

Estruturas de Dados

Módulo 7 – Cadeias de Caracteres

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO



Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel,
Introdução a Estruturas de Dados, Editora Campus
(2004)

Capítulo 7 – Cadeias de caracteres

Tópicos

- Caracteres
- Cadeias de caracteres
 - Leitura de caracteres e cadeias de caracteres
 - Exemplos de funções que manipulam cadeias de caracteres
 - Funções recursivas
 - Constante cadeia de caracteres
- Vetor de cadeias de caracteres

Caracteres

- tipo `char`:
 - tamanho de `char` = 1 byte = 8 bits = 256 valores distintos
 - tabela de códigos:
 - define correspondência entre caracteres e códigos numéricos
 - exemplo: ASCII
 - alguns alfabetos precisam de maior representatividade
 - alfabeto chinês tem mais de 256 caracteres

Códigos ASCII de alguns caracteres (sp representa espaço)

	0	1	2	3	4	5	6	7	8	9
30			sp	!	"	#	\$	%	&	'
40	()	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[\]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}	~			

Exemplo:

82	105	110	32	100	101	32	74	97	110	101	105	114	111
R	i	o		d	e		J	a	n	e	i	r	o

Códigos ASCII de alguns caracteres de controle

0	nul	<i>null</i> : nulo
7	bel	<i>bell</i> : campainha
8	bs	<i>backspace</i> : volta e apaga um caractere
9	ht	<i>tab</i> : tabulação horizontal
10	nl	<i>newline</i> ou <i>line feed</i> : muda de linha
13	cr	<i>carriage return</i> : volta ao início da linha
127	del	<i>delete</i> : apaga um caractere

Caracteres

- Constante de caractere:
 - caractere envolvido com aspas simples
 - exemplo:
 - 'a' representa uma constante de caractere
 - 'a' resulta no valor numérico associado ao caractere *a*

```
char c = 'a';  
printf("%d %c\n", c, c);
```

- printf imprime o conteúdo da variável c usando dois formatos:
 - com o formato para inteiro, %d, imprime 97
 - com o formato de caractere, %c, imprime *a* (código 97 em ASCII)

Cadeias de caracteres

- Representação de cadeia de caracteres:
 - vetor do tipo `char`, **terminado pelo caractere nulo ('\\0')**
 - é necessário reservar uma posição adicional no vetor para o caractere de fim da cadeia
 - função para manipular cadeias de caracteres:
 - recebe como parâmetro um vetor de `char`
 - processa caractere por caractere até encontrar o caractere nulo, sinalizando o final da cadeia

Cadeias de caracteres

- Inicialização de cadeias de caracteres:
 - caracteres entre aspas duplas
 - caractere nulo é representado implicitamente
 - Exemplo:
 - variável cidade dimensionada e inicializada com 4 elementos

```
int main ( void )
{
    char cidade[ ] = "Rio";
    printf("%s \n", cidade);
    return 0;
}
```

≡

```
int main ( void )
{
    char cidade[ ] = {'R', 'i', 'o', '\0'};
    printf("%s \n", cidade);
    return 0;
}
```

Cadeias de caracteres

- Exemplos:

```
char s1[] = "";  
char s2[] = "Rio de Janeiro";  
char s3[81];  
char s4[81] = "Rio";
```

s1 cadeia de caracteres vazia (armazena o caractere '\0')

s2 armazena cadeia de 14 caracteres (em vetor com 15 elementos)

s3 armazena cadeia com até 80 caracteres
dimensionada com 81 elementos, mas não inicializada

s4 armazena cadeias com até 80 caracteres
primeiros quatro elementos atribuídos na declaração {'R', 'i', 'o', '\0'};

Cadeias de caracteres

- Leitura de caracteres e cadeias de caracteres
 - através de `scanf`
 - especificadores de formato definem o comportamento do `scanf`

Cadeias de caracteres


- `scanf` com o especificador de formato `%c`
 - lê o valor de um único caractere fornecido via teclado
 - exemplo:

```
char a;  
...  
scanf("%c", &a);  
...
```

