ZPRO 4. cvičení

Opakování: Funkce

- void

Opakování: String

- Funkce:
  - stoi(string)
  - isdigit(int)
  - isspace(int)

Pole (statická)

- Pole: skupina proměnných stejného typu, s níž pracujeme jako s celkem.
- Jednotlivé proměnné, které pole tvoří, nazýváme *prvky pole*.
- K prvkům přistupujeme pomocí indexů (pořadových čísel těchto prvků v poli), podobně jako k jednotlivým znakům řetězce.
- Indexy: celá čísla, první prvek má vždy index 0.
- Prvky pole mohou být jakéhokoli typu kromě referencí (mohou to být výčtové typy, struktury, s nimiž se seznámíme v následujícím cvičení, objekty nebo třeba opět pole).
- Pole, jehož prvky jsou opět pole, nazýváme *vícerozměrná*.
- Pole, jehož prvky nejsou pole, nazýváme *jednorozměrná*.
- Počet rozměrů = počet indexů potřebných k přístupu k prvku.
- Funkce nemohou vracet pole.
- **Překladač ani program nikdy sám nekontroluje, zda index leží v deklarovaném rozsahu!!**

Příklad: Funkce pro řešení kvadratické rovnice

Je dána rovnice $ax^2 + bx + c = 0$

- Vstupní data: pole obsahující koeficienty a, b, c
- Výstup: funkce vypíše všechna reálná řešení

Příklad: Největší prvek pole

Inicializace jednorozměrného pole

Deklarace jednorozměrného pole

Zjednodušeno.

deklarace jednorozměrného pole:
typ identifikátor specifikace_indexu **;**
typ identifikátor **[** *konstantní_výraz*$_{\text{nep}}$ **]** **=** *inicializátor_pole* **;**

specifikace indexu:
[*konstantní_výraz* **]**

Příkazy

prázdný příkaz:

```
;
{}
```

- Již známe:
  - Blok: **{posloupnost příkazů}**
  - Větvení programu: **if**, **else**

Switch

Výběr z většího počtu možností určených celými čísly

příkaz switch:
switch (*výraz* **)** *tělo_příkazu_switch*

Použití návěští **case** a **default**

Tělo příkazu switch ukončujeme pomocí příkazu **break;**

V dnešní době se příkazu switch spíše vyhýbáme.

Cykly (smyčky)

- Již známe příkaz **while**
- **FOR-cyklus**

  for (počáteční_inicializace; podmínka ; krok)
  {příkazy}
- **Ekvivalentní kód**

  {
  počáteční inicializace
  while (podmínka)
  {příkazy …; krok}
  }

Příklad: Obsahuje řetězec číslo?

Cíl: Napsat funkci, která dostane řetězec (`string`) a vrátí logickou hodnotu říkající, zda obsahuje celé číslo, které může obsahovat znaménko. Před a za číslem mohou být bílé znaky (mezery, tabulátory apod).
Vstup: Proměnná `text` typu `string`.
Výstup: `true`, jestliže řetězec obsahuje právě 1 celé číslo podle výše uvedených pravidel, jinak `false`.
Pomocná proměnná: `i` – index znaku v řetězci.

Příklad: Obsahuje řetězec číslo?

Postup:

1. Do `i` uložíme 0.
2. Přeskočíme případné úvodní mezery (v `i` bude index prvního nebílého znaku nebo délka řetězce).
3. Jsme-li na konci řetězce, vrátíme `false`, konec.
4. Je-li `text[i]` znak `'+'` nebo `'-'`, zvětšíme i o 1.
5. Jsme-li na konci řetězce, vrátíme `false`, konec.
6. Není-li následující znak číslice, vrátíme `false`, konec.
7. Přeskočíme následující číslice (v `i` bude index prvního následujícího znaku nebo délka řetězce).
8. Jsme-li na konci řetězce, vrátíme `true`, konec.
9. Přeskočíme následující mezery (v `i` bude index prvního následujícího znaku nebo délka řetězce).
10. Jsme-li na konci řetězce, vrátíme `true`, jinak vrátíme `false`.

