Jazyk C - vznik, historie, základní pojmy

- Vznik a vývoj UNIX/C/C++
- Jazyk C - charakteristika
- První program
- Základní datové typy a operátory
- Vstup a výstup

Historie - UNIX & C

- 60. léta - vývoj OS **Multics** v Bell Telephone Laboratories (společně s General Electric a MIT)
- 1969 - vývoj zastaven
- Ken Thompson: návrh systému souborů a jádra jednoduchého operačního systému v asembleru pro PDP-7 (DEC). Brian Kernighan: **UNIX**.
- 1970 - Thompson navrhuje pro programování systému **jazyk B**. Vychází patrně z BCPL (Martin Richards, MIT, ~1965).
- 1971 - Thompson neúspěšně žádá nový počítač PDP-11 pro další vývoj, nový počítač přidělen pro projekt systému na zpracování dokumentů.

Historie UNIXu & C - pokračování

- 1972 - **Dennis Ritchie** - 1. verze jazyka **C**
- 1973 - jádro UNIXu na PDP-11 přepsáno do C
- 25 interních instalací
  Unix uvolněn pro použití na univerzitách
- 1977 - přenos systému na počítač Interdata 8/32
  licence komerčním organizacím, 500 instalací
- 1977-82 UNIX System III, 1983 System V
- 1978: B.W.Kernighan, D.Ritchie: The C Programming Language. Popis verze **K&R C**.
- 1988: 2. vydání, **ANSI C**. Norma obsahuje i popis knihovních funkcí a hlavičkových souborů.
- 1999: standard ISO/IEC 9899

Historie - C++

- navrhl Bjarne Stroustrup počátkem 80. let
  - kompatibilita s jazykem C
  - rozšíření o prostředky OOP
    inspirace jazykem Simula 67
- 1986: B. Stroustrup: The C++ Programming Language.
- 1997: 3. vydání, ANSI C++.

Jazyk C - charakterizace

- univerzální programovací jazyk relativně nízké úrovně
- úsporné vyjadřování, strukturovaný, velký soubor operátorů
- základní datové typy: znaky,čísla,adresy
- práce s řetězci,poli,V/V ← pomocí volání funkcí
  - jednoduchost
  - nezávislost
- slabá typová kontrola
- přenositelnost
- velká efektivita kódu (≈ assembleru)

Nejkratší program

```
main(){}
```

První program

```
/*****************************
*    prvni.c
*    19.02.2004
*****************************/

#include <stdio.h>

main()
{
printf("Jednoduchy program v jazyce C\n");
return 0;
}
```

Druhý program

```
/****************************
*    druhy.c
*    19.02.2004
****************************/

#include <stdio.h>

/* ukázka jednoduchého cyklu */

main()
{
int i;

for (i = 0; i < 10; i = i + 1)
        printf("i = %d\n", i);

return 0;
}
```

Komentáře

```
/*
 * PRVNI.C    v.1.0
 *
 * První program v C
 *
 * T.Dvořák 19.2.2004
 */

/* vnořené /* komentáře */ nejsou povoleny */
```

Identifikátory

- Posloupnost písmen, číslic a _ začínající písmenem nebo _
- ANSI C rozeznává prvních 31 znaků
  K&R C 8 znaků
  (ANSI C++ bez omezení)
- malá x VELKÁ písmena

Základní datové typy

- *Datový typ* určuje
  - množinu hodnot,kterých může nabývat
  - množinu operací, které nad ním mohou být prováděny
- Celočíselné typy
  - se znaménkem: kladná i záporná čísla
  - bez znaménka: pouze nezáporná

<prefix> char — závisí na implementaci

<prefix> short int *též* short
<prefix> int
<prefix> long int *též* long
} implicitně signed

