Základy programování (IZP)

Páté počítačové cvičení

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 66 Brno - Královo Pole
Petr Veigend, iveigend@fit.vut.cz, Alena Tesařová, atesarova@fit.vut.cz

Verze 20191022_15:45

5. týden

Organizace

- Projekt
  - Nezapomeňte se přihlásit!
  - Jak překládat a kde?
    - **Windows:**
      - **Codeblock – lokálně**
      - **VS code** -> F1 -> Remote ssh: Connect to host -> Add new SSH host -> ssh xlogin00@merlin.fit.vutbr.cz
      - Putty + WinSCP a otevřít projekt ve vašem editoru
  - Pro načítání použít getchar() nebo getc(stdin)
- Provozní řád CVT test
- Nemoc na cvičení
- Dotazník
- Otázky?

1. projekt – Práce s textem

- Překlad programu

```
gcc -std=c99 -Wall -Wextra -Werror
  proj1.c -o proj1
```

- **`-Werror`** → všechna varování interpretuje jako chyby!
- **Nastavení v CodeBlocks**
  - **Settings -> Compiler**

Debugger

- Pokud není debugger nastavený, je potřeba nastavit cestu:
  - Settings -> Debugger -> Executable path: ***/gdb.exe***
- K debuggování bude potřeba přidat okno s používanými proměnnými
  - *Debug -> Debugging windows -> **Watches***

Náplň cvičení

- seznámit se s datovým typem **struktura**
- seznámit se s **vícerozměrnými** poli
- ovládat implementaci funkcí s parametry **předávanými hodnotou**
- porozumět práci se **standardním vstupem a výstupem**
- ovládat cykly

JAK BY MĚL VYPADAT ZDROJOVÝ KÓD?

Jak by měl vypadat zdrojový kód?

- **Dekomponujte problém na podproblémy** → tvorba vlastních **funkcí**
- Vhodně **pojmenovávejte** funkce a proměnné
- Dbejte na **přehlednost kódu** – **odsazení**!
  - Tělo funkce
  - Příkazy v if, else if, else
  - Těla cyklů
- Pište **komentáře**
  - Vhodné je psát komentáře ke každé definici funkce, a také k určitým částem kódu

Jak by měl vypadat zdrojový kód?

```
/*
  Autor:
  Nazev:
*/
// potrebne knihovny
// definice funkci (dekompozice problemu)

// Hlavni program
int main(int argc, char* argv[])
{
   // zpracovani argumentu prikazove radky
   // provedeni danych akci
   return 0;}
```

Jak by měl vypadat zdrojový kód?

```
/**
 * Funkce scita dve cela cisla.
 * @param a Prvni cislo
 * @param b Druhe cislo
 * @return Vraci soucet dvou cisel.
 */
int soucet(int a, int b)
{
  return a + b;
}
```

Implicitní streamy

- V jazyce C můžete používat **3 implicitní streamy**, které reprezentují vstup a výstup
  - stdin standardní vstup, třeba z klávesnice
  - stdout standardní výstup, třeba na monitor
  - stderr standardní chybový výstup
- **Chybová hlášení programů** je vhodné (a v projektu nutné) vypisovat na standardní chybový výstup pomocí funkce **`fprintf()`**

```
#include <stdio.h>
fprintf(stream, format_retezec, dalsi_param);
fprintf(stderr, "Chybove hlaseni\n");
fprintf(stderr, "Malo parametru: %d", ecode);
```

STRUKTURY, VLASTNÍ DATOVÉ TYPY

Pole x Struktura

- **Pole**

- **Struktura**

Pole x Struktura

- **Pole**
  - **Homogenní**
  - **Musí** obsahovat položky **stejného** datového typu

- **Struktura**
  - **Heterogenní**
  - **Může** obsahovat položky **různého** datového typu

Struktura – příklad

- Mějme např. strukturu, která reprezentuje čas. Lze ji použít dvěma způsoby

```
struct date_t {
   int year;
   int month;
   int day;
};
struct date_t start;
// Alternativa
typedef struct {
   int year;
   int month;
   int day;
} Date; // Date je název datového typu
Date start; // příklad použití
```

Příklad použití struktury

```
typedef struct {
  int year;
  int month;
  int day;
} Date;
int main() {
  Date start;
  start.year = 2020; // přístup k položce .
  start.month = 07; // podobně day
  printf("year:  %d\n", start.year);
  printf("month: %s\n", start.month);
}
```

Co může být například struktura?

- Uveďte další příklady možných struktur
  - čas (atributy: minuty, hodiny, sekundy)
  - třída (atributy: studenti, třídní učitel, místnosti, rozvrh)
  - ….

Jak je to v paměti?

- Kolik bajtů zabere následující struktura?

  Předpokládejme, že sizeof(int) = 2, sizeof(char) = 1

```
typedef struct {
  char c;
  int i, j, k;
  char d;
} Priklad;
Priklad pokus;
```

Jak je to v paměti?

