Universidade Federal do Espirito Santo
Centro Tecnologico

{introcomp}

Working 07 : Vetores 2

Objetivos:

- Compreender a definição de strings e matrizes em C.

Prazo de Envio: sex, 28/10 04:00.

PET
Engenharia de Computação

1 STRINGS E SUAS IMPORTANCIAS

Imagine você precisando digitar seu nome para um formulário, e o computador pedindo para que primeiramente digite o número de caracteres a serem lidos. Isso complicaria uma atividade, aparentemente simples. Uma forma de contornar esse problema seria utilizar um caractere especial como marcador.
Strings são vetores de caracteres (vetores do tipo char) terminados por um caractere especial, o ” (barra zero), que indica o final da palavra ou texto. O uso de strings facilita a manipulação de palavras e textos.

```
int i =0;
char nome[50] = "Hello World";
while (nome[i] != '\0')
i++;
printf ("A frase \"Hello World\" tem %d caracteres \n", i);
```

Para ler e imprimir strings nas funções scanf e printf utilizamos o %s. Para definirmos uma constante do tipo string, utilizamos uma sequência de caracteres entre aspas. Já uma constante do tipo char é um caractere entre apóstrofes.
Exemplos de strings:

```
char nome[] = {'P', 'e', 'd', 'r', 'o', '\0'};
char nome[] = "Pedro";
```

1.1 Declaração de uma string

A declaração de uma string segue a seguinte sintaxe:

tipo nome_variavel[numero de elementos]

- **tipo:** tipo de dados para todos os elementos que compõem o vetor, no caso de char;
- **nome_variável:** nome do vetor;
- **numero de elementos:** valor constante indicando a quantidade de elemento do vetor;

Se temos um vetor com **n** elementos, seu índice varia sempre entre **0** e **n-1**. E cada posição do vetor pode ser acessada através do respectivo índice colocado entre colchetes [ ].

1.2 Escrita/Leitura de uma string

Para ler ou imprimir uma string do teclado usamos o operador especial %s.

```
char nome[10]
 scanf("%s", nome);
printf("\n %s", nome);
```

Exemplo:

```
int main()
{
char nome[30];
int idade;
printf("\nEntre com nome:");
scanf("%s", nome);
printf("\nEntre com idade:");
scanf("%d", &idade);
printf("\n Digitado: %s e %d\n", nome, idade);
}
```

IMPORTANTE: Para strings não é utilizado o operador & na função scanf.

2 STRINGS CONSIDERANDO ESPAÇOS OU QUEBRA DE LINHA

Por padrão, no scanf é feita a leitura da string até se encontrar um espaço ou caractere vazio. Como ler strings considerando espaços e quebras de linhas? Temos duas opções:

1. Utilizar a função de leitura **gets(string)**. A função gets lê uma string até uma quebra de linha, portanto, captura espaços.
2. Utilizar o especificador de extensão no **scanf**.

O especificador de extensão é utilizado no especificador de formato %s, bastando acrescentar um colchetes entre o '%' e o 's'. Com isso:

- %[^\n]s → efetua a leitura de qualquer caractere até que se encontre uma quebra de linha. O '^' designa uma negação, ou seja, não ler a quebra de linha.
- %[a-z]s → efetua a leitura de qualquer caractere enquanto o mesmo esteja na extensão especificada. Portanto, sem o '^', entende-se que deve-se efetuar a leitura enquanto os caracteres forem letras minúsculas, ou seja, estejam entre 'a' e 'z'.

3 BIBLIOTECA string.h

Incluindo no cabeçalho a biblioteca **string.h**, conseguimos comandos para manipulação de strings.

3.1 strcpy (abreviatura de string copy):

A função strcpy() copia a string-origem para a string-destino. Sua forma é:

strcpy(string_destino, string_origem);

Ao usar o **strcpy** o conteúdo antigo da **string_destino** é removido e sobreposto com o conteúdo da **string_origem**. A **string_origem** tem seu conteúdo preservado.

3.2 strlen (abreviatura de string length):

A função strlen() retorna o comprimento da string fornecida. Sua forma é:

strlen(string);

O terminador nulo não é contado, ou seja o comprimento do vetor da string deve ser um a mais que o inteiro retornado por **strlen()**.

