Základy programování v C++
Zadání 1.seminární úlohy (ZS 2019/20)

Zuzana Petříčková

29. října 2019

Rovnice

- Vytvořte ,,povídací" uživatelsky přátelský program pro vypisování a řešení rovnic ve tvaru $ax^2 + bx + c = 0$ pro libovolné hodnoty $a, b, c \in R$.

Program bude mít následující strukturu:

1. Načti z konzole zadání n rovnic (pro pevně dané n, např. n=10).
2. Vypisuj rovnice podle přání uživatele:
   V cyklu:
   1. Načti z konzole pořadové číslo rovnice (i).
   2. Vypiš na konzoli zadání i-té rovnice.
   3. Pokud uživatel chce, tak vypiš na konzoli i řešení této rovnice.

   (cyklus ukonči na přání uživatele)

Rovnice ... ukázka komunikace s programem I.

Zadej 10 rovnic ve tvaru ax^2+bx+c=0
Rovnice c.1:
a= 0.5
b= 0
c= -2
Rovnice c.2:
a= 1
b= 2
c= 1
Rovnice c.3:
a= 1
b= 0
c= 4
Rovnice c.4:
a= 0
b= 2
c= -1
Rovnice c.5:
a= 0
b= 0
c= -2
... (atd. az do cisla 10)

Rovnice ... ukázka komunikace s programem II.

Zadej cislo rovnice, kterou chces vypsat:
i = 1

Rovnice c.1: 0.5x^2+0x-2=0

Chces vypsat reseni rovnice? (a/n): a

Jedna se o kvadratickou rovnici. Ma dva realne koreny 2 a -2

Chces vypsat dalsi rovnici? (a/n): a

Rovnice ... ukázka komunikace s programem III.

Zadej cislo rovnice, kterou chces vypsat:
i = 3

Rovnice c.3: 1x^2+0x+4=0

Chces vypsat reseni rovnice? (a/n): a

Jedna se o kvadratickou rovnici. Ma dva komplexni koreny 0 +- 2i

Chces vypsat dalsi rovnici? (a/n): a

Zadej cislo rovnice, kterou chces vypsat:
i = 2

Rovnice c.2: 1x^2+2x+1=0

Chces vypsat reseni rovnice? (a/n): a

Jedna se o kvadratickou rovnici. Ma jeden dvojnasobny koren -1

Chces vypsat dalsi rovnici? (a/n): a

Rovnice ... ukázka komunikace s programem IV.

Zadej cislo rovnice, kterou chces vypsat:
i = 4

Rovnice c.4: 0x^2+2x-1=0

Chces vypsat reseni rovnice? (a/n): a

Jedna se o linearni rovnici. Resenim je 0.5

Chces vypsat dalsi rovnici? (a/n): a

Zadej cislo rovnice, kterou chces vypsat:
i = 5

Rovnice c.5: 0x^2+0x-2=0

Chces vypsat reseni rovnice? (a/n): a

Rovnice nema reseni

Chces vypsat dalsi rovnici? (a/n): a

Rovnice ... ukázka komunikace s programem V.

Zadej cislo rovnice, kterou chces vypsat:
i = 6

Rovnice c.6: 0x^2+0x+0=0

Chces vypsat reseni rovnice? (a/n): a

Resenim rovnice jsou vsechna realna cisla

Chces vypsat dalsi rovnici? (a/n):n

Rovnice

Další požadavky:

- Program bude kontrolovat vstup od uživatele (např. že zadal čísla) a adekvátně reagovat (např. "prosim, zadej znovu ...")
- Program bude logicky rozdělen na tři části (funkce):
  - část, která načte všechna zadání
    - zadání (popř. ,,Reseni") budou uložena ve statickém poli
  - část, která pro každou rovnici spočítá výsledek ,,v číselné podobě" (ale nic nebude načítat z konzole ani vypisovat)
    - Definujte vhodnou strukturu **Reseni** pro reprezentaci správného řešení rovnice
    - např. pomocna funkce ve tvaru **Reseni vyresRovnici(double a, double b, double c)**
    - Dobře promyslete, jak může vypadat řešení rovnice (je celkem 6 možností, které mohou nastat, viz. poznámky ze cvičení a příklady na těchto slidech).
  - část, která bude vypisovat (předpočítané) výsledky

