APOSTILA DE ALGORITMOS

Algoritmo

O que é Algoritmo?

"... conjunto de comportamento que, obedecidos resultam em uma sucessão finita de ações ...", Harry Farrer

"...um processo de cálculo, ou de resolução de um grupo de problemas semelhantes...", Manzano J., Oliveira J.

Instruções com o objetivo de estabelecer um comportamento previsível

O que é Algoritmo Computacional?

"Na ciência da computação (informática) está associada a um conjunto de regras e operações bem definidas e ordenadas, destinadas à solução de um problema. ou de uma classe de problemas, em um número finito de passos.", Manzano J., Oliveira J.

Dados + Operações

Descrevendo Algoritmos

- Linguagem Natural: Os algoritmos são expressos diretamente em linguagem natural.
- Fluxograma Convencional: Esta é um representação gráfica que emprega formas geométricas padronizadas para indicar as diversas ações e decisões que devem ser executadas para resolver o problema.
- Pseudo-linguagem: Emprega uma linguagem intermediária entre a linguagem natural e uma linguagem de programação para descrever os algoritmos.

Exemplo

Calcule a área de uma mesa retangular

- Cálculo da área de uma mesa.
- Medir a largura da mesa e anotar o resultado.
- Medir o comprimento da mesa e anotar o resultado.
- Multiplicar o comprimento pela largura e anotar o resultado.
- O valor da área da mesa é o resultado anotado no passo anterior.

- Fim do cálculo da área da mesa.

Exercício

- Faça um algoritmo da matricula da faculdade
- Escreva um algoritmo em que uma pessoas saia do trabalho e chegue na sala de aula da faculdade
- Faça a some de dois números e imprima o resultado
- Faça um algoritmo que um aluno perca 20 kg

Linguagem C

Características

- Programas C são compilados e geram executáveis
- Estruturalmente mais simples e seu compilador gera códigos mais enxutos e velozes do que muitas linguagens
- Combina elementos de linguagens de alto nível com a funcionalidade da linguagem assembly, permitindo a manipulação de bits, bytes e endereços
- C é muito portável, pode ser recompilado em várias plataformas
- Não possui muitos tipos de dados mas permite conversão de quase todos os tipos
- C é uma linguagem estruturada , divisão do programa em módulos (funções)
- C é uma linguagem para programadores (criada, influenciada e testada por programadores)
- Os fabricantes de compiladores fornecem várias bibliotecas
- Oferece somente estruturas simples de controle de fluxos

Estrutura de um Programa C

Cabeçalho

Espaço do programa onde é informado as estruturas que serão utilizadas no decorrer do programa. Segue abaixo as estruturas mais utilizadas:

- Include: utilizado para importar funções pré-construídas localizadas em outro arquivo;
- Variáveis Globais: são variáveis que podem ser manipuladas em qualquer lugar do ser programa;
- Protótipos: são assinaturas das funções utilizadas no seu programa.

Exemplo

```
#include <stdio.h>

Int total;

Void calcular(int i, int j);

Main{

    …

}
```

Variáveis

A variável é o local onde se armazena informações que serão manipuladas pelo programa. As informações das variáveis são armazenadas em memória e a fim de facilitar o uso delas é dado um nome que representa o endereço de memória de onde está armazenada a variável.

As variáveis podem ser globais se forem declaradas no cabeçalho, locais se forem declaradas dentro de uma função e parâmetros formais se forem declaradas como parâmetros de uma função.

Declaração Variáveis

As variáveis são declaradas informando seu tipo seguido do nome.

Exemplo

```
Int total;
```

Atribuição

Quando se precisa informar um valor para uma variável utiliza-se o símbolo "=" para fazer a referência da atribuição.

Exemplo:

```
Total = 10;
```

```
Total = total + 15;
```

Printf

Comando utilizado para imprimir alguma informação no console.

Exemplo

```
Printf("Olá Mundo!");

Printf("Eu tenho %i reais", total);
```

Código	Significado
%c	Exibe um caractere
%d	Exibe um inteiro em formato decimal
%i	Exibe um inteiro
%e	Exibe um número em notação científica (com e minúsculo)
%E	Exibe um número em notação científica (com E maiúsculo)
%f	Exibe um ponto flutuante em formato decimal
%g	Usa %e ou %f, o que for menor
%G	O mesmo que %g, só que um E maiúsculo é usado se o formato %e for escolhido
%o	Exibe um número em notação octal
%s	Exibe uma string
%u	Exibe um decimal sem sinal
%x	Exibe um número em hexadecimal com letras minúsculas
%X	Exibe um número em hexadecimal com letras maiúsculas
%%	Exibe um sinal de %
%p	Exibe um ponteiro

Scanf

Comando utilizado para ler alguma informação do console e atribuir a uma variável.

Exemplo:

```
Scanf("%i", &total);
```

Código	Significado
%c	Lê um único caractere
%d	Lê um decimal inteiro
%i	Lê um decimal inteiro (não pode ser octal ou hexadecimal)
%u	Lê um decimal sem sinal
%e	Lê um número em ponto flutuante com sinal opcional
%f	Lê um número em ponto flutuante com ponto opcional
%g	Lê um número em ponto flutuante com expoente opcional (double)
%o	Lê um número em base octal
%s	Lê uma string
%x	Lê um número em base hexadecimal
%p	Lê um ponteiro

Prática

- Imprima a frase "Estou programando em C!"
- Faça a Média aritmética de 3 valores
- Calcule a área e o volume de uma esfera de Raio R.

