

# Programação em C

Prof. Eng. Uerlis Martins

# Objetivo

- Entender os conceitos de linguagem de programação computacional de baixo e alto nível;
- Desenvolver algoritmo estruturado e lógica de programação ;
- Desenvolver no aluno um raciocínio lógico para que ele possa implementar programas em linguagem computacional de alto nível.
- Desenvolver programas utilizando linguagem computacional para auxiliar na elaboração de cálculos e controle de dispositivos.

# INTRODUÇÃO À LÓGICA DE PROGRAMAÇÃO

# A LÓGICA DE PROGRAMAÇÃO

**Lógica de programação é a técnica de encadear pensamentos para atingir determinado objetivo.**

**Sequência Lógica são passos executados até atingir um objetivo ou solução de um problema.**

- Instruções são um conjunto de regras ou normas definidas para a realização ou emprego de algo.
- Exemplo: Receita de bolo

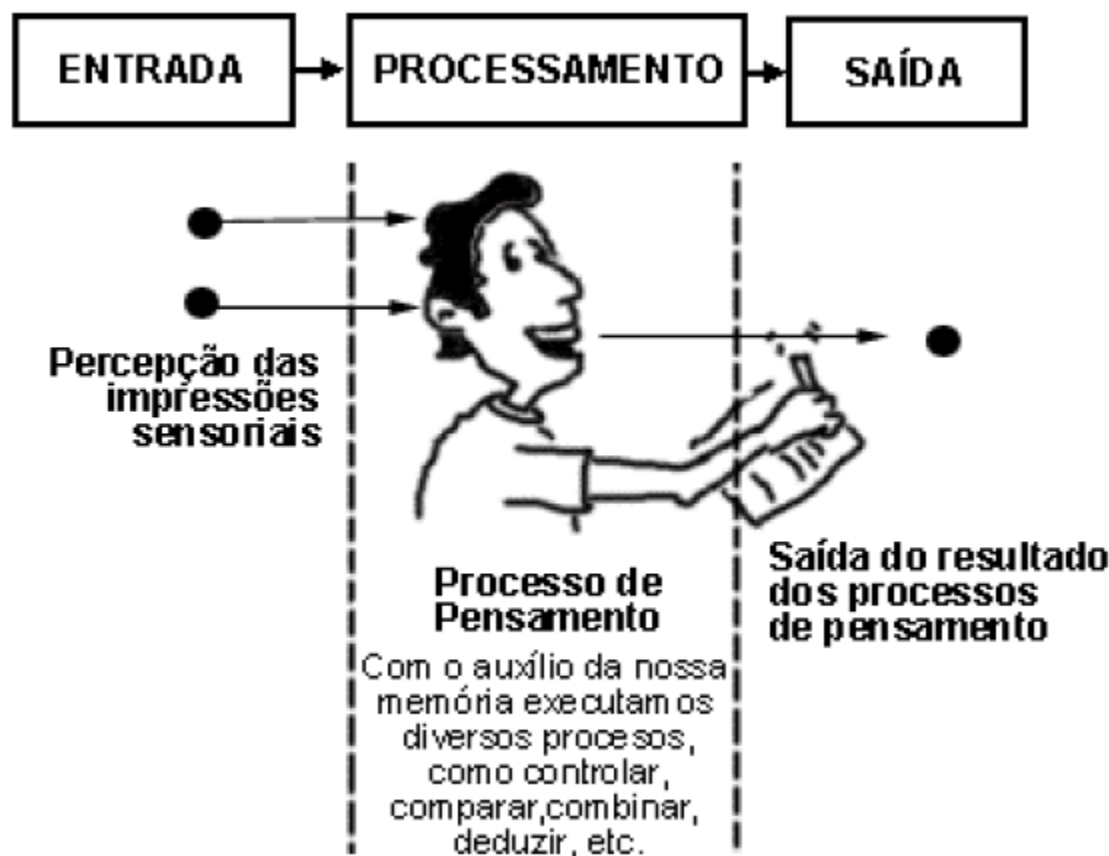
estes passos tem que ser executados um após o outro.

- É uma **seqüência** de procedimentos (passos) **finitos** que, se forem executados em determinado período de tempo, chegará ao seu **objetivo**.
- O **algoritmo** se define por uma **seqüência lógica** de passos que o **computador** executara fielmente para que se obtenha um resultado satisfatório.

# CONSTRUÇÃO DE UM ALGORITMO

1. Identificar o problema (objetivo) mediante leitura atenta de seu enunciado;
2. Retirar do enunciado as **entradas de dados**, ou seja, identificar os dados que devem ser fornecidos e, a partir deles, verificar se desenvolverão os cálculos/processamento;
3. Retirar do enunciado as **saídas de dados** que devem ser gerados como resultado da solução;
4. Determinar o que deve ser feito para transformar (processar) as entradas nas saídas desejadas;
5. Construir o algoritmo;
6. Testar a solução.





- Faça um algoritmo para somar dois números e multiplicar o resultado pelo primeiro número.

# **PORTUGUÊS ESTRUTURADO**

- É uma técnica narrativa denominada pseudocódigo.
- Tem como finalidade mostrar uma notação para elaboração de algoritmos, os quais serão utilizados na definição, criação e desenvolvimento de uma linguagem de programação.

# ESTRUTURA DE UM ALGORITMO

## INICIO DO ALGORITMO

### DECLARE

declaração de variáveis

## INICIO DO CORPO DO ALGORITMO

bloco de comandos

## FIM DO ALGORITMO

Inicio

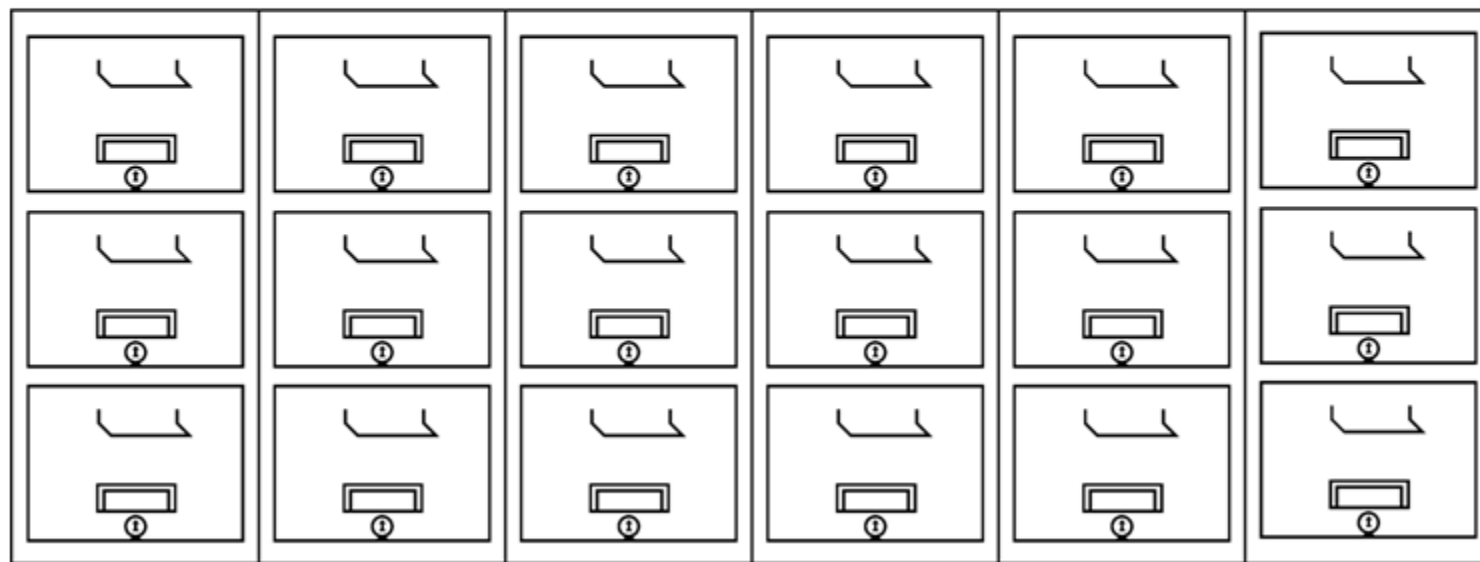
inteiro: n1, n2, d;

escreva("Digite o 1º numero:");  
leia(n1);  
escreva("Digite o 2º numero:");  
leia(n2);  
d=n1+n2;  
escreva("A soma e igual a ", d);

