Capitulo 1

- Introdução a lógica de programação
- Conceitos da linguagem C
- Entendo a diretiva Include
- Funções de entrada e saída de dados
- Representando a informação
- Variáveis e tipos de dados
- Exercícios resolvidos
- Exercícios propostos

Capitulo 2

- Escopo de variáveis
- Operadores aritméticos
- Estrutura condicional
    1. Desvio Condicional simples
    2. Desvio Condicional composto
    3. IF escalonado
    4. IF Sequencial
    5. IF Aninhado
    6. Operadores Relacionais
    7. Operadores Lógicos

Capitulo 3

- Estrutura de comparação
- Estrutura de repetição
    1. while
    2. for
    3. do while
    4. controle fluxo break e continue

Capitulo 4

- Vetores
    1. Dimensionando um vetor
    2. Adicionando elementos a um vetor
    3. Acessando o vetor posição a posição
    4. Acessando o vetor de forma dinâmica

Capitulo 5

- Função

1. Criando sua própria função
2. entendo o conceito de módulos e funções
3. Função void
4. Função não void (Tipadas)
5. Função com passagem de parâmetros.

Capitulo 6

- Matriz
    1. Adicionando elemento a uma matriz
    2. Acessando uma matriz dinamicamente.
    3. Fazendo operações matemáticas em uma matriz.

Capitulo 7

- Recursividade
- Estrutura de dados em C (struct)
    1. Criando uma Estrutura
    2. Entendo o conceito da estrutura
    3. Agregando uma estrutura a outra estrutura
    4. Operações orientada a objetos.

Capítulo 1

Algoritmo

É a sequencia de passos lógicos até a execução de uma determinada tarefa.

Função main()

é a função principal de execução do programa, ela é responsável por executar o programa.

Ex:

```
main(){

 // Colocar aqui dentro todo o código que será executado

}
```

Observação :

Comentários → são blocos de programação que não serão interpretados pelo compilador da linguagem como código executável, serão vistos somente como um texto comum.

Existem duas formas de comentários

Comentários de linha →

Ex:

// Essa linha não será executada

Comentários de blocos

/*

Esse bloco não será executado.

*/

Voltando ao conceito de C todos os programas escritos em C terão uma função de execução chamada main senão o programa não poderá exibir o resultado final da aplicação.

Entendo Uma função

main (){

}

() → parênteses

são utilizados para a passagem de valores para dentro de uma função, esses valores são conhecidos como **parâmetros**. (será revisado no Capitulo 7).

{} → Delimitam o corpo de uma função, seria onde ela começa e termina. Toda o conteúdo criado dentro das chaves pertencem somente aquela função. (Será revisado nos capítulos seguintes).

Include Diretiva do pré-processador

O pré processador é responsável por examinar o programa antes mesmo da linguagem C ele executa certas modificações no arquivos antes do programação principal.

Todo diretiva começa com o símbolo de **#** e deve ser terminada na mesma linha.

Include é uma diretiva responsável por incluir um arquivo em seu programa fonte, ou seja, quando a diretiva include é chamada ela ira adicionar o conteúdo do arquivo selecionado ao seu programa.

Ex:

#include<stdio.h>

#include<conio.h>

Nas duas linhas acima eu peço para incluir ao meu programa dois arquivos de bibliotecas chamado stdio.h e conio.h.

Bibliotecas

É o conjunto de arquivos organizados em pastas responsáveis por conter funções básicas da linguagem, essas funções são usadas para adicionar conteúdo a programação.

stdio.h → tem como base conter as funções básicas de entrada e saída de dados.

Ex:

```
#include<stdio.h>

#include<conio.h>


main(){

// Responsável por fazer a entrada e saída de dados na tela

printf("Saída de dados no  tela");


// Responsável por interromper a execução do  programa

getch();

}
```

A função printf() é responsável por imprimir a mensagem que foi passada como valor na execução da função, isso só foi possível com a inclusão da biblioteca stdio.h,nela contem os dados da função.

Representação da informação

Alguns algoritmos tendem a escrever programas que terão a possibilidade de receber informações externas que serão passadas pelo usuário do sistema. Essas informações deveram ser representadas por tipos de dados específicos, o problema seria; como resolver todos os tipos de dados possíveis.

Tipos de dados

Os tipos de dados em si é a possível representação da informação que será recebida ou passado pelo sistema, pense no seguinte exemplo.

Ex: Escreva um programa que leia o salário, o nome, e a idade de um funcionário do sistema.

