Znaky a reťazce, part 1: znaky

Náhradná Prednáška 6, PROG-2, 2020
Pavol Zajac

1. Znakové premenné a znakové konštanty

Znaky sú v počítači reprezentované postupnosťami bitov na základe dohodnutého kódu. Existuje množstvo kódov (Morseho, telegrafný, Unicode, ...), ale typicky budeme používať ASCII kód.

- úloha: pohľadajte si na internete ASCII tabuľku, a nájdite príklad iného kódu

Štandardný ASCII kód je rovnomerný (každý znak je kódovaný rovnakým počtom bitov), a v jazyku C jeden ASCII kód sa zmestí do 8-bitovej premennej typu `char`.

```
char a = 65;

printf("Kod %i reprezentuje znak %c\n", a, a);
//printf v prvom pripade spracuje a vypise kod ako cislo, %i
//format string %c zabezpeci vypis znaku s predpisanym kodom
```

Aby sme si nemuseli pamätať ASCII kódy, jazyk C poskytuje syntax znakových premenných. Znaková premenná je uzavretá v jednoduchých apostrofoch `'` (na rozdiel od Pythonu, v apostrofoch môže byť len 1 znak). Znakovú konštantu preloží prekladač na príslušný číselný kód sám.

```
char a = 'b';

printf("Kod %i reprezentuje znak %c\n", a, a);
```

Pozn.: Pre prehľadnosť zdrojového kódu je nevhodné písať ASCII kódy priamo, snažte sa vždy použiť znakové konštanty.

Netlačiteľné znaky (napr. koniec riadku) sa dajú zapísať pomocou špeciálnych „escape sekvencií" začínajúcich znakom `\`, napr. už známy zápis konca riadku cez `'\n'`. Každá escape sekvencia predstavuje len 1 kód.

- úloha: vyhľadajte si, aké štandardné escape sekvencie existujú, a ako zapísať špecifický znak s daným ASCII kód vo formáte escape sekvencie

Keďže znaky sú reprezentované číselnými kódmi, môžeme robiť s nimi všetky matematické a logické operácie ako s bežnými číslami (sčítanie, odčítanie, ... porovnanie, ...). Treba však dávať pozor na pretečenie, štandardný typ `char` predstavuje kódy od -128 po 127! Na vyhnutie sa pretečeniu môžeme ukladať znakové kódy aj do premenných typu `int` (short, long, etc.)

Príklad, výpis abecedy:

```
char c;
for (c = 'A'; c < 'Z'; c++)
     printf("%c", c);
```

Príklad, výpis rozšírenej ASCII tabuľky:

```
int c;   //co by sa stalo, keby tu bol char?
for (c = 0; c < 255; c++)
     printf("%i : %c\n", c, c);
```

Na niektoré časté prípady práce so znakmi jazyk C poskytuje knižničné funkcie, sprístupnené cez hlavičkový súbor `ctype.h`:

1. zisťovanie typu znaku: isalpha, isspace, ...
2. konverzie znakov: toupper, tolower

- úloha: naštudujte si dokumentáciu k `ctype.h` a príslušným funkciám

Príklad:

```
int c;
for (c = 0; c < 255; c++)
     if (isprint(c)) //vypiseme iba tlacitelne znaky
         printf("%i : %c\n", c, c);
     else
         printf("%i :   \n", c);


//zlozitejsie priklad, vlastna konverzia zo znakovej cislice
// na cislicu intovu v rozsahu 0-36
int char2int(char c)
{
    //pismena najprv dame jednotne na male cez tolower
    // potom odratanim kodu 'a' ich presunieme do rozsahu 0-25
    // pripocitanim 10 je to 10-36
    if (isalpha(c))
        return tolower(c) - 'a' + 10;

    //predpokladame, ze zvysne su cislice
    // odratanim kodu znaku '0' sa dostaneme do rozsahu 0-9
    return c - '0';
}
```

Pozn.: Funkcie `toupper`, `tolower`, majú špecifikáciu `int toupper(int)`. Tieto funkcie teda dostanú na vstup číselný kód (odovzdávanie hodnotou, nie je tam smerník), a na výstupe majú príslušnú skonvertovanú hodnotu (napr. premenná `c` sa v predošlom príklade neznemí). AK vstup nie je písmeno, vrátia kópiu vstupu.

Pozn.: V ASCII kóde by sa napr. konverzie na malé písmená dala zapísať aj ako `c - 'A' + 'a'`. Takýto zápis je však menej vhodný ako použitie tolower: na rozdiel od tolower nefunguje, ak nie je vstup veľké písmeno, a tiež nemusí fungovať pri iných kódovaniach.

Čítanie znakov z konzoly

Podobne ako pri `printf`, načítať znaky môžete cez `scanf` cez format string `%c`.

```
char znak;
scanf("%c", &znak);
```

V tomto prípade sa do premennej znak zapíše (ASCII) kód načítaného znaku z klávesnice. Problémom je, že vstup z klávesnice je bufferovaný, čo podrobnejšie vysvetlíme až pri súboroch. Zjednodušene to znamená, že keď zadávate vstup z klávesnice, nič sa nedeje, kým nestlačíte ENTER. Vtedy všetko, čo ste napísali sa presunie do pamäte, a z nej sa čísla, znaky, atď. čítajú. Pri číslach to až taký problém nie je, lebo pri načítavaní čísel sa ignorujú biele znaky (medzery, taby, konce riadkov). Ak použijete %c, tak sa načíta aj každý znak konca riadku, čo je niekedy neželané.

- úloha: vyhľadajte si, ako vyriešiť tento problém využitím možností formátovacieho reťazca scanf

Ak chcete načítať jeden znak priamo, môžete tiež použiť funkciu `int getchar()`. Táto funkcia nemá parametre, a vracia výstupný ASCII kód ako celé číslo, čiže ju môžete používať priamo vo výrazoch bez dočasných premenných:

```
//nacitava zo vstupu vsetky znaky, az kym nenarazi na koniec riadku
// nacitane znaky sa nikde neukladaju, iba sa porovnaju s '\n'
// takto sa napr. daju vymazat nepotrebne vstupne znaky...
while (getchar() != '\n') ;
```