If - větvení kódu

A taky čtení vstupu

Knihovna

- Už před dlouho dobou si programátoři uvědomili, že vytvářet všechno od znova není efektivní a je lepší využít kód co existuje
- To vedlo ke vzniku knihoven s jazykem COBOL v roce 1959
- Knihovna je soubor funkcí, typů, objektů určený pro použití dalšími programátory
- Knihovny existují pro všechny možné účely - komunikace se zařízeními, otvírání obrázků, grafika, čtení/zapisování dat, komunikace se sítí, atd.

Standardní knihovna

- Knihovna automaticky používána každým programem
- Umožňuje komunikaci se systémem, obsahuje základní datové typy (třeba textové řetězce nebo seznamy) a užitečné prostředky pro práci s daty
- Každý programovací jazyk má vlastní standardní knihovnu
  - Velikost a obsáhlost se liší jazyk od jazyka
  - C - absolutně minimální
  - Python - velmi obsáhlá
  - Rust, C++ - středně velká
- Dokumentace standardní knihovny Rustu: https://doc.rust-lang.org/std/

Standardní knihovna v Rustu

- Její části se musí importovat, i když část je importována automaticky
- Importování pomocí klíčového slova use:

```
use std::i32::MAX;

fn main() {
    println!("maximální hodnota 32-bitového čísla je {}", MAX);
}
```

Používání cizí knihovny - text_io

1. Do Cargo.toml přidáte toto (na konec):

```
[dependencies]
text_io = “*”
```

2. Do src/main.rs toto (na začátek):

```
#[macro_use]  extern crate text_io;
```

3. Hotovo - při další kompilaci se knihovna automaticky stáhne a připojí k programu

Kalkulačka v4.0

```
#[macro_use] extern crate text_io;

fn main() {
    println!("zadejte první číslo: ");
    let a: i32 = read!();
    println!("zadejte druhé číslo: ");
    let b: i32 = read!();
    println!("vysledek: {}", a + b);
}
```

- Čtvrtá kalkulačka čte vstup “interaktivně” od uživatele
- Knihovna (v rustikální terminologii se taky používá termín crate) text_io přidává dvě makra na čtení vstupu (proto se muselo také přidat #[macro_use], u knihoven co nepřidávají makra se to nedává)

If

- Občas se hodí, aby program dělal různé věci v různých podmínkách
- K tomu slouží if a else

```
if podminka {
    // něco, když podmínka platí
}
else {
    // něco, když podmínka neplatí
}
```

- Blok else je nepovinný
- Součástí může být také ‘else if podminka’, zde se kód pustí pouze, když podmínka u ifu neplatí a podmínka u else ifu platí

Ukázka

```
fn main() {
    let a  = 5;
    let b = 10;

    if a < b {
        println!(“a je vskutku menší než b”);
    }
    else if a > b {
        println!(“a je větší něž b??”);
    }
    else {
        println!(“čísla se rovnají?????”
    }
}
```

- Na podmínky se používají relační operátory
- == - platí, pokud jsou operandy shodné (hodnotou)
- != - platí, pokud operandy nejsou shodné
- < - platí, pokud první operand je menší než druhý
- > - platí, pokud je první operand větší než druhý
- <= - platí, pokud je první operand menší nebo rovný druhému
- >= - platí, pokud je první operand větší nebo rovný druhému

If je výraz, může být přiřazen proměnné

```
let a = if b >= 5 {
    10 * b
}
else {
    100 * b
};
```

Úkol - Kalkulačka v5.0

- Kromě dvou čísel přečtěte i operátor (jako textový řetězec -> typ &str)
- Podle toho, jaký to je operátor, proveďte patřičnou operaci
- Vypište výsledek
- Pokud operace bude neznámá, vypište to