Quando um programa tem a marcação de leitura (leia) , significa que será passada como entrada para o sistema as seguintes informações: nome, salário, e a idade.

Neste exemplo será o salário, o nome e a idade, são informações que serão representadas por tipos de dados específicos para cada informação.

INT (inteiros)

é a representação da informação como números inteiros, números que representam uma unidade positivo ou negativo sem partes decimais.

ex: 10,20,30,40, (representaria a idade do funcionário)

O funcionário poderia ter 30 anos, 20 anos, nunca 2.5 anos, não é a melhor representação da informação.

FLOAT (decimal)

é a representação da informação que permite números decimais, ou seja, a informação pode representar números fracionados.

Ex: 10.2 , 10.4 , 10.5 -20.1 , -23.4

CHAR (caractere)

é a representação da informação como um texto contendo vários caracteres ou apenas um único caractere.

Ex: "Sergio" → uma sequencia de caractere está sempre entre aspas duplas.

Ex: 'M' → uma única letra está sempre em aspas simples.

Ex: "123ser" , "as/'→ caractere especial

Depois de resolver os possíveis tipos de dados que a informação pode representar já podemos receber essa informação no sistema,porém ainda temos um problema como guardar essa informação no sistema ? o que devemos fazer para manter ela durante a execução ? teremos que compreender a criação de vaiáveis para terminar o conceito de informação.

Variáveis

São representadas por um tipo e um nome, são responsáveis por guardar um valor em memória.

Ex:

int numero.

onde i**nt** é o tipo de dado, significa que essa variável por guardar valores numéricos inteiros.

onde numero é o nome da variável, pelo qual ela é acessada.

numero = 10; // Passando valor para a variável.

onde **=** , significa atribuição, o valor da direita está sendo passado para a variável, quando uma variável está recebendo um valor é chamado de **inicialização.**

char nome[20];

float salário;

int idade;

Observação:

a variável nome tem a marcação de até 20 letras para representar um nome de até 20 letras.

Acima foram criadas 3 variáveis que representam tipos de dados diferentes.

Variáveis criadas sem valores na sua declaração são chamadas de **declaradas**.

int num = 20;

Variáveis criadas com valor na sua declaração são chamadas de **inicializadas na criação.**

Exercícios Resolvidos:

Escreva um programa que leia dois números o programa deverá efetuar a soma dos números lidos. O programa devera imprimir o valor encontrado.

```
#include<stdio.h>
#include<conio.h>

// Criação das variáveis somente declaradas

int a;

int b;

main(){

// Pedir ao usuário que entre com os valores a ser efetuados a soma.

printf("Digite um numero : ");

// Pegar o valor digitado e guardar o valor em uma variável.

scanf("%d",&a);

printf("Digite um numero : ");

scanf("%d",&b);

// Criando uma variável para guardar o valor da soma dos números.

int soma = 0;

// Efetuando a soma dos números

soma  =  a + b;

printf("O valor da soma dos numeros : %d  \n", soma);

getch();

}
```

Na primeira linha e adicionado as bibliotecas que serão usadas pelo programa, logo após e criada duas variáveis para guardar os possíveis valores que serão informados pelo usuário do sistema. Dentro do bloco de execução (main()), é pedido que o usuário informe os valores que serão efetuados a soma, assim que o usuário informa o primeiro valor a função scanf() tem como finalidade recuperar o valor e colocar dentro da variável que apontada.

scanf("%d",&a);

onde scanf() → é o nome da função.

%d → formatação de entrada e saída para dados do tipo inteiro (falaremos sobre isso ainda).

& → é o apontador para o endereço físico da variável em questão.

a → é a variável que ira gravar o valor.

Depois do resgate dos valores informados pelo usuário, O programa efetua a soma guardando o valor na variável de nome "soma" e imprimindo o valor encontrado logo após.

soma = a + b; // variável soma

printf("O valor da soma dos numeros : %d \n", soma);

Exercícios Propostos:

1) Escreva um programa que leia dois números, o programa devera efetuar a multiplicação dos números e imprimir o resultado.

2) Escreva um algoritmo para ler um valor (do teclado) e escrever (na tela) o seu antecessor.

3) Escreva um algoritmo para ler as dimensões de um retângulo (base e altura), calcular e escrever a área do retângulo.

4) Faça um algoritmo que leia a idade de uma pessoa expressa em anos, meses e dias e escreva a idade dessa pessoa expressa apenas em dias. Considerar ano com 365 dias e mês com 30 dias.

5) 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.

6) Escreva um algoritmo para ler o salário mensal atual de um funcionário e o percentual de reajuste. Calcular e escrever o valor do novo salário.

