

Základy programování v C++ - 8.-9. cvičení Pole

Zuzana Petříčková

19. října 2019

Přehled

1 Pole v C++

- Jednorozměrná pole
- Příklady z 8. cvičení
- Vícerozměrná pole
- Příklady z 9. cvičení

Pole v C++

Jednorozměrná (statická) pole

- skupina proměnných téhož typu umístěných v paměti za sebou
- deklarace:

```
datovy_typ  nizev[rozmer];
```

- *rozmer* je konstantní výraz představující počet prvků pole

Příklady deklarace a inicializace:

```
int  a[10];           // pole 10 prvku typu int
int  a1[5] = {1,2,3,4,5}; // inicializace
int  a2[]  = {1,2,3,4,5}; /* delka pole podle
                           poctu prvku */
int  a3[5] = {1,2,3};  /* nezadane prvky jsou
                           nastaveny na 0 */
```

Jednorozměrná statická pole v C++

Indexování

- přístup k prvkům pole pomocí operátoru indexování []
 - první prvek má index 0
 - poslední prvek má index $rozmer - 1$
 - při indexování se **nekontroluje**, zda daný index leží ve správných mezích

```
const int rozmer = 10;
int pole[rozmer];
...
int x = pole[5]+3;
pole[0] += x;
for (int i = 0; i < rozmer; i++)
{
    cout << pole[i] << ' ';
}
```

Jednorozměrná statická pole v C++

Kopírování pole

- obsah polí nelze kopírovat přiřazením, ale musíme kopírovat prvek po prvku:

```
int a[10], b[10];  
...  
// nelze a = b;  
for (int i = 0; i < 10; i++)  
{  
    a[i] = b[i];  
}
```

Jednorozměrná statická pole v C++

- délka pole musí být konstantní výraz:

```
#define ROZMER 10           // makro preprocesoru
...
const int rozmer = 10;
...
int a[10];                // rozmer je celociselna konstanta
int a1[rozmer];          // ...      konstantni promenna
int a2[ROZMER];          // ...      makro (symbolicka konst.)
```

Jednorozměrná statická pole v C++

Pole jako parametr funkce

```
void funkce(float pole [], int rozmer)  
void funkce(float *pole, int rozmer)
```

- obě deklarace mají stejný význam (předává se tzv. ukazatel)
- do hranatých závorek nepíšeme rozměr (byl by ignorován)
- délka pole se nedá uvnitř funkce jednoduše zjistit, proto ji musíme předávat samostatně

POZOR:

- na rozdíl od dosavadních datových typů nelze funkci předat lokální kopii pole (tj. předat pole hodnotou).
- předává se ukazatel na pole, v těle funkce lze měnit hodnoty v poli

Jednorozměrná statická pole v C++

Příklad 1 ... funkce přináší ke všem prvkům pole hodnotu.

```
void prinasob(double *pole, int rozmer, double hodnota)
{
    for (int i =0; i < rozmer; i++)
        pole[i] *= hodnota;
}
```

Příklad 2 ... funkce vrátí index maxima pole

- *const* u parametru typu pole zaručuje, že pole uvnitř funkce nebude změněno.

```
int maximum(const double a[], int rozmer)
{
    int im = 0; // index maxima
    for (int i = 1; i < rozmer; i++)
    {
        if (a[i] > a[im])
            im = i;
    }
    return im;
}
```


Jednorozměrná statická pole v C++

Příklady

- 1 Napište funkci **void vypis(const double *a, int n)**, která vypíše na konzoli na jeden řádek všechny prvky pole **a** délky **n**.
- 2 Napište funkci **double skalarniSoucin(const double *a, const double *b, int n)**, která spočítá skalární součin vektorů **a** a **b** délky **n**.
- 3 Napište funkci **bool kolme(const double *a, const double *b, int n)**, která vrátí **true**, pokud jsou vektory **a** a **b** délky **n** kolmé.
- 4 Napište funkci **void prohod(double *a, int i, int j)**, která v poli **a** prohodí prvky na indexech **i** a **j**.
- 5 Napište funkci **void otoc(double *a, int n)**, která otočí pořadí prvků v poli **a** délky **n**.
- 6 Napište funkci **void setridBublinkove(double *a, int n)**, která setřídí pole **a** délky **n** v pořadí od nejmenšího prvku po největší pomocí algoritmu bublinkového třídění (bubble sort).

Jednorozměrná statická pole v C++

Další příklady na procvičení (spíše pro začátečníky)

- 1 Napište a zavolejte funkci **double prumer(const double *a, int n)**, která vrátí aritmetický průměr prvků v poli **a** délky **n**.
- 2 Napište a zavolejte funkci **double geometrickyPrumer(const double *a, int n)**, která vrátí geometrický průměr prvků v poli **a** délky **n**.
- 3 Napište a zavolejte funkci **void absolutniHodnota(double *a, int n)**, která nahradí prvky v poli **a** délky **n** za jejich absolutní hodnotu.
- 4 Napište a zavolejte funkci **int pocetZapornych(const double *a, int n)**, která vrátí počet záporných prvků v poli **a** délky **n**.