Fim

- Variável é uma **região de memória** usada para armazenar um determinado valor por um determinado espaço de tempo.
- Todo dado armazenado na memória de um computador deve ser **identificado segundo seu tipo**.
- O dado armazenado pode ser processado e **usado a qualquer momento**.
- **O armazenamento de um dado é realizado por meio de uma variável.**

Imagine a memória de um computador como um grande arquivo com várias gavetas nas quais é possível guardar apenas um valor por vez, e como em um arquivo, essas gavetas devem estar identificadas por uma “etiqueta” com um nome.



valor pode ser modificado ao longo de sua execução.



- Variável **possui nome de identificação**:
  - ✓ O nome utiliza um ou mais caracteres.
  - ✓ O primeiro caractere do nome deve ser alfabético.
  - ✓ Nome composto não usa espaço em branco. Para separar usa-se "\_" underline.
  - ✓ O nome não deve ser igual ao nome de identificação de um programa ou dos comandos da linguagem de programação em uso.



- Podem ser dos tipos:
  - Caracter
  - Inteiro
  - Real

<Tipo>: <nome>, <nome1>, <nomeN>;

- Regra de nomenclatura: começar por letra e sem caracteres especiais).
- Exemplos:
  - inteiro: x;
  - real: numero;

**Como atribuir um valor a uma variável??**

# COMANDOS

## ❑ Entrada de dados.

**Sintaxe** → leia (<nome da variável>);

Exemplos:

leia (n1) - O valor digitado será armazenado na variável n1.

leia (k) – Um caracter digitado será armazenado na variável k  
(Definida como caracter).

❑ No caso de utilizar leia (<lista de variáveis>), será respeitada a ordem da lista de variáveis, da esquerda para direita.

■ **Exemplo:** leia (n1, k, n2)

# COMANDOS

## ❑ Saída de dados.

**Sintaxe** → escreva (<nome da variável >) ; ou  
escreva (<lista-de-variáveis>);

**Obs:** escreva → imprime a variável na tela e o cursor vai para uma nova linha.

## Exemplos:

**escreva (n1)** - Será mostrado na tela o conteúdo da variável n1.

**escreva (“O caracter digitado foi ”, k)** - Será mostrado o texto entre “” e depois o conteúdo da variável k.

- Possuem operadores aritméticos e operandos são variáveis ou constantes do tipo numérico (inteiro ou real)
- Operadores
  - (+) soma, (-) subtração, (\*) multiplicação, (/) divisão, pot(base,expoente), rad(), div, mod

# OPERADORES

- Aritmético (+, -, \*, /, %)
- Atribuição (=, +=, -=, \*=, /=, %=)
- Relacional (==, !=, <, <=, >, >=)
  - Lógico (&&, ||)



- Soma +
- Subtração -
- Multiplicação \*
- Divisão /

Hierarquia	Operação
1	Parênteses
2	Função
3	-, + (unários)
4	^
5	*, /, \, %
6	+, -

- **Simples =**
  - Incremental +=
  - Decremental -=
  - Multiplicativa \*=
    - Divisória /=
    - Modular %=

- Igualdade ==
- Desigualdade !=
  - Menor <
- Menor ou igual <=
  - Maior >
- Maior ou igual >=

- “E” lógico &&
- “OU” lógico ||

# ESTRUTURAS CONDICIONAIS

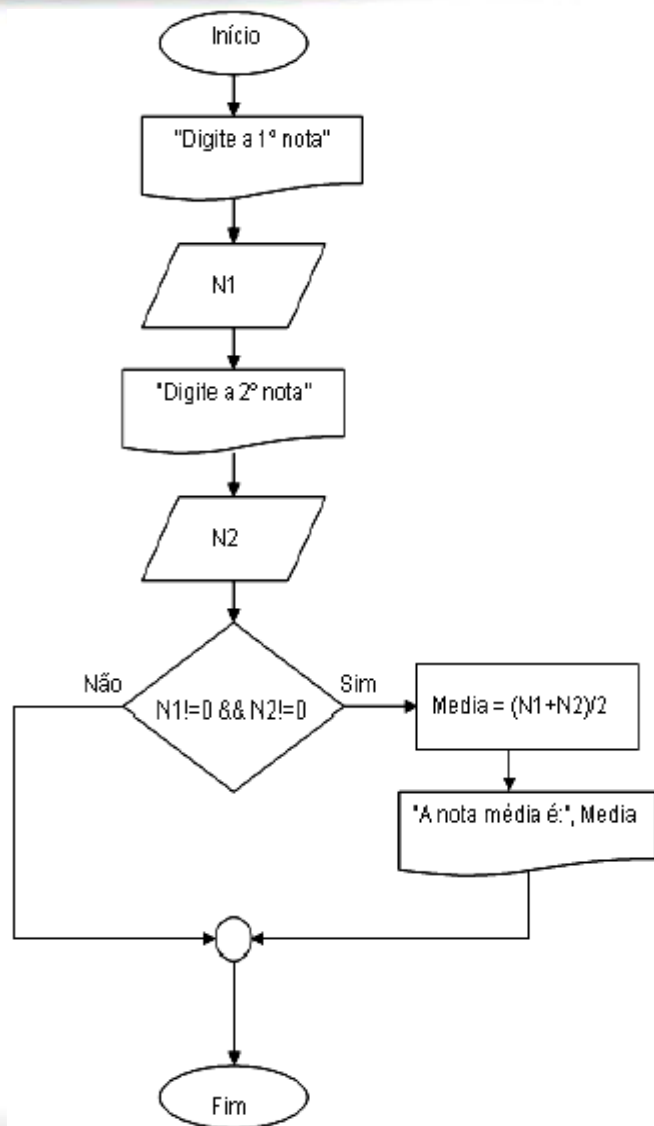


Calcular a média final dos alunos. Pelas provas os alunos receberão 2 notas: N1, N2 sendo que as notas ***devem ser diferentes de zero.***

Observe que temos uma situação condicional nesse caso:

- ❑ **Se os valores** digitados não forem iguais a zero ou seja, se os valores digitados forem diferentes de zero o cálculo será efetuado e o resultado será apresentado. (Exemplo 1)
  
- ❑ **Senão, ou seja, se os valores digitados não atenderem a condição devemos ou não informar** ao usuário que o valor é incorreto para o cálculo. Note que estaremos desconsiderando os valores negativos na lógica pois a operação lógica verifica apenas a condição de igualdade a zero. (Exemplo 2)

