

Základy programování (IZP)

Páté počítačové cvičení

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



- **Odevzdání prvního projektu **dnes ve 23:59:59 !!!****
 - **Před odevzdáním vyzkoušejte na merlinovi**
- **Jak by měl vypadat zdrojový kód? Ukázka**
- **Vlastní datové typy**
- **2D pole**

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

```
/*  
  Autor:  
  Navez:  
*/  
  
// potrebne knihovny  
// definice funkci (dekompozice problemu)  
  
// Hlavni program  
int main(int argc, char** argv)  
{  
    // zpracovani argumentu prikazove radky  
    // provedeni danyh akci  
    return 0;}
```

```
/**  
 * 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;  
}
```

- 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**
 - **Homogenní**
 - **Musí** obsahovat položky **stejného** datového typu
- **Struktura**
 - **Heterogenní**
 - **Může** obsahovat položky **různého** datového typu

- 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
```

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

- Implementujte **funkci** která inicializuje proměnnou typu Date a vrátí její obsah

```
struct date_t {  
    int year;  
    int month;  
    int day;  
};  
// nebo  
typedef struct {  
    int year;  
    int month;  
    int day;  
}Date;
```

```
Date createDate() // struct date_t  
{  
    // vytvoření proměnné typu Date  
    struct date_t datum;  
    // nebo Date datum;  
    // vložení položek do struktury  
    (scanf)  
  
    // vrácení vytvořeného data do  
    volající funkce  
}
```

- Implementujte funkci, která vypíše obsah proměnné typu Date ve formátu <den>.<měsíc>.<rok>, např. 25.10.2025

```
// návratový typ void, funkce nic nevrací  
void printDate(Date d) // struct date_t  
{  
    // vypsání složek proměnné d  
}
```

- Implementujte **funkci**, která ověří, že zadané datum je validní
 - Roky 1900 až 2100
 - Kontrola, zda nemá měsíc moc nebo málo dnů
 - Jen některé měsíce mají 31 dnů, některé 28, navíc lze řešit i přestupné roky (můžete za DÚ přidat)

```
int isDateValid(Date d)
{
    // práce s datem

    // funkce vrátí 0 pokud je datum validní,
    // jinak -1
}
```

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

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

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

- Deklarujte složený datový typ pro měření

```
struct time_t {  
    int hours;  
    int minutes;  
    int seconds; }  
  
struct measurement_t {  
    struct date_t date; // z předchozího příkladu  
    struct time_t time;  
    float temperature; }
```

- Vytvořte funkci `struct measurement_t load_measurement()`, která načte ze standardního vstupu jednotlivé položky měření a vytvořené měření vrátí

- Deklarujte složený datový typ pro měření

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

- Napište funkci, která zkontroluje, zda jsou data v měření validní:
 - Roky 1900 až 2100
 - Kontrola, zda nemá měsíc moc nebo málo dnů
 - Jen některé měsíce mají 31 dnů, některé 28, navíc přestupné roky (kdo chce)
 - Podobná kontrola pro počet hodin

- Deklarujte složený datový typ pro měření

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

- Načtěte 5 měření do pole a vypočítejte
 - Průměrnou teplotu
 - Maximální teplotu
 - Které odpoledne bylo nejteplejší
 - Apod.

2D POLE

- Matice (např. 5x5)

```
int matice[ROWS][COLS] = {  
    {1,2,3,4,5},  
    {6,7,8,9,10},  
    {11,12,13,14,15},  
    {16,17,18,19,20},  
    {21,22,23,24,25}  
};  
  
// indexace prvků matice[i][j]  
printf("%d ",matice[4][0]); // 21
```

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

- Ve funkci **main** napište
 - Inicializaci pole (všechny hodnoty budou 0)
 - Inicializaci pole (čtvercová jednotková matice)
- Napište funkci pro
 - Výpis matice
 - Vyhledání hodnoty v poli (funkce vrátí souřadnici)
- Poznámka: **#define**

- Napište program, který vypíše
 - Všechny prvky matice,
 - Prvky matice, které se nachází na hlavní diagonále,
 - Prvky matice, které se nachází pod hlavní diagonálou

PRÁCE SE SOUBORY

- Všechny vstupy a výstupy v C jsou **soubory**
 - **stdin**, **stdout**, **stderr**
- Datový typ, který reprezentuje soubor: **FILE***
- Otevření souboru se provádí pomocí funkce **fopen**
 - Příklad použití:
`FILE* soubor = fopen("test.txt", "r");`
otevíráme soubor `test.txt` pro čtení - `r`, zápis - `w`
- Před zahájením práce je třeba ověřit, že se podařilo soubor otevřít: `if (soubor != NULL) { //OK }`
- Po ukončení práce je potřeba soubor zavřít pomocí funkce **fclose**
 - `fclose(soubor);`

