31. Strings

DIM0321

2015.1

Outline

Contexto

- Computadores entendem apenas números.
- C é uma linguagem sem muita abstração do hardware
- De fato, caracteres são símbolos a serem representados por números.
- Em C vais ser o tipo `char`.
- Uma codificação explicita a relação entre um caractere e um número.

Strings

- Strings são construídas a partir de um ponteiro e do tipo char.
- Uma *string* é um arranjo de char (i.e. ponteiro) terminado por '\0'.

Codificação

Padrões

- ASCII (*American Standard Code for Information Interchange*), 1960: originalmente codificação de 7 bits
- EBCDIC, 1963 : codificação de 8 bits de IBM.
- ISO-Latin1(8859-1), 1985: codificação de 8 bits baseado no ASCII para o alfabeto latino, suporte terminado em 2004.
- UTF-8 (/Universal coded character set + Transformation Format - 8 bit), 1992: codificação de tamanho variável com unidades de codificação de 8 bits, compatível com ASCII, mais usado desde 2008.

C

- O tipo `char` de C corresponde à codificação ASCII.

Lembretes

O tipo char

- É um tipo de inteiro de 1 byte.
- O tipo `char` pode ser com o sem sinal:
  - `signed char` varia entre $[-128, 127]$
  - `unsigned char` varia entre $[0, 255]$
- Pode ser inicializado de varias vezes.

```
1  #include <stdio.h>
2  int main(void)
3  {
4       char c = 'A';
5       printf("%c\n",c);
6       c = 65;
7       printf("%c\n",c);
8       c =  0x41;
9       printf("%c\n",c);
10      c =  '\x41';
11      printf("%c\n",c);
12      c = 0101;
13      printf("%c\n",c);
14      c = '\101';
15      printf("%c\n",c);
16
17 }
```

Contrabarra

Definição

A contrabarra '\' e o **caractere de escape** (i.e. o único único caractere numa cadeia de caracteres que altera o significado de seu sucessor).

Exemplo

- aspa simples: '\''
- contrabarra: '\\'
- espaço: ' '
- quebra de linha: '\n'
- tabulação: '\t'
- aspas duplas : "\""

Expressões

Equivalência

Caractere = número → podemos fazer operações sobre eles

```c
#include <stdio.h>
int main( void )
{
    char i;
    char c = 'A' + 1; // 'B'
    printf ("%c ", c);
    c = 'A' < 'B' + 66; // 'D' = 1 + 66
    printf ("%c ", c);
    scanf ("%c", i);
    for (i = 'A'; i <= 'Z'; ++i) printf ("%c ", i);
    return 0;
}
```

Texto

Definição (Texto)

- sequência de caracteres (letras, pontuação, . . .)
- C não possui um tipo de dado específico, e usa um arranjo de caracteres *terminado por o caractere nulo ‘\0’*, caractere especial usado somente para delimitar o final de um texto.

Observaçãoj

Se o texto tiver n símbolos, vai ocupar $n + 1$ char em memória, com o primeiro caractere no índice 0. ‘\0’ ocupará o índice n. O código ASCII do caractere nulo e 0.

Exemplo

```c
#include <stdio.h>
int main(void)
{
    char texto[] = {
        'H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '!', '\n',
        '2', '4', '/', '0', '5', '/', '2', '0', '1', '3', '\0' };
    int i = 0;
    while(texto[i] != '\0') {
        printf("%c", texto[i++]);
    }
    printf("\n");
    printf("%s\n", texto);
    return 0;
}
```

- Em C, o texto e do tipo `char *` .
- O **programador** tem que garantir que o bloco de memória tenha espaço suficiente, e que e terminado por o caractere nulo.

Funcionamento das cadeias

Quando há uma constante texto no programa, o compilador gera instruções para:

- reservar um compartimento de memória de tamanho suficiente
- preencher este compartimento com os caracteres do texto
- colocar o caractere nulo logo após o último símbolo do texto
- retornar o endereço onde o primeiro símbolo do texto foi armazenado

putchar & getchar

```c
#include <stdio.h>

int main(void)
{
    char c1, c2;
    scanf("%c", &c1);
    printf("%c\n", c1);
    c2 = getchar(); // ler um caractere
    putchar(c2);    // escrever um caractere
    putchar('\n');
    return 0;
}
```

Observação

- Ver `ctypes.h`
- `man getchar`
- `man putchar`

Biblioteca padrão

```c
#include <stdio.h>

int main(void)
{
    char v1[6], v2[6], v3[6];

    /* Better */
    printf("Fgets\n");
    fgets(v1, 6, stdin);
    printf("%s\n", v1);

    /* ler enquanto não há espaço em branco */
    printf("Scanf\n");
    scanf("%s", &v2);
    printf("%s\n", v2);

    /* Não use! Deprecada! */
    printf("Gets\n");
    gets(v3);
    printf ("%s\n", v3);
}
```

Observações

- Ambas acrescentam um caractere nulo no final do arranjo ao terminar a leitura.
- Veja `string.h`

Referências

Precisões

[?] 7.8 (7.8.1, 7.8.2), 5.5

[?] 7

Perguntas ?

http://dimap.ufrn.br/~richard/dim0321