  Sabendo que: A= 4 x pi x R2 e V= 4/3 x PI x R3
- Escreva um programa que calcule a média de aproveitamento de um aluno. O Aluno possui três avaliações sendo que a Av1 possui peso 1, Av2 possui peso 3 e Av3 possui peso 2. A média M é igual à soma das avaliações levando em consideração os seus respectivos pesos somado a média das notas dividindo tudo isso por 6.

Exercício 1

- Escreva um programa que leia a temperatura em Fahrenheit e calcule o valor em Celsius e vice-versa. Segue a formula: C/5 = (F-32)/9
- Escreva um programa para ler o número de alunos existentes em uma turma, ler a soma das notas destes alunos, e calcular a média aritmética destas notas.
- Escreva um programa que informe a quantidade de anos existente nos número de dias informado pelo usuário.
- Sabendo que o preço da gasolina é R$ 2,50 faça um programa que o usuário informe a quantidade de gasolina desejada e o programa imprima o preço final.

- Faça um programa que o usuário informe o salário do funcionário e o sistema informa o valor de 10% desse salário.
- O custo de um carro novo ao consumidor é a soma do custo de fábrica com a porcentagem do distribuidor e dos impostos (aplicados ao custo de fábrica). Supondo que o percentual do distribuidor seja de 28% e os impostos de 45%, escrever um algoritmo para ler o custo de fábrica de um carro, calcular e escrever o custo final ao consumidor.
-

Estrutura Condicional

As estruturas condicionais permitem tomar decisões diferentes a partir de uma condição. A condição é um teste que poderá retonar o valor VERDADEIRO ou FALSO. Uma condição pode ser composta de vários termos, por exemplo, se a idade for maior que 18 anos e nascido no estado da Bahia.

Estrutura

A estrutura condicional pode ser representada coforme abaixo. O "IF" significa que em seguida virá o teste de uma condição, caso o resultado seja verdadeiro o programa irpa executar os comando existentes entre "{ }" .

```
If (condição)
{
      ....
}
```

A estrutura "IF" possui a variação "IF/ELSE" onde caso o teste da condição seja VERDADEIRO o programa irá executar o comando existente entre "{ }" logo abaixo. Se o resultado do teste da condição for FALSO o programa irá executar o comando existente entre "{ }" da estrutura "ELSE".

```
If (condição)
{
      ....
}else
```

```
{
        ...
}
```

Outra variação é a estrutura “ELSE IF” onde primeiramente é testado a condição “IF”, caso a execução seja VERDADEIRA o programa irá executar o código existente entre “{ }”, senão irá fazer um novo teste na estrutura “ELSE IF”, caso no novo teste seja VERDADEIRO o programa irá executar o código existente entre “{ }” abaixo, se o segundo teste for FALSO será executado o código do “ELSE”.

```
If (condição)
{
        ....
}else if (codição)
{
        ...
}else
{
        ...
}
```

A estrutura Switch/Caseé utilizada para fazer diferentes teste para uma mesma variável. Caso o valor da variável teste verdadeiro para algum “Case” o programa irá executar o código existente nesse “Case”. Se nenhum “Case” possuir o valor da variável o programa irá executar o código do “Default”.

```
Switch(Aux)
{
        Case 1: ... break;
        Case 2: ... break;
        Case 3: ... break;
```

```
    Default:  ...
}
```

Prática

- Ler um valor e escrever a mensagem É MAIOR QUE 10! se o valor lido for maior que 10, caso contrário escrever NÃO É MAIOR QUE 10!
- Faça um algoritmo em C que informe quantas maças foram compradas. Exiba a mensagem "Maior que uma dúzia" se a quantidade for maior ou igual a 12 ou "Menor que uma dúzia" se for menor que 12.
- Escreva um programa que o usuário informe a tamanho dos lados de três triângulos e o sistema exiba se ele é: isosseles, escaleno e equilátero.
- Faça um diagrama que o usuário informe 3 números e o algoritmo exiba os dois maiores números.
- Ler dois valores e imprimir uma das três mensagens a seguir:
    - 'Números iguais', caso os números sejam iguais
    - 'Primeiro é maior', caso o primeiro seja maior que o segundo;
    - 'Segundo maior', caso o segundo seja maior que o primeiro.
- Ler o nome de 2 times e o número de gols marcados na partida (para cada time). Escrever o nome do vencedor. Caso não haja vencedor deverá ser impressa a palavra EMPATE.
- Ler 3 valores (considere que não serão informados valores iguais) e escrevê-los em ordem crescente.

Exercício:

- Escreva um algoritmo para ler o número total de eleitores de um município, o número de votos brancos, nulos e válidos. Calcular e escrever o percentual que cada um representa em relação ao total de eleitores.
- Ler um valor e escrever se é positivo ou negativo (considere o valor zero como positivo).

