veřejným členům lze přistupovat z libovolných funkcí nebo metod.

Jako veřejné ponecháme pouze ty metody ke kterým potřebujeme přistupovat z funkcí nebo z metod jiných tříd, ostatní ponecháme soukromé.

Datové položky ponecháváme téměř vždy jako soukromé, pro nastavení nebo získání jejich hodnoty použijeme veřejné metody.

```
class Circle
{
public:
  void draw();
  void setCentre(int ax, int ay);
  void setRadius(int r);
  int getCentreX();
  int getCentreY();
  int getRadius();
private:
  int x, y;
  int radius;
  void printValues();
};

void Circle::setRadius(int r)
{
  radius = r;
}

int Circle::getRadius()
{
  return radius;
}

int main()
{
  Circle circ;

  // setRadius() je verejna metoda
  circ.setRadius(80);
  // getRadius() je verejna metoda
  cout << "Polomer: " << circ.getRadius();

  // Nasledujici by nefungovalo!!!
  // Prekladac by ohlasil chybu!
  // radius je soukromy clen tridy
  circ.radius = 80;
  cout << "Polomer: " << circ.radius;
  // printValues() je soukromy clen tridy
  circ.printValues();

  return 0;
}
```

Konstruktor a destruktor

Konstruktor je metoda, která je zavolána automaticky po definici instance třídy (nejdříve se přidělí paměť pro datové členy třídy, pak se volá konstruktor).

Konstruktor má vždy stejné jméno jako třída (podle toho překladač pozná že to je konstruktor).

Konstruktory se využívají zejména pro inicializaci datových členů třídy (tj. nastavený výchozích hodnot).

Konstruktory nemají žádnou návratovou hodnotu.

Konstruktor může přijímat parametry, které mu lze předat při definici instance třídy (předávané hodnoty uvádíme v závorkách za jménem definované proměnné).

Destruktor je metoda, která je automaticky zavolána po zrušení instance třídy (např. lokální proměnné jsou rušeny po opuštění funkce).

Jméno destruktoru je tvořeno znakem ~ a jménem třídy (podle toho překladač pozná že to je destruktor).

Destruktory se využívají zejména pro uvolnění dynamicky alokované paměti.

Destruktory nemají žádnou návratovou hodnotu ani nemohou přijímat žádné parametry.

```
class Circle
{
public:
  Circle();  // Konstruktor bez parametru
  // Konstruktor s parametry:
  Circle(int ax, int ay, int r);
  ~Circle();       // Destruktor
private:
  int x, y;
  int radius;
};

Circle::Circle()
{
  x = 0;
  y = 0;
  radius = 100;
}

Circle::Circle(int ax, int ay, int r)
{
  x = ax;
  y = ay;
  radius = r;
}

Circle::~Circle()
{
  // Tady muze byt kod napr. pro uvolneni
  // dynamicky alokovane pameti
}

int main()
{
  Circle circ1; // V tomto okamziku se
  // alokuje pamet pro promennou a pote se
  // automaticky zavola konstruktor bez
  // parametru Circle()
  Circle circ2(120, 150, 80); // Definice
  // objektove promenne s inicializaci.
  // Parametry jsou predany konstruktoru
  // Circle(int ax, int ay, int r)
  return 0;
}
```

Předávání instancí třídy jako parametry funkcí a metod

S instancemi třídy se pracuje podobně jako s běžnými proměnnými, lze je předávat jako parametry do funkcí a metod.

```
// Nekde na zacatku je definovana trida
// Circle a jeji metody

// Funkce vykresli dve kruznice, ktere
// jsou ji predany jako parametr
void drawCircles(Circle circ1,
                 Circle circ2)
{
  circ1.draw();
  circ2.draw();
}

int main()
{
  Circle circ1, circ2;

  drawCircles(circ1, circ2);

  return 0;
}
```

Instance třídy jako návratová hodnota funkce nebo metody

Instance třídy lze vrátit z funkce příkazem **`return`**.

```
// Tady nekde na zacatku je definovana
// trida Circle

// Funkce vrati kruznici s vetsim
// polomerem
Circle getLargerCircle(Circle circ1, Circle
circ2)
{
  if (circ1.getRadius()
      > circ2.getRadius())
    return circ1;
  else
    return circ2;
}

int main()
{
  Circle circ1, circ2;
  Circle largerCirc;

  largerCirc =
      getLargerCircle(circ1, circ2);

  return 0;
}
```

