Úvod do programování

Lekce 6

Pole

definice statického pole

`typ identifikátor[velikost]`

- velikost statického pole je známa už při překladu
- `velikost` je konstantní výraz, často symbolická konstanta (makro)
- první prvek pole má vždy index `0`. Například `int a[10]` definuje pole deseti celých čísel počínaje prvkem `a[0]` a konče prvkem `a[9]`.

příklad: program přečte ze vstupu řádek a vypíše ho na obrazovku obráceně

```
#include<iostream>
using namespace std;
#define MAXDELKA 100  // max delka radku je sto znaku
int main(void){
  char radek[MAXDELKA];
  int i, c, delka;
  i = 0;                // zaciname od prvku s indexem 0
  while((c = cin.get()) != '\n'){
    radek[i] = c;     // ukladame precteny znak do pole
    i++;
    if(i >= MAXDELKA) break;  // ochrana proti preteceni
  }
  delka = i;          // tolik znaku bylo precteno
  for(i = delka - 1; i >= 0; i--)   // prochazime pole obracene
    cout.put(radek[i]);
  cout << endl;
}
```

příklad: program spočítá frekvence číslic na řádku, ostatní znaky jsou ignorovány

```
#include <iostream>
using namespace std;
#define POCET ('9'-'0'+1)
int main(void){
  int cislice[POCET];
  int c, i;
  for(i = 0; i < POCET; i++)  // nulovani pole
     cislice[i]=0;
  while((c = cin.get()) != '\n'){
    if((c >= '0') && (c <= '9'))   // byla prectena cislice?
       cislice[c - '0']++;   // dalsi vyskyt cislice "c"
  }
  for(i = 0; i < POCET; i++)
    cout << i << " - " <<  cislice[i] << endl;
}
```

poznámka:

- hodnota makra musí být v závorkách
- přečtené číslo je použito jako index pole

inicializace pole

```
int a[4]={1,5,3,-5}; // definuje a inicializuje pole
```

pole jako parametr funkce

```
double maxim(double a[]) // hlavička funkce pracující s polem
maxim(a)   // volání funkce maxim()
```

příklad: funkce `soucet()` sečte prvky pole

```
#include<iostream>
using namespace std;
#define DELKA 5

double soucet(double x[]){
  int i;
  double suma = 0;
  for(i = 0; i < DELKA; i++)
    suma += x[i];
  return(suma);
}

int main(void){
  double a[DELKA] = {1,2,3,4,5};   // inicializace
  double b[DELKA] = {-13.6,1e-5,-9.999,3.3,1};
  cout << "soucet prvniho: " << soucet(a) << endl; // predani do
funkce
  cout << "soucet druheho: " << soucet(b) << endl;
}
```

poznámka:

- do funkce je předána adresa pole
- prvky pole mohou být tedy ve funkci změněny

příklad: program přečte ze souboru deset čísel, seřadí je, a zapíše do výstupního souboru

```
#include<iostream>
#include<fstream>
using namespace std;
#define POCET 10

void cteni(int a[]){
  ifstream fr;
  int i;
  fr.open("nesetrideny.txt");
  for(i = 0; i < POCET; i++)
    fr >> a[i];            // uklada prectena data
  fr.close();
}

void trideni(int a[]){
```

```
  int opakuj, i, pom;
  for(opakuj = 1; opakuj < POCET; opakuj++){
// porovnavani dvojic se musi opakovat alespon POCET-1 krat,
// aby pripadny nejmensi prvek na krajni prave pozici mohl
// doputovat na krajni levou pozici, kam patri
    for(i = 0; i < POCET - 1; i++)  // postupne porovna vsechny
dvojice
      if(a[i] > a[i+1]){ // vymeni levy a pravy prvek, pokud treba
        pom = a[i];      // ulozi levy do pomocne promenne
        a[i] = a[i+1];   // pravy nahradi levy
        a[i+1] = pom;    // ulozeny nahradi pravy
      }
  }
}

void zapis(int a[]){
  ofstream fw;
  int i;
  fw.open("setrideny.txt");
  for (i = 0; i < POCET; i++)
    fw << a[i] << " ";  // zapisuje data
  fw << endl;
  fw.close();
}

int main(void){
  int seznam[POCET];
  cteni(seznam);
  trideni(seznam);
  zapis(seznam);
}
```

poznámky:

- jednotlivé činnosti jsou naprogramovány jako samostatné funkce; zlepší se tak čitelnost programu
- funkce spolu komunikují přes parametry

příklad: program vypíše seznam prvočísel - Erathostenovo síto

```
#include<iostream>
#include<cmath>
using namespace std;
#define MAX 100
int main(void){
  int prvocisla[MAX+1], i, j;
  for(i = 2; i <= MAX; i++)   // inicializace seznamu
    prvocisla[i] = 1;
  for(i = 2; i <= sqrt(MAX); i++)  // alespon jeden delitel bude
<=sqrt(MAX)
    for(j = i*i; j <= MAX;j += i) // slozena cisla < i^2 jiz byla
```

```
vyskrtana drive
      prvocisla[j] = 0;
  for(i = 2; i <= MAX; i++)    // vypis prvocisel
    if(prvocisla[i]) cout << i << " ";
  cout << endl;
}
```

poznámky:

- čísla jsou reprezentována indexem pole
- hodnota `1` uložená v `prvocisla[i]` informuje, že číslo `i` je na seznamu

příklad: program vypočítá a zobrazí histogram z dat uložených v souboru

```
#include<iostream>
#include<fstream>
#include<iomanip>
using namespace std;
#define POCET 10000 // cisel na radku
#define MAX 100  // pocitame histogram do 99

// definice globalnich promennych
ifstream fr;
int histogram[MAX], data[POCET];
int maximum;   // bude obsahovat maximum z data[]

void cti_radek(void){   // precte jeden radek, nezavira soubor
  int i;
  for(i = 0; i < POCET; i++)
    fr >> data[i];
}

// pocita histogram a nalezne maximum
int pocitej_histogram(void){
  int i;
  for(i = 0; i < MAX; i++)
    histogram[i] = 0;       // nuluji histogram
  maximum = 0;
  for(i = 0; i < POCET; i++){
    histogram[data[i]]++; // dalsi vyskyt hodnoty data[i]
    if(data[i] > maximum)
      maximum = data[i];    // nove maximum
  }
}

void hvezdy(int n){   // kresli "n" hvezd
  int i;
  for(i = 0; i < n; i++)
    cout << '*';
}
```

```
void kresli_histogram(void){
   int i;
   for(i = 0;i <= maximum; i++){
     cout << setw(3) << i << "  ";
     hvezdy(histogram[i]/50); // hvezdicka = 50 vyskytu
     cout << endl;
   }
}

int main(void){
  int maximum;
  // nejprve nacteme data
  fr.open("random.txt");
  cti_radek(); cti_radek(); cti_radek(); // preskoci tri radky
  cti_radek(); // cte jeden radek ze souboru
  fr.close();
  // ted data zpracujeme
  pocitej_histogram();
  kresli_histogram();
}
```

poznámka:

- na rozdíl od předchozího příkladu zde funkce spolu komunikují přes globální proměnné

vícerozměrné pole

```
double x[2][3];          // deklarace 2D pole 2x3
x[0][0]=x[0][1]+x[1][0]; // práce s polem
double maxim(double x[][3]);  // pole jako parametr funkce
maxim(x);                // volání funkce
double x[][3]={{1,2,3},{4,5,6}}; // deklarace s inicializací
```

dynamicky alokované pole

```
double pole[DELKA]   // statické pole, velikost dána při překladu
double *pole         // dynamické pole - ukazatel na typ pole

pole = new typ [pocet_prvku];  // alokace paměti
```

- paměť pro uložení pole je nutno alokovat
- pokud se alokace paměti nezdaří (např. málo paměti), dojde k výjimce; alternativně `new typ (nothrow) [pocet_prvku]` vrací `NULL`; `nothrow` je definováno v `<new>`
- s prvky pole se pracuje stejně jako u statického pole: `pole[i]`

```
delete [] pole;  // uvolnění paměti
```

řetězce

- řetězec je pole znaků
- poslední znak řetězce je `'\0'`
- třída `string`

```
string radek;         // vytvoří řetězec
string radek("abcd"); // vytvoří a inicializuje řetězec
string radek("abcd",2); // nastaveni delky

getline(cin, radek); // načte celý řádek do řetězce
```

formátovaný zápis a čtení

- `<sstream>` definuje řetězcové buffery `istringstream` a `ostringstream`
- zápis a čtení je obdobné jako u `ifstream/ofstream`
- získání/nastavení obsahu bufferu je pomocí funkce `str()`

příklad práce s řetězci

```
#include<iostream>
#include<sstream>
using namespace std;
int main(void){
  string a("auto");
  cout << a << " " << endl;
  string b("auto",2);
  cout << b << " " << endl;
  cout << endl;

  ostringstream str_out;
  string retezec;
  str_out << 1.34 << ' ' << " kg"; // zapis do bufferu
  retezec = str_out.str();    // vysledny retezec
  cout << retezec << endl;

  istringstream str_in;
  double x;
  str_in.str("5.37 metru");  // nastavi retezec
  str_in >> x >> retezec;   //  cteni z bufferu
  cout << x+1 << " " << retezec << endl;
  cout << endl;

  a = "leva";
  b = " ruka";
  string c;
  c = a ;                 // kopirovani
  cout << c << endl;
  c = a + b;              // slouceni retezcu
```

```
  cout << c << endl;
  cout << (a == b) << endl;  // porovnani retezcu
  cout << endl;

  string text="tyden ma sedm dni";
  cout << text.length() << endl;      // delka retezce
  cout << text.find("sedm") << endl;  // hledani retezce
  cout << text.substr(2,11) << endl;  // vraci cast retezce
  cout << text.replace(9,8,"pet dni?") << endl; // nahrazeni
}
```

příklad: vytvoří deset souborů

```
#include<iostream>
#include<sstream>
#include<fstream>
using namespace std;
int main(void){
  int i;
  string koren("file");  // koren jmena
  string filename;   // zde bude vysledne jmeno
  ostringstream strout; // retezcovy buffer
  ofstream fw;          // souborovy buffer
  for(i = 0; i <= 9; i++){
    strout.str("");    // inicializace bufferu
    strout <<  koren << i << ".txt"; // zapis do bufferu
    filename = strout.str();  // vrati vysledne jmeno
    cout << filename << endl; // kontrolni vypis
    fw.open(filename);
    fw << "soubor cislo " << i << endl; // zapis do souboru
    fw.close();
  }
}
```

poznámka:

- existují i funkce pro převody řetězců na číselné typy a naopak, které lze využít k podobnému účelu
- automatické generování jmen je užitečné pro načtení dat z velkého množství podobně pojmenovaných souborů např. obsahujících experimentální data