3.3 strcmp (abreviatura de string compare):

A função strcmp() verifica se o conteúdo de duas strings são rigorosamente iguais, analisando tanto os valores atribuidos quanto a ordem destes valores. A comparação é realizada até se encontrar o conteúdo '\0' em ambos vetores. Sua forma é:

strcmp(string1, string2);

Se as duas strings forem idênticas a função retorna zero (FALSO), se elas forem diferentes a função retorna um valor:

- > 0, quando a string1 é maior que a string2
- < 0, quando a string1 é menor que a string2

Os valores para quando maior ou menor que zero são referentes à diferença da numeração ASCII do último caractere comparado. Por exemplo:

- strcmp("abc","abc"); // retorna 0
- strcmp("abc","abe"); // retorna 'c' - 'e' = -2
- strcmp("abf","aba"); // retorna 'f' - 'a' = 5

3.4 strcat:

A função **strcat()** une (concatena) uma string na outra. A string de origem permanecerá inalterada e será anexada ao fim da string de destino. Sua forma é:

strcat(string_destino, string_origem);

IMPORTANTE: Deve-se tomar cuidado não ultrapassar o tamanho da string ao somar dois conteúdos na string destino.

4 MATRIZ E SUA IMPORTÂNCIA

Uma matriz é uma coleção de variáveis de mesmo tipo, referenciada por nome comum. Para acessar um elemento específico em uma matriz, utilizamos índices. C armazena os elementos da matriz de forma sequencial na memória.

Uma matriz é um tipo de dado composto homogêneo na qual seus elementos são organizados em uma estrutura multidimensional. Uma matriz com m linhas e n colunas é chamada

matriz **m** por **n**.

Podemos citar como exemplo de matrizes de nosso dia a dia: um cartão de bingo, uma agenda de compromissos. Temos também matrizes com mais de três dimensões, porém é difícil visualizá-las, mas é possível declará-las em C.

Uma matriz é uma estrutura de dados que pode armazenar vários valores do mesmo tipo.

4.1 Matriz unidimensionais:

São matrizes de uma única dimensão. Já vimos estas, elas são chamadas de vetores. A declaração é da seguinte forma:

tipo nome_vetor[tamanho];

Como vimos o tamanho representa a quantidade de elementos que esta matriz irá conter. Lembrando que os índices começam do 0.

4.2 Matrizes bidimensionais:

São matrizes linha-coluna, onde o primeiro índice indica a linha e o segundo a coluna. Esse tipo de matriz é considerado o caso mais simples de matriz multidimensionais. A declaração é da seguinte forma:

tipo nome_matriz[numero_linhas][numero_colunas];

4.3 Matrizes multidimensionais

Podemos utilizar também matrizes de três ou mais dimensões, que requerem uma maior quantidade de memória e os acessos aos seus elementos são mais lentos. A declaração é da seguinte forma:

tipo nome [tamanho1][tamanho2][tamanho3]...[tamanho n];

5 INICIALIZAÇÃO DE UMA MATRIZ

Na declaração de uma matriz você deve ficar atento a inicialização das células, pois sua não inicialização pode gerar resultados inesperados. Observe o exemplo abaixo:

```
int main(){
    int i,j;
    int matriz[2][2] = {0};
    return 0;
}
```

Temos a inicialização de todos os elementos da matriz com o valor 0.

6 PREENCHENDO UMA MATRIZ

É muito importante ressaltar que, nesta estrutura, o índice da esquerda indexa as linhas e o da direita indexa as colunas. Abaixo temos um exemplo da matriz mtrx sendo preenchida:

```
#include <stdio.h>
int main()
{
    int mtrx[3][3];
    int i,j,count;
    cont = 1;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
        mtrx[i][j] = cont;
        cont++;
        }
    }
    return (0);
}
```

7 MANIPULAÇÃO DE UMA MATRIZ BIDIMENSIONAL

Podemos manipular os elementos de uma matriz bidimensional. Observe o código abaixo:

```
// manipulando uma matriz bidimensional
#include <stdio.h>
int main(void)
{
int pesos[3][5] = {{10,30,45,70,36},{86,44,63,82,80},{70,61,52,63,74}};
int linha,coluna;
for(linha=0;linha<3;linha++)
    for(coluna=0;coluna<5;coluna++)
    printf("elemento[%d][%d] = %d\n", linha+1, coluna+1, pesos[linha][coluna]);
return 0;
}
```

8 MATRIZES COMO PARÂMETROS DE FUNÇÃO

Vetores unidimensionais são declarados como parâmetros de função da seguinte forma:

```
<tipo> <nomedafuncao>(<tipo> <nomedovetor>[], <...>){
//codigo
}
```

Para matrizes bidimensionais, a passagem é declarada da seguinte maneira:

```
<tipo> <nomedafuncao>(<tipo> <nomedamatriz>[][<ncol>], <...>){
//c digo
}
```

Como pôde-se observar, na declaração de uma matriz como parâmetro de função é necessário especificar o número de colunas para a correta utilização. A explicação para essa obrigatoriedade abrange o conteúdo de ponteiros, tópico contido em um dos próximos Workings.

Praticando

Agora vamos praticar! Para todos os praticando em que é pedido que se escreva um programa, escreva o código do seu programa e nos envie o .c (código fonte) correspondente.

