UPR
Cvičení 6

Struktury

- Známe datové typy pro uložení řetězců a čísel
- Představme si, že vytváříme hru, kde si budeme chtít pamatovat žebříček nejlepších hráčů, tedy:

```
char* name;

int high_score;
```

- Co když máme víc hráčů?

Struktury

```
char* name1;
int high_score1;
char* name2;
int high_score2;
...
```

Nebo pole?

- Museli bychom vytvořit zvlášť pole pro jména a pole pro skóre

Struktury

- Řešení: struktura
  - Seskupíme více datových typů a vytvoříme tak nový datový typ

```
struct Player {
   char* name;
   int high_score;
};
```

a ve funkci `main` vytvoříme proměnnou typu `Player` jedním ze způsobů:

```
struct Player player;
struct Player player = {"Joe", 5};
struct Player player = {.name="Joe", .high_score=5};
```

Struktury

- Přístup k atributům pomocí .

```
player.name = "Joe";
player.high_score = 10;
```

- Můžeme předat jako parametr funkce

```
void player_info(struct Player p){...}
```

Struktury

- Úkol č.1

  Vytvořte funkci `player_new_score`, která nastaví nové maximální skóre u hráče, pokud je větší než aktuální skóre hráče

Struktury

- Předání struktury do funkce ukazatelem
- Zápis `(*p).high_score` lze nahradit `p→high_score`

- Je nutné psát všude `struct Player` ?
  - alias pro datový typ:

```
typedef <datovy typ> <alias>
typedef struct Player Player
```

Struktury

- Úkol č.2

  Vytvořte pole hráčů, naplňte ho daty a vypište hráče s nejvyšším skóre.

Třídění

- Chceme vytvořit žebříček hráčů
- Potřebujeme seřadit hráče podle skóre od největšího po nejmenší
  - Můžeme si napsat vlastní algoritmus
  - Nebo použít funkci `qsort` (součást `stdlib.h`)

Třídění

- ```
  void qsort (void* base,
              size_t num,
              size_t size,
              int (*compar)
              (const void*,const void*));
  ```
  ```

Struktury + třídění

- Úkol č.3

  Vytvořte pole hráčů, naplňte ho daty a vypište hráče seřazené podle skóre od největšího po nejmenší

Struktury

- Struktura, jehož prvkem je samotná struktura

```
struct Player
 {
   char* name;
   int high_score;
   struct Player* next;
 };
```

Struktury

- Úkol č.4

  Vytvořte funkci `player_score(struct Player* p)`, která vrátí součet skóre hráčů, kteří jsou spoluhráči hráče `p`.

  Jinými slovy, všechny hráče, kteří jsou nějak “napojeni” na hráče `p`.