## Resolvendo com Seleção Simples



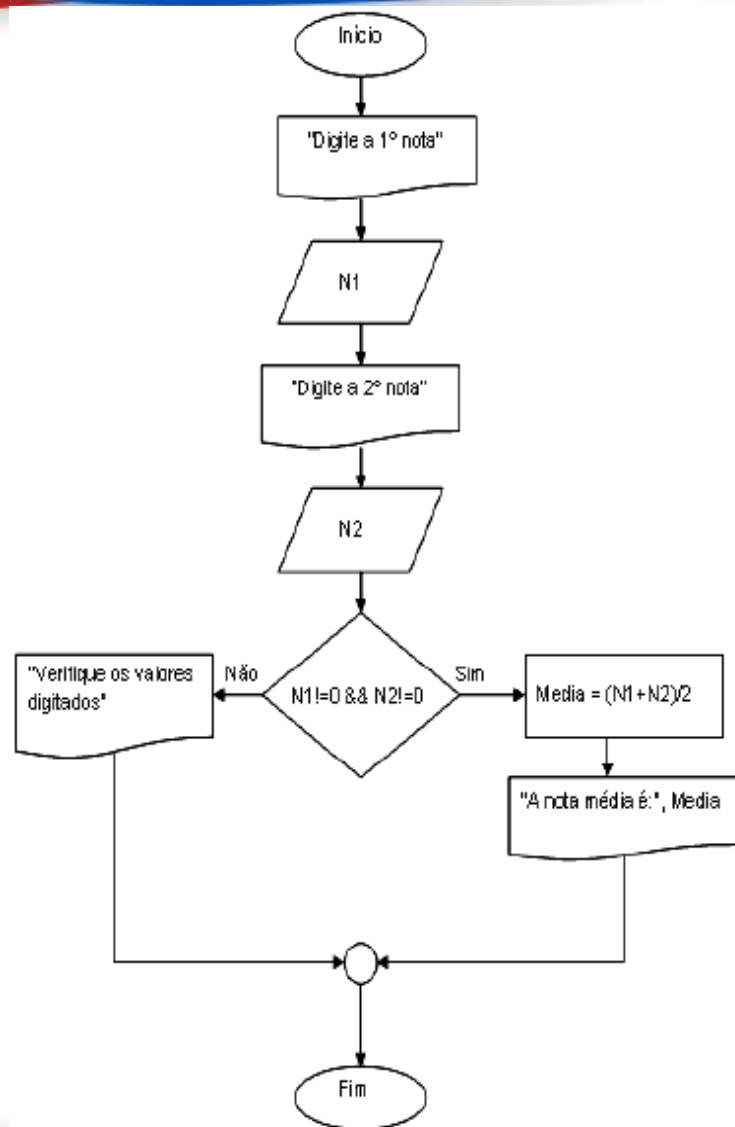
### INICIO do algoritmo

```
| Real: N1,N2,Media;  
| Escreva ("Digite a 1º nota");  
| Leia (N1);  
| Escreva ("Digite a 2º nota");  
| Leia (N2);  
| Se (N1 !=0 && N2!=0)  
| { Media = (N1+N2)/2;  
|   Escreva ("A nota média é:",Media);  
| };
```

### FIM do algoritmo



## Resolvendo com seleção composta



### INICIO do algoritmo

```
| Real: N1,N2,Media;  
| Escreva ("Digite a 1º nota");  
| Leia (N1);  
| Escreva ("Digite a 2º nota");  
| Leia (N2);  
| Se (N1 !=0 && N2!=0)  
| { Media = (N1+N2)/2;  
|   Escreva ("A nota média é:",Media);  
| };  
| Senao  
| {Escreva("Verifique os valores digitados");  
| };
```

### FIM do algoritmo

# EXERCÍCIOS

- 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!**”.

- Da questão anterior identifique quais são os dados de entrada, o processamento e as informações de saída.
- Faça um algoritmo para “Calcular o estoque médio de uma peça”, sendo que:

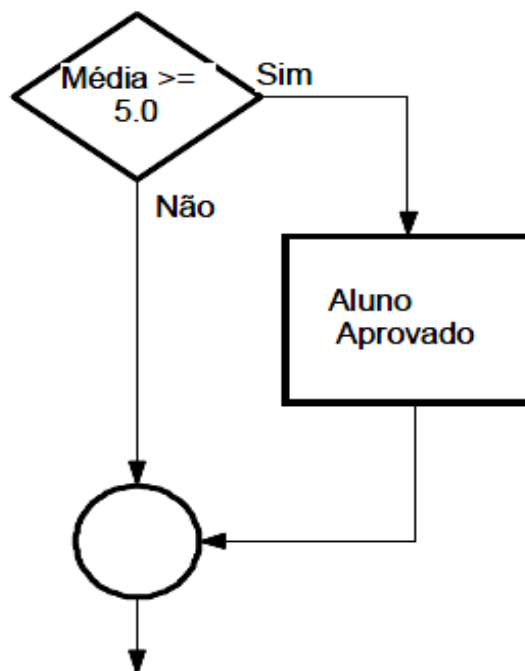
$$\text{ESTOQUE M\u00c9DIO} = (\text{QUANTIDADE M\u00cdNIMA} + \text{QUANTIDADE M\u00c1XIMA}) / 2.$$

- Desenvolva um programa que leia um valor em Dólar, converta para reais e exiba o resultado.
- Elabore um programa que leia 4 números, calcule o quadrado de cada um e some todos os resultados. Exibir resultado final.

