Úvod do programování

Lekce 3

Řízení běhu programu - pokračování

příklad: program vypisuje hodnotu $\int_0^{\pi/2} \sin(x)dx$ pro různé délky integračního kroku

```
#include<iostream>
#include<iomanip>
#include<cmath>  // budeme pouzivat funkci sin()
using namespace std;
#define RADKU 6    // zde zadam pocet radku tabulky
#define KROK_INI 0.1
main(){
  double x,krok,integral;
  int radek;
  cout << "  krok        integral\n";  // zahlavi tabulky
  cout << "======================\n";
  krok = KROK_INI;   // nastavi pocatecni delku kroku
  for(radek = 1;radek < RADKU + 1; radek++){
    integral = 0;
    for(x = 0;x < M_PI/2.;x += krok)
      integral += sin(x);  // vytvari sumu
    integral *= krok;    // nasobeni dx
    cout << fixed << setprecision(6) << krok;
    cout << setw(15) << setprecision(8) << integral << endl;
    krok *= 0.1;    // zmenseni kroku pro dalsi radek tabulky
  }
}
```

poznámky:

- v programu často používané konstanty je dobré definovat jako makra na začátku programu nebo v hlavičkovém souboru; změna všech výskytů dané konstanty se pak snadno provede změnou hodnoty makra
- náhradu maker provádí preprocesor před vlastním překladem zdrojového kódu
- za definicí makra není středník: pokud bychom psali `#define pi 3.14;` vytvořil by preprocesor z `x=pi+x` chybný výraz `x=3.14;+x`
- proměnná `krok` by mohla přímo sloužit jako řídící proměnná cyklu; pokud ale požadujeme určitý přesně daný počet opakování, je lépe pro řízení cyklu definovat novou proměnnou celočíselného typu, jako ve výše uvedeném příkladu
- konstanta π je definovana makrem `M_PI` v `cmath`

příklad: program vypíše Fibonacciho posloupnost, vypočítá zlatý řez z posledních dvou členů

```
#include<iostream>
#include<cmath>  // budeme pouzivat funkci sqrt()
using namespace std;
int main(void){
  int clenu,fi1,fi2,i,pom;
  double zlatyrez;
  cout << "kolik clenu Fibonacciho rady: ";  // vyzva
```

```
  cin >> clenu;
  fi1 = fi2 = 1;                    //prvni dva cleny
  cout << fi1 << endl << fi2 << endl;
  for(i = 3; i <= clenu; i++){
    pom = fi2;                    // uschova druhy clen
    fi2 += fi1;                   // scita cleny
    fi1 = pom;                    // ulozeni fi2 do fi1
    zlatyrez = fi2/ (double) fi1; // nechceme celociselne deleni
    cout << fi2 << "  " << zlatyrez << endl;
  }
  cout << "spravne: " << (sqrt(5)+1)/2. << endl; // presna hodnota
}
```

přepínač

```
switch (výraz) {
  case  hodnota_a : příkaz_a; break;
  case  hodnota_b : příkaz_b; break;
  ...
  default : příkaz_def; break;
}
```

- mnohonásobné větvení podle hodnoty výrazu (typu `int` nebo `char`)
- pokud není větev ukončena příkazem `break` program neopouští příkaz `switch` a zpracovává následující větev
- pokud nevyhovuje žádná hodnota uvedená za `case`, vykoná se větev `default`
- obvykle použijeme příkaz switch pro volání různých funkcí v závislosti na vstupu - např. zpracování vstupu textovým editorem

příklad: program podle zadaného znaku vypočte x^2 , x^3 , nebo $\sqrt{x}$

```
#include<iostream>
#include<cmath>
using namespace std;
int main(void){
  double x,vysledek;
  char druh;
  int chyba=1;
  cout << "zadej x: ";
  cin >> x;
  while(chyba){      // opakuje dokud je chyba v zadani
    cout << "zvol druh vypoctu\n";
    cout << "a: druha mocnina\n";
    cout << "b: treti mocnina\n";
    cout << "c: druha odmocnina\n";
    cin >> druh;
    switch(druh){
      case 'a': cout << "x^2=" << x*x << endl; chyba=0; break;
      case 'b': cout << "x^3=" << pow(x,3) << endl; chyba=0;
break;
```

```
      case 'c': cout << "sqrt(x)=" << sqrt(x) << endl; chyba=0;
break;
      default: cout << endl << "chyba: spatny vstup!\n\n"; break;
    }
  }
}
```

poznámky:

- větve `case` ruší příznak chyby, větev `default` nikoliv: to vede k opakované volbě druhu výpočtu v případě výskytu chyby
- pokud je příkaz switch vnořen do cyklu (`for`, `while`, ...), ukončuje `break` ve větvi příkaz `switch` a nikoli cyklus
- více příkazů ve větvích není nutno uzavírat do složených závorek
- větev `default` nemusí být uvedena

Práce se soubory

hlavičkový soubor `<fstream>` implementuje vstupní a výstupní operace se soubory

```
ifstream infile;  // vstupní proud
ofstream outfile; // výstupní proud
```

otevření souboru

přiřazení souboru k datovému proudu - volání metody `open()`
`open(jméno, režim)`

základní režimy

`ios::in` – textový soubor pro čtení (defaultní pro ifstream)
`ios::out` – textový soubor pro zápis (defaultní pro ofstream)
`ios::app` – textový soubor pro připsání na konec

příklady pro výše definované `outfile`, `infile`

```
outfile.open("vysledky.dat");
```

- otevře soubor `"vysledky.dat"` pro zápis
- pokud soubor `"vysledky.dat"` již existuje je jeho obsah zničen!

```
infile.open("vysledky.dat");
```

- otevře existující soubor `"vysledky.dat"` pro čtení od začátku souboru
- úspěšné otevření lze testovat funkcí `infile.is_open()`

uzavření souboru

`close()`

- pokud byl soubor měněn zapíší se fyzicky všechny změny do souboru
- po skončení práce se souborem je vhodné soubor uzavřít

formátovaný vstup a výstup

obdobně jako u standardního vstupu a výstupu

např.

```
outfile << x  // zápis proměnné do souboru
infile >> x   // čtení proměnné ze souboru
```

příklad: program přečte délky odvěsen ze souboru a vypočtenou přeponu zapíše do souboru

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
int main(void){
  ifstream fr;      // objekty pro pristup k souboru
  ofstream fw;
  double a,b,c;
  fw.open("prepona.txt");    // oteviram soubor pro zapis
  fr.open("odvesny.txt");    // oteviram soubor pro cteni
  if(!fr.is_open()){       // kontrola uspesneho otevreni
    cout << "chyba otevreni souboru! \n";
    exit(1);               // ukonceni programu
  }
  fr >> a >> b;          // ctu delky odvesen
  cout << "precteno a=" << a << ", b=" << b << endl; // kontrolni
vypis
  c=sqrt(a*a+b*b);
  fw << "delka odvesny: " << c << endl;   // zapis vysledku
  fr.close(); fw.close();
}
```

poznámka:

- funkce `is_open()` je použita pro kontrolu, zda byl soubor úspěšně otevřen
- při chybě, např. pokud soubor neexistuje, je program ukončen