Příkaz for pro rozsahy

příkaz for pro rozsahy:
for (*deklarace_proměnné* **:** *inicializace* **)** *příkaz*

Do *proměnné* se budou postupně ukládat kopie hodnot z rozsahu uvedeného v *inicializaci*.

- V případě polí musí překladač vidět definiční deklaraci!!

Cyklus s podmínkou na konci – do while

příkaz do–while:

do *příkaz* **while (** *podmínka* **) ;**

- Příkazy v bloku se vykonají alespoň jednou

- Příklad: Funkce, která si od uživatele vyžádá celé číslo

Skoky – přenos řízení

- **break;** - ukončení cyklů a switche
- **continue;** - předčasný přechod k další iteraci uvnitř cyklů
- **return;** - ukončení funkce

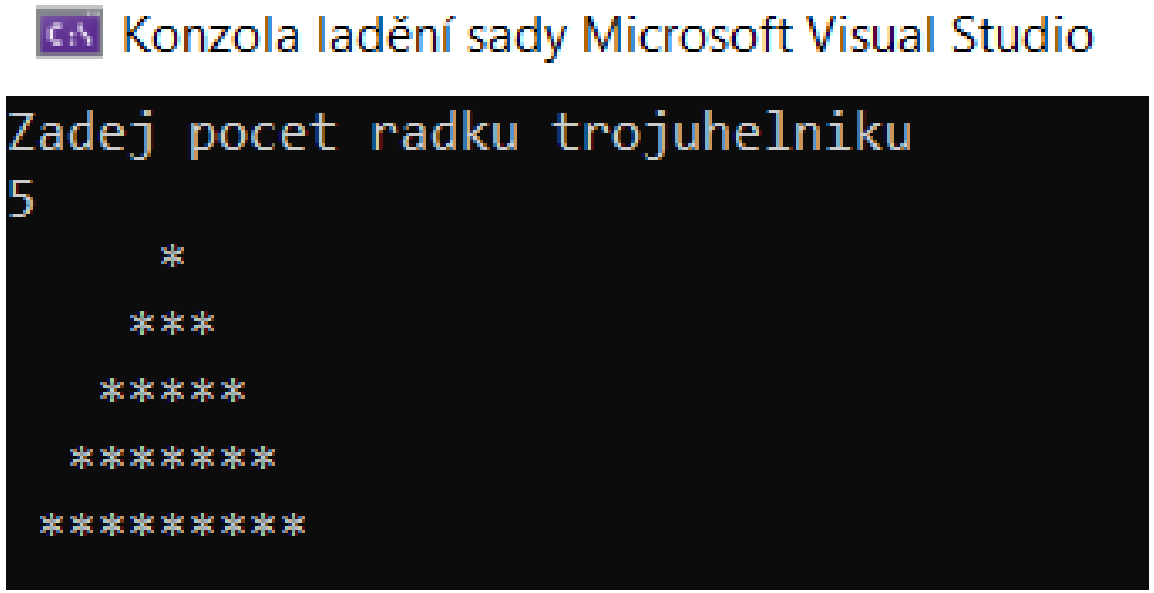
příkaz goto:

goto *návěští* **;**

Návěští = identifikátor připojený k příkazu dvojtečkou.

Domácí úkol: Trojúhelník

- Napište povídací program, který načte z klávesnice výšku rovnostranného trojúhelníku a následně nakreslí trojúhelník na obrazovku.



Domácí úkol: Eratosthenovo síto

```
algorithm Sieve of Eratosthenes is
    input: an integer n > 1.
    output: all prime numbers from 2 through n.

    let A be an array of Boolean values, indexed by integers 2 to n,
    initially all set to true.

    for i = 2, 3, 4, ..., not exceeding √n do
        if A[i] is true
            for j = i², i²+i, i²+2i, i²+3i, ..., not exceeding n do
                set A[j] := false

    return all i such that A[i] is true.
```