<prefix> ::= prázdný | signed | unsigned

  - unsigned int *se zkracuje na* unsigned

Velikosti celočíselných typů

- sizeof(`short`) ≤ sizeof(`int`) ≤ sizeof(`long`)
  totéž pro `unsigned`

typ	16b	32b
`short`	-32768-32767	-32768-32767
`int`	-32768-32767	-2147483648-2147483647
`long`	-2147483648-2147483647	-2147483648-2147483647
`unsigned short`	0-65535	0-65535
`unsigned`	0-65535	0-4294967295
`unsigned long`	0-4294967295	0-4294967295
`unsigned char`	0-255	0-255
`signed char`	-128-127	-128-127
`char`	závisí na implementaci	

rozsahy typů v souboru `limits.h`

Logické hodnoty, reálná čísla

- Logické hodnoty jsou reprezentovány typem `int`
  - nulová hodnota (0) FALSE
  - nenulová hodnota TRUE
- Reálná čísla (čísla s pohyblivou řádovou čárkou)
  float
  double
  long double v K&R C není

rozsahy typů v souboru `float.h`

- sizeof(`float`) ≤ sizeof(`double`) ≤ sizeof(`long double`)

typ	paměť	rozsah	# platných cifer
`float`	4B	3.4E-38-3.4E+38	7-8
`double`	8B	1.7E-308-1.8E+308	15-16
`long double`	10B	3.4E-4932-1.1E+4932	19-20

Konstanty

- Celočíselné konstanty
  - *dekadické*: posloupnost číslic, první z nichž není 0
  - *oktalové*: 0 následovaná posloupností číslic 0 - 7
  - *hexadecimální*: 0x nebo 0X následovaná posloupností číslic 0 - 9, a - f, A - F
- Příklady
  - desítkové 15, 0, 1
  - osmičkové 065, 015, 0, 01
  - šestnáctkové 0x12, 0X3A, 0Xcd, 0x15, 0x0, 0x1
  - špatně 019, 1F5

Konstanty

- Typ celočíselné konstanty určen implicitně
  - první z typů, do jehož rozsahu se hodnota konstanty vejde
  - dekadické: `int,long,unsigned long`
  - oktalové,hexadecimální: `int, unsigned, long, unsigned long`
- Explicitně
  - `long(unsigned long)`: příponou `L` *nebo* `l`, např. `12345678L`
  - `unsigned(unsigned long)`: příponou `U` *nebo* `u`, např. `129u`, `100UL`

Konstanty

- Reálné konstanty: `15. 56.8 .84 .0 0. 5e6 7E23`
- Implicitní typ `double`
  - `float`: přípona `f` nebo `F`, např. `2.1f`
  - `long double`: přípona `L` nebo `l`
- Znakové konstanty
  - znak mezi apostrofy: `'A','.','%','ř'`
  - jsou typu `int` !
- Speciální znaky (řídící symboly) jak znakové konstanty - escape sekvence:
  - `'\ddd'` osmičkový kód znaku, např. `'\012'`
  - `'\0xHH'` *nebo* `'\0XHH'`, např. `'\0x0A'`

Znakové ekvivalenty často používaných řídících znaků

sekvence	ASCII	význam
\n	NL (LF)	nový řádek
\t	HT	horizontální tabulátor
\v	VT	vertikální tabulátor
\b	BS	posun doleva
\r	CR	počátek řádky
\f	FF	konec stránky
\a	BEL	zvonek
\\	\	obrácené lomítko
\?	?	otazník
\'	'	apostrof
\"	"	uvozovky
\0	NUL	nulový znak

Řetězcové konstanty

- `"Toto je ukázková řetězcová konstanta"`
- `"Ahoj, \nlidi!"`
- Automatické zřetězení dvou konstant oddělených bílými znaky:
  - `"Toto je velmi" "dlouhý řetězec"`