7) 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.

8)Efetuar o calculo de uma prestação em atraso, Utilizando a seguinte fórmula:

PRESTACAO ← VALOR + (VALOR + TAXA/100) * TEMPO).

9)Efetuar a leitura de um numero inteiro e apresentar o resultado do quadrado do numero.

10)Ler um temperatura em graus fahrenheit e apresenta-la convertida em graus Celsius.

C ← (f-32) * (5/9).

11) Uma revendedora de carros usados paga a seus funcionários vendedores um salário fixo por mês, mais uma comissão também fixa para cada carro vendido e mais 5% do valor das vendas por ele efetuadas. Escrever um algoritmo que leia o número de carros por ele vendidos, o valor total de suas vendas, o salário fixo e o valor que ele recebe por carro vendido. Calcule e escreva o salário final do vendedor.

12) Faça um algoritmo que leia três notas de um aluno, calcule e escreva a média final deste aluno. Considerar que a média é ponderada e que o peso das notas é 2, 3 e 5. Fórmula para o cálculo da média final é:

$$\text{mediafinal} = \frac{n1 * 2 + n2 * 3 + n3 * 5}{10}$$

Capítulo 2

Escopo de variáveis

É o local onde as variáveis podem ser criados dentro do programa, existem duas formas de se criar variáveis podem ser locais(dentro de blocos) ou globais(fora de blocos de programação).

Local

São variáveis criadas dentro de blocos de programação, precisam ser inicializadas e são visíveis somente no bloco em que foram declaradas.

Globais

São variáveis criadas fora de blocos de programação, não precisam ser inicializadas e têm visibilidade por todos os blocos de programação.

Formatação de dados em C

%d → representa uma informação inteira (int)

%f →representa uma informação decimal(float)

%s → representa uma informação texto(sequencia de caracteres) (char).

%c → representa uma informação texto (uma só letra)(char).

Operadores aritméticos

+ , adição

- , Subtração

/, Divisão

*, multiplicação

Estrutura condicional

Representam um fluxo alternativo na execução do programa.

Em um programa haverá momentos em que uma determinada informação necessita ser tratada antes de executar um processamento associado.

Imagine a seguinte situação :

Escreva um programa que leia o preço de um produto, o programa deverá verificar se o preço do produto é superior a 300 reais, caso seja o programa deverá aplicar um desconto de 10% e imprimir o novo valor a ser pago com o desconto ou não.

Deverá notar que nessa situação o processamento do novo valor a ser pago deverá primeiro ser verificado se o programa pode ou não aplicar o desconto desejado.

Desvio Condicional Simples

É quando uma condição tem somente uma única resposta associada.

Pseudo- código

Se(condição)

// executar Bloco de controle caso condição verdadeira

então

Fim se // Finalizando a execução da condicional

Codificação em C

IF(condição)**{**

// bloco do IF (se a condição for verdadeira esse bloco será executado)

}

Resolvendo a questão anterior:

```
#include<stdio.h>

#include<conio.h>

float preco;

main(){

 float valorApagar = 0;

float desconto = 0;

 printf("Digite o preco do produto  :");

scanf("%f",&preco);

if (preco > 300){

  desconto = (10 * preço )/100;

}

valorApagar = preco - desconto;

printf("O valor a ser pago : %f  \n", valorApagar);


}
```

Analise do algoritmo:

primeiro e criado uma variável para representar o valor do produto (preço → tipo decimal), depois é pedido ao usuário para informar o valor a ser guardado na variável. Na segunda parte que consiste em

gerar o novo valor a ser pago, primeiro é feita uma verificação no preço para saber se ele esta de acordo com a condição definida.

if (preco > 300){

desconto = (10 * preço)/100;

}

Caso o preço do produto seja realmente maior que 300, o desconto será calculado, mas somente se ele passar na verificação. O calculo do novo valor a ser pago e feito com a retirada do possível desconto, mas somente se o valor do produto for maior que 300.

Desvio Condicional Composto

É quando um condição tem duas possíveis respostas associadas.

Pseudo - código.

se(condição)

então

// Bloco se a condição for verdadeira

senão

// Bloco se a condição for falsa

fim se

if(condição){

// Bloco se a condição for verdadeira

}else{

// Bloco se a condição for falsa

}

Exemplo:

Escreva um programa que leia duas notas de um aluno. O programa deverá calcular a media do aluno, Com o valor da media o programa deverá imprimir a situação do aluno de acordo com a seguinte tabela:

media >= 7, aprovado

senão, reprovado.