Cadeias de caracteres

- `scanf` com o especificador de formato `%c` (cont.):
 - não pula os “caracteres brancos”
 - “caractere branco” = espaço (' '), tabulação ('\t') ou nova linha ('\n')
 - se o usuário teclar um espaço antes da letra:
 - o código do espaço será capturado
 - a letra será capturada apenas na próxima chamada de `scanf`
 - para pular todos os “caracteres brancos” antes do caractere:
 - basta incluir um espaço em branco no formato, antes do especificador

```
char a;  
...  
scanf(" %c", %a); /* o branco no formato pula brancos da entrada */  
...
```



Cadeias de caracteres

- `scanf` com o especificador de formato `%s`
 - lê uma cadeia de caracteres não brancos
 - pula os eventuais caracteres brancos antes da cadeia
 - exemplo:

```
char cidade[81];  
...  
scanf("%s", cidade);  
...
```

- `&cidade` não é usada pois a cadeia é um vetor
- o código acima funciona apenas para capturar nomes simples
 - se o usuário digitar **Rio de Janeiro**, apenas **Rio** será capturada, pois `%s` lê somente uma seqüência de caracteres não brancos


Cadeias de caracteres

- `scanf` com o especificador de formato `%[...]`
 - `%[...]` lista entre os colchetes todos os caracteres aceitos na leitura
 - `%[^...]` lista entre os colchetes todos os caracteres **não** aceitos na leitura
 - exemplos:
 - `%[aeiou]`
 - lê seqüências de vogais
 - leitura prossegue até encontrar um caractere que não seja uma vogal
 - `%[^aeiou]`
 - lê seqüências de caracteres que não são vogais
 - leitura prossegue até encontrar um caractere que seja uma vogal

Cadeias de caracteres

- Exemplo:
 - lê uma seqüência de caracteres até que seja encontrado o caractere de mudança de linha ('\n')
 - captura linha fornecida pelo usuário até que ele tecle “Enter”
 - inclusão do espaço no formato garante que eventuais caracteres brancos que precedam a cadeia de caracteres sejam descartados

```
char cidade[81];  
...  
scanf(" %[^\\n]", cidade);  
...
```



Cadeias de caracteres

- Exemplos de funções para manipular cadeias de caracteres:
 - “comprimento”
 - “copia”
 - “concatena”
 - “compara”

Cadeias de caracteres

- Função “comprimento”:
 - retorna o comprimento de uma cadeia de entrada s
 - conta o número de caracteres até encontrar o caractere nulo
 - o caractere nulo em si não é contado

```
int comprimento (char* s)
{
    int i;
    int n = 0; /* contador */
    for (i=0; s[i] != '\0'; i++)
        n++;
    return n;
}
```

```
#include <stdio.h>

int comprimento (char* s)
{
    int i;
    int n = 0; /* contador */
    for (i=0; s[i] != '\0'; i++)
        n++;
    return n;
}

int main (void)
{
    int tam;
    char cidade[] = "Rio de Janeiro";
    tam = comprimento(cidade);
    printf("A string \"%s\" tem %d caracteres\n", cidade, tam);
    return 0;
}
```

Cadeias de caracteres

- Função “copia”:
 - copia os elementos de uma cadeia de origem (orig) para uma cadeia de destino (dest)
 - cadeia de destino deverá ter espaço suficiente

```
void copia (char* dest, char* orig)
{
    int i;
    for (i=0; orig[i] != '\0'; i++)
        dest[i] = orig[i];
    /* fecha a cadeia copiada */
    dest[i] = '\0';
}
```

Cadeias de caracteres

- Função “concatena”:
 - copia os elementos de uma cadeia de origem (orig) para o final da cadeia de destino (dest)

```
void concatena (char* dest, char* orig)
{
    int i = 0; /* índice usado na cadeia destino, inicializado com zero */
    int j;     /* índice usado na cadeia origem */
    /* acha o final da cadeia destino */
    i = 0;
    while (dest[i] != '\0')
        i++;
    /* copia elementos da origem para o final do destino */
    for (j=0; orig[j] != '\0'; j++)
        { dest[i] = orig[j]; i++; }
    /* fecha cadeia destino */
    dest[i] = '\0';
}
```