Definice a deklarace proměnných

- každá proměnná musí být před použitím deklarována
- *definice* (definiční deklarace) přidělí proměnné určitého typu jméno a paměť
- *deklarace* (informativní deklarace) udává jméno a typ proměnné
- ```
  char c;
  int i;
  float f;
  ```
- ```
  int i1, i2;
  ```
- součástí definice může být inicializace
- ```
  int i = 1;                    /* inicializace */
  int i1 = 10, i2 = 20;
  ```
  ```

Přiřazení

- operátor přiřazení =
- l-hodnota - výraz, který může stát na levé straně přiřazení, např. proměnná (představuje adresu)
- přiřazení *l-hodnota = výraz* `i = j * 2 + 5`
- přiřazení je výraz, jehož hodnotou je hodnota na pravé straně
- přiřazení se stává příkazem, je-li ukončeno středníkem
- přiřazovací příkaz

  l-hodnota = výraz; `i = j * 2 + 5;`
- `c = a = b = 2;` se vyhodnocuje `c = (a = (b = 2));`

Výrazový příkaz

- Připojením středníku se z výrazu stane příkaz
- ```
  main()
  {
   5;6*9;sin(7);
   return 0;
  }
  ```
  ```

Aritmetické operátory a výrazy

operátor	počet operandů	asociativita	priorita
+ -	1	←	↑
* / %	2	→	
+ -	2	→	

- / celočíselné nebo reálné dělení dle typu operandů
- % zbytek po celočíselném dělení (jen celočíselné operandy)
-
```
int i = 5, j = 13;

j = j / 4;
j = i % 3;
```

Další operátory

- inkrement ++ *++l-hodnota* *l-hodnota++*
  dekrement -- *--l-hodnota* *l-hodnota--*
- ~~`45++ --(i+j)`~~
- prefix × postfix
  - `++i` inkrementace před použitím
  - `i++` inkrementace po použití
- Příklad:

```
int i = 5, j = 3, k;
i++;
j = ++i;
j = i++;
k = --j + 2;
```

Lexikální konvence

- `a+++b;`
- Možné interpretace?

  `(a++)+b;` `a+(++b);` `a+(+(+b));`
- Od konce posledního nalezeného symbolu překladač rozpozná vždy nejdelší řetězec, který může utvořit další symbol
- Interpretace: `(a++)+b;`

Rozšířený přiřazovací operátor

- *l-hodnota* = *l-hodnota operátor výraz*

  ⇓

  l-hodnota operátor = *výraz*
- rozšířené přiřazovací operátory:
  `+=  -=  *=  /=  %=  >>=  <<=  &=  |=  ^=`
- stejná priorita, asociativita ←
- Příklad:

```
int i = 5, j = 4;

j += i;
j /= --i;
j *= i-2;
```

Vstup a výstup

- ```
  #include <stdio.h>
  main()
  {
    int c;

    c = getchar();
    putchar(c);
    putchar('\n');
  }
  ```
- V/V jednoho znaku
- proměnné typu **`int`**
  ```

Formátovaný vstup a výstup

- vstup: **`scanf()`** výstup: **`printf()`**
- výstup: **`printf("%d",i);`**
- vstup: **`scanf("%d",&i);`**
- Příklad:

```
#include <stdio.h>
main()
{
  int i,j;

  scanf("%d", &i);
  scanf("%d", &j);

  printf("Soucet cisel %d a %d", j, i);
  printf(" je %d.\n", i+j);
}
```

Formátovaný V/V

- proměnný # parametrů
- 1. parametr - formátový řetězec
  - formátové specifikace začínají znakem %
  - znakové posloupnosti
- Příklady:

```
printf("\007Soucet je %d%%", i+j);
printf("Toto je \"backslash\": '\\'\n");
printf("Je přesně %2d:%2d\n", hod, min);
printf("Utratili jsme: %6.2f Kc\n",
pocet*cena);
```

Formátové specifikace

`c` znak
`d` dekadické číslo typu `int`
`ld` dekadické číslo typu `long`
`u` dekadické číslo typu `unsigned`
`lu` dekadické číslo typu `unsigned long`
`f` číslo typu `float`
`lf` číslo typu `double`
`Lf` číslo typu `long double`
`x` hexadecimální číslo malými písmeny
`X` hexadecimální číslo velkými písmeny
`o` oktalové číslo
`s` řetězec