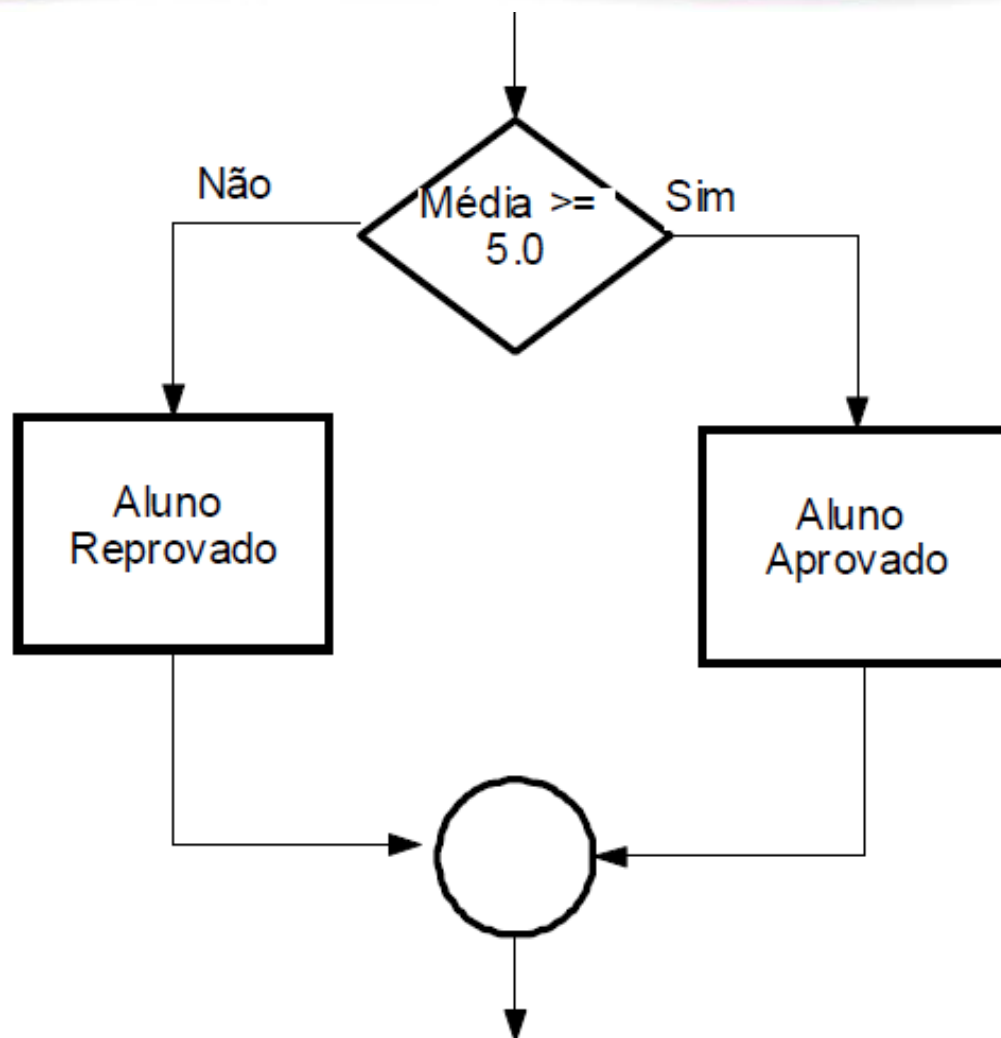
# EXEMPLO 3

Imagine um algoritmo que determinado aluno somente estará aprovado se sua média for maior ou igual a 5.0, veja no exemplo de algoritmo como ficaria.

**SE MEDIA  $\geq$  5.0 ENTÃO ALUNO APROVADO**



# EXEMPLO 4





A nota final de um estudante é calculada a partir de três notas atribuídas, respectivamente, a um trabalho de laboratório, a uma avaliação semestral e a um exame final. As média das três notas mencionadas obedece aos seguintes pesos:

<b>Nota</b>	<b>Peso</b>
Trabalho de laboratório	2
Avaliação semestral	3
Exame final	5

Elabore um algoritmo para um programa que receba as três notas, calcule a média ponderada do aluno e classifique seu desempenho (conceito) de acordo com a tabela seguinte:

<b>Média</b>	<b>Conceito</b>
[8,0; 10,0]	A
[7.0; 8,0)	B
[6.0; 7.0)	C
[5.0; 6.0)	D
[0.0; 5.0)	E



Desenvolva um programa que leia duas notas dos alunos, calcule a media e verifique:

- Se  $\text{media} \geq 7$  aluno aprovado;
- Se  $\text{media} < 4$  aluno reprovado;
- Se  $4 \leq \text{media} < 7$  aluno na final;

Caso o aluno tenha ido a final, pergunte-o se o mesmo deseja calcular o quanto precisará na final ou a situação final dele.

Se  $M \geq 7$  Aprovado

Se  $4 \leq M < 7$  Prova Final

Se  $M < 4$  Reprovado direto

$$M_f = \frac{6xM + 4xP_f}{10}, \quad P_f = \frac{10M_f - 6M}{4}$$

Se  $M_f \geq 5$  Aprovado

Se  $M_f < 5$  Reprovado

Desenvolver um algoritmo para calcular uma conta de água. O custo da água varia dependendo do tipo do consumidor: residencial, comercial ou industrial.

A regra para calcular a conta é:

- **‘R’-residencial:** R\$ 5,00 de taxa mais R\$ 0,05 por m<sup>3</sup> gastos;
- **‘C’-comercial:** R\$ 500,00 para os primeiros 80 m<sup>3</sup> gastos mais R\$ 0,25 por m<sup>3</sup> gastos acima dos 80 m<sup>3</sup>;
- **‘I’-industrial:** R\$ 800,00 para os primeiros 100 m<sup>3</sup> gastos(taxa fixa) e mas R\$ 0,04 por m<sup>3</sup> gastos acima dos 100 m<sup>3</sup>;

O algoritmo devera ler o tipo do consumidor (residencial, comercial e industrial) e o seu consumo de água em metros cubos. Como resultado imprimir a conta do cliente e o valor em real a ser pago pelo mesmo.

- Faça um programa que receba dois números e execute uma das operações listadas a seguir de acordo com a escolha do usuário. **Se for digitada uma opção inválida mostrar a mensagem de erro e terminar a execução do programa.** As opções são:

Escolha do usuário	Operação
+ Adição	Media entre os números digitados
- Subtração	Diferença do maior pelo menor
* Multiplicação	Produto entre os números digitados
/ Divisão	Divisão do primeiro pelo segundo

OBS: Lembre-se de que na operação de divisão o segundo número deve ser diferente de zero.

- 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.

- Elabore um algoritmo que leia 3 valores e escrever o maior deles.



# ESTRUTURAS DE REPETIÇÃO



As estruturas de repetição permitem **executar mais de uma vez um mesmo trecho de código**. Trata-se de uma forma de executar blocos de comandos somente **sob determinadas condições**, mas com a opção de **repetir o mesmo bloco quantas vezes for necessário**.

As estruturas de repetição são úteis, por exemplo, para repetir uma série de operações semelhantes que são executadas para todos os elementos de uma lista ou de uma tabela de dados, ou simplesmente para repetir um mesmo processamento até que uma certa condição seja satisfeita.

Faça {  
Executa enquanto a proposição for verdadeira  
} Enquanto ( **condição de repetição** );

Enquanto ( **condição de repetição** ) {  
Executa enquanto a proposição for verdadeira  
}

# EXEMPLO:

- Desenvolva um programa que leia um valor em Dólar, converta para reais e exiba o resultado. Logo após escrever a mensagem **“Converter outro valor? [S]im [N]ão?”** e solicitar um resposta. Se a resposta for "S", o programa deve ser executado novamente, caso contrário deve ser encerrado.

# EXEMPLO II:

- Escreva um algoritmo que leia uma quantidade indeterminada de números. Se o usuário digitar o número 0 (zero) o algoritmo mostrará a soma dos números e a quantidade de números digitada.

# PREENCHA A TABELA ABAIXO:

Inicio

Inteiro: x,y;

x=1;

y=5;

Enquanto(x<y){

x=x+ 2;

y=y+ 1;

}

Escreva("valores:",x,y);

Fim

X	Y
1	5

Inicio

Inteiro: x,y;

x=1;

y=5;

{

x=x+ 2;

y=y+ 1;

} Enquanto(x<y);

Escreva("valores:",x,y);

Fim

X	Y
1	5

- Escreva um programa para ler 2 notas de um aluno, calcular e imprimir a média final. Logo após escrever a mensagem "**Calcular a média de outro aluno [S]im [N]ão?**" e solicitar uma resposta. Se a resposta for "S", o programa deve ser executado novamente, caso contrário deve ser encerrado imprimindo a quantidade de alunos aprovados.



# Exercício 2

- Reescreva o programa do anterior, para que seja impressa no final, a quantidade de alunos aprovados, reprovados e que ficaram em exame.

