Cvičení #6

Ukazatele a pole

Co je ukazatel

- Celočíselná proměnná
- Místo hodnoty obsahuje pouze adresu do paměti
- Index ve hranaté závorce za proměnnou ukazatel značí posun v paměti od adresy obsažené v této proměnné

Výhody ukazatelů

- Nepracujeme s celým objektem, ale s odkazem
- Rychlý, přímý přístup k prvkům pole

Operátor *

- Unární operátor ***** má dvě funkce, záleží na kontextu
  - Při deklaraci proměnné určuje, zda se jedná o ukazatel (neboli adresu objektu v paměti)
  - Uvedený před proměnnou typu ukazatel vrací přímo objekt, na který proměnná ukazuje (= **dereference**) (transformuje adresu do paměti na příslušný objekt)

Operátor &

- Unární operátor **&** má také dvě funkce
  - Při deklaraci proměnné určuje, že proměnná je typu reference
  - Uvedený před proměnnou vrací ukazatel na tuto proměnnou (transformuje objekt na jeho adresu v paměti)

Ukazatel vs int

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

int main() {
   int cislo;
   int *p_cislo;
   long int fakePointer;

   cislo = 5;
   cout << cislo << endl;  // Tiskne 5

   p_cislo = &cislo; // Adresu "cisla" ulozime do ukazatele
   *p_cislo = 10; // Dereference
   cout << cislo << endl;  // Tiskne 10

   fakePointer = (long int) &cislo; // A co ted?
   *(int *)fakePointer = 15;  // !!!
   cout << cislo << endl;  // Tiskne 15

   return 0;
}
```

Ukazatel vs reference

- Reference je v podstatě pouze alias na objekt, který odkazuje
- Oproti ukazateli je usnadněna práce:
  - Není třeba uvádět operátor dereference *****
  - Přístup k metodám a datovým složkám strukturovaných datových typů je přes operátor **.** narozdíl od ukazatelů, kde je třeba použít **->**
- Referenci nelze deklarovat samostatně, pouze s inicializací a pak ji nelze změnit

Předávání parametrů

```
// hodnotou - cely objekt v parametru kopirovan
vector<int> serad1(vector<int> cisla){
   sort(cisla.begin(), cisla.end());
   return cisla;
}

// odkazem - pomoci reference
void serad2(vector<int> &cisla){
   sort(cisla.begin(), cisla.end());
}

// odkazem - pomoci ukazatele
void serad3(vector<int> *cisla){
   sort(cisla->begin(), cisla->end());
}
```

Vícerozměrné ukazatele

Ukazatel může ukazovat i na ukazatel:

```
int main() {
  int a = 10;
  int *p_a = &a;
  int **p_p_a = &p_a;

  cout << a << endl; // Tiskne 10
  cout << *p_a << endl; // Tiskne 10
  cout << **p_p_a << endl; // Tiskne 10
  return 0;
}
```

Pole v C++

- Složený datový typ
- Umožňuje ukládat prvky stejného typu.
- Nemůžeme libovolně měnit jeho velikost.
- Velikost pole se specifikuje při jeho deklaraci.
- Pole si nepamatuje svoji velikost.
- Prvky v poli jsou indexovány od nuly.

Pole v C++

- Přístup k jednotlivým prvkům pomocí indexu.
- Typ prvku pole – libovolný, kromě:
  - Referencí,
  - funkcí (ale mohou být ukazatele na funkci).
- Prvkem pole může být opět pole (vícerozměrná pole).
- Alokace pole:
  - statická, pokud je velikost známa v době překladu,
  - dynamická.

Statická pole v C++

Příklady

int pole_int[3]; // deklarace pole velikosti 3 obsahující neinicializované prvky typu int

int pole_int[3] = {1,2,3}; //deklarace pole a jeho inicializace

int pole_int[] = {1,2,3}; //výsledek stejný jako předchozí

Procházení pole

Příklad výpisu pole

```
int a[5] = {1, 2, 3, 4, 5}; //indexy prvků 0..4 !
for (int i=0; i < 5; ++i) {
    cout << “Prvek a[” << i << “] = ” << a[i] << endl;
}
```