Reťazce, Preprocesor

Základy procedurálneho programovania 1, 2020

Ing. Marek Galinski, PhD.

Reťazce

- **Jednorozmerné polia typu char ukončené znakom ‘\0’**
- Dĺžka reťazca 0 – veľkosť poľa-1
- Reťazec musí byť ukončený ukončovacím znakom
- Pole znakov nemusí byť ukončené ukončovacím znakom

Definícia a inicializácia reťazca

- **Inicializácia – len v definícii**
- Ukončovací znak implicitne

```
char s[6] = "ahoj";
```

```
char s[10];
...
printf("%s", s);
```

- Nedá sa to pre “statické pole”

```
char s[10];
s = "ahoj";
```

```
char s[10];
int i;

for (i = 0; i < 10-1; i++)
    s[i] = '*';
s[10-1] = '\0';
```

Prevody reťazcov na čísla

- **Ako skonvertovať textový reťazec na číslo?**
- Definované v stdlib.h

```
int atoi(char *s);
```
Reťazec na int

```
long atol(char *s);
```
Reťazec na long

```
float atof(char *s);
```
Reťazec na float

Prevody reťazcov na čísla

- **Ako skonvertovať textový reťazec na číslo?**
- Definované v stdlib.h

```
char s1[100], s2[100];
int i;

scanf("%s %s", s1, s2);
if (!strcmp(s1, "int")) {
   i = atoi(s2);
   printf("Nacitalo sa cele cislo: %d\n", i);
}
else {
   printf("Nacital sa retazec znakov: %s\n", s2);
}
```

Formátovaný vstup a výstup

- **Používanie výhod formátovaného vstupu a výstupu, avšak voči reťazcom, nie súboru či obrazovke**

```
int sprintf(char *s, char *format, ...);
```

pracuje ako **fprintf**, ale zapisuje do reťazca **s**

```
int sscanf(char *s, char *format, ...);
```

pracuje ako **fscanf**, ale číta z reťazca **s**

Formátovaný vstup a výstup

- **Podobne aj z terminálu**

```
char *gets(char *s);
```

číta celý riadok do **s**: na koniec nezapíše **\n**, ale **\0**, vracia ukazovateľ na **s**, ak je riadok prázdny dáva do **s** **\0** a vráti **NULL**

```
int puts(char *s);
```

vypíše reťazec a odriadkuje (**\n**), vráti nezáporné číslo ak sa podarilo vypísať, inak **EOF**

Funkcie pre prácu s reťazcom

- **Nie sú súčasťou jazyka C samotného**
- Sú definované v <string.h>

- Zistenie dĺžky reťazca `int strlen(char *s);`
- Kopírovanie reťazca znakov `char *strcpy(char *s1, char *s2);`
- Spojenie reťazcov `char *strcat(char *s1, char *s2);`
- Porovnávanie reťazcov `int strcmp(char *s1, char *s2);`
- Nájdenie prvého výskytu znaku `char *strchr(char *s, char c);`
- Nájdenie prvého výskytu reťazca `char *strstr(char *s1, char *s2);`

Činnosť preprocesora

Preprocesor

- **Spracováva zdrojový kód pred kompilátorom, pred prekladom**
- Vypúšťa zo zdrojového kódu komentáre
- Zamieňa text – identifikátor konštánt za číselné hodnoty
- Rozvíja makrá
- Vkladá súbory
- Prevádza podmienený preklad

- Nekontroluje syntax
- Direktívy pre preprocesor začínajú znakom #

Konštanty

- **Konštante je v podstate makro bez parametrov**

- Používajú sa často
- Definujú sa na začiatku modulu
- Platia do konca modulu
- Náhrady konštanty hodnotou

```
#define meno_konst konst
```

Kedy sa makro nerozvinie?

- **Nerozvinie sa, ak je uložené v úvodzovkách**

```
#define MENO   "Katka"

...
printf("Volam sa MENO");
```

```
printf("Volam sa %s", MENO);
```

Prekrývanie definícií

- **Nová definícia prekrýva starú**
- Dobrým zvykom je zrušiť starú definíciu a definovať nanovo

```
#undef meno_makra
```

```
#define POCET 10
#undef POCET
#define POCET 20
```

Makro ako časť programu

- **Pri použití sa makrá neukončujú bodkočiarkou**

```
#define ERROR { printf("Chyba v datach.\n"); }
```

```
if (x == 0)
    ERROR
else
    y = y / x;
```

Makro s parametrom

- **Krátka a často používaná funkcia, napr. Jednoduchý výpočet**
- Niekedy je to efektívnejšie, než používať skutočné funkcie – rozdiel v tom, ako sa vykonajú.

