Základy programování v C++ 13. cvičení Ukazatele

Zuzana Petříčková

7. listopadu 2019

Přehled

Ukazatele v C/C++

Ukazatel (pointer)

- **laicky:** proměnná, která ,,ukazuje" na jinou proměnnou v paměti
- **lépe:** proměnná, jejíž hodnotou je adresa v paměti počítače
  - ukazatel na data (např. na jinou proměnnou)
  - ukazatel na funkci

Kdy ukazatele využijeme?

- efektivnější práce s daty (např. s poli)
- předávání parametrů funkcí odkazem
- dynamická alokace paměti

Ukazatele v C/C++

Ukazatel (pointer) - deklarace:

```
domenovy_typ *jmeno; // kolem * mohou a nemusi byt mezery
```

Příklady

```
int i = 16;
cout << &i << endl; // adresa promenne i v pameti
int j = (int)(&i);  // adresa jako cislo v desitk. soust.

int *pi;            // pi je ukazatel na datovy typ int
pi = &i;            // do ukazatele pi vlozime adresu i
int *pj = &i;       // inicializace ukazatele pj na adresu i
double *pd, *pe;    // deklarace dvou ukazatelu na double
                    // (bez inicializace)
```

- operátor **&** ... získání adresy proměnné v paměti

Ukazatele v C/C++

Příklady - pokračování:

```
int i = 16;
int *pi;          // pi je ukazatel na datovy typ int
pi = &i;          // do ukazatele pi vlozime adresu i
int *pj = &i;     // inicializace ukazatele pj na adresu i

cout << &i << endl; // vypise adresu promenne i v pameti
cout << pi << endl; // vypise adresu promenne i v pameti

cout << i << endl; // vypise hodnotu promenne i
cout << *pi << endl; // vypise hodnotu promenne i
```

- operátor **&** ... získání adresy proměnné v paměti
- operátor * ... přístup k paměti, na kterou ukazatel ukazuje (tzv. dereferencování)
  symbol * je použit také v deklaraci ukazatele

Ukazatele v C/C++

Příklady - pokračování:

```
int i = 3, j = 5;
int *pi;            // pi je ukazatel na datovy typ int
pi = &i;            // do ukazatele pi vlozime adresu i

*pi = 10;           // do promenne i vlozime hodnotu 10
          // i = 10;

*pi = *pi + j;      // v promenne i bude hodnota 15
          // i = i + j;

pi = &j;            // do ukazatele pi vlozime adresu j
(*pi)++;            // v promenne j bude hodnota 6
          // j++;
```

Ukazatele v C/C++

Ukazatel nikam (nullpointer)

- neukazuje na žádnou platnou adresu

```
int *pi;
...
pi = nullptr; // od C++11, doporuceno
pi = 0;       // starsi zapis
pi = NULL;    // makro, knihovna stdef.h
```

Potenciální problémy ukazatelů

- ukazatel ukazuje „na náhodnou adresu v paměti" → mohu přepsat paměť, která mi nepatří
- ukazatel ukazuje „nikam" (nullpointer) → pokus o dereferenci vyvolá výjimku

```
int *pi;
...
pi = nullptr;
cout << (*pi) << endl // CHYBA!
```

Ukazatele v C/C++

Automatická konverze na logickou hodnotu

- ukazatel lze použít všude tam, kde se očekává logická hodnota
- ukazatel nikam (nullptr) se konvertuje na **false**
- nenulový ukazatel se konvertuje na **true**

```
int *pi;
...
if (pi)
    cout << (*pi) << endl;
```

Ukazatele v C/C++

Ukazatel bez doménového typu

- lze mu přiřadit hodnotu libovolného ukazatele
- pro opačné přiřazení je třeba použít přetypování
- nelze dereferencovat

```
int i=1, j=2;
int *pi = &i; // pi ukazuje na i
void *v = &j; // v ukazuje na j
pi = (int*)v; // pi ukazuje na j
// nelze i = (*v);
i = (*pi);    // i bude 2
...
```

Ukazatele v C/C++

Ukazatel jako parametr funkce

```
void vypis(int *x)
{
    cout << "Hodnota promenne je " << (*x) << endl;
    cout << "Adresa promenne je " << x << endl;
}
int main()
{
    int a = 56;
    int *pa = &a;
    vypis(&a);
    vypis(pa);
    ...
}
```

Ukazatel jako parametr funkce

Ukazatele lze (podobně jako reference) použít pro předávání parametrů odkazem

```
void funkce(int *a, int *b)
{
    ...
}
```

- funkce může měnit skutečné hodnoty, na které ukazují **a** a **b** (změna se projeví „i venku")

Ukazatel jako parametr funkce - příklad:

```
void zmen(int &x)
{
    x++;
    cout << "Hodnota promenne uvnitr je " << x << endl;
}
void zmen(int *px)
{
    (*px)++;
    cout << "Hodnota promenne uvnitr je " << (*px) << endl
}
int main()
{
    int a = 56;
    zmen(a);
    cout << "Hodnota promenne je " << a << endl;
    zmen(&a);
    cout << "Hodnota promenne je " << a << endl;
    ...
}
```

Funkce a předávání hodnot odkazem

1. **pomocí ukazatelů** (bylo již v jazyce C)
   - trochu nepohodlné (jiný zápis, musíme dereferencovat)
   - potenciálně nebezpečné (co když je **px** nullptr nebo ,,ukazuje jinam než by měl"?)
2. **pomocí tzv. referencí** (v jazyce C++)
   - bezpečnější a pohodlnější způsob
     reference z principu ,,nezná" nullptr (někdy výhoda)
   - interně se jedná o konstantní statické ukazatele, ale syntaxe je stejná jako pro typ, na který reference odkazuje

Ukazatele v C/C++ ... cvičení (základy)

1. vytvořte proměnnou typu double, vypište na konzoli její adresu v paměti, vytvořte ukazatel ukazající na tuto proměnnou a pomocí něj změňte hodnotu této proměnné a vypište ji
2. sečtěte dvě proměnné typu double pomocí ukazatelů na ně a ukazatele na výsledek, výsledek vypište
3. napište a zavolejte funkci **void prohod(int *x, int *y)**, která prohodí obsah dvou proměnných typu int
4. pomocí ukazatelů napište funkci **void serad(int *x, int *y)**, která ,,seřadí" dvě proměnné typu int dle velikosti (první bude menší)