Cadeias de caracteres

- Função “compara”:
 - compara, caractere por caractere, duas cadeias dadas
 - usa os códigos numéricos associados aos caracteres para determinar a ordem relativa entre eles
 - valor de retorno da função:
 - 1 se a primeira cadeia preceder a segunda
 - 1 se a segunda preceder a primeira
 - 0 se ambas as cadeias tiverem a mesma seqüência de caracteres

```

int compara (char* s1, char* s2)
{
    int i;
    /* compara caractere por caractere */
    for (i=0; s1[i]!='\0' && s2[i]!='\0'; i++) {
        if (s1[i] < s2[i])
            return -1;
        else if (s1[i] > s2[i])
            return 1;
    }
    /* compara se cadeias têm o mesmo comprimento */
    if (s1[i]==s2[i])
        return 0;      /* cadeias iguais */
    else if (s2[i]!='\0')
        return -1;     /* s1 é menor, pois tem menos caracteres */
    else
        return 1;      /* s2 é menor, pois tem menos caracteres */
}

```

Cadeias de caracteres

- Biblioteca de cadeias de caracteres [string.h](#)

“comprimento”	strlen
“copia”	strcpy
“concatena”	strcat
“compara”	strcmp

Cadeias de caracteres

- Função “duplica”:
 - copia os elementos de uma cadeia de origem (s) para uma cadeia de destino (d), alocada dinamicamente

```
include <stdlib.h>
#include <string.h>

char* duplica (char* s)
{
    int n = strlen(s);
    char* d = (char*) malloc ((n+1)*sizeof(char));
    strcpy(d,s);
    return d;
}
```

Cadeias de caracteres

- Funções recursivas para manipular cadeias de caracteres:
 - baseiam-se em uma definição recursiva de cadeias de caracteres:

Uma cadeia de caracteres é:

 - *a cadeia de caracteres vazia; ou*
 - *um caractere seguido de uma cadeia de caracteres*

Cadeias de caracteres

- Implementação recursiva da função “comprimento”:

```
int comprimento_rec (char* s)
{
    if (s[0] == '\0')
        return 0;
    else
        return 1 + comprimento_rec(&s[1]);
}
```

Cadeias de caracteres

- Constante cadeia de caracteres:
 - representada por seqüência de caracteres delimitada por aspas duplas
 - comporta-se como uma expressão constante, cuja avaliação resulta no ponteiro para onde a cadeia de caracteres está armazenada

Cadeias de caracteres

- Exemplo:

```
#include <string.h>

int main ( void )
{
    char cidade[4];
    strcpy (cidade, "Rio" );
    printf ( "%s \n", cidade );
    return 0;
}
```

- quando a cadeia "Rio" é encontrada:
 - uma área de memória é alocada com a seqüência de caracteres: 'R', 'i', 'o', '\0'
 - o ponteiro para o primeiro elemento desta seqüência é devolvido
- função **strcpy** recebe dois ponteiros de cadeias:
 - o primeiro aponta para o espaço associado à variável cidade
 - o segundo aponta para a área onde está armazenada a cadeia constante **Rio**

Cadeias de caracteres

- Exemplo:
 - (código equivalente ao anterior)

```
int main (void)
{
    char *cidade; /* declara um ponteiro para char */
    cidade = "Rio"; /* cidade recebe o endereço da cadeia "Rio" */
    printf ( "%s \n", cidade );
    return 0;
}
```

Cadeias de caracteres

- Exemplos:

```
char s1[] = "Rio de Janeiro";
```

- s1 é um vetor de char, inicializado com a cadeia **Rio de Janeiro**, seguida do caractere nulo
- s1 ocupa 15 bytes de memória
- é válido escrever s1[0]='X', alterando o conteúdo da cadeia para **Xio de Janeiro**, pois s1 é um vetor, permitindo alterar o valor de seus elementos

```
char* s2 = "Rio de Janeiro";
```

- s2 é um ponteiro para char, inicializado com o endereço da área de memória onde a constante **Rio de Janeiro** está armazenada
- s2 ocupa 4 bytes (espaço de um ponteiro)
- não é válido escrever s2[0]='X', pois não é possível alterar um valor constante

Vetor de cadeia de caracteres

- Alocação de vetor de cadeia de caracteres:
 - alocação estática:
 - alocação como matriz estática de elementos do tipo char
 - alocação dinâmica:
 - alocação como vetor de ponteiros
 - cada cadeia de caracteres (elemento do vetor) é alocada dinamicamente

Vetor de cadeia de caracteres

- Exemplo:
 - função para imprimir os nomes dos alunos de turmas com 50 alunos, onde o nome dos alunos possui no máximo 80 caracteres

```
char alunos[50][81];  
...  
  
void imprime (int n, char alunos[][81])  
{  
    int i;  
    for (i=0; i<n; i++)  
        printf("%s\n", alunos[i]);  
}
```

