

Métodos Computacionais



Strings (Vetor de Caracteres)

Caracteres

Em C, o tipo *char* :

- é usado para representar caracteres
- pode armazenar valores inteiros (em 1 byte), representando assim, 256 valores distintos
- Uma constante *char* é escrita entre aspas simples

```
char letraA = 'A';  
char letraC;  
letraC = 'C';  
printf ( " %c %c ", letraA , letraC) ;
```

Caracteres

- ◆ São representados internamente na memória do computador por códigos numéricos
 - A correspondência entre os caracteres e os seus códigos numéricos é feita por uma tabela ASCII
 - Na tabela ASCII:
 - os dígitos são codificados em seqüência
 - as letras minúsculas e maiúsculas também formam dois grupos sequenciais

```
char letraA = 65; /* letra A*/  
char letraC;  
letraC = 67;  
printf ( "%c %c ", letraA , letraC) ;
```

Tabela ASCII

	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	{		}	~			

Impressão de Caracteres

- ◆ Podem ser impressos de duas formas diferentes usando o *printf*:

```
char lc = 97 ;  
printf( "%d %c" ,lc,lc);
```

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

Saída: 97 c

Saída: 95 a

- ◆ Existe a função *putchar* da biblioteca `stdio.h` que permite a impressão de um caractere

```
char la = 'a'; /* ou la = 97; */  
putchar(la);
```

Leitura de Caracteres

- ◆ Leitura de caracteres com a função *scanf*

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

OU

```
char a ;  
/* sem brancos */  
scanf ( " %c" , &a ) ;
```

- ◆ Existe a função *getchar* da biblioteca `stdio.h` que permite a leitura de um caractere

```
char a ;  
a = getchar ( ) ;
```

Leitura de Caracteres

- ◆ Função *scanf* e *getchar* obriga que a tecla <enter> seja pressionada após a entrada dos dados
- ◆ Existem funções para ler dados sem esperar pelo <enter> em C **para ambientes Windows:**
 - Função *getche* – definida em *conio.h*
 - Lê **um** caractere e o exibe na tela

```
char letra ;  
letra = getche( );
```

- Função *getch* – definida em *conio.h*
 - Lê **um** caractere e **não** o exibe na tela (invisível)

```
char letra ;  
letra = getch( );
```

Escrevendo Funções que Manipulam Caracteres

- ◆ Pode-se tirar proveito da codificação seqüencial da tabela ASCII
 - Escrevendo programas que usam a tabela
 - A função abaixo verifica se um dado caractere é um dígito entre '0' e '9'

```
int digito (char c){
    int ehDigito;
    if(( c >= '0')&&(c <= '9')) {
        ehDigito = 1;
    } else{
        ehDigito = 0;
    }
    return ehDigito;
}
```


Escrevendo Funções que Manipulam Caracteres

- ◆ Função para converter uma letra em maiúscula

```
char maiuscula(char c) {  
    char maiusc;  
    if((c >= 'a') && (c <= 'z')) {  
        maiusc = c - 'a' + 'A';  
    }  
    return maiusc;  
}
```

Diferença entre qualquer caracter minúsculo e a letra 'a' é a mesma do equivalente maiúsculo e a letra 'A'

Vetor de Caracteres (String)

- ◆ É representada por um vetor do tipo *char* e terminada **obrigatoriamente**, pelo **caractere nulo** ‘\0’
- ◆ O especificador de formato %s da função *printf* permite imprimir uma cadeia de caracteres
- ◆ A partir do endereço para o primeiro caractere, as funções processam caractere a caractere até que ‘\0’ seja encontrado

```
int main(){
    char cidade[4];
    cidade[0]='R';
    cidade[1]='I';
    cidade[2]='\0';
    cidade[3]='\0';
    printf("%s",cidade);
}
```

Inicialização de Strings

- ◆ Inicialização do vetor de caracteres na declaração

```
int main(){  
    char cidade[]={ 'R', 'I', 'O', '\0' } ;  
    printf ( "%s\n", cidade );  
}
```

- ◆ Inicialização do vetor na declaração através da escrita dos caracteres entre aspas duplas

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

Caractere nulo
é representado
implicitamente

Declarando Strings

```
char s1[];
```

↳ Consiste de uma cadeia vazia, representando um vetor com apenas o caracter nulo (' \0 ')

```
char s2[] = "Rio de Janeiro";
```

↳ Representa um vetor com 15 elementos

```
char s3[81];
```

↳ Representa um vetor de no máximo, 80 elementos

```
char s4[81] = "Rio";
```

↳ Representa um vetor de no máximo 80 elementos, mas com um valor inicial já atribuído

Constantes do Tipo String

```
printf("Uma string constante!\n");  
printf("Eu moro em %s ", "Recife");
```

106	\0
105	e
104	f
103	i
102	c
101	e
100	R

São criados strings
contantes na memória

Constantes do Tipo String

ERRADO

```
char capital[7];  
capital = "Recife";
```

Tentativa de atribuir endereço da constante "Recife" à constante capital

Declaração da constante do tipo vetor de caracteres capital

Já foi atribuído um endereço à constante capital (endereço inicial do vetor)