Neste exemplo temos duas possíveis formas de imprimir o a saída da informação de acordo com o valor processado na media, antes de imprimir a situação do aluno temos que primeiro saber se ele foi ou não foi aprovado ou reprovado.

Resolvendo:

```
#include<stdio.h>
#include<conio.h>

float nota1,nota2; // Variáveis globais

main(){

float media = 0;  // variável local

printf("Digite a nota do aluno : ");

scanf("%f",&nota1);

printf("Digite a nota do aluno : ");

scanf("%f",&nota2);

media =  (nota1 + nota2 )/ 2;

IF(media >= 7){

printf("Aluno Aprovado!")

}else{

 printf("Aluno Reprovado!");

}

getch();

}
```

IF ESCALONADO

É quando uma condição tem mais de duas possíveis resposta.

estrutura escalonada:

```
else if(condição){
```

}

Pseudo - código.

senão se(condição)

fim se

Exemplo:

Escreva um programa que leia um numero. O programa deverá verificar se o numero é positivo, negativo ou neutro.

```
#include<stdio.h>

#include<conio.h>


int numero;

main(){

printf("Digite um numero : ");

scanf("%d",&numero);

if(numero > 0){

  printf("Positivo");

}else if(numero < 0){

 printf("Negativo");

}else{

printf("Neutro");

}


getch();

}
```

O IF escalonado é usado para agrupar possíveis respostas a uma mesma condição, no exemplo anterior notamos que somente duas respostas não era o suficiente para a condição passada então colocamos mais uma possível resposta escalonada dentro da mesma condição, uma das vantagens é que quando

qualquer bloco responder verdade ele automaticamente saíra da estrutura condicional não precisando testar as outras condições.

IF SEQUENCIAL

É quando um programa tem mais de uma condição que não tem ligação uma com a outra.

```
if(condição){
  //resposta da primeira condição
}
// Sequencial
if(condição){
// resposta da segunda condição
}
```

A segunda condição não depende em nada na existência da primeira condição as duas são executadas de forma independentes.

Exemplo:

Escreva um programa que leia o preço e a quantidade de produtos comprados.

A) O programa deverá calcular o valor a ser pago.

B) O programa deverá aplicar um desconto de 10% caso o valor do produto seja maior que 300 reais.

C) O programa deverá aplicar um desconto de 8% caso a quantidade de produtos seja maior que 10.

D) O programa deverá calcular o novo valor a ser pago aplicando os possíveis descontos calculados.

Resolvendo:

```
#include<stdio.h>
#include<conio.h>

float preco;

int quantidade;
```

```
main(){

float valorTotal = 0;

float valorTotalDesc = 0;

float desc1 = 0;

float desc2 = 0;

printf("Digite o preço do produto  : ");

scanf("%f",&preco);

printf(''Digite a quantidade de produtos  :'');

scanf("%d",&quantidade);

valorTotal = preco * quantidade;

if(preco > 300){

desc1 =( preco * 10 )/100;

}

If(quantidade > 10){

desc2 = (preco * 8)/ 100;

}

valorTotalDesc  =  valorTotal - (desc1 + desc2);

printf("Valor total a ser pago : %0.2f \n",valorTotalDesc);

getch();
}
```

IF ENCADEADO

É quando uma condição depende de uma outra condição para executar o seu processamento.

```
// condição 1

IF(condição){
```

```
// condição 2
        IF(condição){
                }
}else{


}
```

Escreva um programa que leia um numero.

A) O programa deverá verificar se o mesmo é valido ou não.

obs.: numero valido é todo numero maior que 10 e par.

Resolvendo:

```
#include<stdio.h>
#include<conio.h>

int numero;
main(){
printf("Digite um numero : ");
scanf("%d",&numero);
IF(numero >10){
        IF(numero % 2 == 0){
                printf("Valido");
                }


}
getch();
```

}

Neste exemplo Foi necessário fazer duas verificações para saber se o numero e valido ou não. A primeira é saber se o numero é maior que 10 (o que não garante ele ser valido) e a segunda é saber se ele é par(quando passado nas duas verificações podemos garantir que ele é valido).

Operadores Relacionais

< menor

> maior

<= menor igual

>= maior igual

== igual há

<> diferente de

!= não igual há

Operadores Lógicos

São operados usados para unir condicionais.

&& → AND lógico, Retorna verdadeiro quando as condições testadas forem verdadeiras.

|| → OR lógico, Retorna verdadeiro quando pelo menos uma condição testada for verdadeira.

