

# Základy programovania (IZP)

## Dátové štruktúry (5. cvičenie)

Ing. Pavol Dubovec

Vysoké Učení Technické v Brně, Fakulta informačních technologií  
Božetěchova 1/2. 612 66 Brno- Královo Pole  
[pdubovec@fit.vutbr.cz](mailto:pdubovec@fit.vutbr.cz)



Na čo slúžia štruktúry v jazyku C?

- Dátové štruktúry sú potrebné na organizáciu a spravovanie dát v programe.
- Umožňujú efektívny prístup a manipuláciu s dátami, čo zlepšuje výkon a flexibilitu programu.
- Pomáhajú riešiť komplexné problémy a implementovať rôzne algoritmy.
- **Definícia štruktúry** – určuje aké členy bude štruktúra obsahovať a aké dátové typy.
- **Deklarácia štruktúry** – vytvorenie premennej, ktorá bude používať túto štruktúru
  - Po definovaní štruktúry môžeme vytvoriť jej inštanciu.
- Prístup k prvkom štruktúry vykonávame pomocou operátoru ( . )

```
// definícia
struct Person {
    char name[50]; // Meno osoby, reprezentované ako reťazec znakov
    int age; // Vek osoby, reprezentovaný ako celé číslo
    char address[100]; // Adresa osoby, reprezentovaná ako reťazec znakov
};
struct Person person1; // Deklarácia premennej person1 typu struct Person
// Nastavenie veku pre fera
person1.age = 30;
```

- Hodnoty môžu byť priamo priradené jednotlivým členom štruktúry.

```
person1.age = 35;
```

- Hodnoty môžu byť priradené už pri inicializácii štruktúry.

```
struct Person person1 = {"Julia", 28, 180.0};
```

- typedef umožňuje zjednodušenie názvov štruktúr, aby sa zlepšila čitateľnosť kódu.
- Pomocou typedef môžeme definovať nový názov pre existujúcu štruktúru (napr. Person namiesto struct Person).
- Rozdiel medzi použitím struct Person a Person je v tom, že druhá forma je kratšia a čistejšia.
- Použitie typedef pre zjednodušenie názvu štruktúry:

```
typedef struct {  
    int day;  
    int month;  
    int year;  
} Date;
```

- Priradenie hodnôt členom štruktúry môže byť realizované aj pomocou ukazovateľov.
- V tom prípade využívame operator šípky (->)

```
struct Person *ptr = &person1; ptr->age = 30;
```

- Pre správu viacerých inštancií štruktúr naraz je možné použiť pole štruktúr.

```
struct Person people[2] = {
    {"Fero", 25, 203.3},
    {"Julia", 28, 180.0}};
```

- **Self-reference štruktúra** – je štruktúra, ktorá obsahuje ukazovateľ na premennú rovnakého typu. To umožňuje vytváranie reťazových zoznamov, stromov a iných zložitejších dátových štruktúr.

```
#include <stdio.h>
struct Node {
    int data;
    struct Node *next;
};
int main() {
    struct Node node1, node2;
    node1.data = 10;
    node2.data = 20;
    node1.next = &node2; // node1 ukazuje na node2
    node2.next = NULL;   // node2 je posledným prvkom
    printf("Data in node1: %d\n", node1.data);
    printf("Data in node2: %d\n", node1.next->data);
    return 0;
}
```

# Ako pracujeme s maticami v jazyku C?

- Matica je dvojrozmerné pole čísel usporiadané v riadkoch a stĺpcoch.
- V jazyku C je reprezentovaná dvojrozmerným poľom.

```
#define MX_ROWS 3  
#define MX_COLS 3  
int matrix[MX_ROWS][MX_COLS];
```

- Umožňujú efektívnu manipuláciu s dátami v riadkoch a stĺpcoch.
- Prístup k jednotlivým prvkom pomocou indexov riadkov a stĺpcov.

```
matrix[1][2] = 5; // priradí hodnotu 5 k prvku v druhom  
                 // riadku a treťom stĺpci  
int value = matrix[1][2]; // načítanie tejto hodnoty z matice
```

- **Transpozícia matice** prehodí riadky a stĺpce matice.
- Pomocou funkcií malloc a free môžeme alokovať pamäť pre matice dynamicky, čo je užitočné, ak veľkosť matice nie je vopred známa.



Akým spôsobom pracujeme v jazyku C so vstupom?

