Programovací jazyk C

1.1. Preprocesor

Konstrukce preprocesoru C:

- Definování makra:
  `#define` *jméno* text rozvoje
- Zrušení definice:
  `#undef` *jméno*
- Podmíněný preklad na hodnotě *konstanta*
  `#if` *konstanta*
  `#elif #else #endif`
- Vložení souboru z adr. uživatele
  `#include "soubor"`
- Vložení systémového souboru
  `#include <soubor>`
- Podmíněný překlad, pokud je makro definováno
  `#ifdef` *makro*
  `#elif #else #endif`
- Podmíněný překlad, pokud je makro nedefinováno
  `#ifndef` *makro*
  `#elif #else #endif`
- Předdefinovaná makra: `__LINE__`, `__FILE__`, `__TIME__`, `__DATE__`, `__STDC__`

1.2. Proměnné

- Celočíselné typy:
- Se znaménkem (`signed`):
  - short (short int, signed short, signed short int)
  - int (signed int, signed)
  - long (long int, signed long, signed long int)
- Bez znaménka (`unsigned`):
  - unsigned short
  - unsigned int
  - unsigned long
- Znakový typ:
  - char
  - signed char
  - unsigned char
- Reálný typ:
  - float
  - double (long float)
  - long double
- Ukazatel:
  - `void *prt` – generický ukazatel
  - `(void *)0` – NULL
- Pole:
  - *typ* pole[] –
- Výčtový typ:
  - enum
- Struktura:

```
struct název {
        typ název1;
        typ název2;
};
```

- Bitové pole: unsigned a:4; – bitové pole o délce 4 bity
- Union
- Prázdný typ: `void`
- Paměťové třídy (před typ proměnné):
  - `auto` – lokální prom. na zásobníku,
  - `extern` – globální prom. v dat. oblasti,
  - `static` – lok. prom., ponechání hodnoty mezi jednotlivými voláními funkce,
  - `register` – lok. prom., uložení v registru počítače.

1.3. Výrazy

Čím větší číslo v indexu tím vyšší priorita.

- Unární výrazy:
  - $!_{15}$ – log. negace
  - $\tilde{}_{15}$ – bit. negace (`~0xF0F0=0x0F0F`)
  - $\&_{15}$ – adresový operátor
  - $*_{15}$ – nepřímý operátor
- Binární výrazy:
  - multiplikativní: $*_{13}$ (násobení), $/_{13}$ (dělení), $\%_{13}$ (zbytek po dělení)
  - aditivní: $+_{12}$ (plus), $-_{12}$ (mínus)
  - posuny: $<<_{11}$ (doleva), $>>_{11}$ (doprava)
  - rovnosti: $==_9$ (je roven), $!=_9$ (není roven)
  - relace: $<_{10}$, $<=_{10}$, $>_{10}$, $>=_{10}$
  - bitové: $\&_8$ and, $|_6$ or, $\hat{}_7$ xor
  - logické: $\&\&_5$ and, $||_4$ or
- Podmíněný výraz$_3$: *výraz-logické-or* ? *výraz* : *podmíněný výraz*
  - `r = a ? b : c;` $\Longleftrightarrow$ `if(a!=0) r=b; else r=c;`
  - `a ? b : c ? d : e ? f : g` $\Longleftrightarrow$ `a ? b : (c ? d : (e ? f : g))`
- Dosazovací výrazy$_2$:
  `= += -= *= /= %= <<= >>= &= ^= |=`
  (priorita 2, asociativní z prava)

a op= b	*a = a op b*
`x*=y=z`	`x*=(y=z)`
`a=b=d+7`	`a=(b=(d+7))`

- Příklady:
  - `++e` $\Longleftrightarrow$ `e+=1` – inkrem. *před* použitím
  - `e++` – inkrementace *po* použití
  - `--e` $\Longleftrightarrow$ `e-=1` – dekrem. *před* použitím
  - `j = ++i` $\Longleftrightarrow$ `j = (i = (i + 1))`
  - `j = i--` $\Longleftrightarrow$ `j = i; i = i - 1;`

