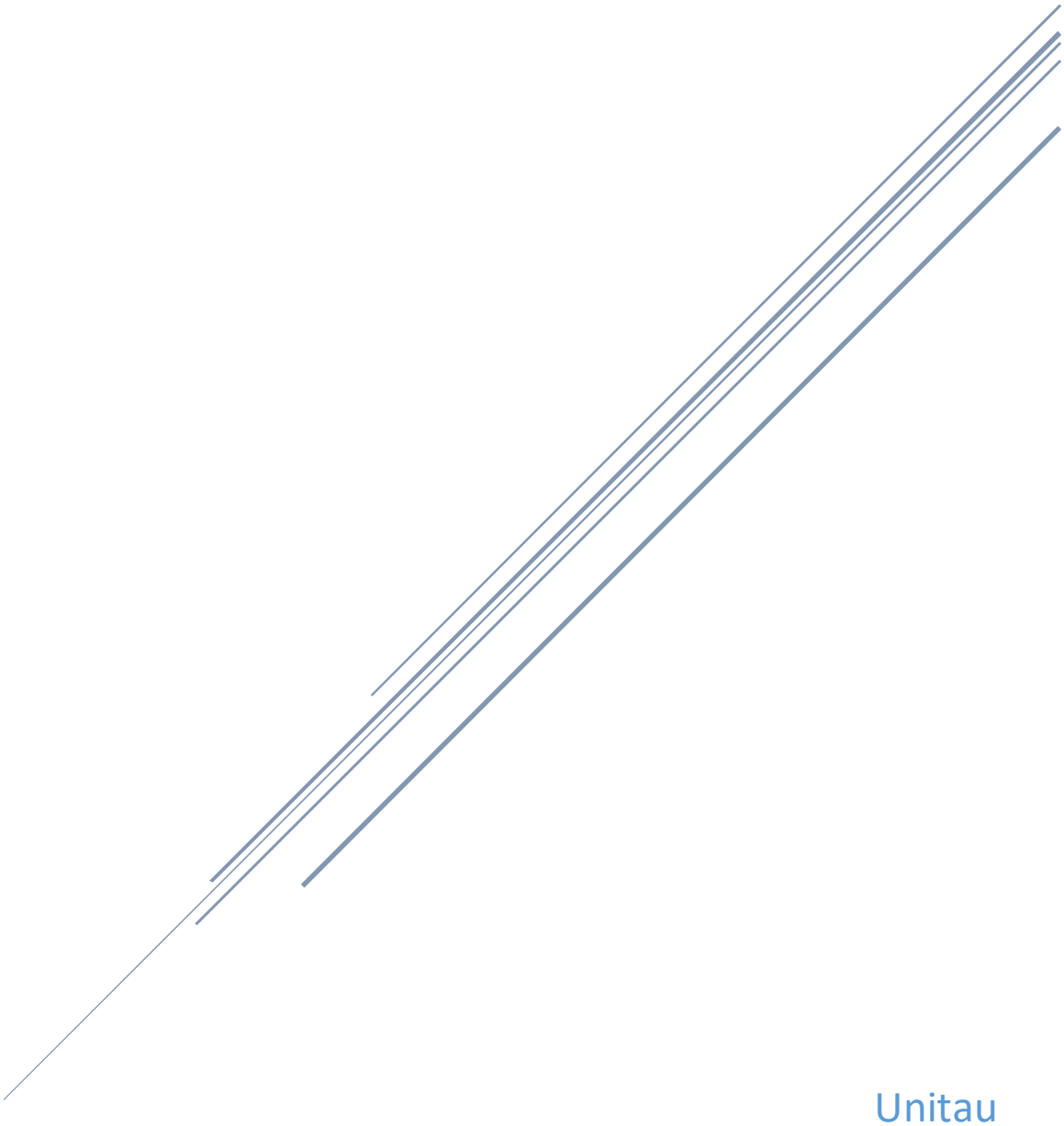
MATERIAL DE APOIO – MODULO 2

Seven Online Jugde



Unitau
Engenharia de Computação

Lista de figuras

Lista de tabelas

Sumário

Resumo

Este material visa auxiliar o aluno no aprendizado de programação assistido por computador. Neste modulo é apresentado ao estudante, funções e seus tipos, vetores e matrizes.

1 Introdução

Nesta apostila, no capítulo 2, você aprenderá funções, como chama-la, quais os seus parâmetros, funções do tipo *void* e funções que geram números aleatórios.

Enquanto que no capitulo 3, será apresentado vetores e como manipula-los.

Por fim, no capitulo 4, será mostrado as matrizes, como trabalhar com matrizes de uma dimensão e com matrizes bidimensionais, além de mostrar como percorrer a matriz e imprimir os seus dados.