APOSTILA DE ALGORITMOS

- As maçãs custam R$ 1,30 cada se forem compradas menos de uma dúzia, e R$ 1,00 se forem compradas pelo menos 12. Escreva um programa que leia o número de maçãs compradas, calcule e escreva o custo total da compra.
- Ler as notas da 1a. e 2a. avaliações de um aluno. Calcular a média aritmética simples e escrever uma mensagem que diga se o aluno foi ou não aprovado (considerar que nota igual ou maior que 6 o aluno é aprovado). Escrever também a média calculada.
- Ler o ano atual e o ano de nascimento de uma pessoa. Escrever uma mensagem que diga se ela poderá ou não votar este ano (não é necessário considerar o mês em que a pessoa nasceu).
- Ler 3 valores (considere que não serão informados valores iguais) e escrever o maior deles.
- Ler 3 valores (considere que não serão informados valores iguais) e escrever a soma dos 2 maiores.
- Ler a hora de início e a hora de fim de um jogo de Xadrez (considere apenas horas inteiras, sem os minutos) e calcule a duração do jogo em horas, sabendo-se que o tempo máximo de duração do jogo é de 24 horas e que o jogo pode iniciar em um dia e terminar no dia seguinte.
- Ler 3 valores (considere que não serão informados valores iguais) e escrever o maior deles.
- Ler 3 valores (considere que não serão informados valores iguais) e escrever a soma dos 2 maiores.
- Ler 3 valores (considere que não serão informados valores iguais) e escrevê-los em ordem crescente.
- Ler 3 valores (A, B e C) representando as medidas dos lados de um triângulo e escrever se formam ou não um triângulo. OBS: para formar um triângulo, o valor de cada lado deve ser menor que a soma dos outros 2 lados.
- Faça um programa que mostre o menu abaixo, receba a opção do usuário e os dados necessários para executar cada operação.
    - Somar dois números
    - Multiplicar dois números

- Faça um programa que determine a data cronologicamente maior entre duas datas fornecidas pelo usuário. Cada data deve ser composta por três valores inteiros, em que o primeiro representa o dia, o segundo, o mês e o terceiro, o ano.

Laços

Os laços são estruturas que permitem a execução de um bloco de código diversas vezes, portanto que uma condição seja satisfeita. As condições são testadas para iteração executada, se a condição for FALSA o laço para de se testado.

A estrutura "FOR" é utilizada normalmente quando se tem um laço caracterizado pelo controle a partir de um contador. Quando se sabe o estado inicial e final da contagem. Por exemplo, faça a soma de todos os números entre 1 e 10. Sabemos que a contagem começará em 1 e terminará em 10. A estrutura do "FOR" possui 3 termos. O primeiro é executado somente no início inicializando a variável utilizada para ser o contador. O segundo termo é onde se tem a condição para poder executar o código existente dentro do "FOR". Por fim, o terceiro termo é onde é incrementado/decrementado a variável contador. Abaixo segue o formato da estrutura "FOR".

```
For(i =0; i<10; i++)
{
    ...
}
```

A estrutura "WHILE" é utilizada normalmente quando se tem um laço caracterizado pelo controle a partir de uma condição lógica. Quando não se sabe a quantidade de vezes que irá ser executado o laço porém sabe-se da condição de parada do laço. A esrtutura "WHILE" possui somente um termo que é a condição.

```
While (condição)
{
    ...
}
```

A estrutura do "DO/WHILE" é parecida com a estrutura "WHILE", exceto que o código existente dentro dos "{ }" é executado pelo menos uma vez e irá ser executado novamente se a condição for testada VERDADEIRO.

```
Do
{
    ...
} While (condição);
```

Pratica

- Escreva um algoritmo para imprimir os números de 1 (inclusive) a 10 (inclusive) em ordem crescente.
- Escreva um algoritmo para imprimir os números de 10 (inclusive) a 1 (inclusive) em ordem decrescente.
- Escreva um algoritmo para imprimir os 10 primeiros números inteiros maiores que 100.
- Ler um valor N e imprimir todos os valores inteiros entre 1 (inclusive) e N (inclusive). Considere que o N será sempre maior que ZERO.
- Modifique o exercício anterior para aceitar somente valores maiores que 0 para N. Caso o valor informado (para N) não seja maior que 0, deverá ser lido um novo valor para N.

Exercício

- Ler um valor inteiro (aceitar somente valores entre 1 e 10) e escrever a tabuada de 1 a 10 do valor lido.
- Ler 10 valores e escrever quantos desses valores lidos são NEGATIVOS.
- Ler 10 valores e escrever quantos desses valores lidos estão no intervalo [10,20] (incluindo os valores 10 e 20 no intervalo) e quantos deles estão fora deste intervalo.
- Ler 10 valores, calcular e escrever a média aritmética desses valores lidos.