# EXERCÍCIO 3

Construa um algoritmo que leia 500 valores inteiros e positivos e:

- **Encontre o maior valor**
- **Encontre o menor valor**
- **Calcule a média dos números lidos**

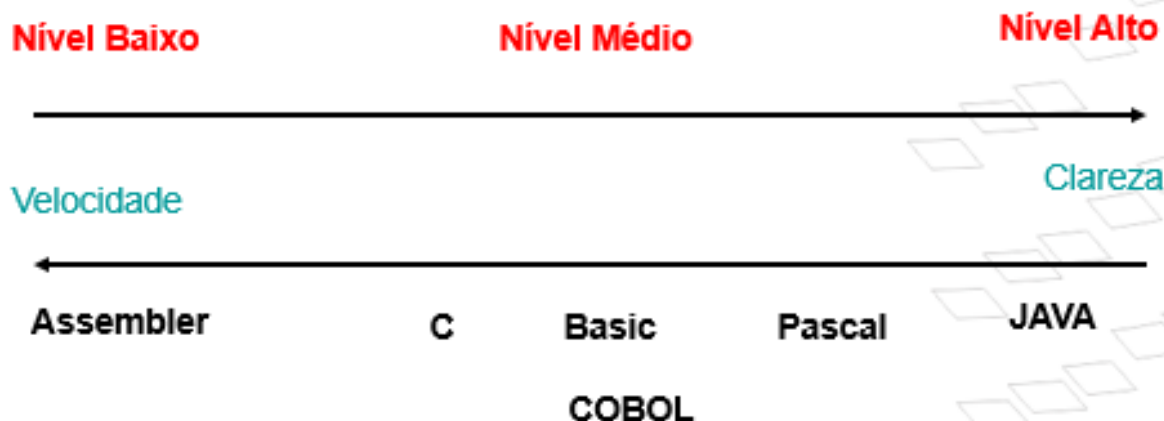
A primeira versão de C foi criada por Dennis Ritchie em 1972 nos laboratórios Bell.

Linguagem de programação genérica que é utilizada para a criação de programas diversos como:

- Processadores de texto, planilhas eletrônicas;
- Sistemas operacionais, programas para automação;
- Gerenciadores de bancos de dados, programas de projeto assistido por computador, etc...
- Linguagem que pode ser utilizada atualmente na programação de quase todos os microcontroladores;
- Há microcontroladores com instruções otimizadas para programação em C;
- O compilador transforma as instruções em C no código em assembly;

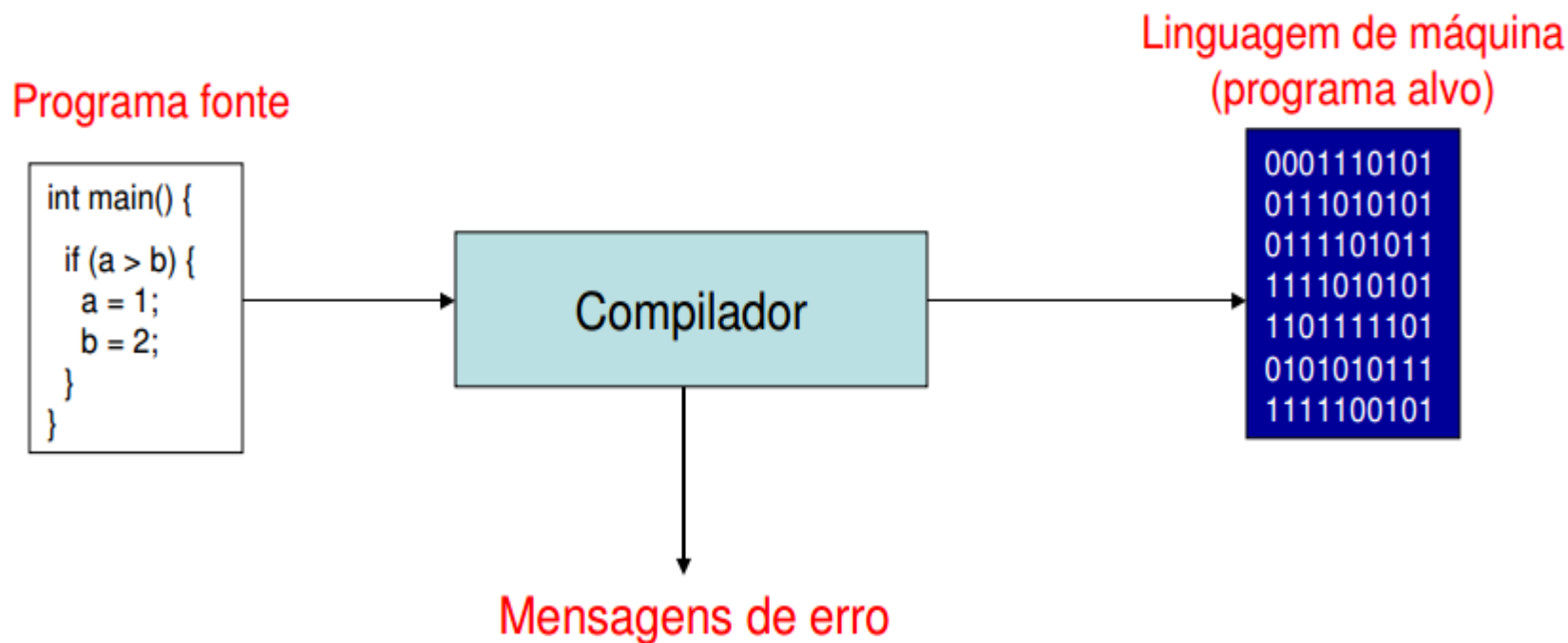
Programar em linguagens de alto nível, como a C, apresenta como principal vantagem uma menor exigência na interação do projetista com o hardware, no que diz respeito ao controle do mesmo (ajuste de bancos de registradores, sequências de inicialização e etc ), permitindo, desta forma, que o projetista dedique o seu tempo basicamente à lógica do problema e não aos detalhes internos do chip.

Deve-se entender **NÍVEL ALTO** como sendo a capacidade da linguagem em compreender instruções escritas em “dialetos” próximos do inglês (Ada e Pascal, por exemplo) e **NÍVEL BAIXO** para aquelas linguagens que se aproximam do **Assembly**, que é a linguagem própria da máquina, compostas por instruções binárias e outras incompreensíveis para o ser humano não treinado para este propósito.