Constantes do Tipo String

Declaração da constante do tipo vetor de caracteres capital

Vetor de caracteres inicializado com as letras que fazem parte de Recife

CORRETO

```
char capital[7] = "Recife";
```

A constante capital armazena o valor inicial da String (100)

106	\0
105	e
104	f
103	i
102	c
101	e
100	R

Constantes do Tipo String

Declaração da
constante do tipo
vetor de caracteres
capital

Vetor de caracteres
inicializado com as letras
que fazem parte de Recife

CORRETO

```
char capital[7] = "Recife";  
char* cidade;  
cidade = "Recife";
```

Atribuição do
endereço da
constante "Recife"
à variável cidade

Leitura de Strings

- ◆ Especificador %s na função *scanf* captura somente uma seqüência de caracteres não brancos

- **Limitação:** somente nomes simples podem ser lidos

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

& não é necessário pois cidade já armazena um endereço (endereço inicial do vetor)

- Um caracter branco pode ser um:
 - espaço (' ')
 - caractere de tabulação (' \t ')
 - caractere de nova linha (' \n ')

Permitindo Ler Mais de um Nome com o *scanf*

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

O caracter ^ informa
que o caracter \\n
não pode ser lido

- ◆ A função acima lê uma seqüência de caracteres até que seja digitado um <enter>
- ◆ A inclusão do espaço antes de % descartam espaços em brancos que precedem o nome

Permitindo Ler Mais de um Nome com o *scanf*

- ◆ Para limitar o número máximo de caracteres capturados:

```
char cidade [81] ;  
scanf ( " %80[^\n]" , cidade ) ;
```



No máximo, 80 caracteres são lidos

Funções de Manipulação de Strings

```
void    imprime (char[] s) {  
    int i;  
    for (i = 0; s[i] != '\0'; i++) {  
        printf ("%c",s[i]) ;  
    }  
    printf ("\n");  
}
```

Imprime caracter a
caracter

● Função análoga

```
void    imprime (char[] s) {  
    printf ("%s",s);  
    printf ("\n");  
}
```

Funções de Manipulação de Strings

- Calcula o comprimento da cadeia

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

- Função análoga definida em string.h:

strlen (**char* str**) ;

Equivalente a char[]

Funções que Fazem Cópias de Strings

```
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';  
}
```

Funções que manipulam Strings
assumem que toda String
termina com o '\0'

- Função análoga definida em `string.h`:

```
strcpy (char* dest , char* orig) ;
```

- Copia os elementos do 2º parâmetro no 1º
- Supõe que o 2º parâmetro tem espaço suficiente

Copiando Strings

ERRADO

```
char capital[7];  
capital = "Recife";
```

Não se usa atribuição para copiar uma String na constante do tipo String capital

CORRETO

```
char capital[7];  
char cidade[7];  
strcpy(capital, "Recife");  
strcpy(cidade, capital);
```

Para copiar Strings deve-se utilizar uma função que faz a cópia!

Funções que Concatenam Strings

```
void concatena (char[] dest, char[] orig) {  
    int i = 0 ;    int j ;  
    while (dest[i] != '\\0') {  
        i++ ;  
    }  
    for (j = 0; orig[j] != '\\0 ' ; j++){  
        dest[i] = orig[j];  
        i++;  
    }  
    dest[i] = '\\0';  
}
```

Acha o final da
String destino

Copia elementos

Fecha a String
destino

- Função análoga definida em string.h:

```
strcat (char* dest , char* orig) ;
```

- Concatena as duas cadeias e o resultado é atribuído ao 1º parâmetro

Funções que Comparam Cadeias 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;
    }
    if (s1[i] == s2[i]) return 0 ;
    else if (s2[i] != '\0')
        return -1 ; /* s1 é menor */
    else return 1 ; /* s2 é menor */
}
```

- Função análoga definida em string.h:

```
int strcmp (char* s1 , char *s2) ;
```

Outras Funções para Strings

◆ Definidas em *string.h*:

- *strcmp(char *str1, char *str2);*

- Retorna um inteiro positivo se *s1* é lexicamente posterior que *s2*; zero se as duas são idênticas; e negativo se *s1* é lexicamente anterior que *s2*

- *strncpy(char *dest, char *origem, int n)*

- Copia *n* caracteres de origem para destino

- *strncat(char *dest, char *origem, int n);*

- Concatena *n* caracteres da origem em destino

Vetor de Strings

- ◆ Vetor de Strings equivale a um vetor de vetores
 - Matriz
 - Cada linha da matriz corresponde a uma string
- ◆ Útil quando queremos armazenar uma coleção de strings

Exemplo de Vetor de Strings

```
#define    MAX    50 ;
int  main (){
    int  i , numAlunos ;
    char alunos[MAX][121] ;
    do {
        printf( "Digite o numero de alunos:\n" ) ;
        scanf ("%d",&numAlunos);
    } while ( numAlunos  >  MAX );
    for (i = 0; i < numAlunos; i++) {
        gets(alunos[i]) ; /* Lê uma string*/
    }
    return  0 ;
}
```

Cada posição do vetor
guarda uma String