Informační a komunikační technologie

Práce s polem

David Weber
Kabinet K13

weber3@spsejecna.cz

Jak jsme zatím pracovali...

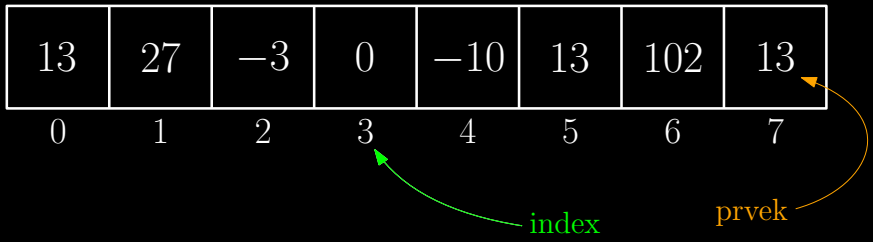
- Zatím jsme pracovali se samostatnými proměnnými
- Např. načtení rozměrů obdélníku

```
float width, height;
scanf("%d %d", &width, &height);
```

- Funguje pro pevný počet hodnot. 🙂
- Nefunguje pro proměnný počet hodnot. 😨

Co použít?

- Můžeme se dostat do situace, kde počet hodnot, které potřebujeme uchovat, závisí buď přímo na uživateli, nebo na jiných okolnostech
- $\Rightarrow$ pro uchování většího množství hodnot **stejného datového typu** používáme tzv. **pole** (anglicky *array*).
- Lze si představit jako „řadu přihrádek“



Vlastnosti pole I

- Každý prvek má v poli svoji pozici (index), které se číslují **od nuly**.
- Pro obecný počet prvků n máme tak indexy $0, 1, 2, \ldots, n-1$.
- Pole deklarujeme pomocí **datového typu prvků** a jeho **velikosti (tj. počtu prvků)**. Např.

```
int array[10];
```

- Případně lze rovnou deklarovat pole i s uchovanými hodnotami:

```
int array[] = { -1, 0, 2023, 40 };
```

- V takovém případě nemusíme uvádět velikost.

Vlastnosti pole II

- K prvkům přistupujeme pomocí jejich **indexu**, který uvádíme do hranatých závorek.

```
int array[10];
array[0] = 1;
array[3] = -10;
```

- Podobně můžeme např. vypsat daný prvek v poli či načíst hodnotu na vstupu:

```
scanf("%d", &array[7]);
printf("%d", array[7]);
```

Příklad (1/2)

Výpis všech zadaných prvků:

```
// Load array length
int count;
scanf("%d", &count);

int numbers[count];
```

Příklad (2/2)

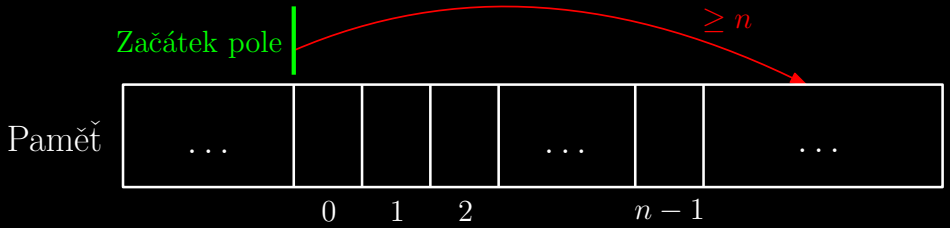
```c
// Fetch numbers into array
int i;
for (i = 0; i < count; i++){
    scanf("%d", &numbers[i]);
}

// Print array
for (i = 0; i < count; i++){
    printf("%d ", numbers[i]);
}

return 0;
```

Vlastnosti pole III

- Co když zasáhnu indexem mimo pole?

```
int arr[5];
arr[7] = ...;
```

- Jazyk C toto neošetřuje ⇒ zasáhneme mimo blok paměti pole!



- Může tak dojít k přepisu jiné proměnné

Otázky?