2 Funções

Uma função é um conjunto de instruções desenhadas para cumprir uma tarefa particular e agrupadas numa unidade com um nome para referencia-la.

Funções dividem grandes tarefas de computação em tarefas menores, e permitem as pessoas trabalharem sobre o que outras já fizeram, em vez de partir do nada (Treinamento em linguagem C – 2ª edição, Victorine Viviane Mizrahi, 2008).

Cada função deve ser limitada a realizar uma única tarefa bem-sucedida, e o nome dela deve expressar essa tarefa (Eliane Cecilia Gatto, 2013).

2.1 Chamando uma função

Chamar uma função pode ser comparado a contratação de uma pessoa para a execução de um trabalho especifico. Algumas vezes a interação com essa pessoa é bem simples; outras vezes, mais complexa.

Este é o meio pelo qual solicitamos que o programa desvie o controle e passe a executar as instruções da função; e que, ao término desta, volte o controle para a posição seguinte a da chamada (Odair Soares, 2014).

No modulo 1, você já escreveu programas que chamam funções.

Várias funções, como **printf(), scanf()** e **system()**, são desenvolvidas por outros programadores e fornecidas pelo sistema (Bruno Anastacio, 2011). Outras funções podem ser escritas por nós mesmos.

Vamos a um exemplo de como chamar uma função.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int main(){
5       char nome;
6       printf("Escreva o seu nome: \n");
7       scanf("%c",&nome);
8       system("PAUSE");
9       return 0;
10  }
```

Figura 1 - Chamando uma função em C

No exemplo da figura 1, utilizamos 3 funções fornecidas pelos sistemas, que são elas: **printf(), scanf()** e **system().**

2.2 Função simples

Um programa em C, pode conter uma ou mais funções, das quais uma deve ser **main().** A execução do programa sempre começa em **main()**, e quando o controle do programa encontra uma nova instrução que inclui o nome de uma função, esta é chamada.

O código C que descreve o que a função faz é denominado *definição da função*. Sua forma geral é a seguinte:

Tipo nome (declaração dos parâmetros)
{
    Instruções;
}

2.2.1 Parâmetros de uma função

As variáveis que receberão as informações enviadas a uma função são chamadas parâmetros. A função deve declarar essas variáveis entre parênteses, no cabeçalho de usa definição ou antes das chaves que marcam o início da função. Os parâmetros podem ser utilizados livremente no corpo da função.

```
float celsius(fahr)
float fahr; // Declaração de parâmetros
{
    return (fahr - 50);
}
```

Figura 2 - Parâmetros de uma função

Na figura 2, é mostrado um exemplo de função, na primeira linha, a função é do tipo **float** e o seu nome é *celsius*, a qual recebe como parâmetro a variável *fahr.*

Note que na segunda linha, a variável *fahr* é declarada e dentro das chaves a mesma é utilizada para a realização de um cálculo.

2.3 Função que retorna nada

Em C, é possível criar funções que não retornam nenhum valor. Normalmente isto é feito quando queremos executar um bloco de comandos, mas estes comandos precisam retornar nada.

Para o exercício 3 deste módulo, é pedido que o usuário imprima uma frase colocando borda sólida.

Dentro do **printf**, o aluno deve colocar a seguinte linha:

printf("\xDB texto \xDB\n");

Ao passo que dentro da função do tipo **void**, o estudante também deve imprimir uma linha da seguinte forma:

printf("\xDB");

Agora, vamos a um exemplo de funções que não retornam nada.

```c
void SOMA(float a, int b)
{
    float result;
    result = a+b;
    printf("A soma de %.3f com %d é %.3f\n, a,b,Result);
}

int main()
{
     int a;
     float b;

     a = 10;
     b = 12.3;
     SOMA(b,a);  // Chamada da função

     return 0;
}
```

Figura 3 - Exemplo de função do tipo void

Na figura 3, nos é mostrado uma função do tipo ***void***, chamada **SOMA**, a qual, calcula a soma entre dois números. Note que, dentro da função main(), a função SOMA é chamada, e esta recebe os parâmetros **a** e **b** para cálculo da soma entre estes dois números.

2.4 Função que gera números aleatórios

A função **rand()** gera números aleatórios na faixa entre 0 e **RAND_MAX.** O valor de **RAND_MAX** é um valor que pode variar de máquina para máquina (C Progressivo, 2013).

Abaixo, vamos a um exemplo que gera 10 valores aleatórios.