- Ler o número de alunos existentes em uma turma e, após isto, ler as notas destes alunos, calcular e escrever a média aritmética dessas notas lidas.
- Escreva um algoritmo para ler 10 números e ao final da leitura escrever a soma total dos 10 números lidos.
- Escreva um algoritmo para ler 10 números. Todos os números lidos com valor inferior a 40 devem ser somados. Escreva o valor final da soma efetuada.
- Ler 2 valores, calcular e escrever a soma dos inteiros existentes entre os 2 valores lidos (incluindo os valores lidos na soma). Considere que o segundo valor lido será sempre maior que o primeiro valor lido.
- Escreva um algoritmo para ler 2 valores, se o segundo valor informado for ZERO, deve ser lido um novo valor, ou seja, para o segundo valor não pode ser aceito o valor zero e imprimir o resultado da divisão do primeiro valor lido pelo segundo valor lido.
    - Acrescentar uma mensagem de 'VALOR INVÁLIDO' no exercício anterior caso o segundo valor informado seja ZERO.
- Um funcionário de uma empresa recebe, anualmente, aumento salarial. Sabe-se que:
    - Esse funcionário foi contratado em 2005, com salário inicial de R$ 1.000,00
    - Em 2006, ele recebeu aumento de 1,5% sobre seu salário inicial
    - A partir de 2007 (inclusive), os aumentos salariais sempre corresponderam ao dobro do percentual do ano anterior.

    Faça um programa que determine o salário atual desse funcionário.
- Faça um programa que leia um número N que indica quantos valores inteiros e positivos devem ser lidos a seguir. Para cada número lido, mostre uma tabela contendo o valor lido e o fatorial desse valor.
- Faça um programa que mostre os oito primeiros termos da sequencia Fibonacci.
- Faça um programa qu receba duas notas de seis alunos. Calcule e mostre:

- A média aritmética das duas notas de cada aluno;e
- Mostre:
  - A mensagem “Reprovado” se a nota for menor e igual que 3;
  - A mensagem “Final” se a nota estiver entre 3 e 67;
  - Aprovado se for maior e igual a 7;

- Em um campeonato de futebol existem cinco times e cada um possui onze jogadores. Faça um programa que receba a idade, o peso e a altura de cada um dos jogadores, calcule e mostre:
  - a quantidade de jogadores com idade inferior a 18 anos;
  - a média das idades dos jogadores de cada time;
  - a média das alturas de todos os jogadores do campeonato; e
  - a porcentagem de jogadores com mais de 80 kg entre todos os jogadores do campeonato.

APOSTILA DE ALGORITMOS

Vetor

Definição:

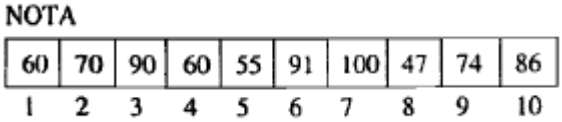
"Variável composta homogênea unidimensional. Isso quer dizer que se trata de um conjunto de variáveis de mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas em sequencia na memória."(Ascenio, A., Campos, E., 2012)

"Variáveis compostas homogêneas correspondem a posições de memória, identificadas por um mesmo nome, individualizadas por índices e cujo conteúdo é de mesmo tipo." (FARRER, H., BECKER, C., 1999)

Vetores são variáveis compostas por serem organizada uma ao lado da outra, conforme exemplo. Unidimensional por possui somente uma dimensão, ou seja, uma linha. O fato de possuírem somente uma declaração para representar todas as variáveis compostas faz com que as variáveis compostas (vetores) possuam o mesmo tipo informado na declaração. Porém cada variável pode assumir valores diferentes, para isso cada elemento do vetor possui um identificador único, chamado de índice.

Exemplo:

O vetor abaixo possui 10 posições. Podemos dizer também que a figura representa 10 variáveis organizadas uma ao lado da outra. Essas 10 variáveis juntas forma a variável NOTA. Essa variável possui somente numero inteiros. Dentro da variável NOTA posso guardar 10 valores, conforme ilustrado abaixo. Na posição 1 possui o valor 60, na 2 o valor 70 até chegar a posição 10 com o valor 86.



Declaração:

Um vetor ou variável composta é uma variável que possui a seguinte declaração:

```
VAR

vetor: array [1..10] of integer;
```

vetor: nome da variável;

array: palavra que indica que a variável é um vetor;

[1..10]: definição do tamanha do vetor;

of integer: tipo de dado dos elementos dos vetores;

Exercício de fixação:

a. Declare três variáveis vetor do tipo INTEGER, CHAR e FLOAT;
b. Um ônibus possuem 48 lugares (24 janelas e 24 corredor). Como representaríamos esses lugares separando janela de corredor?
c. A turma possui 45 alunos como armazenaria as notas desses alunos? E o nome dos alunos?
d. Preciso guardar 10 números e depois guardar e depois a soma doa elementos 1+2, 3+4, 5+6, 7+8 e 9+10.

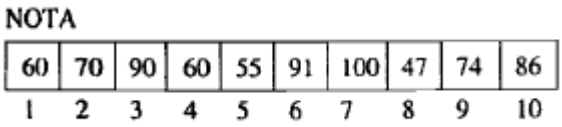
Dica: A declaração de um vetor é parecida de com a declaração de uma variável normal, "vetor: integer;". Porém é acrescentado "array [1..10] of" para indicar que é um vetor e o seu tamanho .

Atribuição de valores:

Como foi mostrado na definição, uma variável vetor pode guardar "n" valores. "Isso é realizado por meio da atribuição ":=". Porém para fazer a atribuição diferentemente das variáveis normais, temos que além de informar o nome referenciamos também a posição do vetor que se quer armazenar o valor.