Vícerozměrná pole v C++

- pole, jehož prvky jsou pole

```
int A[2][3]; // matice s 2 radky a 3 sloupci  
int B[2][3][4]; // trojrozměrné pole
```

- ve skutečnosti: A je pole o 2 prvcích, jehož prvky jsou pole o 3 prvcích typu int

```
A[0][0]  A[0][1]  A[0][2]  
A[1][0]  A[1][1]  A[1][2]
```

- v paměti jsou prvky matice umístěny za sebou:

```
A[0][0] A[0][1] A[0][2] A[1][0] A[1][1] A[1][2]
```

Vícerozměrná pole v C++

Inicializace

```
int A[3][2] = {{1,2},{3,4},{5,6}};
int B[2][3] = {{1,2,3},{4,5,6}};
int C[][3] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
int D[2][3] = {{1,2,3}};
int E[2][3] = {1,2,3,4,5,6}; /* vnitřní závorky
                             lze vynechat */
```

Indexování

```
double A[3][2] = {{1,2},{3,4},{5,6}};
double x = A[2][0]; // x == 5
                // x = A[2,0] nelze
A[1][1] += x;
```

Vícerozměrná pole v C++

Příklad 1 ... program sečte dvě matice a vypíše výsledek

```
float a[3][2] = {{1,0},{0,-1},{0,1}};  
float b[3][2] = {{-1,-2},{0,3},{-5,2}};  
float c[3][2];      // vysledek souctu  
...
```

Příklad 2 ... program vynásobí dvě matice a vypíše výsledek

```
float a[3][2] = {{1,0},{0,-1},{0,1}};  
float b[2][4] = {{0,1,0,0},{0,-1,-1,0}};  
float c[3][4];      // vysledek soucinu  
...
```

Vícerozměrná pole v C++

Příklad 1 ... program sečte dvě matice a vypíše výsledek

```
#include <iomanip> // setw (formatovani vystupu)
```

```
...
```

```
void prikklad1 ()
```

```
{
```

```
    float a[3][2] = {{1,0},{0,-1},{0,1}};
```

```
    float b[3][2] = {{-1,-2},{0,3},{-5,2}};
```

```
    float c[3][2]; // vysledek souctu
```

```
    for(int i = 0; i < 3; i++)
```

```
        for(int j = 0; j < 2; j++)
```

```
            c[i][j] = a[i][j] + b[i][j];
```

```
    // vypis:
```

```
    for(int i = 0; i < 3; i++)
```

```
{
```

```
    for(int j = 0; j < 2; j++)
```

```
        cout << setw(3) << c[i][j] << " ";
```

```
    cout << endl;
```

```
}
```

```
}
```

Vícerozměrná pole v C++

Příklad 2 ... program vynásobí dvě matice a vypíše výsledek

```
float a[3][2] = {{1,0},{0,-1},{0,1}};
float b[2][4] = {{0,1,0,0},{0,-1,-1,0}};
float c[3][4]; // vysledek soucinu
...
```

- na procvičení

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

- výsledek:

```
0  1  0  0
0  1  1  0
0 -1 -1  0
```

Vícerozměrná statická pole v C++

Vícerozměrné statické pole jako parametr funkce

```
const int m = 5;  
void nejaka_funkce(double matice[][m], int n)
```

- parametrem funkce může být matice s libovolným počtem řádků, ale s fixním počtem sloupců ($m=5$)

Vícerozměrné pole jako parametr funkce

Příklad: funkce pro výpis prvků matice

```
#include <iostream>
using namespace std;
#include <iomanip> // setw

const int m = 3;

void vypis(const float a[][m], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m ; j++)
            cout << setw(3) << a[i][j] << " ";
        cout << endl;
    }
    cout << endl;
}
```

Příklady ... 9. cvičení

Příklad 1: Vícerozměrná pole v C++

- 1 Dokončete příklady na sčítání a na násobení matic

Příklad 2: Eratosthenovo síto

- Napište program, který nalezne všechna prvočísla menší než dané pevné $n=500$. Použijte následující algoritmus známý jako Eratosthenovo síto:
 - 1 připrav tabulku pro čísla od 2 do n (např. pole delky n s prvky typu `bool/int`).
 - 2 najezni nejmenší nevyškrtnuté číslo (x), jedná se o prvočíslo.
 - 3 vyškrtni všechny násobky tohoto čísla x větší (nebo rovny) než jeho čtverec (tj., x^2) (menší už byly vyškrtnuty).
 - 4 opakuj kroky 2 a 3 až do dosažení \sqrt{n} .
 - 5 nevyškrtnutá čísla v tabulce jsou prvočísla.

3. Další příklady na třídění pole

- Napište funkci **void setridBublinkove(double *a, int n)**, která setřídí pole **a** délky **n** v pořadí od nejmenšího prvku po největší pomocí algoritmu bublinkového třídění (bubble sort).

Dobrovolně:

- Napište funkci **int indexMinima(double *a, int od, int po)**, která vrátí index minima pole **a** mezi indexy **od** a **po**.
- Napište funkci **void setridVyberem(double *a, int n)**, která setřídí pole **a** délky **n** v pořadí od nejmenšího prvku po největší pomocí algoritmu třídění výběrem minima (selection sort).
- Napište funkci **void setridVkladanim(double *a, int n)**, která setřídí pole **a** délky **n** v pořadí od nejmenšího prvku po největší pomocí algoritmu třídění vkládáním (insertion sort).