- V jazyku C môžeme so súbormi pracovať pomocou knižnice `stdio.h`, ktorá poskytuje základné funkcie na čítanie a zápis do súborov.
- **Rozlišujeme:**
  - Textové súbory: Obsahujú dáta vo forme čitateľnej pre človeka (napr. súbor `.txt`).
  - Binárne súbory: Obsahujú dáta vo forme čitateľnej pre počítač (napr. súbor `.bin`).
- Pre otváranie súborov používame funkciu `fopen`:


```
FILE *file_ptr;  
file_ptr = fopen("example.txt", "r");
```

- **Parametre:**
  1. **Názov súboru** – určuje názov súboru, ktorý chcete otvoriť,
  2. **Režim otvorenia** – reťazec, určujúci typ operácie, ktorú chcete vykonať so súborom.
    1. `"r"` – Otvorenie na čítanie. Súbor musí existovať.
    2. `"w"` – Otvorenie na zápis. Vytvorí nový súbor alebo vymaže stávajúci.
    3. `"a"` – Otvorenie na vkladanie. Vytvorí nový súbor, ak neexistuje.
- **Výstup:** prúd s dátovým typom `FILE*` (adresa otvoreného priestoru)

- Obvykle nasledujeme testom na neprázdnosť `if(file_ptr!=NULL) {...}`
- Otvorený súbor uzatvárame pomocou funkcie `fclose(FILE *file_ptr)`, pričom
  - Prúd s dátovým typom `FILE*`, ktorý určuje súbor, ktorý sa má uzavrieť.
  - Funkcia vracia 0, ak bol úspešne uzatvorený, inak EOF (obvykle 1).
  - **Súbory otvorené pre zápis, spôsobujú pri padnutí programu strátu dát.**

```
fclose(file_ptr);
```

- Čítanie a zápis do súboru:
  - Využívame funkcie podobné ako sme používali do teraz:
    - `fscanf(FILE* stream, ...)` – Načítanie formátovaných údajov zo súboru.
    - `fprintf(FILE* stream, ...)` – Zápis formátovaných údajov do súboru.
- Štandardná knižnica `stdio.h` poskytuje 3 základné prúdy:
  - `stdin` – štandardný vstup – obvykle klávesnica
  - `stdout` – štandardný výstup – obvykle konzola
  - `stderr` – štandardný chybový výstup – obvykle konzola (napr. `fprintf(stderr, "Error - ...")`)

1. **Zadanie:** Implementujte funkciu pre detekciu validného dátumu. Najskôr si treba definovať **štruktúry** pre **dátum** a pre **čas**. Reštrikcia rokov na interval  $\langle 1582, 2500 \rangle$ .
  2. **Zadanie:** Implementujte funkciu, ktorá rozhodne, ktorý z dvoch *dat* je *dřívější* (*dátumov* je *skorší*).
- **Riešenie:** (  ).

## Ďalšie informácie

- Prestupný rok je každý rok, ktorý je deliteľný štyrmi, okrem rokov, ktoré sú deliteľné sto, ale nie sú deliteľné štyristo.
- Niektoré mesiace majú menej ako 31 dní. Apríl, jún, september a november majú 30 dní. Február má 28 dní, alebo 29 dní počas prestupného roka.
- Ak zadáte neplatný dátum, napríklad 31. apríla, funkcia `is_valid_date` by mala ten dátum odmietnuť ako neplatný.
- Funkcia `earlier_date` postupuje od najvyššieho významného údajá k najnižšiemu (rok  $\rightarrow$  mesiac  $\rightarrow$  deň). Ak sú roky rovnaké, porovnávajú sa mesiace; ak sú mesiace rovnaké, porovnávajú sa dni.

```
// Funkcia na validáciu dátumu
int is_valid_date(struct date_t date) {
    // Reštrikcia rokov na interval 1582 - 2500
    if (date.year < 1582 || date.year > 2500 ||
        date.month < 1 || date.month > 12 ||
        date.day < 1 || date.day > 31) {
        return 0;
    }


    // Určité mesiace môžu mať až 31 dní
    // Nezabúdajte na prestupné roky!
    if (date.day == 31 && date.month != 1 &&
        date.month != 3 && date.month != 5 &&
        date.month != 7 && date.month != 8 &&
        date.month != 10 && date.month != 12) {
        return 0;
    } else if (date.month == 2) {
        // Február môže mať 28 alebo 29 dní
        // (v závislosti od prestupného roku)
        if (date.day == 30) {
            return 0;
        } else if (date.day == 29 && (!(date.year % 4 == 0 &&
            date.year % 100 != 0) || date.year % 400 != 0)) {
            return 0;
        }
    }
    return 1;
}
```

```
// Funkcia na porovnanie dvoch dátumov
// Vrátí 0 = date1 je skorší, 1 = date2 je skorší, 2 = rovnaký dátum
int earlier_date(struct date_t date1, struct date_t date2) {
    // Porovnanie rokov
    if (date1.year < date2.year) {
        return 0;
    } else if (date1.year > date2.year) {
        return 1;
    }

    // Porovnanie mesiacov
    if (date1.month < date2.month) {
        return 0;
    } else if (date1.month > date2.month) {
        return 1;
    }

    // Porovnanie dní
    if (date1.day < date2.day) {
        return 0;
    } else if (date1.day > date2.day) {
        return 1;
    }

    // Dátumy sú rovnaké
    return 2;
}
```