NOTA[1] := 60;

NOTA[2] := 90;



Dica: Toda atribuição deve ser informado o índice do vetor!

Exercício de fixação:

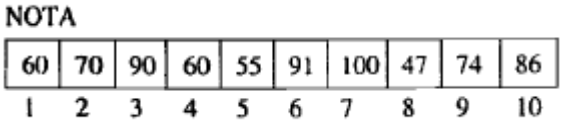
a. Armazene em um vetor de inteiro com 3 posições os números 3, 8 e 6;
b. Armazena em um vetor de char a frase "Belo dia!";
c. Declare uma variável "i" inteira, atribua o valor 3 a ela e altera o elemento "i" do vetor da questão a para 15.

Preenchendo um vetor:

O preenchimento de um vetor é o processo pelo qual são realizados diversas atribuições de valores aos elementos de um vetor. Normalmente esse processo é sequencial, onde o vetor é preenchido um elemento de cada vez. Coma a declaração do vetor sabemos qual o seu tamanho exato (vetor: array **[1..10]** of integer;). Então para se fazer uma atribuição sequencial, onde o processo será repetitivo e o inicio e fim da repetição é conhecido é utilizado a estrutura de repetição **FOR**. O exemplo abaixo mostra como preencher o vetor NOTA.

```
For i:= 1 to 10 do

Begin

    writeln('Digite um número');

    readln(NOTA[i]);

End;
```

O "For i:= 1 to 10 do" irá repetir a instrução dentro o "Begin/End" várias vezes. A instrução "NOTA[i]" é responsável por atribuir um valor a um elemento do vetor referenciado pela variável "i" que vai de 1 até 10.



Dica: Toda vez que utilizar atribuir um valor ao um vetor usando o "realdn" informar a posição do vetor. Caso esteja dentro de um for a referência será a variável contador do laço "readln(NOTA[i]);".

Exercício de fixação:

a. Cadastre as notas de 20 alunos de uma sala de aula;
b. Cadastre os nomes de 20 alunos de uma sala de aula;
c. Um ônibus possuem 48 lugares (24 janelas e 24 corredor). Faça um programa que utilize dois vetores para controlar as cadeiras ocupadas no cerredor e na janela.

Percorrendo um vetor:

Depois de se preencher um vetor é será necessário em algum momento obter as informações que foram guardadas. Normalmente esse processo é sequencial, onde o vetor é percorrido um elemento de cada vez. Coma a declaração do vetor sabemos qual o seu tamanho exato (vetor: array

[1..10] of integer;). Então para se percorrer sequencialmente, onde o processo será repetitivo e o inicio e fim da repetição é conhecido é utilizada a estrutura de repetição **FOR**. O exemplo abaixo mostra como percorrer o vetor NOTA.

O "For i:= 1 to 10 do" irá repetir a instrução dentro o "Begin/End" várias vezes. A instrução "writeln('O numero ', i , ' é igual a: ', NOTA[i]);" é responsável por exibir o valor de cada elemento do vetor referenciado pela variável "i" que vai de 1 até 10.

```
For i:= 1 to 10 do

Begin

    writeln('O numero ', i , ' é igual a: ', NOTA[i]);

End;
```

Dica: Toda vez que exibir elementos de um vetor percorrendo ele com um for utilizar "writeln(NOTA[i]);" ou uma variação dessa estrutura, referenciado sempre o elemento do vetor com uso da variável contador "NOTA[i]".

Exercício de fixação:

a. Exiba as notas de 20 alunos de uma sala de aula;
b. Exiba os nomes de 20 alunos de uma sala de aula;
c. Sabendo as notas da posição 1 da questão "a" equivale ao aluno da posição 1 da questão "b" e assim respectivamente, exiba os alunos e suas notas;
d. Exiba somente os lugares ocupados da janela e livres do corredor.

Exercício

1. Elaborar um algoritmo que lê um conjunto de 10 valores e os coloca em 2 vetores conforme estes valores forem pares ou ímpares. O tamanho do vetor é de 5 posições. Se algum vetor encher informar ao usuário.

2. Faça um algoritmo que leia um vetor N[20]. A seguir, encontre o menor elemento do vetor N e a sua posição dentro do vetor, mostrando: "O menor elemento de N é", M, "e sua posição dentro do vetor é:",P.

3. Escreva um algoritmo que leia dois vetores de 10 posições e faça a multiplicação dos elementos de mesmo índice, colocando o resultado em um terceiro vetor. Mostre o vetor resultante.

4. Faça um algoritmo que leia um vetor K[30]. Troque a seguir, todos os elementos de ordem ímpar do vetor com os elementos de ordem par imediatamente posteriores.

5. Faça um algoritmo que leia um vetor S[20] e uma variável A. A seguir, mostre o produto da variável A pelo vetor.

6. Faça um algoritmo que leia dois vetores: F[20] e G[20]. Calcule e mostre, a seguir, o produto dos valores de F por G.

7. Escreva um algoritmo que leia dois vetores de 10 posições e faça a multiplicação dos elementos de mesmo índice, colocando o resultado em um terceiro vetor. Mostre o vetor resultante.

8. Escreva um algoritmo que leia e mostre um vetor de 20 números. A seguir, conte quantos valores pares existem no vetor.