Exemplos.

Escreva um programa que leia um numero.

A) O programa deverá verificar se o numero e par e maior que 10, caso seja imprimir valido.

```
#include<stdio.h>
#include<conio.h>

int numero;

main(){

printf("Digite um numero : ");

scanf("%d",&numero);

IF(numero % 2 == 0 && numero > 10){
```

```
printf("Valido!");

}

getch();

}
```

No exemplo acima , perguntamos ao numero se ele é maior que 10 e par ao mesmo tempo, caso ele retorne verdadeiro para as duas condições testadas o and lógico retornara verdadeiro para o IF, que executara o bloco com a mensagem ("Valido").

OR → Ou lógico.

Escreva um programa que leia um numero.

A) O programa devera verificar se o numero valido ou invalido.

OBS:

Numero valido deve estar entre 0 e 10

```
#include<stdio.h>
#include<conio.h>

int numero;

main(){

printf("Digite  um numero : ");

scanf("'%d",&numero);


IF(numero < 0 || numero > 10){

printf("Invalido");

}else {


printf("Valido");

}
```

```
getch();
}
```

Exercícios:

12) 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!

13) Ler um valor e escrever se é positivo ou negativo (considere o valor zero como positivo).

14) 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.

15) 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.

16) 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).

17) Ler dois valores (considere que não serão lidos valores iguais) e escrever o maior deles.

18) Ler dois valores (considere que não serão lidos valores iguais) e escrevê-los em ordem crescente.

19) 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.

20) A jornada de trabalho semanal de um funcionário é de 40 horas. O funcionário que trabalhar mais de 40 horas receberá hora extra, cujo cálculo é o valor da hora regular com um acréscimo de 50%. Escreva um algoritmo que leia o número de horas trabalhadas em um mês, o salário por hora e escreva o salário total do funcionário, que deverá ser acrescido das horas extras, caso tenham sido trabalhadas

21) Ler o salário fixo e o valor das vendas efetuadas pelo vendedor de uma empresa. Sabendo-se que

ele recebe uma comissão de 3% sobre o total das vendas até R$ 1.500,00 mais 5% sobre o que ultrapassar este valor, calcular e escrever o seu salário total.

22) Faça um algoritmo para ler: número da conta do cliente, saldo, débito e crédito. Após, calcular e escrever o saldo atual (saldo atual = saldo - débito + crédito). Também testar se saldo atual for maior ou igual a zero escrever a mensagem 'Saldo Positivo', senão escrever a mensagem 'Saldo Negativo'.

23) Faça um algoritmo para ler: quantidade atual em estoque, quantidade máxima em estoque e quantidade mínima em estoque de um produto. Calcular e escrever a quantidade média ((quantidade média = quantidade máxima + quantidade mínima)/2). Se a quantidade em estoque for maior ou igual a quantidade média escrever a mensagem 'Não efetuar compra', senão escrever a mensagem 'Efetuar compra'.

(considere que o mês possua 4 semanas exatas).

Capítulo 3

Estrutura de comparação

Switch (Variável) {

case 'valor':

// Bloco do case

break; // Interrompe a execução do switch.

```
case 'valor':
        // Bloco do case
    break; // Interrompe a execução do switch.
default:
  // bloco do default
    break;
}
```

Switch → usado para criar menus, o switch é uma estrutura de comparação que tem como objetivo testar os possíveis valores de contido em uma variável.

Case → cada estrutura case do switch testa um possível valor que a variável a ser testada pode conter.

Break → é um comando aninhado a estrutura case usado para finalizar a execução do switch caso ele execute um case.

Default → É uma estrutura do switch usada para ser executada caso nenhum case seja executado.

Obs.: O switch é usado com números inteiros e char de uma única letra.

Exemplo:

```
#include<stdio.h>
#include<conio.h>


main(){

int opcao = 0;

printf("MENU :  \n");

printf("1- Adicionar :   \n");

printf("2 - Excluir ");

scanf("%d",&opcao);

switch(opcao){

        case 1 :

                printf("Adicionando um item !");

        break;

        case 2:

                printf(''Excluindo um item");

        break;

        Default:

                printf("Opcao invalida");
```

```
        Break;
}
getch();
}
```

No exemplo acima, nos criamos um menu para o usuário escolher entre duas opções de ações possíveis a serem feitas, pegamos esse valor e colocamos na variável opção, logo depois passamos a variável para o switch e o switch fica encarregado de testar as possíveis respostas de acordo com cada valor dentro da variável.

Exercícios:

24) Escreva um programa que leia dois números.

