Struktury

DELTA - Střední škola informatiky a ekonomie, s.r.o.

Ing. Luboš Zápotočný

04.12.2025

Struktury

Struktury

Struct (struktura) je klíčové slovo v C pro vytváření vlastních (komplexních) datových typů

Proč používat struktury?

- Seskupení souvisejících dat do jednoho celku
- Lepší organizace kódu
- Reprezentace objektů z reálného světa

Struktury

Příklad: Student má věk a studijní průměr

```
struct Student {
    int age;
    float average;
};
```

Způsoby deklarace struktury

```c
// 1. Type definition only
struct Person {
    int age;
    float average;
};

struct Person john;
john.age = 20;
john.average = 1.3f;

printf("Age: %d, Average: %f\n", john.age, john.average);
```

Způsoby deklarace struktury

```c
// 2. Type definition + immediate variable declaration
struct Point {
    int x, y; // shorthand for int x; int y;
} center; // center is a variable of type struct Point

center.x = 10;
center.y = 20;

printf("Center: %d, %d\n", center.x, center.y);
```

Způsoby deklarace struktury

```c
// 3. Type definition + variable declaration with typedef (modern)
typedef struct {
    int age;
    float balance;
} Person;  // Now we can write "Person" instead of "struct Person"

Person john;
john.age = 20;
john.balance = 1000.0f;

printf("Age: %d, Balance: %f\n", john.age, john.balance);
```

Omezení inicializace struktury

Časté chyba začátečníků:

```
struct Point {
    int x = 0;  // ERROR! Cannot initialize members in type definition
    int y = 0;  // ERROR! Cannot initialize members in type definition
};
```

Proč to nefunguje?

- Struct je pouze **šablona/blueprint** pro vytvoření proměnné
- Hodnoty se nastavují až při vytvoření konkrétní proměnné

Struct - Správná inicializace

Správné způsoby inicializace:

```
struct Point {
    int x, y;
};
```

Struct - Správná inicializace

```c
int main() {
    // 1. Initialization in order of definition
    struct Point p1 = { 5, 10 }; // x = 5, y = 10

    // 2. Designated initializers (C99+)
    struct Point p2 = { .y = 20, .x = 15 }; // x = 15, y = 20

    // 3. Partial initialization (remaining is 0)
    struct Point p3 = { 5 }; // x = 5, y = 0
}
```

Uložení struktury v paměti

```
typedef struct {
    int x; // 4 bytes
    int y; // 4 bytes
} Point;

Point p = { 100, 200 };
```

Uložení struktury v paměti

Adresa	0x1000	0x1004
Proměnná	`p.x`	`p.y`
Hodnota	100	200
Velikost	8 bytes	

Důležité: Struktura je uložena v paměti jako souvislý blok

Velikost struktury

Velikost struktury (paměťové reprezentace) se vypočítá jako součet velikostí všech jejích členů

- **Skoro**

Velikost struktury

```c
struct Point {
    int x, y; // 2x 4 bytes = 8 bytes
};

int main() {
    printf("Size of Point: %zu bytes\n",
        sizeof(struct Point)
    );
    return 0;
}
```

```
Size of Point: 8 bytes
```

Velikost struktury

```c
struct Temperature {
    int x, y;   // 2x 4 bytes = 8 bytes
    char value; // 1x 1 byte  = 1 byte
};

int main() {
    struct Temperature t = { 10, 20, 30 };
    printf("Size of Temperature: %zu bytes\n",
        sizeof(struct Temperature)
    );
    return 0;
}
```

```
Size of Temperature: 12 bytes
```

Padding

Padding (výplň) je dodatečná paměť, která je přidávána do struktury, aby byly její položky „zarovnány“ v paměti

Padding

```
struct Temperature {
    char value; // 1x 1 byte  = 1 byte
    int x, y;   // 2x 4 bytes = 8 bytes
};
```

```
Size of Temperature: 12 bytes
```

```
struct Temperature {
    int x;      // 1x 4 bytes = 4 bytes
    char value; // 1x 1 byte  = 1 byte
    int y;      // 1x 4 bytes = 4 bytes
};
```

```
Size of Temperature: 12 bytes
```

Padding

```
struct Temperature {
    char value;     // 1x 1 byte  = 1 byte
    int x;          // 1x 4 bytes = 4 bytes
    char precision; // 1x 1 byte  = 1 byte
    int y;          // 1x 4 bytes = 4 bytes
};
```

```
Size of Temperature: 16 bytes
```

```
struct Temperature {
    int x, y;               // 2x 4 bytes = 8 bytes
    char value, precision; // 2x 1 byte  = 2 bytes
};
```

```
Size of Temperature: 12 bytes
```

Padding

Ve struktuře velmi **záleží na pořadí** členů a jejich velikosti

Kvůli **efektivitě práce s pamětí** bude kompilátor optimalizovat strukturu tak, jak byste možná na první pohled neočekávali

Přidaný padding může být uvnitř (mezi prvky) struktury nebo na konci struktury

Velikost paddingu se může lišit mezi **různými platformami** nebo kompilátory a je definována takzvaným **ABI** (Application Binary Interface)