9.Escreva um algoritmo que leia um vetor de 100 posições e mostre-o ordenado em ordem crescente.

10. Escreva um algoritmo que leia um vetor de 20 posições e mostre- o. Em seguida, troque o primeiro elemento com o último, o segundo com o penúltimo, o terceiro com o antepenúltimo, e assim sucessivamente. Mostre o novo vetor depois da troca.

11. Escreva um algoritmo que leia 50 valores para um vetor de 50 posições. Mostre depois somente os positivos.

12. Escreva um algoritmo que leia um vetor inteiro de 30 posições e crie um segundo vetor, substituindo os valores nulos por 1. Mostre os 2 vetores.

APOSTILA DE ALGORITMOS

Matriz

Definição:

"Variável composta homogênea **multidimensional**. Isso quer dizer que se trata de um conjunto de variáveis de mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas em sequencia na memória."(Ascenio, A., Campos, E., 2012)

"variáveis compostas bidimensionais, que necessitam de dois índices para individualização de seus elementos. São muito úteis quando se tem que manipular matrizes e tabelas. O primeiro índice representa o número da linha e o segundo, o número da coluna." (FARRER, H., BECKER, C., 1999)

Matrizes é uma extensão dos vetores pois são armazenadas sequencialmente, possuem o mesmo tipo e uma declaração para somente. Porém quando os vetores possuem somente uma linha as matrizes permitem a criação de várias linhas. A representação mais comum dessa estrutura é uma tabela. Como uma tabela possui varias linhas e colunas cada elemento da tabela é indicado pela sua posição nela. Um tabuleiro de xadrez cada casa possui sua indicação com a linha e a coluna, o mesmo é utilizando para as matrizes.

Exemplo:

A matriz abaixo 3 linhas e 3 colunas totalizando 10 posições. Podemos dizer também que a figura representa 9 variáveis organizadas no formato de uma tabela. Essas 9 variáveis juntas forma a variável MATRIZ. Essa variável possui somente numero inteiros. Dentro da variável MATRIZ posso guardar 9 valores, conforme ilustrado abaixo. Na posição de linha 1 e coluna 1 possui o valor 12, na posição (1,2) o valor 19.

MATRIZ

	1	2	3
1	12	19	32
2	69	14	37
3	11	112	42

Declaração:

Uma matriz é uma variável que possui a seguinte declaração:

```
VAR

vetor: array [1..10, 1..10] of integer;
```

vetor: nome da variável;

array: palavra que indica que a variável é uma matriz;

[1..10,1..10]: definição do tamanha da matriz, o elemento "[**1..10**,1..10]" indica a quantidade de linhas e o elemento "[1..10,**1..10**]" indica a quantidade de colunas.

of integer: tipo de dado dos elementos da matriz;

Dica: A declaração de uma matriz é parecida de com a declaração de um vetor. A diferença é a referencia a quantidade de linhas "matriz: array [**1..10**, 1..10] of integer;".

Exercício de fixação:

a. Declare três variáveis matriz do tipo INTEGER, CHAR e FLOAT;
b. Um ônibus possuem 48 lugares (24 janelas e 24 corredor). Como representaríamos esses lugares separando janela de corredor?
c. Você foi convocado para criar um jogo de xadrez como armazenaria as posições do tabuleiro?
d. Crie uma matriz 4x4, 5x7, 3x5.

Atribuição de valores:

Como foi mostrado na definição, uma variável matriz pode guardar "n" valores. "Isso é realizado por meio da atribuição ":=". Porém para fazer a atribuição diferentemente dos vetores, temos que informar a linha e a coluna que se quer armazenar o valor.

```
MATRIZ[1,1] := 12;

MATRIZ[1,2] := 19;
```

MATRIZ

	1	2	3
1	12	19	32
2	69	14	37
3	11	112	42

Dica: Toda atribuição deve ser informado a linha e coluna da matriz "MATRIZ[1,2]" !

Exercício de fixação:

a. Armazene em um matriz de inteiro 2x2 os números de 1 até 4;
b. Armazena em uma matriz de char o primeiro nome, o segundo nome e o terceiro nome de uma pessoa;
c. Declare uma variável "i" e "j" inteira, atribua o valor 1 e 2 respectivamente e altere o elemento "i,j" da matriz da questão a para 15.

Preenchendo uma matriz:

O preenchimento de uma matriz é o processo pelo qual são realizados diversas atribuições de valores aos elementos de uma matriz. Normalmente esse processo é sequencial, onde a matriz é preenchida um elemento de cada vez. Com a declaração da matriz sabemos qual o seu tamanho exato (vetor: array **[1..10, 1..10]** of integer;). Então para se fazer uma atribuição sequencial, onde o processo será repetitivo e o inicio e fim da repetição é conhecido é utilizado a estrutura de repetição **FOR**. Porém existe uma diferença em relação ao vetor. No vetor criamos um laço para controlar os elementos da coluna no vetor. Para a matriz além de controlar os elementos da coluna teremos que controlar os elementos das linhas. Para isso teremos um FOR para as linhas e um FOR para as colunas. O exemplo abaixo mostra como preencher a matriz.

```
For i:= 1 to 10 do

Begin

    For j:= 1 to 10 do

    Begin

        writeln('Digite um número');

        readln(MATRIZ[i,j]);

    End;

End;
```

O "For i:= 1 to 10 do" irá repetir a instrução dentro o "Begin/End" várias vezes. Para cada linha irá executar o "For j:= 1 to 10 do" para inserir em

todas as colunas da linha "i". A instrução "MATRIZ[i,j]" é responsável por atribuir um valor a um elemento da matriz referenciado pela variável "i" que vai de 1 até 10 e pela variável "j" que vai de 1 até 10.