A) O programa deverá conter um menu para as 4 operações básicas da matemática. O usuário ira escolher entre uma das 4 e o programa deverá imprimir o resultado da operação escolhida.

25) Um posto está vendendo combustíveis com a seguinte tabela de descontos:

até 20 litros, desconto de 3% por litro Álcool

acima de 20 litros, desconto de 5% por litro

até 20 litros, desconto de 4% por litro Gasolina

acima de 20 litros, desconto de 6% por litro

Escreva um algoritmo que leia o número de litros vendidos e o tipo de combustível (codificado da seguinte forma: A-álcool, G-gasolina), calcule e imprima o valor a ser pago pelo cliente sabendo-se que o preço do litro da gasolina é R$ 3,30 e o preço do litro do álcool é R$ 2,90.

26) Escreva um algoritmo que leia as idades de 2 homens e de 2 mulheres (considere que as idades dos homens serão sempre diferentes entre si, bem como as das mulheres). Calcule e escreva a soma das idades do homem mais velho com a mulher mais nova, e o produto das idades do homem mais novo com a mulher mais velha.

27) Uma fruteira está vendendo frutas com a seguinte tabela de preços:

Até 5 Kg Acima de 5 Kg

Morango R$ 2,50 por Kg R$ 2,20 por Kg

Maçã R$ 1,80 por Kg R$ 1,50 por Kg

Se o cliente comprar mais de 8 Kg em frutas ou o valor total da compra ultrapassar R$ 25,00, receberá ainda um desconto de 10% sobre este total. Escreva um algoritmo para ler a quantidade (em Kg) de morangos e a quantidade (em Kg) de maçãs adquiridas e escreva o valor a ser pago pelo cliente.

Estrutura de repetição

Representam rotinas executadas automaticamente.

Loop → é uma ideia de repetição executada de acordo com uma condição verdadeira, se essa condição permanecer verdadeira o bloco continuara sendo executado até que a condição seja falsa.

Loop infinito → é uma repetição que não tem uma condição de saída.

Enquanto (condição)

// Bloco será repetido enquanto a condição for verdadeira

Fim Enquanto

While (condição){

Bloco que será repetido enquanto a condição for verdadeira

}

Ex:

```
main(){
        int i  = 0;
        while ( i < 10){
                printf("O valor do I : %d", \n);
                I = i + 1;
                }
```

getch();

}

Estrutura de repetição

FOR

Para i de 1 a 10 faça

// bloco do para

Fim para

Arg1 → variável que inicializa o loop

Arg2 → condição de existência.

Arg3 → incremento ou decremento da variável

```
For (arg1; arg2; arg3){

        // Bloco do for
}
```

Ex:

```
Main(){
For( int i = 0; i < 10;i++){
 Printf("%d \n",i);

}
Getch();
}
```

Break → Em uma estrutura de repetição o comando break interrompe a execução da estrutura.

Continue → Em uma estrutura de repetição ele desvia o fluxo para o inicio do estrutura.

Exercícios:

28) Escreva um algoritmo para ler 2 valores e 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. (utilizar a estrutura REPITA).

29) Reescreva o exercício anterior utilizando a estrutura ENQUANTO.

30) Acrescentar uma mensagem de 'VALOR INVÁLIDO' no exercício [44] caso o segundo valor informado seja ZERO.

31) Acrescentar uma mensagem de 'VALOR INVÁLIDO' no exercício [45] caso o segundo valor informado seja ZERO.

32) Escreva um algoritmo para ler as notas da 1a. e 2a. avaliações de um aluno, calcule e imprima a média (simples) desse aluno. Só devem ser aceitos valores válidos durante a leitura (0 a 10) para cada nota.

33) Acrescente uma mensagem 'NOVO CÁLCULO (S/N)?' ao final do exercício [48]. Se for respondido 'S' deve retornar e executar um novo cálculo, caso contrário deverá encerrar o algoritmo.

34) Escreva um algoritmo para imprimir os números de 1 (inclusive) a 10 (inclusive) em ordem crescente.

35) Escreva um algoritmo para imprimir os números de 1 (inclusive) a 10 (inclusive) em ordem decrescente.

36) Escreva um algoritmo para imprimir os 10 primeiros números inteiros maiores que 100.

37) Ler um valor N e imprimir todos os valores inteiros entre 1 (inclusive) e N (inclusive). Considere

Capítulo 4

Vetores

Serve para agrupar variáveis do mesmo tipo através do uso de índices.

Declaração de um vetor

int números[5];

int → tipo de dado do vetor.

