Programação 2

Cadeias de caracteres

Rivera

Caracteres

- Tipo char
  - Sizeof (char) = 1 byte
    - 256 tipos de caracteres
  - Tabela de códigos
    - Correspondência entre caractere e códigos numéricos
      - ASCII
    - Alguns alfabetos precisam de maior representatividade
      - Alfabeto chinês (mais de 256 caracteres)

Um caractere ocupa 1 byte de espaço, e existem, dentro da tabela ASCII, 256 tipos de caracteres, entretanto alguns alfabetos requerem mais do que isso.

Código ASCII (alguns caracteres)

	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

A tabela ASCII assimila 256 caracteres a 256 números, tornando possível montar palavras e frases apenas com códigos numéricos

Proveito de Caracteres Representados de Forma sequencial na tabela ASCII

```
char maiuscula(char c)
{
   // Verifica de é letra minúscula

   if (c >= ‘a’ && c <= ‘z’)
      c = (c – ‘a’) + ‘A’;

   return c;
}
```

O código acima, visualmente, utiliza apenas letras para sua manipulação, entretanto, 'a', 'z' e 'A' são representações de seus respectivos números da tabela ASCII, por isso é possível realizar operações como a acima, que caso o caracter seja uma letra do alfabeto, subtrai-se o código do número que dá inicio a letras minúsculas, e adiciona-se o número correspondente ao A maiúsculo, retornando o caracter original em sua forma maiúscula.

Cadeias de Caracteres

- Representação de cadeias de caracteres
  - Vetor do tipo CHAR
    - Termina em caractere nulo ('\0')
    - É necessário reservar uma posição adicional para caractere de fim da cadeia
  - Funções
    - Parâmetro um vetor CHAR
    - Processa caractere por caractere até encontrar o caractere nulo

Vetores do tipo caracter são processados linearmente, dado por dado, até o fim, que é, necessariamente, um caractere nulo

Cadeias de Caracteres

- Inicialização de cadeias de caracteres
  - Caracteres entres aspas duplas
  - Caractere nulo é representado implicitamente
  - Exemplo
    - Variável CIDADE inicializada com 4 elementos

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

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

cadeias de caracteres podem ser iniciadas com os dados em sequência, dentro de aspas duplas, ou separados, em aspas simples. O caractere nulo pode ser declarado implicitamente.

Cadeias de Caracteres

- Exemplos

```
Char  s1[ ] = “ ”;
Char  s2[ ] = “Rio de Janeiro”;
Char  s3[81];
Char  s4[81] = “Rio”;
```

S1: cadeia vazia (apenas armazena o caractere ‘\0’)

S2: cadeia de 14 caracteres em um vetor de 15 elementos (último ‘\0’)

S3: para cadeia com 80 caracteres e 81 elementos (último ‘\0’)

S4: para cadeia de 80 caracteres, mas só 4 são usados

Uma cadeia vazia possui apenas o caracter nulo, uma cadeia de caracteres declarada sempre possui +1 caractere nulo implícito, e uma vez iniciada com um número específico de caracteres, uma cadeia não precisa ser completamente preenchida, porém mantém seu tamanho original.

Cadeias de Caracteres

- Exemplos

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

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

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

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

diferentes implementações para manipular cadeias de caracteres, imprimindo, copiando e medindo seu comprimento, por meio de loops que percorrem os caracteres um a um dentro da cadeia.

Cadeias de Caracteres

- Exemplos

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

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

   for (j=0; orig[j] != '\0'; j++) {
     dest[i] = orig[j]; i++;
   }
   dest[i] = '\0';
}
```

percorre-se a primeira cadeia até o final, e
adiciona-se cada caracter da segunda cadeia, um a um,
concatenando-os.

Cadeias de Caracteres

- Exemplos
  - -1: 1ª precede 2ª; 1: 2ª precede a 1ª; 0: ambas mesma seq.

```
void compara (char* s1, char* s2)
{
   int i;
   // compara
   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 (menos caracteres)
   else
     return 1;    // s2 é menor (menos caracteres)
}
```

contador de caracteres que percorre ambas cadeias com mesmo index, até que se ache a maior cadeia, ou ambas possuem o mesmo tamanho.

Cadeias de Caracteres

- Biblioteca de cadeias de caracteres string.h
  - "comprimento" strlen
  - "copia" strcpy
  - "concatena" strcat
  - "compara" strcmp

biblioteca string.h implementa módulos
de funções exemplificadas anteriormente,
já prontas.

Trabalho

- Escreva um programa que calcule a frequencia de cada símbolo que aparecem no texto *.txt seguinte:

A ciencia da computacao busca construir uma base cientifica para uma diversidade de topicos, tais como a construcao e a programacao de computadores, o processamento de informacoes, a solucao algoritmica de problemas e o estudo dos algoritmos propriamente ditos. Nesse sentido, estabelece os fundamentos para as aplicacoes computacionais existentes, assim como as bases para as futuras aplicacoes. Nao se pode, no entanto, aprender ciencia da computacao pelo simples estudo de alguns topicos isolados, ou pela utilizacao das ferramentas computacionais existentes. Mais do que entender a ciencia da computacao, é preciso comprender o escopo e a dinamica grande variedade de topicos de que trata esta importante disciplina.