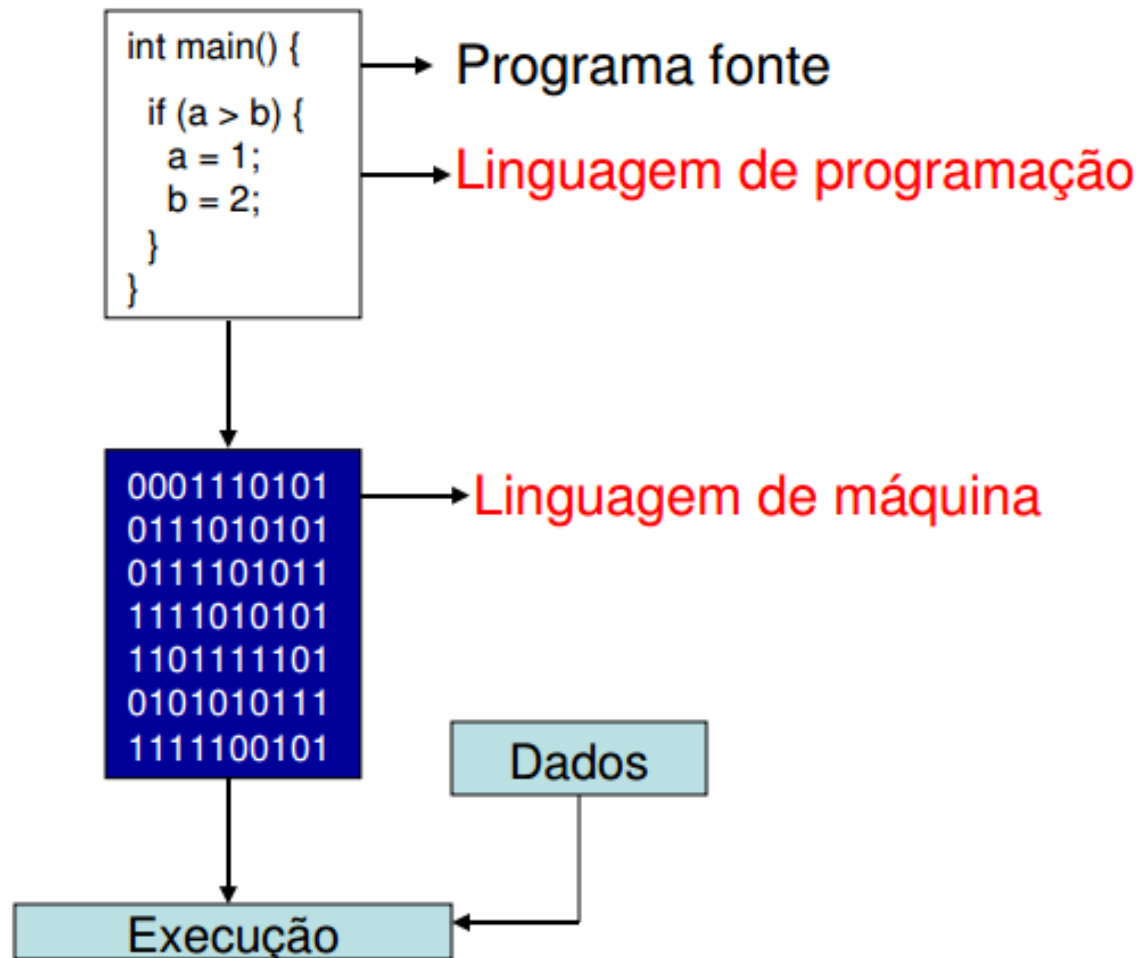


## VISÃO GERAL DE UM COMPILADOR.

Classicamente, um compilador traduz um programa de uma linguagem textual facilmente entendida por um ser humano para uma linguagem de máquina, específica para um processador e sistema operacional.



## VISÃO GERAL DE UM COMPILADOR.





## CARACTERÍSTICAS DA LINGUAGEM C

- Programas estruturados.
- Total interação com o Sistema Operacional.
- Código compacto e rápido, quando comparado ao código de outras linguagens de complexidade análoga.
- C é uma linguagem compilada: lê todo o código fonte e gera o código objeto (ling. de máquina) uma única vez.
- Sempre que o código fonte for alterado ele deve ser novamente compilado.
- **C** é “case sensitive”, assim a palavra **pedra** é diferente de **PEDRA**.
- Deve-se listar antecipadamente todas as variáveis utilizadas no programa.
- C possui palavras reservadas que possuem significado especial para a linguagem e tais palavras devem ser escritas com letras minúsculas

**auto, break, case, if, for, while, begin, end, continue, return, const,....**



## ESTRUTURA BÁSICA

Quatros elementos estão presentes num programa em linguagem C, são eles:

DIRETIVAS DE COMPILAÇÃO, DEFINIÇÃO DE DADOS, COMENTÁRIOS E BLOCOS COM INSTRUÇÕES E FUNÇÕES.

EXEMPLO:

```
# include<.....>           ( Diretiva de compilação )  
// estas barras servem para comentários que ocupam uma linha  
/* aqui podem ser escritos comentários que podem ocupar  
mais de uma linha */  
int    i, tempo; / definição de dados ou declaração de variáveis  
float  temperatura / definição de dados ou declaração de variáveis  
main() / função principal e obrigatória para o início do programa em C  
{  
  Instruções do programa principal  
}
```

## ESTRUTURA BÁSICA

### DIRETIVA DE COMPILAÇÃO

São instruções para o compilador, e não para o programa que será gerado.

As diretivas informam, por exemplo, o processador para o qual o código deverá ser gerado, o valor do clock que será usado pela cpu e etc.

As diretivas sempre começam com # e são exemplo de diretivas:

```
# include<stdio.h> /* esta diretiva inclui no processo as configurações padrões de entrada e saída */
```

```
# include<p18f4550.h> /*é a diretiva que inclui no processo de compilação as defini-ções do chip */
```

O termo .h que compõe a diretiva, indica um Header File, que é o mesmo que cabeçalho , da linguagem C.

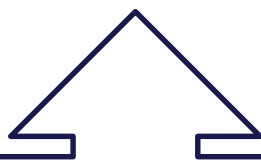
## DEFINIÇÃO DE DADOS OU DECLARAÇÃO DE VARIÁVEIS

Uma variável em C é um espaço de memória reservado, para armazenar um certo tipo de dado, que a cada tempo pode assumir valores diferentes .

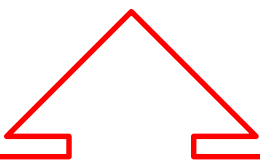
Sintaxe simples para declaração de uma variável:

**<tipo da variável>**

**<lista\_de\_variáveis>**



Tipo da informação  
que a variável vai  
guardar; existem  
5 tipos básicos.



Lista com o nome de  
todas as variáveis  
sendo criadas com  
este tipo, separadas  
por vírgulas.

**Exemplo:**

```
int    i, tempo;  
float  temperatura;  
char   Ana, João;
```

## DEFINIÇÃO DE DADOS OU DECLARAÇÃO DE VARIÁVEIS

Existem cinco tipos primitivos de dados na linguagem C e um tipo de variável ocupará uma quantidade de memória diferente da outra

TIPO	BIT	BYTES	ESCALAS
char	8	1	-128 a 127
int	16	2	-32768 a 32767
float	32	4	3.4E-38 a 3.4E+38
double	64	8	1.7E-308 a 1.7E+308
void	0	0	Nenhum valor

## DEFINIÇÃO DE DADOS OU DECLARAÇÃO DE VARIÁVEIS

Com exceção do void, os tipos de dados básicos podem ter vários modificadores precedendo-os e objetivos destes é modificar o tipo primitivo para adaptá-lo as necessidades da situação.

Tipo	Num de bits	Intervalo	
		Início	Fim
char	8	-128	127
unsigned char	8	0	255
signed char	8	-128	127
int	16	-32.768	32.767
unsigned int	16	0	65.535
signed int	16	-32.768	32.767
short int	16	-32.768	32.767
unsigned short int	16	0	65.535
signed short int	16	-32.768	32.767
long int	32	-2.147.483.648	2.147.483.647
signed long int	32	-2.147.483.648	2.147.483.647
unsigned long int	32	0	4.294.967.295
float	32	3,4E-38	3.4E+38
double	64	1,7E-308	1,7E+308
long double	80	3,4E-4932	3,4E+4932

## DEFINIÇÃO DE DADOS OU DECLARAÇÃO DE VARIÁVEIS

Conforme já mostrado, definir uma variável é criá-la na memória (alocá-la), dar a ela um nome e especificar o tipo de dado que nela vai armazenar. As variáveis podem ter qualquer nome até 32 caracteres, entretanto, deve começar com uma letra ou sublinhado ( \_ ) e os caracteres subsequentes devem ser letras, números ou sublinhados.

