Obsah

22 Vstup/výstup, podmínky

22.1 Vstup/výstup do/z programu

- tzv. IO (input output) operace
- způsob komunikace programu se systémem, uživatelem a dalšími programy
- funkce poskytnuty knihovnou `stdio.h`

22.1.1 Uživatelský vstup

- čtení vstupu z `stdin`[1]
- funkce `getchar(<variable>);`, `gets(<variable>);`, `scanf(<format>, <pointer>);`
  - `getchar` – načtení jednoho znaku
  - `gets`
    - načtení stringu s mezerami, nelze číst čísla
    - konec inputu novým řádkem nebo EOF
    - unsafe, chybí ochrana před buffer overflow
  - `scanf` – načtení vstupu v zadaném formátu, konec inputu mezerou, `\n` nebo EOF

```c
#include <stdio.h>

int main(){
  // init variables
  char character, word[20], sentence[20];
  int number;

  // different input methods
  getchar(character);
  gets(sentence);
  scanf("%s %d", &word, &number);

  return 0; // exit
}
```

Kód 1: Načtení vstupu od uživatele

22.1.2 Výstup programu

- výpis textu do `stdout` nebo `stderr`
  - `stdout` – standard stream pro output
  - `stderr` – standard stream pro errory

[1] standard stream pro vstup v terminálu

- funkce `putchar`, `puts`, `printf`, `fwrite`
  - `putchar(<variable>);` – vypsání jednoho znaku
  - `puts(<variable>);`
    - funkce z knihovnyinlistc
    - prostý výpis proměnné do `stdout` bez formátování
    - na konci automaticky `\n`
  - `printf(<string and format>, <variable>);`
    - interpretace prvního stringu jako formátu
    - následné proměnné specifikují hodnotu dat ve formátu
    - nekončí automaticky novým řádkem
  - `fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);`
    - `*ptr` – buffer na vypsání, `size, nmemb` – délka a počet dat
    - zápis binárních dat do souboru nebo streamu

22.1.3 Formát printf/scanf

- využití formátovací značek pro definici formátu outputu – `%d, %i, %u, %c`…
- zápis do stringu
- formátovací značky různé pro každý data typ (viz otázka 21)

```c
#include <stdio.h>

int main() {
  int a,b,c;

  printf("Zadejte 3 cisla: ");
  scanf("%i %i %i", &a, &b, &c);

  printf("Sumy kazde dvojice: %i %i %i", a+b, b+c, a+c);

  return 0;
}
```

Kód 2: Použíti printf a scanf

22.2 Podmínky

- způsob spouštění kódu za pomocí podmínek
- příkaz `if (rule){...} else {...}`
- pokud je podmínka splněna, je spuštěn kód v bloku; pokud není splněna, blok je přeskočen
- `else` blok spuštěn pouze za nedodržení podmínky
- podmínka – boolenový výraz

22.2.1 Booleovské výrazy

- v C použit `int`, případně `bool` ze `stdbool.h`
- `true` – nenulová hodnota (nejčastěji 1), `false` – 0

22.2.2 Kombinace podmínek

- kombinování podmínek za pomocí logických operací `&&` a `||`
- první vyhodnocení podmínek, následně vyhodnocení kombinací (viz tab. 22.2)
- vyhodnocení zleva doprava
- logické výrazy možno skládat, závorky pro přednost…
- pokud v AND je první argument 0, další hodnoty již nejsou vyhodnoceny

Znak	Význam
==	rovnost
!=	nerovnost
<	menší než
<=	menší rovno
>	větší než
>=	větší rovno
&&	AND
||	OR
!	negace

Tab. 22.1: Logické operátory v C

```c
#include <stdio.h>

int main () {
  int i;
  printf("Zadejte cislo: ");
  scanf("%d", &i);
  if (i % 2 == 0) puts("Sude");
  else puts("Liche");
  return 0;
}
```

Kód 3: Příklad programu s podmínkou a if

Operátor	Směr vyhodnocení
! ++ -- - +	⟵
* / %	⟶
+ -	⟶
< <= >= >	⟶
== !=	⟶
&&	⟶
||	⟶
? :	⟵
= += -= *= …	⟶
,	⟶

Tab. 22.2: Priorita vyhodnocování logických výrazů

```c
int compare(a,b,c) {
  if (a == 0) return 0;
  if (a < b) return -1;
  if (a >= b) return 1;
  if ((a == c) && (b < c) || !(c == 1)) return 2;
  return 3;
}
```

Kód 4: Příklady boolenových operací