- Zadanie:** Implementujte funkciu `struct measurement_t load_measurement()`, ktorá číta zo `stdin` jednotlivé položky merania teploty a vracia zodpovedajúce meranie teploty.
- Zadanie:** Implementujte funkciu pre detekciu validných dát merania, tj. ktorá vezme ako parameter meranie a skontroluje, či sú dané položky validné (`date` zodpovedá dátumu, `time` zodpovedá času).
- Zadanie:** Načítajte 5 meraní do poľa (vo funkcii `main`) a spočítajte: priemernú teplotu, maximálnu teplotu, najteplejšie dopoludnie.
  - **Riešenie:** ().

## Ďalšie informácie:

- Hodiny musia byť v intervale 0 – 23, minúty a sekundy v intervale 0 – 59.
- Najteplejšie dopoludnie, získáme ako teplotu nameranú pred 12:00, pričom určíme najvyššiu z nich.


```
/**
 * @brief Skontroluje platnosť merania
 *
 * @param measurement Štruktúra reprezentujúca meranie
 * @return 1, ak je meranie platné, inak 0
 */
int is_valid_measurement(struct measurement_t measurement) {
    struct date_t date = measurement.date; /**< Dočasná štruktúra pre datum */
    struct time_t time = measurement.time; /**< Dočasná štruktúra pre čas */

    // Reštrikcia rokov na interval 1900 - 2100
    if (date.year < 1900 || date.year > 2100 || date.month < 1 ||
        date.month > 12 || date.day < 1 || date.day > 31) {
        return 0;
    }

    // Určité mesiace môžu mať až 31 dní
    if (date.day == 31 && date.month != 1 && date.month != 3 && date.month != 5 &&
        date.month != 7 && date.month != 8 && date.month != 10 && date.month != 12) {
        return 0;
    } else if (date.day == 29 && date.month == 2 &&
        (!(date.year % 4 == 0 && date.year % 100 != 0) || date.year % 400 != 0)) {
        return 0;
    }

    // Skontrolujte aj platnosť časovej pečiatky
    if (time.hour < 0 || time.hour > 23 || time.min < 0 || time.min > 59 || time.sec < 0 || time.sec > 59) {
        return 0;
    }
    return 1;
}

struct measurement_t load_measurement() {
    struct measurement_t measurement;
    printf("Format: rok mesiac den hodina\n");
    scanf("%d %d %d %d %d %d %f",
        &measurement.date.year,
        &measurement.date.month,
        &measurement.date.day,
        &measurement.time.hour,
        &measurement.time.min,
        &measurement.time.sec,
        &measurement.temperature);
    return measurement;
}
```

1. **Zadanie:** Implementujte funkciu na vytvorenie a výpis matice 3x3 (riadky x stĺpce).
  2. **Zadanie:** Implementujte funkciu pre vyhľadávanie hodnoty v matici. Ak sa nájde, vráťte hodnotu, inak -1.
  3. **Zadanie:** Implementujte funkciu pre vyhľadávanie hodnoty v matici, ktorá vráti štruktúru s riadkovými a stĺpcovými súradnicami hodnoty. Ak hodnota nie je v matici, vráťte riadok = -1 a stĺpec = -1.
- **Riešenie:** ()

