

3. Proseminář

Pole, řetězce ve stylu C, ukazatele

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 překladač,
 - dynamická.

Statické 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;  
}
```

Ukazatele

- Ukazatel - abstrakce adresy ve vyšším programovacím jazyce.
- Proměnná obsahuje adresu, na které se nachází zpřístupňovaná data:
 - proměnná,
 - objekt,
 - funkce,
 - Jiný ukazatel.

Ukazatele

- Operace s ukazatelem:
 - dereference (*) – zpřístupnění místa, kam ukazatel směřuje,
 - reference (&) – získání adresy paměťového místa (pořízení ukazatele).

Ukazatele

- Deklarace proměnné typu ukazatel:
int * dataPtr;
- Vyhradí v paměti prostor pro adresu:
 - typ. 4B pro 32-bit prostředí,
 - typ. 8B pro 64-bit prostředí.
- Proměnná **dataPtr** je ukazatel, který může ukazovat na hodnotu typu **int**.
- Deklarací není nastavena adresa – **dataPtr** zatím ukazuje "někam do paměti".

Ukazatele

- Příklad

int a = 10; //deklarace proměnné typu **int**

int *b; //deklarace proměnné typu **ukazatel na int**

b = &a; //do **b** přiřadíme adresu proměnné **a** (operátor reference)

cout << *b << endl; //výpis hodnoty proměnné **a** na kterou ukazuje **b** (operátor dereference)

Řetězce ve stylu C

- Pole prvků typu char
- Příklad

char r1[] = "Ahoj"; // řetězec ve stylu C

char r2[] = {'A', 'h', 'o', 'j'}; // **není řetězec ve stylu C**, chybí „nula“ na konci

char r3[] = {'A', 'h', 'o', 'j', '\0'}; // řetězec ve stylu C

Řetězce ve stylu C

<code>strlen(s)</code>	Returns the length of <code>s</code> , not counting the null.
<code>strcmp(s1, s2)</code>	Compares <code>s1</code> and <code>s2</code> for equality. Returns 0 if <code>s1 == s2</code> , positive value if <code>s1 > s2</code> , negative value if <code>s1 < s2</code> .
<code>strcat(s1, s2)</code>	Appends <code>s2</code> to <code>s1</code> . Returns <code>s1</code> .
<code>strcpy(s1, s2)</code>	Copies <code>s2</code> into <code>s1</code> . Returns <code>s1</code> .
<code>strncat(s1, s2, n)</code>	Appends <code>n</code> characters from <code>s2</code> onto <code>s1</code> . Returns <code>s1</code> .
<code>strncpy(s1, s2, n)</code>	Copies <code>n</code> characters from <code>s2</code> into <code>s1</code> . Returns <code>s1</code> .

```
#include <cstring>
```

Knihovny funkce, zdroj C++ Primer, 4th Edition, Addison-Wesley

Řetězce ve stylu C

- Nedoporučují se používat, avšak programátor v C++ se s nimi může často setkat (starší programy, nepřepsané části kódu z C apod.).
- Jsou častým zdrojem bezpečnostních chyb (Buffer over-flow)

Dynamická alokace pole

- Použitím operátoru new:

```
int size = 10;
```

```
int *pa = new int[size](); // pa ukazuje na  
                             první prvek pole
```

```
delete [] pa; // pokud pole již nepotřebujeme,  
                uvolníme paměť
```

Průchod polem pomocí ukazatele

```
int a[] = {1, 2, 3, 4, 5};
```

```
int *zacatek_a = a;
```

```
int *konec_a = a + 5; //Mohu dereferencovat?
```

```
for(int *p=zacatek_a; p!=konec_a; p++) {  
    cout << *p << endl;  
}
```

Průchod polem pomocí ukazatele

```
int a[] = {1, 2, 3, 4, 5};
```

```
int *i = a;
```

```
int *konec_a = a + 5;
```

```
while(i!=konec_a) {
```

```
    cout << *i << endl;
```

```
    i++;
```

```
}
```

Zadání prosemináře

- Implementujte zásobník pomocí dynamicky alokovaného pole v C++, který rozumí příkazům push <int>, pop, size.
- Vstup:
push 10
pop
push 20
size
- Výstup:
10
1

Zadání domácího úkolu

- Na stránkách předmětu v sekci Domácí úkoly
- <https://edux.feld.cvut.cz/courses/A7B36PJC/homeworks/02>
- Na vypracování je opět 14 dní od zadání