MATRIZ

	1	2	3
1	12	19	32
2	69	14	37
3	11	112	42

Dica1: Toda vez que atribuir um valor a uma matriz usando o "realdn" informar a posição da matriz readln(MATRIZ[i,j]).

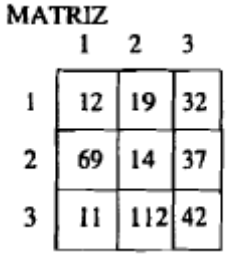
Dica2: Para preencher uma matriz sequencialmente **sempre** será necessário utilizar dois FOR uma para a linha "i" e outro para a coluna "j".

Exercício de fixação:

a. Cadastre uma matriz 10x10 valores inteiros;
b. Cadastre uma matriz 10x10 valores inteiros pares, caso o numero informado não for par cadastre o valor -1;
c. Cadastre uma matriz 10x10 valores inteiros pares, caso o numero informado não for par o novo cadastramento deverá ser no mesmo elemento da matriz.

Percorrendo uma matriz:

Depois de se preencher uma matriz será necessário em algum momento obter as informações que foram guardadas. Normalmente esse processo é sequencial, onde a matriz é percorrida um elemento de cada vez. Com a declaração da matriz sabemos qual o seu tamanho exato (vetor: array **[1..10, 1..10]** of integer;). Então para se percorrer sequencialmente, onde o processo será repetitivo e o inicio e fim da repetição é conhecido é utilizado a estrutura de repetição **FOR**. Porém existe uma diferença em relação ao vetor. No vetor criamos um laço para controlar os elementos da coluna no vetor. Para a matriz além de controlar os elementos da coluna teremos que controlar os elementos das linhas. Para isso teremos um FOR para as linhas e um FOR para as colunas. O exemplo abaixo mostra como preencher a matriz.




```
For i:= 1 to 10 do

Begin

    For j:= 1 to 10 do

    Begin

        writeln('O elemento', i,j,'é igual a:', MATRIZ[i,j]);

    End;

End;
```

O "For i:= 1 to 10 do" irá repetir a instrução dentro o "Begin/End" várias vezes. Para cada linha irá executar o "For j:= 1 to 10 do" para exibir os elementos de das colunas da linha "i". A instrução "MATRIZ[i,j]" é responsável por mostrar o valor do elemento da matriz referenciado pela variável "i" que vai de 1 até 10 e pela variável "j" que vai de 1 até 10.

Dica1: Toda vez que atribuir um valor a uma matriz usando o "realdn" informar a posição da matriz readln(MATRIZ[i,j]).

Dica2: Para percorrer uma matriz sequencialmente **sempre** será necessário utilizar dois FOR uma para a linha "i" e outro para a coluna "j".

Exercício de fixação:

a. Exiba uma matriz 10x10 valores de inteiros;
b. Exiba somente os valores maiores que 10 de uma matriz de inteiros;
c. Exiba a diagonal principal e secundária de uma matriz 10x10.

Função

Função é uma estrutura de que é utilizada para encapsular um conjunto de código fonte que junto tem o objetivo de chegar a um resultado. As funções são criadas separadamente do bloco principal *main*. Podemos eleger trechos de códigos para formarem uma junção e substituir-lo pela chamada da função.

Os atores W. Celes e J. L. Rangel dizem que As funções dividem grandes tarefas de computação em tarefas menores. A criação de funções evita a repetição de código, de modo que um procedimento que é repetido deve ser transformado numa função que, então, será chamada diversas vezes.

O uso de funções traz inúmeros benefícios no desenvolvimento de programas, como por exemplo:

1. Evita que um trecho de código que seja repetido várias vezes dentro de um mesmo programa;
2. Permite a alteração de um trecho de código de uma forma mais rápida. Com o uso de uma função é preciso alterar apenas **dentro** da função que se deseja;
3. Permite que os blocos do programa não fiquem grandes demais e, por conseqüência, mais difíceis de entender;
4. Facilita a leitura do programa-fonte de uma forma mais fácil;
5. Separa o programa em partes(blocos) que possam ser logicamente compreendidos de forma isolada.

Forma Geral

```
tipo_retornado nome_da_função (lista de parâmetros...)
{
        corpo da função
}
```

Nome da Função

Pode ser dado qualquer nome a função que represente a computação que a mesma se propõem.

Parâmetros

Os parâmetros possibilitam que seja enviados à função valores que são utilizados internamente. Para definir os parâmetros de uma função o programador deve explicitá-los como se estive declarando uma variável, entre os parênteses do cabeçalho da função. Caso precise declarar mais de um parâmetro, basta separá-los por vírgulas. No exemplo a seguir temos a função SOMA que possui dois parâmetros, sendo o primeiro um float e o segundo um int.

```
void SOMA(float a, int b)  // basta separar por vírgulas
{
   float result;      // a declaração de variáveis é igual ao que
                      // se faz na função main
   result = a+b;
   printf("A soma de %6.3f com %d é %6.3f\n, a,b,Result);
}
```

Os parâmetros da função na sua declaração são chamados **parâmetros formais**. Na chamada da função os parâmetros são chamados **parâmetros atuais**.