- Je potrebné rozhodovať sa:
  - Funkcia = kratší, ale pomalší program
  - Makro = dlhší, ale rýchlejší program

Makro s parametrom

- **Rozvinutie makra znamená len to, že meno makra sa nahradí jeho telom na príslušnom mieste v kóde**

- Zápis

```
#define je_velke(c) ((c) >= 'A' && (c) <= 'Z')
```

```
ch = je_velke(ch) ? ch + ('a' - 'A') : ch;
```

- Výsledný kód

```
ch = ((ch) >= 'A' && (ch) <= 'Z') ? ch + ('a'-'A') : ch;
```

Makro s parametrom

- **Viacero parametrov v makre a využitie ternárneho operátora**
- Vždy používať zátvorky, aby nevznikol chaos

```
#define max(a, b) ((a) > (b) ? (a) : (b))
```

```
#define na_velke(c) ((c)>= 'a' && (c)<='z' ?
   (c)-'a'+'A' : (c))
```

```
#define PI 3.14
#define obsah_kruhu(r) (PI * (r) * (r))
```

Skrytá časť programu

- **Načítanie čísla a zistenie, či je hodnota 0**

```
#include <stdio.h>

#define citaj_int(i) (scanf("%d", &i), i)

int main() {
   int j, k;

   printf("Zadajte cele cislo: ");
   if((j = citaj_int(k)) == 0)
      printf("Bola nacitana nula.\n");
   printf("Bolo nacitane cislo %d", k);
   return 0;
}
```

Preddefinované makrá

- **Napr. v ctype.h – konverzia znaku**

`tolower` - konverzia na malé písmeno
`toupper` - konverzia na veľké písmeno
`toascii` - prevod na ASCII - len najnižších 7 bitov je významných

Vkladanie súborov

- **Systémové súbory vs. súbory v aktuálnom adresári**
- Viac pri oddelenom preklade, ďalší semester

```
#include <stdio.h>
#include <ctype.h>
#include "KONSTANTY.H"
```

Podmienený preklad

- Ladiace časti, voliteľné časti program napr. podľa prepínača, konfigurovateľný program na základe súboru config.h

```
#if konstantny_vyraz
    cast_1
#else
    cast_2
#endif
```

```
#define PCAT 1

#if PCAT
    #include <conio.h>
#else
    #include <stdio.h>
#endif
```

Podmienený preklad

- Ladiace časti, volitelné časti program napr. podľa prepínača, konfigurovateľný program na základe súboru config.h

```
#define PCAT 1

#if PCAT
    #include <conio.h>
#else
    #include <stdio.h>
#endif
```

```
#define PCAT

#ifdef PCAT
    #include <conio.h>
#else
    #include <stdio.h>
#endif
```

Podmienený preklad

- **Ladiace časti, voliteľné časti program napr. podľa prepínača, konfigurovateľný program na základe súboru config.h**

```
#if defined(ZAKLADNY)   &&   defined(DEBUG)
    #define VERZIA_LADENIA  1
#elif defined(STREDNY)   &&   defined(DEBUG)
    #define VERZIA_LADENIA 2
#elif !defined(DEBUG)
    #error Ladiacu verziu nie je mozne pripravit!
#else
    #define VERZIA_LADENIA  3
#endif
```

Podmienený preklad

```
#include <stdio.h>
#define LADENIE
int main() {
   int x, y, nasobok = 0;

   printf("Zadajte dve cisla: ");
   scanf("%d %d", &x, &y);

   printf("%d * %d = ", x, y);
   for(; y>0; y--) {
      nasobok += x;

#ifdef LADENIE
   printf("\n(y: %d, nasobok: %d)\n", y, nasobok);
#endif

   }
   printf("%d\n", nasobok);
```

Footnotes

- **Prednáška je dostupná na YouTube v dvoch dieloch:**
  - **Úvod k reťazcom**
    https://www.youtube.com/watch?v=PmzBAFRTHYQ
  - **Pokračovanie reťazcov, preprocessor**
    https://www.youtube.com/watch?v=6STnf5uxHng

V prednáške boli použité materiály zo slidov prednášok ZPrPr1 od Gabriely Grmanovej.

Q?

26