## Ďalšie informácie:

- Dvojrozmerné pole (matica) je zoznam zoznamov, kde každý zoznam reprezentuje jeden riadok matice.
- Súradnice (riadok, stĺpec) určujú polohu prvku v matici. Riadky a stĺpce sú indexované od 0.
- Matica identity je špeciálna matica, kde prvky na hlavnej diagonále sú 1 a ostatné prvky sú 0.
- Pri vyhľadávaní hodnoty v matici kontrolujte každý prvok a ak sa nájde zhodný, vráťte súradnice.




```
struct mx_coords_t {
    int row;    /**< Riadok */
    int col;    /**< Stĺpec */
};

/**
 * @brief Vytlačí obsah matice MX_ROWS x MX_COLS
 *
 * @param arr Maticové pole
 */
void print_2d(int arr[MX_ROWS][MX_COLS]) {
    for (int riadok = 0; riadok < MX_ROWS; riadok++) {
        for (int stlpec = 0; stlpec < MX_COLS; stlpec++) {
            printf("%d ", arr[riadok][stlpec]);
        }
        printf("\n");
    }
    printf("\n");
}

/**
 * @brief Vyhľadá hodnotu v matici
 *
 * @param arr Maticové pole
 * @param value Hodnota na vyhľadanie
 * @return Hodnotu, ak sa nájde, inak -1
 */
int find_value(int arr[MX_ROWS][MX_COLS], int value) {
    for (int riadok = 0; riadok < MX_ROWS; riadok++) {
        for (int stlpec = 0; stlpec < MX_COLS; stlpec++) {
            if (arr[riadok][stlpec] == value) {
                return value;
            }
        }
    }
    return -1;
}
```

```
/**
 * @brief Vyhľadá hodnotu v matici a vráti jej súradnice
 *
 * @param arr Maticové pole
 * @param value Hodnota na vyhľadanie
 * @return Štruktúra mx_coords_t so súradnicami hodnoty. Ak hodnota nie je v
 * matici, vráti row = -1, col = -1.
 */
struct mx_coords_t find_item(int arr[MX_ROWS][MX_COLS], int value) {
    for (int riadok = 0; riadok < MX_ROWS; riadok++) {
        for (int stlpec = 0; stlpec < MX_COLS; stlpec++) {
            if (arr[riadok][stlpec] == value) {
                // Tento spôsob inicializuje štruktúru
                struct mx_coords_t position = {.row = riadok, .col = stlpec};
                return position;
            }
        }
    }
    // Kompaktnejší spôsob priameho vrátenia inicializovanej štruktúry
    return (struct mx_coords_t) {-1, -1};
}
```

1. **Zadanie:** Implementujte funkciu pre uloženie dvojrozmernej matice do súboru.
2. **Zadanie:** Implementujte funkciu pre načítanie 2D matice zo súboru. Predpokladajte fixnú veľkosť matice.
3. **Zadanie:** Vytvorte program, ktorý spočíta slová v zadanom súbore.
4. **Zadanie:** Vytvorte program, ktorý spočíta počet výskytov zadaného znaku v danom súbore.
  - **Riešenie:** ()

## Ďalšie informácie:

- Funkcia `fopen` otvára súbor v režime zápisu (`"w"`). Ak súbor neexistuje, vytvorí sa; ak existuje, prepíše sa.
- Funkcia `fopen` otvára súbor v režime čítania (`"r"`). Ak súbor neexistuje, operácia zlyhá.
- Pomocou `fscanf` načítate prvky matice zo súboru.
- Môžete použiť funkciu `fgetc` na čítanie znakov zo súboru a detekciu slov pomocou medzery, nového riadku alebo tabulátoru.
- Použite `fgetc` na čítanie znakov a porovnajte každý znak so zadaným znakom, pričom počítajte výskyty.
- Po ukončení práce so súborom nezabudnite zavrieť súbor pomocou `fclose`.

```
/**
 * @brief Uloží dvojrozmernú maticu do súboru
 *
 * @param mx Maticové pole
 * @return 0, ak je operácia úspešná, inak 1
 */
int save_to_file(int mx[MX_ROWS][MX_COLS]) {
    FILE *output;
    output = fopen("matrix_save.txt", "w");
    if (output == NULL) {
        return 1;
    }
    for (int riadok = 0; riadok < MX_ROWS; riadok++) {
        for (int stlpec = 0; stlpec < MX_COLS; stlpec++) {
            fprintf(output, "%d ", mx[riadok][stlpec]);
        }
        fprintf(output, "\n");
    }
    fclose(output);
    return 0;
}
```

```
/**
 * @brief Načíta dvojrozmernú maticu zo súboru
 *
 * @param mx Maticové pole
 * @return 0, ak je operácia úspešná, inak 1
 */
int load_from_file(int mx[MX_ROWS][MX_COLS]) {
    FILE *input;
    input = fopen("matrix_save.txt", "r");
    if (input == NULL) {
        return 1;
    }
    for (int riadok = 0; riadok < MX_ROWS; riadok++) {
        for (int stlpec = 0; stlpec < MX_COLS; stlpec++) {
            fscanf(input, "%d", &mx[riadok][stlpec]);
        }
    }
    fclose(input);
    return 0;
}
```