```c
int main(void)
{
    int i;

    printf("Gerando 10 valores aleatorios:\n\n");

    for( i = 0; i < 10; i++)
    {
        printf("%d ", rand() );
    }

    system("PAUSE");
    return 0;
}
```

Figura 4 - Função que gera 10 números aleatórios

Na página anterior vimos um exemplo de um programa que gera 10 números aleatórios utilizando a função **rand().**

A cada iteração do loop, é gerado um número aleatório para composição da sequência de 10 números.

Iteração	Saída
i = 0	41
i = 1	18467
i = 2	6334
i = 3	26500
i = 4	19169
i = 5	15724
i = 6	11478
i = 7	29358
i = 8	26962
i = 9	24464

Tabela 1 - Iterações para gerar números aleatórios

3 Vetores

Vetores, também chamados ***arrays*** (do inglês) ou arranjo ou ainda matrizes, são uma maneira de armazenar vários dados num mesmo nome de variável através do uso de índices numéricos (Wikilivros, 2013). Tem como facilidade, a manipulação de um grande conjunto de dados do mesmo tipo declarando-se apenas uma variável.

```
int vetor[7] = {7, 14, 21, 28, 35, 42, 49};
```

Figura 5 - Exemplo de inicialização de um vetor

Você também pode declarar um vetor e não inicializa-lo, conforme exemplo abaixo:

```
int vetor[7];
```

Figura 6 - Declaração de um vetor

Há também a possibilidade de abreviar as declarações, podendo se omitir o número de elementos quando os valores são inicializados; o tamanho do vetor será o número de valores inicializados. Por exemplo, as duas notações abaixo são equivalentes.

```
int vetor[7] = {7, 14, 21, 28, 35, 42, 49};
int vetor[] = {7, 14, 21, 28, 35, 42, 49};
```

Figure 7 - Abreviando as declarações

Por fim, vamos a um exemplo de aplicação com vetores:

```
int main(){
    float notas[7] = {7, 7.7, 8.7, 9.7, 6.7, 7, 7.7};

    printf("Exibindo os valores do vetor: \n");
    printf("notas[0] = %.1f\n", notas[0]);
    printf("notas[1] = %.1f\n", notas[1]);
    printf("notas[2] = %.1f\n", notas[2]);
    printf("notas[3] = %.1f\n", notas[3]);
    printf("notas[4] = %.1f\n", notas[4]);
    printf("notas[5] = %.1f\n", notas[5]);
    printf("notas[6] = %.1f\n", notas[6]);

    system("PAUSE");
    return 0;
}
```

Figura 8 - Imprimindo valores do vetor

O mesmo exemplo escrito de outra forma, é mostrado na página seguinte.

```c
int main(){
    int i;
    float notas[7] = {7, 7.7, 8.7, 9.7, 6.7, 7, 7.7};

    printf("Exibindo os valores do vetor: \n");
    for(i = 0; i <= 6; i++){
        printf("notas[%d] = %.1f\n", i, notas[i]);
    }

    system("PAUSE");
    return 0;
}
```

Figura 9 - Imprimindo valores do vetor, utilizando laço for

4 Matrizes

Matriz é uma coleção de variáveis do mesmo tipo que compartilham um mesmo nome. São estruturas bidimensionais utilizadas para armazenar dados de um mesmo tipo (IMEUSP).

A matriz é um vetor com mais de uma dimensão, para declara-la, deve-se usar a seguinte construção:

Tipo_da_matriz nome_da_matriz [numero_linhas] [numero_colunas]

No exemplo acima vimos a sintaxe de declaração de uma matriz, na figura 10 veremos com se declara uma matriz utilizando a linguagem C.

```
int main()
{
    int matriz[][]; //declarando uma matriz em C

    int matriz1[7][7]; //declarando uma matriz de tamanho 7 por 7

    int matriz2[2][4] = {{1, 2, 3, 4}, {1, 2, 3, 4},{1, 2, 3, 4},{1, 2, 3, 4}} //iniciando valores na matriz
}
```

Figura 10 - Declarando matrizes em C

A figura 10 nos mostra 3 diferentes formas de se declarar matriz em C, na primeira linha é declarada uma matriz do tipo ***int*** e vazia, já na segunda linha, é declarada uma matriz com 7 linhas e 7 colunas, enquanto que na terceira linha, é declarada uma matriz com 2 linhas e 4 colunas e ela já está inicializada com valores.

Para atribuir valores a matriz, você deve colocar os valores entre chaves, como é mostrado abaixo:

float matiz [2][2] = {{2.7, 7.5}, {3.7, 5.7};

Note que acima, foi declarada uma matriz do tipo **float** de tamanho 2 por 2. E assim ficará a matriz:

A linha [0][0] = 2,7
A linha [0][1] = 7,5
A linha [1][0] = 3,7
A linha [1][1] = 5,7

OBS: Lembre-se que em C, a contagem começa em 0.

4.1 Matrizes de uma dimensão

As matrizes de uma dimensão são equivalentes aos vetores, sendo assim, quando for declarar uma matriz, lembre-se da forma que se declara um vetor, como mostra o exemplo abaixo:

```
int main()
{
    int matriz[7];
}
```

Figura 11 - Matriz de uma dimensão

4.2 Percorrendo e imprimindo matrizes

Para percorrer uma matriz, utilizamos o loop **for**, porém, como normalmente, uma matriz possui 2 dimensões, são necessários dois loops.

Vamos a um exemplo em que é percorrida uma matriz de tamanho n x n (lê-se n por n).