Números → nome da variável vetorial.

[] → tamanho do vetor

“5” → significa que o vetor em questão pode guardar somente cinco variáveis do tipo inteiro.

Organização da informação do vetor

Os valores guardados em um vetor são acessados via índice onde o índice 0 representa a primeira posição do vetor e o ultimo índice é criado a partir da diferença do tamanho do vetor menos uma unidade (tamanho -1).

Primeiro índice → números [0].

Ultimo índice → números [tamanho -1] [4];

38) Escreva um algoritmo que permita a leitura dos nomes de 10 pessoas e armaze os nomes lidos em um vetor. Após isto, o algoritmo deve permitir a leitura de mais 1 nome qualquer de pessoa e depois escrever a mensagem ACHEI, se o nome estiver entre os 10 nomes lidos anteriormente (guardados no vetor), ou NÃO ACHEI caso contrário.

39) Escreva um algoritmo que permita a leitura das notas de uma turma de 20 alunos. Calcular a média da turma e contar quantos alunos obtiveram nota acima desta média calculada. Escrever a média da turma e o resultado da contagem.

40) Ler um vetor Q de 20 posições (aceitar somente números positivos). Escrever a seguir o valor do maior elemento de Q e a respectiva posição que ele ocupa no vetor.

41) O mesmo exercício anterior, mas agora deve escrever o menor elemento do vetor e a respectiva posição dele nesse vetor.

42) Ler um vetor A de 10 números. Após, ler mais um número e guardar em uma variável X. Armazenar em um vetor M o resultado de cada elemento de A multiplicado pelo valor X. Logo após, imprimir o vetor M.

43) Faça um algoritmo para ler 20 números e armazenar em um vetor. Após a leitura total dos 20 números, o algoritmo deve escrever esses 20 números lidos na ordem inversa.

44) Faça um algoritmo para ler um valor N qualquer (que será o tamanho dos vetores). Após, ler dois

vetores A e B (de tamanho N cada um) e depois armazenar em um terceiro vetor Soma a soma dos elementos do vetor A com os do vetor B (respeitando as mesmas posições) e escrever o vetor Soma.

Capítulo 5

Funções ou Módulos

Consiste em subdividir um programa mais complexo em programas menores que executam rotinas que são partes da rotina principal. Uma sub-rotina ou modulo pode ser visto como uma parte de um programa maior é como se o programa fosse divido em módulos e cada modulo executa-se somente uma parte do programa.

Utilização → Uma sub-rotina pode ser reutilizada varias vezes em um sistema e o programa fica mais organizado (reutilização e organização).

Ex:

```
 void nome_f(){


}
```

Void → palavra reservada para indicar que o a função não retorna um valor.

Nome_f → nome da função.

() → usado para a passagem de parâmetros para um função.

{} → bloco da função.

Função void ou procedimento

São funções que executam uma ação e não precisam retornar um valor de saída.

```
Void nome_f (){


}
```

Função Tipadas

São funções que executam uma ação e precisam retornar um valor de saída.

Tipo_de_dado **nome_f (){**

    return ***tipo_de_dado.***

}

Parâmetros

É a passagem de valor para uma função.

Ex:

Void nome_f (**int a, float b, char c**){

}

Tipo_de_retorno nome_f (**int a, float b, char c**){

    return tipo_de_retorno.

}

Exercícios :

Reescrever todos os exercícios feito nos capítulos anteriores.

Capítulo 6

Matriz

São Variáveis bidimensionais usadas para guardar valores em posições de linhas e colunas. Matrizes são usadas para agrupar variáveis de mesmo tipo e a disposição dos itens são feitas preenchendo linhas e colunas.

Declaração de uma matriz

int matriz[2][2];

int → Tipo de dado.

Matriz → nome da variável.

[][] → definir tamanho de linha e coluna

"2" → Quantidade de linhas, primeiro colchete.

"2" → Quantidade de colunas, define a quantidade elementos por linha, segundo colchete.

Os elementos de uma matriz são preenchidos linha a linha onde o primeiro colchete representa a linha e o segundo a coluna, o colchete da coluna representa cada elemento por linha.

Primeiro elemento de uma matriz → [0][0], é o elemento na posição zero de Linha e zero de coluna.

Ultimo elemento de uma matriz → [Tamanho -1][Tamanho -1], é o elemento na posição que é a diferença do tamanho da Linha (primeiro colchete) e o tamanho da coluna(segundo colchete).

Int matriz[2][2] → Quantidade de elementos é o produto de linha por coluna.