Na chamada da função deve-se passar os parâmetros atuais obedecendo a ordem que foi declarado os parâmetros formais, ou seja, os valores passados deve ser do mesmo tipo que os valores recebidos.

```
void SOMA(float a, int b)  // basta separar os parâmetros por vírgulas
{
   float result;      // a declaração de variáveis é igual ao que
                      // se faz na função main
   result = a+b;
   printf("A soma de %d com %6.3f é %6.3f\n", a,b,result);
}

main()
{
    int a;
    float b;

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

}
```

Protótipos

Em C pode-se trabalhar de duas formas com funções: a primeira declarando as funções antes do bloco principal *main* e a segunda informando o protótipo da função e declarando a função depois do bloco principal *main*.

Para declarar o protótipo da função tem que informar o trecho de código que especifica o nome e os parâmetros da função.

```
void SOMA(float a, int b);  // Protótipo da função SOMA

void main()
{
    SOMA(16.7,15);  // Chamada da função SOMA antes de sua definição,
}                   // mas após sua prototipação

void SOMA(float a, int b)  // Definição da função SOMA
{
   float result;     // a declaração de variáveis é igual ao que
                     // se faz na função main
   result = a+b;
   printf("A soma de %d com %6.3f é %6.3f\n", a,b,result);
}
```

Retorno da Função

Uma função ao final da sua execução poderá retornar algum valor ou não. Para não retornar valor tem que informar no retorno da função a palavra *void*.

```
void SOMA(float a, int b)  // basta separar os parâmetros por vírgulas
{
   float result;     // a declaração de variáveis é igual ao que
                     // se faz na função main
   result = a+b;
   printf("A soma de %d com %6.3f é %6.3f\n, a,b,Result);
}
```

Porém caso seja necessário que a função retorne algum valor deverá informar o tipo de dados que será retornado. Quando a função encerrar a sua execução o valor retornado será substituído pela chamada da função. Para retornar um valor deve-se usar *return*.

```
int SOMA(int a, int b);  // Protótipo da função SOMA

main()
{
    int s,a,b;

    a=15;

    b=16;

    SOMA(a,b); // função retorna valor ao final da execução

    printf("A soma de %d com %d é %d \n", a,b,s);

}

int SOMA(int a, int b)  // Definição da função SOMA
{
   int result;     // a declaração de variáveis é igual ao que
                     // se faz na função main
   result = a+b;

   return result;
}
```

Escopo da Variáveis

Variáveis declaradas internamente nas funções só poderá ser manipuladas dentro das funções. Ao final da execução do programa todas as variáveis declaradas local serã destruídas.

Exercício de Fixação

1. Faça um programa que leia 2 valores. Crie uma função SOMA_INTEIRO que calcule retorne a soma dos inteiros existentes entre os 2 valores lidos (incluindo os valores lidos na soma). Considere que o segundo valor lido será sempre maior que o primeiro valor lido.
2. Fazer uma função que calcula o fatorial de um número. Testar a função.
3. Fazer um programa que verifica se um número é primo, usando uma função.
4. Fazer uma função que receba um caractere como parâmetro e retorne verdadeiro caso o caractere seja uma consoante, e falso caso contrário.
5. Fazer uma função para verificar se um caracter é uma vogal minúscula, retornando verdadeiro ou falso. Faça um programa para testar a função.

6. Fazer uma função que calcula o número de arranjos de n elementos p a p . A fórmula do arranjo é a seguinte:
7. Criar uma função que calcula o número de combinações de n elementos p a p . A fórmula da combinação é a seguinte
8. Um estacionamento cobra uma taxa mínima de R$2,00 para estacionar por três horas. Um adicional de R$0,50 por hora não necessariamente inteira é cobrado após as três primeiras horas. A valor máximo para qualquer dado período de 24 horas é R$10,00. Suponha que nenhum carro fica estacionado por mais de 24 horas por vez. Escreva um aplicativo que calcule e exiba as taxas do estacionamento para cada cliente que estacionou nessa garagem ontem. Você deve inserir as horas de estacionamento para cada cliente. O prograva deve exibir a cobrança para o cliente atual e calcular e exibir o total recebido no final do dia. O programa deve usar uma função *valorAPagar* para determinar a cobrança para cada cliente.

9. Dizemos que um número inteiro é número perfeito se a soma de seus fatores, incluindo 1, mas não incluindo o próprio número, é igual ao número. Por exemplo, 6 é um número perfeito, pois 6 = 1 + 2 + 3. Escreva uma função perfeito que determina se o seu parâmetro numero é um número perfeito. Utilize essa função em uma aplicação que determina e exibe todos números perfeitos entre 1 e 1000. Exiba os fatores de cada número perfeito confirmando que o número é de fato perfeito. Desafie o poder de computação do seu computador testando números bem maiores que 1000. Exiba os resultados.