Além disso, o nome das variáveis não pode ser igual a uma palavra reservada, nem igual ao nome de uma função declarada pelo programador ou pelas bibliotecas do C.

Sintaxe completa a para criação de uma variável em C;

**<modificador> <tipo da variável> nome\_da\_variavel;**

```
Int soma ;  
unsigned char i,j,k ;  
float salario;  
unsigned int idade;  
short int y;
```



## DEFINIÇÃO DE DADOS OU DECLARAÇÃO DE VARIÁVEIS

Agora que já se sabe criar uma variável, resta inicia-la. Para atribuir valores utiliza-se o sinal de igualdade. Além disso, a depender do local onde as variáveis são criadas e iniciadas, elas podem ser classificadas em variáveis locais ou globais.

- Pode ser feita na própria declaração:

**int x = 0;**

**float num = 4.58;**

- É possível inicializar mais de uma variável ao mesmo tempo:

**int x=0, y=10;**

- É possível misturar declaração e inicialização na mesma linha:

**int x, y=10, z;**

Obs.: Uma variável recém criada armazena um valor indeterminado



## COMENTÁRIOS

Comentários são informações anexadas ao programa fonte e ignoradas pelo compilador, que permitem ao programador ou outros entenderem o significado do que esta sendo feito.

É reconhecido como uma boa prática de programação comentar o máximo possível as linhas e os grandes blocos de funções, explicando o quê está sendo feito e o porquê, pois após algum tempo, normalmente, nem mesmo o criador do programa será capaz de lembrar de tudo, além disso, outros podem precisar compreender o quê foi desenvolvido para efeitos de melhoria ou adaptação.

## BLOCOS COM INSTRUÇÕES E FUNÇÕES

Uma função ( as vezes chamada de rotina) pode ser entendida como um “pequeno programa” que realiza determinada operação.

Exemplo:

```
LeTeclas()  
{  
    ---  
    ---  
    X = ABS(tecla_lida);  
    ---  
    ---  
}  
  
main ()  
{  
    ---  
    ---  
    LeTeclas();  
    ---  
    ---  
}
```

No programa ao lado a função main “chama” uma rotina (lerTeclas) e a rotina “chama” a função ABS para calcular o valor absoluto da tecla lida.

## COMANDO DE ENTRADA E SAÍDA

Funções mais comuns (**exigem `stdio.h`**):

- **printf :**

escreve na tela texto e valores com um formato especificado e retorna o número de caracteres escritos.

- **scanf :**

lê dados do teclado, converte os valores de acordo com o formato especificado e coloca-os nas variáveis cujos **endereços são** passados na lista de argumentos (retorna o número de caracteres lidos).

## COMANDO DE SAÍDA

- ***printf*** :

### Sintaxe:

`printf ("string_de_controle", lista_de_variáveis);`

Onde a string de controle :

- representa **tudo que vai ser exibido na tela;**
- contém: texto, os tipos das variáveis com suas respectivas posições e caracteres de controle de texto;

Onde a lista de variáveis:

- utilização de códigos de controle, com a notação **%** ;
- para cada código de controle deve haver uma variável na lista de variáveis.

## COMANDO DE SAÍDA

- ***printf*** :

- **Formatações de variável:**

- %d - impressão de inteiros decimais
- %ld - impressão de inteiros longos
- %o - impressão de octal
- %x - impressão de hexadecimal
- %u - impressão de decimal sem sinal
- %c - impressão de caracteres
- %f - impressão de real, com 6 decimais
- %s - impressão de cadeia de caracteres (strings)
- %% - impressão do símbolo %

## COMANDO DE ENTRADA

### **scanf :**

- Sintaxe:

**scanf** (“**controle**”, &**endereços\_de\_variáveis**);

Onde o controle:

- contém apenas a formatação das variáveis utilizando os códigos de controle, com %

Onde a lista de endereços:

- indica o endereço de memória de cada variável a ser lida com **& na frente**



## OPERADORES ARITMÉTICOS

+ Adição      - subtração      \* multiplicação      / divisão      % resto da divisão

++ incremento na variável ( x++ é equivalente a x=x+1)

-- decremento na variável ( x-- é equivalente a x = x- 1)

Obs. O operador de divisão pode ser usado com **inteiros e reais**. Porém, para os inteiros, apenas a parte inteira é retornada como resultado.

## OPERADORES DE ATRIBUIÇÃO

Sintaxe:

*<variável> = <expressão>;*

a = 8;

c = b = a = 2;

Este operador atribui o valor de expressão **a variável e retorna este valor**; por isso o segundo exemplo é válido.

## OPERADORES PARA COMPARAÇÃO OU OPERADORES RELACIONAIS

< Menor que

> Maior que

== igual a

<= Menor ou igual que

>= Maior ou igual que

!= diferente de



## OPERADORES LÓGICOS

- && (E)** AND lógico ou relacional (todas as condições verdadeiras)
- || (OU)** OR lógico ou relacional (uma das condições é verdadeira)
- ! (NÃO)** NOT lógico ou relacional (vê se a condição é TRUE ou FALSE)

## OPERADORES BINÁRIOS

- & (E)** AND binário (bit a bit nas variáveis)
- | (OU)** OR binário (bit a bit nas variáveis)
- ~ (NÃO)** NOT binário (inverte o estado de cada bit da variável)

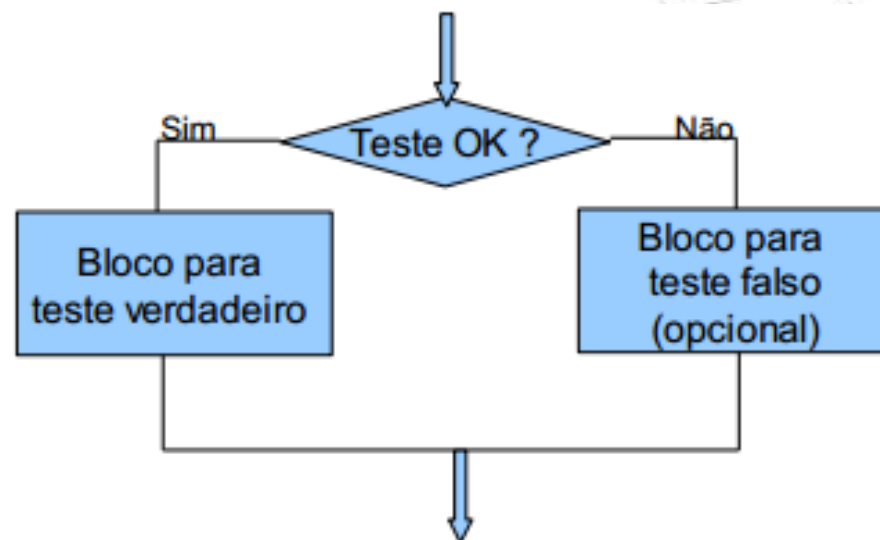
# Estruturas em C

If... Else  
Switch...case  
while  
Do... While

## A ESTRUTURA IF

Pode-se entender a estrutura if como um teste simples, onde o programa testará uma condição estabelecida e caso a mesma seja verdadeira, uma ação será executada, caso contrário, executará uma outra ação.

- SE (teste = ok!) *"executa esta(s) declaração(ões)"*
- SE (teste = ok!) *"executa esta(s) declaração(ões)"*  
SENÃO *"executa esta(s) outras declaração(ões)"*



## A ESTRUTURA FOR

Com esta estrutura é possível executar uma instrução ou blocos de instruções por um número determinado de vezes. Devido a essa característica, o comando for também é conhecido como laço ( loop ).