(`alunos[i][j]` acessa a (j+1)-ésima letra do nome do (i+1)-ésimo aluno)

Vetor de cadeia de caracteres

- Exemplo:
 - função para capturar os nomes dos alunos de uma turma
 - lê o número de alunos da turma
 - captura os nomes fornecidos por linha, fazendo a alocação correspondente
 - usa função auxiliar que captura uma linha e fornece como retorno uma cadeia alocada dinamicamente com a linha inserida

Vetor de cadeia de caracteres

- Exemplo (cont.):
 - função “lelinha”
 - captura uma linha
 - retorna uma cadeia alocada dinamicamente com a linha inserida
 - usa a função “duplica”

```
char* lelinha (void)
{
    char linha[121];    /* variável auxiliar para ler linha */
    printf("Digite um nome: ");
    scanf(" %120[^\n]",linha);
    return duplica(linha);
}
```

Vetor de cadeia de caracteres

- Exemplo (cont.):
 - função “lenomes”
 - captura os nomes dos alunos e preenche o vetor de nomes
 - retorna o número de nomes lidos

```
int lenomes (char** alunos)
{ int i, n;
  do { printf("Digite o numero de alunos: ");
        scanf("%d",&n);
        } while (n>MAX);
  for (i=0; i<n; i++)
    alunos[i] = lelinha();
  return n;
}
```

Vetor de cadeia de caracteres

- Exemplo (cont.):
 - função “liberanomes”
 - liberar os nomes alocados na tabela

```
void liberanomes (int n, char** alunos)
{
    int i;
    for (i=0; i<n; i++)
        free(alunos[i]);
}
```

Vetor de cadeia de caracteres

- Exemplo (cont.):
 - função “imprimenomes”
 - imprime os nomes dos alunos

```
void imprimenomes (int n, char** alunos)
{
    int i;
    for (i=0; i<n; i++)
        printf("%s\n", alunos[i]);
}
```

Vetor de cadeia de caracteres

- Exemplo (cont.):
 - programa principal

```
#define MAX 50

int main (void)
{
    char* alunos[MAX];
    int n = lenomes(alunos);
    imprimenomes(n,alunos);
    liberanomes(n,alunos);
    return 0;
}
```

Vetor de cadeia de caracteres

- Parâmetros da função `main`:
 - `main` pode ser definida para receber zero ou dois parâmetros:

```
int main (int argc, char** argv)
{
    ...
}
```

- `argc`:
 - recebe o número de argumentos passados para o programa quando este é executado
- `argv`:
 - vetor de cadeias de caracteres que armazena os nomes passados como argumentos

Vetor de cadeia de caracteres

- Exemplo:

- declaração da função main (no arquivo “teste.c”):

```
int main (int argc, char** argv) { ... }
```

- geração do programa executável, com o nome “mensagem”:

```
c:\Program Files\INF1620\PROGRAMAS\gcc teste.c -o mensagem
```

- chamada ao através da linha de comando:

```
c:\Program Files\INF1620\PROGRAMAS\mensagem estruturas de dados
```

- `argc` receberá o valor 4
- `argv` será inicializado com os seguintes elementos:
 `argv[0]`="mensagem" `argv[1]`="estruturas"
 `argv[2]`="de" `argv[3]`="dados"

Resumo

caracteres

- representados pelo tipo `char`, com o auxílio de tabela de códigos (de caracteres)

cadeia de caracteres

- representada por vetor do tipo `char`, terminada pelo caractere nulo (`'\0'`)

inicialização de cadeia de caracteres

- caracteres entre aspas duplas

Resumo

leitura de caracteres e cadeias de caracteres

– através de `scanf` com especificadores de formato

<code>%c</code>	lê o valor de um único caractere fornecido via teclado não pula os “caracteres brancos”
<code>%s</code>	lê uma cadeia de caracteres não brancos pula os eventuais caracteres brancos antes da cadeia
<code>%[...]</code>	lista entre os colchetes todos os caracteres aceitos na leitura
<code>%[^...]</code>	lista entre os colchetes todos os caracteres não aceitos

Resumo

Biblioteca de cadeias de caracteres [string.h](#)

“comprimento”	strlen
“copia”	strcpy
“concatena”	strcat
“compara”	strcmp