

31. Strings

DIM0321

2015.1

Outline

- 1 Introdução
- 2 Caracteres
- 3 Texto & Biblioteca padrão

DIM0321

31. Strings

2015.1 1 / 17

DIM0321

31. Strings

2015.1 2 / 17

- 1 Introdução
- 2 Caracteres
- 3 Texto & Biblioteca padrão

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'`.

DIM0321

31. Strings

2015.1 3 / 17

DIM0321

31. Strings

2015.1 4 / 17

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.

DIM0321

31. Strings

2015.1

5 / 17

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 }
```

DIM0321

31. Strings

2015.1

7 / 17

1 Introdução

2 Caracteres

3 Texto & Biblioteca padrão

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: '\"'

DIM0321

31. Strings

2015.1

8 / 17

Expressões

Equivalência

Caractere = número → podemos fazer operações sobre eles

```
1 #include <stdio.h>
2 int main( void )
3 {
4     char i;
5     char c = 'A' + 1; // 'B'
6     printf ("%c ", c);
7     c = 'A' < 'B' + 66; // 'D' = 1 + 66
8     printf ("%c ", c);
9     scanf ("%c", i);
10    for (i = 'A'; i <= 'Z'; ++i) printf ("%c ", i);
11    return 0;
12 }
```

DIM0321

31. Strings

2015.1

9 / 17

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.

Observação

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 é 0.

DIM0321

31. Strings

2015.1

11 / 17

1 Introdução

2 Caracteres

3 Texto & Biblioteca padrão

Exemplo

```
1 #include <stdio.h>
2 int main(void)
3 {
4     char texto[] = {
5         'H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '!', '\n',
6         '2', '4', '/', '0', '5', '/', '2', '0', '1', '3', '\0' };
7     int i = 0;
8     while(texto[i] != '\0') {
9         printf("%c", texto[i++]);
10    }
11    printf("\n");
12    printf("%s\n", texto);
13    return 0;
14 }
```

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

DIM0321

31. Strings

2015.1

12 / 17

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

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     char c1, c2;
6     scanf("%c", &c1);
7     printf("%c\n", c1);
8     c2 = getchar(); // ler um caractere
9     putchar(c2);    // escrever um caractere
10    putchar('\n');
11    return 0;
12 }
```

Observação

- Ver ctype.h
- man getchar
- man putchar

DIM0321

31. Strings

2015.1 13 / 17

Biblioteca padrão

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     char v1[6], v2[6], v3[6];
6
7     /* Better */
8     printf("Fgets\n");
9     fgets(v1, 6, stdin);
10    printf("%s\n", v1);
11
12    /* ler enquanto não há espaço em branco */
13    printf("Scanf\n");
14    scanf("%s", &v2);
15    printf("%s\n", v2);
16
17    /* Não use! Deprecada! */
18    printf("Gets\n");
19    gets(v3);
20    printf ("%s\n", v3);
21 }
```

Observações

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

DIM0321

31. Strings

2015.1 15 / 17

Referências

Precisões

[?] 7.8 (7.8.1, 7.8.2), 5.5

[?] 7

DIM0321

31. Strings

2015.1 16 / 17

Perguntas ?



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