1. Dada uma lista de N nomes (apenas nomes simples com no máximo 10 caracteres, sem espaço) verificar qual seria o primeiro em ordem alfabética. A entrada inicia com um inteiro N que representa a quantidade de nomes lidos. Após esse inteiro, serão dados os N nomes. A entrada termina quando N = 0. Na saída, os nomes são seguidos por uma quebra de linha. Na saida do pragrama, cada numero é seguido de um espaço em branco.

Exemplo de Entrada	Exemplo de Saida
4 Bruno Carlos Andre Bernardo 6 Bruno Bruno Carlos Ricardo Pedro Joao 0	Andre Bruno

2. Considere duas matrizes quadradas NxN, denominadas A e B. Faça um programa para gerar a soma dessas duas matrizes. A entrada do programa inicia com um inteiro N que representará o tamanho das matrizes a serem lidas e as próximas 2*N linhas descreverão as matrizes de dimensão NxN. Após isso será dado um novo N, e assim sucessecivamente. O programa acaba quando N = 0.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, B = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}, SOMA = \begin{bmatrix} 2 & 6 & 10 \\ 6 & 10 & 14 \\ 10 & 14 & 18 \end{bmatrix}$$

Exemplo de Entrada	Exemplo de Saida
3 1 2 3 3 4 5 1 2 3 3 4 5 1 2 3 4 5 6 0	4 6 8 4 6 8 5 7 9

3. Faça um programa que leia uma lista de N nomes e formate os nomes lidos seguindo o seguinte padrão:

   (a) A primeira letra de cada nome começa com letra maiúscula. Sobrenomes também seguem este padrão.

   (b) As demais letras de cada nome devem ser minúsculas. Sobrenomes também seguem este padrão.

   A entrada do programa consiste de um numero inteiro N que informa a quantidade de nomes que serão lidos seguido de N nomes, um por linha. O programa termina quando N = 0.

Exemplo de Entrada	Exemplo de Saida
2 andre ferreira breno santos 2 lucas Silva lopES vitor 0	Andre Ferreira Breno Santos Lucas Silva Lopes Vitor

Desafios

1. **(Olímpiada Brasileira de Informática – 2005 – Nível 1, Fase 2)**

   Minhocas são muito importantes para a agricultura e como insumo para produção de ração animal. A Organização para Bioengenharia de Minhocas (OBM) é uma entidade não governamental que promove o aumento da produção, utilização e exportação de minhocas. Uma das atividades promovidas pela OBM é a manutenção de uma fazenda experimental para pesquisa de novas tecnologias de criação de minhocas. Na fazenda, a área destinada às pesquisas é de formato retangular, dividida em células quadrangulares de mesmo tamanho. As células são utilizadas para testar os efeitos, na produção de minhocas, de variações de espécies de minhocas, tipos de terra, de adubo, de tratamento, etc. Os pesquisadores da OBM mantêm um acompanhamento constante do desenvolvimento das minhocas em cada célula, e têm uma estimativa extremamente precisa da produtividade em cada uma das células. A figura abaixo mostra um mapa da fazenda, mostrando a produtividade estimada de cada uma das células.

   Um pesquisador da OBM inventou e construiu uma máquina colhedeira de minhocas, e quer testá- la na fazenda. A máquina tem a largura de uma célula, e em uma passada pelo terreno de uma célula colhe todas as minhocas dessa célula, separando-as, limpando-as e empacotando-as. Ou seja, a máquina eliminara uma das etapas mais intensivas de mão de obra no processo de produção de minhocas. A máquina, porém, ainda está em desenvolvimento e tem uma restrição: não faz curvas, podendo movimentar se somente em linha reta.

Decidiu-se então que seria efetuado um teste com a máquina, de forma a colher o maior número possível de minhocas em uma unica passada, em linha reta, de lado a lado do campo de minhocas. Ou seja, a máquina deve colher todas as minhocas de uma 'coluna' ou de uma 'linha' de células do campo de minhocas (a linha ou coluna cuja soma das produtividades esperadas das células é a maior possível).

Escreva um programa que, fornecido o mapa do campo de minhocas, descrevendo a produtividade estimada em cada célula, calcule o número esperado total de minhocas a serem colhidas pela máquina durante o teste, conforme descrito acima.

A primeira linha da entrada contém dois números inteiros N e M, representando respectivamente o número de linhas ($1 \leq N \leq 100$) e o número de colunas ($1 \leq M \leq 100$) de células existentes no campo experimental de minhocas. Cada uma das N linhas seguintes contém M inteiros, representando as produtividades estimadas das células correspondentes a uma linha do campo de minhocas.

A saída deve ser composta por uma unica linha contendo um inteiro, indicando o número esperado total de minhocas a serem colhidas pela máquina durante o teste.

Exemplo de Entrada	Exemplo de Saida
3 4 81 28 240 10 40 10 100 240 20 180 110 35	450