**UEMG – Universidade do Estado de Minas Gerais – Unidade Frutal**

**Curso – Sistemas de Informação, 1o Período, Noturno**

**Disciplina – Introdução à Programação**

**Docente – Prof. Sérgio Carlos Portari**

**Discente – Luiz Felipe dos Santos Freitas**

**Trabalho – Matrizes e Vetores**

**Data – 27/08/2021**

**Matrizes e Vetores**

Vetores e Matrizes são estruturas de dados simples, que facilita o trabalho quando se tem muitas variáveis do mesmo tipo em um mesmo algoritmo.

O que diferencia vetores de matrizes é que vetores contém uma única dimensão, enquanto matrizes podem ter duas ou mais dimensões.

Exemplo disso é se precisarmos receber uma única nota de 100 pessoas, usando o vetor faríamos o código de forma mais rápida e organizada. No caso da matrizes, usaríamos se tivéssemos mais de uma nota por aluno. Supondo que precisamos calcular 4 notas dos 100 alunos, basicamente as matrizes teriam 100 posições e dentro de cada posição teria mais um vetor com 4 posições.

**String**

Na linguagem C uma string é um vetor de caracteres. Obrigatoriamente um dos caracteres do vetor deve ser o caractere nulo ('\0'). O caractere nulo vem após o último caractere válido, ou seja, se sua string usar 10 caracteres, no final serão ocupado 11 espaços. Na prática, as strings são usadas para representar textos. Para retratar uma string, devemos criar um vetor do tipo char.

Exemplo: char nome[10];Se inicializarmos a string de 10 posições, colocando a palavra Luiz, da seguinte forma: char nome[10] = “Luiz”; teremos então 4 caracteres destinados as letras “L”, “U”, “I” e “Z” e 1 destinado ao “\0”, o resto fica vazio.

**Declarando string**

Para declarar string em C é bem fácil, usa-se a mesma sintaxe de declarar um vetor de variáveis do tipo char.

Exemplo: char nome\_da\_string[tamanho];

Sempre lembrando de declarar o tamanho da string com um ou mais caracteres além dos que serão usados, pois não podemos nos esquecer do caractere nulo.

**Inicializando string**

Para inicializar uma string, basta apenas adicionar a frase/palavra desejada na frente da declaração, mas dentro de aspas. O caractere nulo é inserido automaticamente.

Exemplo: char nome\_da\_string[5] = “Luiz”;

**String na prática**

#include <stdio.h>

int main()

{

char nome[20] = "Luiz"; // Declarando e inicializando a string

int i;

printf("\nO programa ira mostrar a posicao do '\\0' na string ");

for(i=0 ; i < 20 ; i++) //O primeiro for mostra os caracteres da string Luiz, antes do '\0'

{

printf("%c",nome[i]);

}

for(i=0 ; nome[i] != '\0' ; i++) //O segundo for conta do 0 até encontrar o '\0',assim nos mostra em qual posição ele está

{}

printf("\n\nO '\\0' esta na posicao %d da string\n", i);

return 0;

}

**strcpy( )**

Essa função copia o valor da string na segunda posição do argumento para uma variável string na primeira posição. O segundo argumento pode ser uma variável, uma expressão string ou um valor literal string contido entre aspas.

É importante ressaltar que essa função não verifica se o destino tem tamanho suficiente para armazenar todos os caracteres presentes na fonte. Portanto o programador precisar fazer com que a string destino consiga encaixar todos os caracteres da string origem

**Sintaxe**: strcpy(string\_destino, string\_origem);

#include <stdio.h>

#include <string.h>

int main ()

{

char nome[20];

strcpy(nome, "Luiz Felipe");

printf("Nome: %s\n", nome);

return 0;

}

**strcat( )**

A função strcat junta uma string ao final de outra. Ao realizar o comando envolvendo a strcat, o valor de string2 é colocado ao final de string1. O primeiro parâmetro da função portanto deve ser uma variável e possuir o espaço suficiente para o resultado. A função não irá testar se existe espaço fazendo a movimentação de caracteres do segundo parâmetro para o final do primeiro.

**Sintaxe:** strcat (string1, string2);

#include <stdio.h>

#include <string.h>

int main ()

{

char string1[30] = "Ola,";

strcat(string1, " tudo bem?");

printf("%s\n", string1);

return 0;

}

**strchr( )**

Localiza a primeira ocorrência de um caractere especificado em uma string de origem e retorna um ponteiro para esse local. Strchr retorna um ponteiro nulo se o caractere especificado não foi encontrado.

**Sintaxe:** strchr(const char \*s, char ch);

#include <stdio.h>

#include <string.h>

int main ()

{

const char str[] = "String simples";

const char ch = 'm';

char \*p;

p = strchr(str, ch);

printf("String comecando de %c: %s\n", ch, p);

return 0;

}

**strcmp( )**

Essa função irá fazer a comparação de duas string e mostrar os seguintes resultados: 0 se as strings são iguais; < 0 se a string1 é menor que a string2 e > 0 se a string1 é maior do que a string2.

**Sintaxe:** strcmp(string1, string2);

#include <stdio.h>

#include <string.h>

int main ()

{

char string1[12] = "Luiz Felipe";

char string2[5] = "Luiz";

int res;

res = strcmp(string1, string2);

printf("Valor retornado: %d\n", res);

return 0;

}

**stricmp( )**

Essa função é usada para comparar duas strings sem diferenciar as letras maiúsculas e minúsculas. Se as strings forem iguais, ele retorna 0. Caso contrário, retorna um valor diferente de zero.

**Sintaxe:** stricmp(string1,string2);

#include <stdio.h>

#include <string.h>

int main ()

{

char string1[5] = "Luiz";

char string2[5] = "luiz";

int res;

res = stricmp(string1, string2);

printf("Valor retornado: %d\n", res);

return 0;

}

**strlen( )**

A função strlen retorna um valor inteiro, que é o comprimento da string, sem contar com o caractere nulo.

**Sintaxe:** strlen(string);

#include <stdio.h>

#include <string.h>

int main(void)

{

char palavra[50];

int caracteres;

printf("Digite uma palavra: ");

gets(palavra);

caracteres = strlen(palavra);

printf("\nA palavra digitada tem %d caracteres.\n", caracteres);

return 0;

}