Funkce vs. metody

Metody upřednostňujeme před funkcemi, zejména pokud funkce/metoda pracuje s datovými položkami dané třídy.

```
class Circle
{
public:
  void printValues();
};

// Metoda tridy Circle:
void Circle::printValues()
{
  cout << "Stred kruznice: "
     << x << ", " << y << endl;
  cout << "Polomer kruznice: "
     << radius << endl;
}

// Funkce
void printCircleValues(Circle circ)
{
  cout << "Stred kruznice: "
     << circ.getCentreX()
     << ", " << circ.getCentreY() << endl;
  cout << "Polomer kruznice: "
     <<   circ.getRadius() << endl;
}

int main()
{
  Circle circ;

  // Optimalni reseni – volani metody:
  circ.printValues();

  // Mene vhodne reseni – volani funkce:
  printCircleValues(circ);
  return 0;
}
```

Dodržujte následující pravidla

- Na konci definice třídy nezapomeňte uvádět středník.
- Pro každou třídu definujte její konstruktor.
- V konstruktoru vždy inicializujte všechny datové členy třídy (pokud konstruktor přijímá parametry, budou zpravidla inicializovány pomocí nich, jinak je inicializujeme vhodnou hodnotou, zpravidla nulou a pod.)
- Datové členy třídy uvádějte vždy jako soukromé (tj. v sekci `private`).

Úloha 1 — 2 body

Vytvořte program pro kreslení kružnice. V programu definujte třídu Circle která bude mít konstruktor (bez parametrů), a následující veřejné metody:

- `setCentre()` nastavující souřadnice kružnice
- `setRadius()` nastavující poloměr kružnice
- `setColor()` nastavující číslo barvy kružnice
- `printValues()` vypisující hodnoty datových členů (souřadnice středu, poloměr, číslo barvy)
- `draw()` vykreslující kružnici na obrazovku (pomocí knihovny g2)

Ve funkci `main()` definujte objektovou proměnnou pro kružnici. Potom si program si vyžádá od uživatele souřadnice středu kružnice (v pixelech), poloměr kružnice (v pixelech) a číslo barvy (1, 3, 7, 19 nebo 25). Hodnoty se načtou do lokálních proměnných a potom se nastaví příslušné hodnoty v objektové proměnné kružnice. Nakonec se zavolá metoda `printValues()`, která vypíše hodnoty a pak metoda `draw()`, která vykreslí kružnici.

Úloha 2 — 1 bod

Program modifikujte tak aby konstruktor přijímal čtyři parametry (souřadnice středu, poloměr, číslo barvy). Při definici proměnné kružnice předejte do konstruktoru vhodné hodnoty. Dále implementujte ve třídě `Circle` metody `readCentre()`, `readRadius()` a `readColor()` které od uživatele vyžádají příslušné hodnoty. Dále implementujte metodu `readValues()`, ktrá postupně zavolá tři výše zmíněné metody. Tuto metodu použijte v programu pro načtení dat od uživatele.

Úloha 3 — nepovinná, 1 bod

Modifikujte program z úlohy 2 tak, aby barva nebyla zadávána jako číslo ale formou textu (*black*, *blue*, *green*, *red*, *yellow*). Podobně při výpisu hodnot se barva vypíše jako text. Ve třídě `Circle` však bude bude hodnota barvy uchovávána i nadále jako číslo. Pro tento účel implementujte ve třídě `Circle` dvě soukromé metody `getColorNumberFromString()` a `getColorStringFromNumber()`.

Úloha 4 — 1 bod

Vytvořte program, který si od uživatele vyžádá souřadnice a poloměr pro dvě kružnice. Potom v jednom okně vykreslí tyto dvě kružnice a navíc třetí kružnici, jejíž střed bude ležet na spojnici středů těchto dvou kružnic a velikost poloměru bude průměrem z poloměru dvou zadaných kružnic. Použijte stejnou třídu `Circle` jako v předchozí úloze. Navíc v ní implementujte metody `getCentreX()`, `getCentreY()` a `getRadius()` pro získání hodnot příslušných členských proměnných. Dále mplementujte metodu `setAverageCircle()`, která přijme jako parametr dvě kružnice a z nich spočítá hodnoty svého středu a poloměru, jak je uvedeno výše. První kružnice bude vždy zelená, druhá modrá a třetí zprůměrovaná kružnice bude červená.