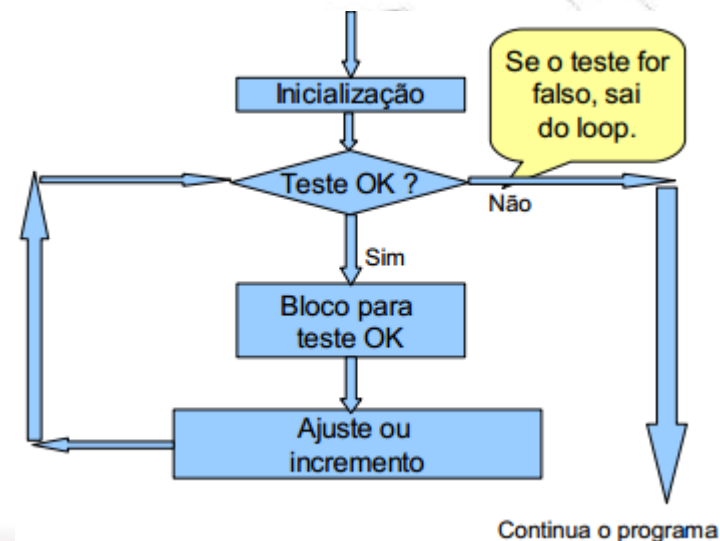
## SINTAXE DO COMANDO FOR

```
for ( inicialização ; condição de teste ; ajuste ou incremento )  
    instrução ;
```

```
ou  
for ( inicialização ; condição de teste ; ajuste ou incremento )  
{  
    ( grupo de instruções )  
}
```

Para gerar um loop infinito:

```
for ( ; ; )    instrução ou função que será executada indefinidamente  
ou  
for ( ; ; )  
{  
    Grupo de instruções que serão executadas indefinidamente  
}
```



Continua o programa

## A ESTRUTURA WHILE

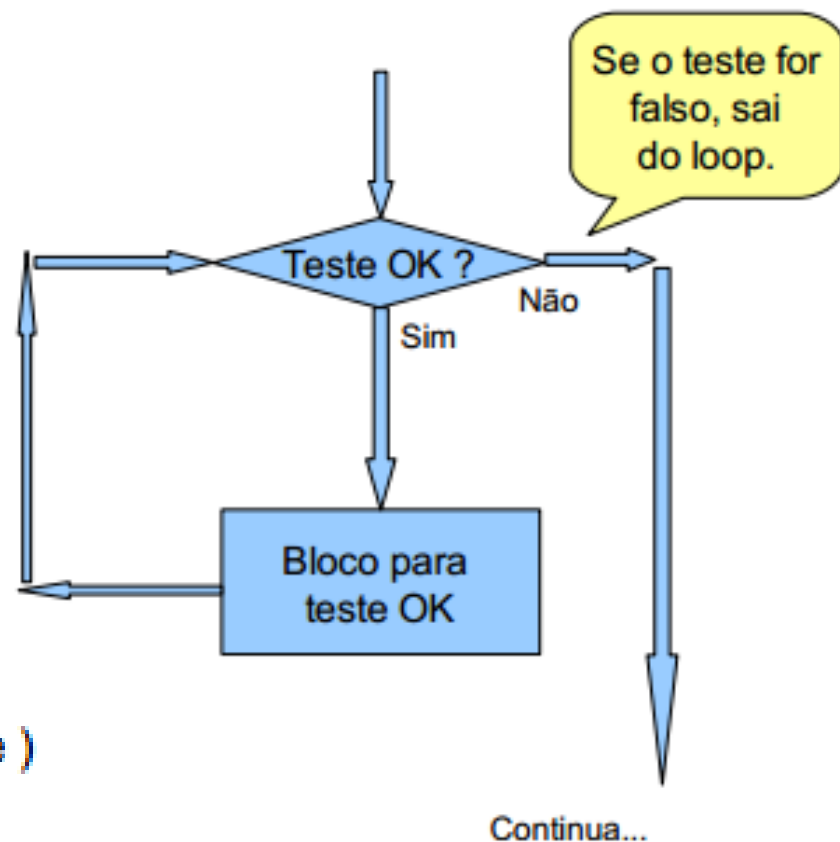
Esta estrutura executa uma instrução ou um bloco de instruções **enquanto** a condição de teste for verdadeira

```
while ( teste )  
    instrução para teste verdadeiro  
ou
```

```
while ( teste )  
{  
    ( grupo de instruções para teste verdadeiro )  
}
```

### Loop infinito com while

```
while ( 1 )  
{  
    ( declarações executadas eternamente )  
}
```



## EXERCÍCIO

1) Implemente um programa em Linguagem C para calcular a área de um quadrado. Esse programa deve receber o tamanho do lado do quadrado e imprimir a sua área na tela.

2) O índice de Massa Corporal (IMC) é uma fórmula que indica se um adulto está acima do peso, se está obeso ou abaixo do peso ideal considerado saudável. A fórmula para calcular o Índice de Massa Corporal é:  $IMC = \text{peso} / (\text{altura})^2$ .

A Organização Mundial de Saúde usa a seguinte tabela para determinar a condição de um adulto:

Condição	IMC em adultos
Abaixo do peso	Abaixo de 18.5
No peso normal	Entre 18.5 e 25
Acima do peso	Entre 25 e 30
Obeso	Acima de 30

Implemente um programa em Linguagem C para calcular o IMC de um adulto. Esse programa deve receber a altura e o peso da pessoa e exibir o IMC na tela.

## EXERCÍCIO

- 3) Desenvolva um programa para contar até 100, a partir de um número menor que 100.
- 4) Desenvolva um programa para contar de 0 até 100, sendo que o intervalo da contagem deve ser escolhido pelo usuário.
- 5) Desenvolver um programa para calcular a média entre duas notas de uma aluno e informa se ele está aprovado ou em prova final, baseado na média de aprovação igual a 7,0.
- 6) Desenvolva um programa que a partir da idade de uma pessoa, mostre se ela não pode votar ( idade inferior a 16 anos ), se o voto é facultativo ( idade igual a 16 e 17 anos, ou maior do que 70 anos), ou se o voto é obrigatório (idade entre 18 e 70 anos, incluindo estes valores).



## EXERCÍCIO

- 7) Desenvolver um programa que permita mostra a tabuada de um número fornecido pelo usuário, como no exemplo a seguir. Faça utilizando a estrutura for e também a while.

5 x 1 = 5	5 x 6 = 30
5 x 2 = 10	5 x 7 = 35
5 x 3 = 15	5 x 8 = 40
5 x 4 = 20	5 x 9 = 45
5 x 5 = 25	5 x 10 = 50

## AS MATRIZES

Define-se como matriz um grupo de dados que podem ser agrupados num mesmo nome, diferenciando-se apenas pela posição no grupo.

Em C, a definição de uma variável ou de uma constante como matriz é feita apenas pela inclusão de seu tamanho entre colchetes [ ].

Exemplo 1:

Matriz para os 20 valores de temperatura lidos.

```
char temperatura[20]; /* reserva espaço de memória para 20 bytes que indicarão a
                        temperatura.O primeiro elemento é temperatura[0] ,O último
                        elemento é temperatura[19]*/
```

Exemplo 2:

```
Char nome[7]="Joao";      /* Joao é atribuido a string nome de 7 posições
```