Int matriz[2][2] → Matriz Quadrática é a matriz onde o numero de linhas e igual ao numero de colunas.

Diagonal Principal de uma matriz é a aquela onde o índice da linha é igual ao índice da coluna.

Preenchendo uma matriz:

Int matriz[2][2]:

→ matriz[0][0] = 10;

→ matriz[0][1] = 20;

→ matriz[1][0] = 30;

→ matriz[1][1] = 40;

Matriz

1—Escreva um programa que leia uma matriz 2x2.

A. O programa deverá efetuar o calculo do determinante da matriz.

2—Escreva um programa que leia uma matriz 3x3.

a) O programa deverá efetuar o calculo da primeira linha.
b) O programa deverá efetuar o calculo da ultima coluna.
c) O programa deverá imprimir a diferença entre a primeira linha e a ultima coluna.

3—Escreva um programa que leia valores para uma matriz M 3x3.

a) Imprima a matriz criada e encontre a quantidade de números pares, a quantidade de números ímpares.

4—Escreva um programa que leia uma matriz 2x2.

a) O programa deverá criar multiplicar por dois os elementos com valores maiores que 10.
b) O programa deverá multiplicar por quatro os valores menores que 10.
c) O programa deverá imprimir a nova matriz.

5 – Escreva um programa que leia um vetor A de três posições e um vetor B de três posições.

a) O programa deverá gerar uma matriz 3x3 a partir da multiplicação do vetor A x Vetor B.
b) O programa deverá imprimir a configuração da matriz encontrada

Capítulo 7

Recursividade

É quando uma função tem uma chamada interna a ela mesma.

Ex:

```
Void nome_f(){

        IF(condição){

          Nome_f(); // chamada recursiva

                }


}
```

Toda função escrita de forma recursiva deve ter uma estrutura condicional para verificar se é possível efetuar a chamada da função ou não, se a função recursiva não tiver uma condicional para ocorrer um loop infinito.

Estrutura

São usados para agrupar variáveis de tipos diferentes de dados. Estruturas representam registros de dados que são coleções de dados heterogêneos.

Ex:

```
struct  Teste{
```

Int a;

Char b;

Float c;

};

A estrutura do tipo teste representa um tipo de dado novo o "Teste", o nome da estrutura é a referencia ao agrupamento das variáveis.

Um programa que use a estrutura Teste poderá conter 3 variáveis de tipos diferentes.

Declaração

Struct Teste teste.

Struct → palavra reservada para representar uma estrutura.

Teste → A estrutura criada.

teste → variável do tipo teste.

Recursividade

1 – Escreva um algoritmo que leia um numero.

A) O programa deverá executar a soma do numero digitado até o valor chegar a zero.

2 – Escreva um programa que leia um numero 'n' e execute o fatorial do numero lido.

3—Escreva um programa que leia um número e imprima a serie do fibonacci.

4 – Escreva um algoritmo que faça a entrada dos números digitados de forma recursiva e execute a soma dos números.

5 – Faça uma função recursiva que receba um vetor de 100 posições e retorne o somatório dos elementos pares (ou ímpares) do vetor.

Estrutura

1-- Escreva um programa que leia uma estrutura do tipo aluno (nome, matrícula e duas notas).

a) O programa deverá calcular a media do aluno.
b) O programa deverá imprimir a situação do aluno em relação a media encontrada:
Media >= 7 aluno aprovado, media < 4 aluno reprovado, senão aluno em recuperação.

2 – Escreva um programa que leia uma estrutura do tipo funcionário (nome,salário e função).

a) O programa deverá verificar qual a função do funcionário e de acordo com a função aplicar um bônus.
b) Funções: Programador, Gerente de projetos ou estagiário.
c) Os bônus serão: 12%, 17%, 7%.
d) O programa deverá imprimir o novo salário corrigido.

3—Escreva um programa que leia uma estrutura do tipo Cliente (nome, idade) e uma estrutura do Endereço (bairro, cidade, rua), relacionando uma estrutura com a outra.

a) O programa deverá imprimir os valores lidos.

4—Escreva um programa que leia quatro produtos (nome, preço, descrição) de um carrinho de compras.

a) O programa deverá calcular o valor a ser pago nas compras.
b) Caso o valor seja maior que 500 reais o programa deverá aplicar um desconto de 10%.
c) O programa deverá imprimir o novo valor a ser pago.

5—Escreva um programa que leia uma estrutura do tipo pessoa (nome, idade, sexo).

a) O programa deverá verificar se a pessoa em questão é maior ou menor de idade (idade >=18).