```
FILE* soubor = fopen("test.txt", "r");  
if (soubor != NULL) { // kontrola  
    // práce se souborem }  
fclose(soubor); // jsme hotovi
```

- Do souboru můžeme zapisovat podobně jako do terminálu (také **soubor**) pomocí funkce **fprintf**
 - **fprintf(soubor, "ABC %d\n", 1);**
Zapisujeme do otevřeného souboru **soubor** řetězec **ABC 1** a vkládáme **nový řádek**
- Názvy funkcí, které pracují se soubory začínají na **f**
 - **fprintf, fscanf, ...**

- Vyzkoušejte si čtení a zápis do souborů
- Zápis "w"
 - Do souboru zapište libovolné slovo pomocí funkce `fprintf(soubor, %s, "retezec")`
- Čtení "r"
 - Pomocí funkce `fscanf` zapsané slovo v souboru opět přečtěte, slovo může mít max. 99 znaků (`%99s`)
 - Načtený řetězec uložte do pole `char buff[100]` ;
 - Vypište načtené slovo

```
FILE* soubor = fopen("test.txt", "r"); // w
if (soubor != NULL) { // kontrola
// práce se souborem }
fclose(soubor); // jsme hotovi
```

- Implementujte funkci pro uložení 2D matice do souboru
- Uložení "w"
 - `fprintf(soubor, %d, "cislo")`
- Načtení: "r"
 - Pomocí funkce `fscanf` opět uloženou matici načtěte.

```
FILE* soubor = fopen("test.txt", "r"); // w
if (soubor != NULL) { // kontrola
// práce se souborem }
fclose(soubor); // jsme hotovi
```

- Napište program, který
 - Spočítá slova v zadaném souboru
 - Spočítá počet výskytů zadaného znaku v daném souboru

DALŠÍ ÚKOLY

Napište funkci, která vypočítá **n-tý prvek Fibonacciho posloupnosti**.

Fibonacciho posloupnost je definována:

$$f(0) = 0$$

$$f(1) = 1$$

$$f(n) = f(n-1) + f(n-2)$$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, atd.

Napište program, který načte ze souboru 2D pole pixelů.

- Každý pixel je specifikovaný R,G a B barevnou složkou
- Navrhněte vhodnou datovou strukturu pro uložení jednotlivých složek RGB
- Implementujte funkce např.:
 - Nalezení všech pixelů, které jsou sytě červené
 - Transformace barevného obrázku na černobílý
 - Uložení obrázku zpět do souboru
 - Atd.

- Napište funkci, která ověří, zda je zadané číslo rodné číslo (tzn. zda má správný počet číslic a zda je jeho číselná reprezentace dělitelná 11).
- Napište funkci, která otočí řetězec.

JEDNODUCHÉ CYKLY

- Se **známým počtem** průchodů – cyklus **for**
 - Vzestupně - interval $[0;n)$
 - Vzestupně - interval $[a;b]$
 - Sestupně - interval $[a;b)$
 - Vzestupně ob jedno - interval $[a;b]$
 - Po jednotlivých znacích řetězce
- **Zkuste si stejné cykly pomocí cyklu while**

- Pozor na = a ==

```
for (int i=0; i=5; i++) { ... }
```

- Pozor na pořadí jednotlivých částí cyklu `for`

```
for (int i=0; i++; i<5) { ... }
```

```
for (int i=5; i--; i>0) { ... }
```

- Se **známým počtem** průchodů – cyklus **for**
 - Vzestupně - interval $[0;n)$
 - Vzestupně - interval $[a;b]$
 - Sestupně - interval $[a;b)$
 - Vzestupně ob jedno - interval $[a;b]$
 - Po jednotlivých znacích řetězce
- **Zkuste si stejné cykly pomocí cyklu while**

- Napište program, který bude procházet 2D pole
 - Po diagonále shora
 - Po diagonále zdola
 - Po sloupcích
 - Po řádcích
 - Pod hlavní diagonálou
 - Nad hlavní diagonálou
 - A další... (podobné příklady se objevují pravidelně na zkoušce !!!)

```
for (inicializace; test; inkrementace)  
{ }
```



```
inicializace;  
while (test)  
{ inkrementace; }
```

KARTOTÉKA

- Napište program, který načítá osobní údaje pacientů a ukládá je do souborů.
- Soubory jsou pojmenované ve formátu PACRR.txt (RR představuje rodné číslo pacienta)
 - Název souboru lze vytvořit např. pomocí funkce `sprintf`.

```
Zadejte rodne cislo: 1234569876
Zadejte jmeno pacienta (ukoncene teckou): Zaphod Beeblebrox .
Zadejte bydliste (ukoncene teckou): Planeta Betelgeuse .
Zadejte rodne cislo: 0123459999
Zadejte jmeno pacienta (ukoncene teckou): Fort Prefect .
Zadejte bydliste (ukoncene teckou): Somewhere in the galaxy .
Zadejte rodne cislo: .
```

Děkuji Vám za pozornost!