1.4. Příkazy

if (výraz$_1$)
  příkaz$_1$;
else if (výraz$_2$)
  příkaz$_2$;
else if (výraz$_3$)
  příkaz$_3$;
...
else
  příkaz$_n$;

while (výraz)
  příkaz;

do příkaz
  while (výraz);
(oproti while se tělo cyklu provede alespoň jednou)

for (výraz$_1$; výraz$_2$; výraz$_3$)
  příkaz;

switch (výraz){
  case *konstanta*: příkaz;
  break;
  default : příkaz;
}

goto *návěští*;
návěští:
  příkaz;

1.5. Některé funkce

1.5.1. Řídící znaky:

sekvence	hodnota	význam
\n	0x0A	nová řádka (LF)
\r	0x0D	návrat vozíku (CR)
\f	0x0C	nová stránka (FF)
\t	0x09	tabulátor (HT)
\b	0x08	posun doleva (BS)
\a	0x07	písknutí (BELL)
\\	0x5C	zpětné lomítko
\'	0x2C	apostrof (signle quote)
\0	0x00	nulový znak (NULL)

1.5.2. Formátové specifikace:

% [příznaky][šířka][.přesnost][modifikátor]konverze

- Konverze:

%c	znak
%d	desítkové číslo `signed int`
%ld	desítkové číslo `signed long`
%u	desítkové číslo `unsigned int`
%lu	desítkové číslo `unsigned long`
%f	float
%Lf	long double
%lf	double
%x	hexadecimální číslo, např. 1a2b
%X	hexadecimální číslo, např. 1A2B
%o	osmičkové číslo
%s	řetězec
%p	adresa argumentu
%n	netiskne, ukládá počet znaků

- Modifikátor konverze:

`h`, `l`	`d`, `i` → signed short int; `u`, `o`, `x`, `X` → unsigned short int
`L`	`f`, `e`, `E`, `g`, `G` → long double

- Šířka:

n	tiskne alespoň *n* znaků, mezery zprava
0*n*	tiskne alespoň *n* znaků, nuly doplňuje zleva
*	počet znaků udává předchozí argument

- Přesnost:

`d`, `i`, `u`, `o`, `x`, `X`	minimální počet cifer na výstupu
`f`, `e`, `E`	počet cifer za desetin. tečkou
`g`, `G`	maximální počet významových cifer
`s`	počet tištěných znaků
*	počet znaků udává předchozí argument

- Příznak:

-	zarovnává doleva a zprava doplňuje mezery
+	tiskne číslo vždy se znaménkem +/−

1.5.3. Ukazatele

Příklady:
`*poi = 5;` – na adresu se zapíše 5
`i = *poi;` – získá se obsah na adrese
`poi = &i;` – ukazatel na adresu `i`
`i = poi;` – do `i` se uloží adresa `poi`
`(* poi)++;` – hodnota na adrese se inkrementuje
`double (* p_fd)();` – ukazatel na funkci vracející double

`int **xd;` – `xd` je ukazatel na ukazatel na `int`,
`*xd` je ukazatel na `int`,
`**xd` je prvek `int`

Pole:
vícerozměrné – `pole`[*řádek*][*sloupec*]
`x[i]` $\Longleftrightarrow$ `*(x+i)`
`x[i][j]` $\Longleftrightarrow$ `*(x[i]+j)` $\Longleftrightarrow$ `*(*(x+i)+j)`
Alokace paměti:
`int *poi;`
`poi = (int *) malloc(4*sizeof(int))` – přidělení
`poi[0]` $\Longleftrightarrow$ `*p_i`, `poi[1]` $\Longleftrightarrow$ `*(poi+1)`
`free(poi)` – uvolnění paměti

1.6. Příkazy gdb

Při použití `gcc` kompilovat s parametrem `-ggdb3` a bez optimalizací.

`run` – spustí program
`bt` – (backtrace) vypíše pouzžité funkce před pádem
`b` *funkce* – breakpoint na jméno funkce
`b` *n* – breakpoint na řádku číslo *n*
`tb` ... – dočasný breakpoint
`CTRL+C` – zastaví spuštěný program
`info breakpoints` – seznam nastavených breaků
`delete` *n* – zruší break číslo *n*
`p` *jméno* – vypíše obsah proměnné *jméno*
`p*` *ukazatel* – vypíše obsah na adrese ukazatele
`up/down` – pohyb po stack-framu
`next` – krokování programu
`thread` – přepínání mezi vlákny