```
int main() {
int n;
int matriz[n][n]; //iniciando uma matriz de tamanho n x n

for(int i=0;i<n;i++) //loop para percorrer a linha
{
 for(int j=0;j<n;j++) //loop para percorrer a coluna
  {
  printf("%d", matriz[i[j]]); //imprime os valores da matriz
  }
}

}
```

Figura 12 - Percorrendo e imprimindo matrizes em C

A figura 12 nos mostra um exemplo de como se percorre e imprime-se uma matriz, primeiramente, o primeiro for vai percorrer cada linha da matriz.

Então, na primeira interação, ele entra na linha 0 da matriz, e em seguida entra no segundo loop, o qual começa com j =0 e vai até chegar em n, e cada iteração imprime-se o valor daquela posição da matriz.

A tabela abaixo mostra como é feita esta interação.

Iteração	Valor exibido na tela
i = 0	
j = 0	
printf matriz[0][0]	1
j = 1	
printf matriz[0][1]	2
j = 2	
printf matriz[0][2]	3
i = 1	
j = 0	
printf matriz[1][0]	4

Tabela 2 - Laço for na matriz

A Tabela 2 mostra como funciona a matriz, primeiro começa em i = 0, e, dentro deste loop existe outro loop, somente depois que este segundo loop terminar é que se incrementará 1 a i, e volta-se o processo com i = 1, j = 0.

Novamente, i só será incrementado depois que j atingir o valor de n, que é o limite estabelecido no loop for.

Referências Bibliográficas

[1] Só Matemática –**Vetores.** Disponível em: <http://www.somatematica.com.br/emedio/vetores/vetores6.php>. Acesso em: 06. Out 2017
[2] Elaine Cecilia Gatto – **Funções, 2013.** Disponível em: <https://pt.slideshare.net/elainececiliagatto/pc-funes>. Acesso em: 06. Out 2017
[3] Bruno Anastacio – **Linguagem C – Bibliotecas, Funções main, printf, scanf e mais – Drops I, 2011.** Disponível em: <https://blogdecodigo.wordpress.com/2011/07/12/linguagem-c-bibliotecas-funcoes-main-printf-scanf-e-mais-drops-i/>. Acesso em: 06. Out 2017
[4] Eduardo Casavella – **Funções em C, 2015.** Disponível em: <http://linguagemc.com.br/funcoes-em-c/>. Acesso em: 12. Out 2017
[5] **Programação em C – Uso de Funções.** Disponível em: <http://www.inf.pucrs.br/~pinho/LaproI/Funcoes/AulaDeFuncoes.htm>. Acesso em: 12. Out 2017
[6] **Gerando números aleatórios em C: *rand, srand e seed*.** Disponível em: <http://www.cprogressivo.net/2013/03/Como-gerar-numeros-aleatorios-em-C-com-a-rand-srand-e-seed.html>. Acesso em: 12. Out 2017
[7] Eduardo Casavella. **Valores aleatórios em C com a função rand, 2013.** Disponível em: <http://linguagemc.com.br/valores-aleatorios-em-c-com-a-funcao-rand/>. Acesso em: 12. Out 2017
[8] **Programar em C/Vetores, 2016.** Disponível em: <https://pt.wikibooks.org/wiki/Programar_em_C/Vetores>. Acesso em: 12. Out 2017
[9] Eduardo Casavella. **Vetores – arrays em linguagem C, 2015.** Disponível em: <http://linguagemc.com.br/vetores-ou-arrays-em-linguagem-c/>. Acesso em 12. Out 2017
[10] **Matrizes.** Disponível em: <https://www.ime.usp.br/~elo/IntroducaoComputacao/Matrizes.htm>. Acesso em 21. Out 2017
[11] C Progessivo.net. **Vetores multidimensionais, ou Matrizes: o que são e para que servem, 2013.** Disponível em: <http://www.cprogressivo.net/2013/03/Vetores-multidimensionais-Matrizes-em-C--vetor-de-vetores.html>. Acesso em 21. Out 2017
[12] Treinamento em Linguagem C 2ª Edição – Victorine Viviane Mizrahi. São Paulo. Editora Pearson. 2008.