Jak by tedy mohla např. vypadat funkce main:

```
/*
void nactiZadani(double zadani[][3], int n);
void vyres(const double zadani[][3], Reseni *reseni, int n);
void vypisReseni(const double zadani[][3], const Reseni *rese
*/
int main()
{
    const int n = 10;
    double zadani[n][3]; // zadani v poli
    Reseni reseni[n];    // reseni jako struktura

    nactiZadani(zadani, n);
    vyres(zadani, reseni, n);
    vypisReseni(zadani, reseni, n);

    return 0;
}
```

Jak by tedy mohla vypadat funkce main (alternativně):

```
/*
void nactiZadani(Zadani *zadani, int n);
void vyres(const Zadani *zadani, Reseni *reseni, int n);
void vypisReseni(const Zadani *zadani, const Reseni *reseni,
*/
int main()
{
    const int n = 10;
    Zadani zadani[n]; // zadani jako struktura
    Reseni reseni[n]; // reseni jako struktura

    nactiZadani(zadani, n);
    vyres(zadani, reseni, n);
    vypisReseni(zadani, reseni, n);

    return 0;
}
```

Jak by tedy mohla vypadat funkce main (alternativně):

```
/*
void nactiZadani(Reseni *reseni, int n);
void vyres(Reseni *reseni, int n);
void vypisReseni(const Reseni *reseni, int n);
*/
int main()
{
    const int n = 10;
    Reseni reseni[n]; // zadani a reseni v jedne strukture

    nactiZadani(reseni, n);
    vyres(reseni, n);
    vypisReseni(reseni, n);

    return 0;
}
```

Jak by tedy mohla vypadat struktura Reseni:

```
enum Typ {dva, jeden, komplexni, nema, nekonecne, linearni};
struct Reseni
{
    double x1;
    double x2;
    Typ typ;
    //  double a, b, c; // volitelne vcetne zadani
};

// alternativne:
struct Reseni
{
    double x1;
    double x2;
    int typ; // ciselnik
    //  double a, b, c; // volitelne vcetne zadani
};
```

Rovnice

Dovednosti, které byste měli na této úloze ukázat (a tedy by bylo fajn je do řešení vhodně zakomponovat):

- Kontrola vstupu uživatele.
- Struktura
- Statické pole.
- Správné rozdělení programu do funkcí a do několika souborů (vč. hlavičkového souboru), umět oddělit vstup z konzole a výpis na konzoli od logiky aplikace
  - Funkce, které mají na starosti logiku aplikace a nic nečtou z konzole ani nevypisují (v mém příkladu **vyres, vyresRovnici**), budou v druhém zdrojovém souboru.
  - Definice datovych typu budou v hlavičkovém souboru.

První seminární úloha - obecně

- Používejte jen ty konstrukty jazyka C/C++, které jsme doposud probírali (nepoužívejte knihovnu STL apod.)
- Programy by měly být přehledné (používejte odsazení, rozdělte program smysluplně do kratších funkcí)
- Program by měl být přeložitelný v MS Visual Studiu

První seminární úloha - obecně

- Řešení úloh pošlete emailem (jen soubory .cpp a .h v archivu .zip). V konkrétních případech bude následovat osobní konzultace programu (např. při závažnějších nebo obtížně popsatelných nedostatcích nebo když budu mít podezření, že jste program nevypracovali sami)
- V případě, že program nebude v pořádku, popíšu vám seznam nedostatků a budete mít další čas (cca. týden) na poslání opravy
- V případě, že dva studenti pošlou nápadně identická řešení, uznávám jen to chronologicky první (druhý student dostane náhradní (obtížnější) úlohu