- Kolik bajtů zabere následující struktura?

  Předpokládejme, že sizeof(int) = 2, sizeof(char) = 1

```
typedef struct {
  char c;
  int i, j, k;
  char d;
} Priklad;
Priklad pokus;
```

- Položky se obsazují v paměti shora dolů
- Položky jsou většinou zarovnány na sudé adresy (memory alignment)
- Správná odpověď: 8, 9 nebo 10 bajtů

Příklad

- Implementujte funkci, která porovná dvě proměnné typu Date a zjistí, které datum je dřívější

```
Date compare2Dates(Date first, Date second)
{
  // práce s jednotlivými daty

  // funkce vrátí dřívější datum
  // tj. return first nebo return second
}
```

Příklad

- Implementujte funkci pro detekci validního data
  - Led, Bře, Kvě, Červ, Srp, Říj, Pro – 31 dní
  - Únor **pro jednoduchost** řekněme, že má 28 dní
  - Rok od **1900 do 2021** bude validní

```
int is_day_valid(Date day)
{

  // funkce vrátí 1 jestli je validní, jinak 0
  // tj. return 1 if day is valid, 0 otherwise
}
```

Další příklad

- Deklarujte strukturu pro časovou značku (obsahuje **datum**, **čas** a **teplotu**)

```
struct time_t
{
    …
    …
    …
};

struct measurement_t
{
    …
    …
    …
};
```

Další příklad

- Deklarujte strukturu pro časovou značku (obsahuje **datum**, **čas** a **teplotu**)

```
struct time_t
{
   int hour;
   int min;
   int sec;
};

struct measurement_t
{
    Date date;
    struct time_t time;
    float temperature;
};
```

Příklad

- Udělejte funkci, která zkontroluje, jestli je čas validní
  - Parametr ale bude `struct measurement_t`

```
int is_time_valid(struct measurement_t
  mesurement)
{
  // funkce vrátí 1 jestli je validní, jinak 0
  // tj. return 1 if time is valid, 0 otherwise
}
```

Příklad

- Načtěte 5 měření do pole (ve funkci main) a spočítejte: průměrnou teplotu.

```
int main(){
   // structure definition
   struct measurement_t measurements[5];

   // example of scanf
   scanf("%d", & measurements[0].temperature);

}
```

DVOUROZMĚRNÉ POLE

Dvourozměrné pole

```
int pole2d[4][5]; // pole 4 řádků s 5 sloupci
pole2d[0][0] = 0;
pole2d[3][4] = -1;
```

- Inicializujte dvourozměrné pole (samé 0).
- Vytvořte funkci pro **vyhledání prvku** v dvourozměrném poli. Vrátit hodnotu prvku.

	0	1	2	3	4
0	0	?	?	?	?
1	?	?	?	?	?
2	?	?	?	?	?
3	?	?	?	?	-1

Dvourozměrné pole

```
int pole2d[4][5]; // pole 4 řádků s 5 sloupci
pole2d[0][0] = 0;
pole2d[3][4] = -1;
```

- Inicializujte dvourozměrné pole (samé 0).
- Vytvořte funkci pro **vyhledání prvku** v dvourozměrném poli. Vrátit hodnotu prvku. Jak bychom museli funkci předělat, aby vracela souřadnice, kde se prvek nachází?

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

PRÁCE SE SOUBORY

Práce se soubory

```
// soubor pro čtení
FILE *soubor;
soubor = fopen("jmeno_souboru.txt", "r");
if (soubor != NULL) {
  char s[100];
  fscanf(soubor, "%99s", s);
  printf("Prvni slovo souboru je '%s'\n", s);
  fclose(soubor);
}

// soubor pro zápis
soubor = fopen("log.txt", "w");
if (soubor != NULL) {
  fprintf(soubor, "Hello, world!\n");
  fclose(soubor);
}
```

Vždy zkontrolujte, jestli se podařilo soubor otevřít !!

```
// soubor pro čtení
FILE *soubor;
soubor = fopen("jmeno_souboru.txt", "r");
if (soubor != NULL) {
   // podařilo!!
}
```

Příklad

- Implementujte funkci pro uložení **dvourozměrné** matice do souboru. Předpokládejte fixní velikost matice.

```
int num_cols = 4;
int num_rows = 3;
// počet sloupců musí být uveden
int arr2d[][num_cols] = {
   {1,2,5,7},
   {4,6,0,4},
   {1,10,2,3}
};
```

Příklad

- Navrhněte strukturu **studenta** (musí mít jméno a věk)
- Načtěte třídu 5 studentů a spočítejte průměrný věk ve třídě.
- Př: Adam (14 let), Jana (10 let), Venda (13 let), Pavel (12 let), Jindra (10 let)
  - Průměrný věk je: 59/5 = 11.8

- **Dále**: načtěte studenty ze souboru (na jednom řádku bude vždy **jméno** a **věk**)

Děkuji Vám za pozornost!