IT doupě

# Jak na… První program v Rustu

Jak si jistě pamatujete, v minulém dílu měsíčníku jsem vám ukazoval Rust, jazyk, který se pravděpodobně stane nástupcem céček a který má široké využití i v té „vyšší“ sféře programování. Zaměřil jsem se převážně na nějakou základní teorii, to dnes ovšem změníme a napíšeme si v něm první program, abyste si jeho syntax trochu osvojili.

Předpokládám, že už máte Rust stažený a vytvořený nový cargo projekt (pokud ne, koukněte na poslední díl). Myslím, že ideální na ukázku většiny funkcí jazyka bude hádací hra – na začátku se vygeneruje číslo a vy ho budete muset uhodnout.

Váš nový projekt by měl zatím obsahovat jen funkci **main()** a příkaz **println!()**. Jak jste si asi všimli, funkce se označují **fn**, používají se složené závorky na ohraničení a také tu jsou středníky. Funkce **main()** se spouští jako první, stejně jako v céčkách nebo Javě. **Println!()** vypíše do konzole text, stejně jako jeho alternativa **print!()**, který nezalamuje řádky.

Abychom mohli vygenerovat náhodné číslo, potřebujeme externí knihovnu. Ty se budou hodit celkem často, takže se za jejich používání nemusíte stydět. Tuto konkrétní přidáte do carga pomocí **rand = "0.8.5"** a do programu importujete přes **use rand::Rng**.Nezapomeňte program zkompilovat, aby se knihovna stáhla.

Teď by se hodilo uložit vygenerované číslo do nějaké proměnné. To, že existuje víc typů vás asi nepřekvapí, co je ale novinka je jejich měnitelnost – mutability. Všechny proměnné jsou v základu kvůli rychlosti a bezpečnosti neměnné, čili se jednou zapíšou a pak se z nich dá pouze číst. Pokud je plánujete přepisovat, musíte mezi klíčové slovo **let** a jméno proměnné dopsat **mut**. Za rovnítko napište **rand::thread\_rng().gen\_range(0..100)**, kde rand je hlavní knihovna a dvojitými dvojtečkami slézáte postupně dolů až k funkci **gen\_range()**, která bere parametr od..do.

Ještě prosím neusínejte, blížíme se ke konci – potřebujeme získat uživatelský vstup. K tomu nám poslouží knihovna *io*, která je v základu a k přidání jí tudíž stačí jen **use std::io**. Vytvořte si měnitelnout proměnnou *input* typu String (**let mut input = String::new()**) a pomocí **io::stdin().read\_line(&mut input).expect("Neplatný vstup")** do ní uložíte vstup – **stdin().read\_line()** funguje podobně jako input() v pythonu, vše se zapisuje do proměnné input (& znamená referenci, ale tím vás zatěžovat nebudu) a **expect()** se zavolá, pokud vstup zadaný uživatelem nebude platný. Často se místo něj používá funkce **unwrap()**, ale to doporučuji jen v případě, že víte, že vám daná věc nevyhodí chybu. String s číslem by se nám ovšem porovnával špatně, proto vytvoříme ještě proměnnou guess, do které pomocí **input.trim().parse::<i32>().expect("Vstup není číslo")** uložíme proměnnou input s novým typem i32. Nesmíme zapomenout ani na funkci **trim()**, která string ořeže od neviditelných znaků a samozřejmě už známý **expect()**, pokud by se konverze nepovedla.

A teď už zbývá jen čísla porovnat a říct, zda je větší, menší nebo rovno. Je více způsobů jak na to, dá se použít i příkaz **match**, já vám to ale ukážu na jednoduchém ifu. Tady se to píše úplně stejně jako v pythonu, jen s tím rozdílem, že se celý obsah podmínky stejně jako u funkcí uzavře do složených závorek. Samozřejmě můžeme použít i else if nebo else.

Nu, a to by bylo k těm „opravdovým“ základům jakžtakž všechno. Ještě bychom si mohli probrat cykly, v Rustu jsou tři: klasický **while**, **for** a nakonec **loop** neboli while true s jiným jménem. Můžete je zkusit v programu použít a například udělat hádání do doby, než se uživatel trefí. S tímhle vším vám hodně pomůže oficiální Rust knížka pro začátečníky, kterou najdete buď na <https://doc.rust-lang.org/stable/book/>, nebo lokálně ve vaší instalačce. Má 20 kapitol a v jedné z nich je i tento program, který jsme právě dělali. Existuje i spousta komunitních discord serverů, kde vám s vašim kódem rádi pomůžou. Přeji hodně štěstí s učením, v příštím čísle se rovnou vrhneme na okenní aplikace.

//img1,2