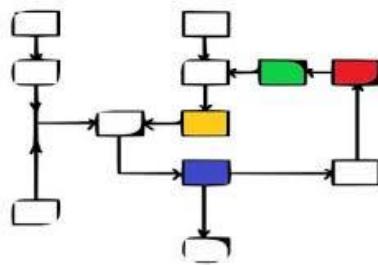


Algoritmos e Programação

Sarah de Oliveira Alcântara

sarah.alcantara01@etec.sp.gov.br

ESTRUTURA DE DECISÃO



Algoritmo



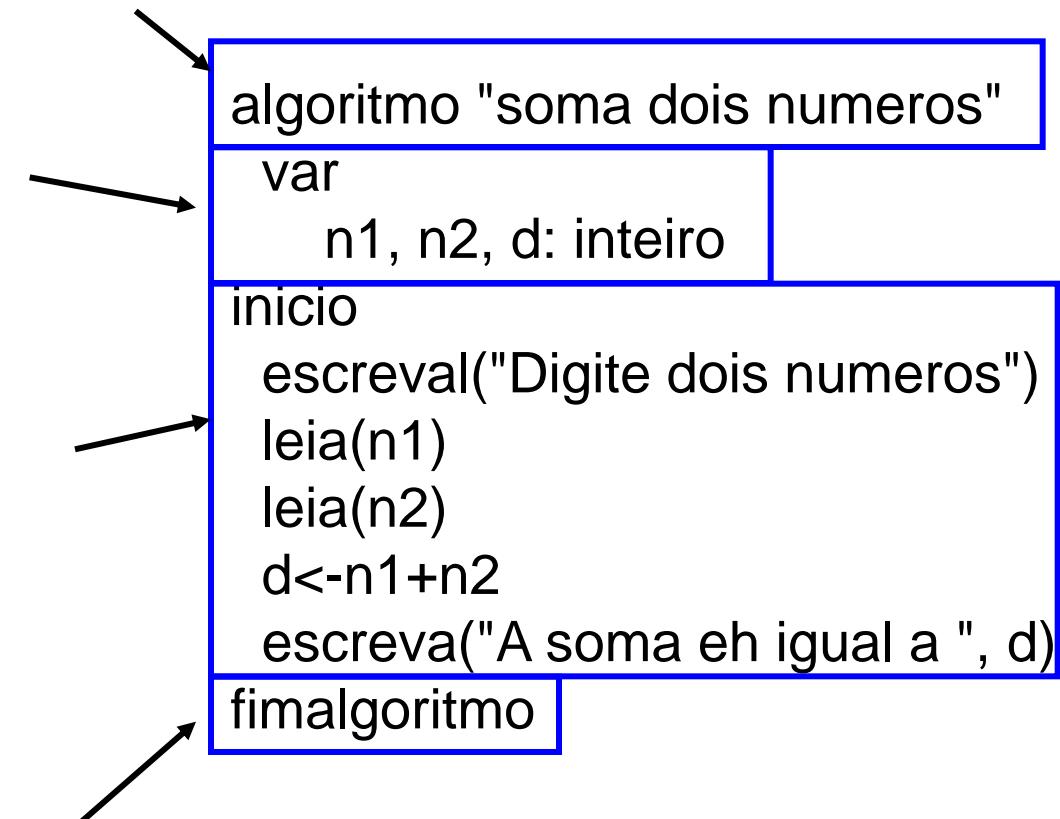
Linguagem C

Estrutura de um Algoritmo

NOME DO ALGORITMO

VAR

declaração de variáveis

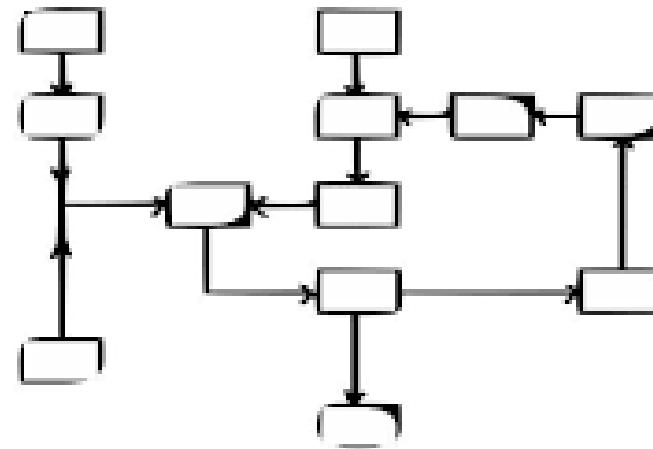


INICIO DO ALGORITMO

bloco de comandos

FIM DO ALGORITMO

Estruturas de Controle de Fluxo

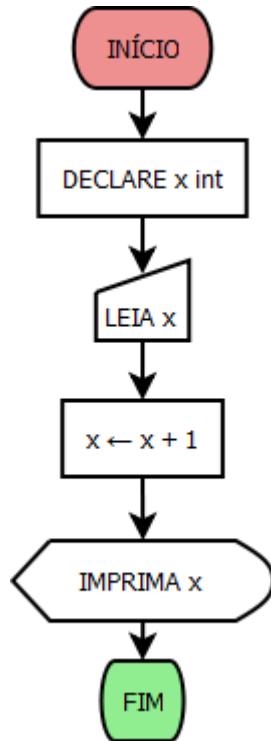


Estrutura de Controle de Fluxo

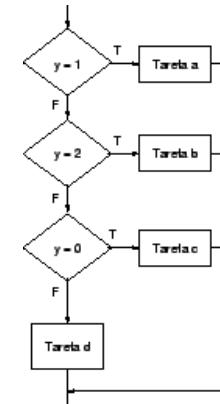
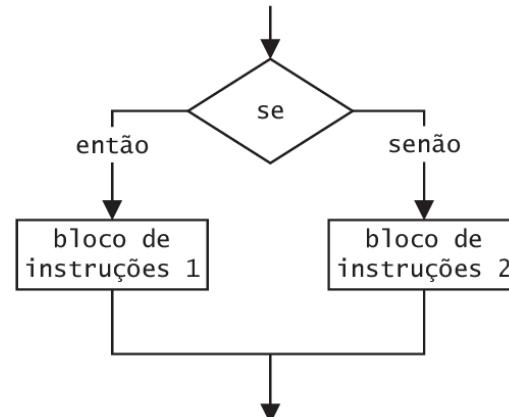
- ✓ Para se desenvolver sistema é necessário controlar o fluxo de execução das instruções (a sequência em que as instruções são executadas num algoritmo) em função dos dados fornecidos como entrada ao mesmo.
- ✓ Em muitos casos pode ser necessário executar um mesmo conjunto de instruções um número repetido de vezes.
- ✓ A partir de agora e nas próximas aulas serão estudadas as estruturas básicas de controle do fluxo de instruções de um algoritmo. De acordo com o modo como este controle é feito, estas estruturas são classificadas em:
 - Estruturas sequenciais;
 - Estruturas de decisão;
 - Estruturas de repetição.

Estruturas de Controle de Fluxo

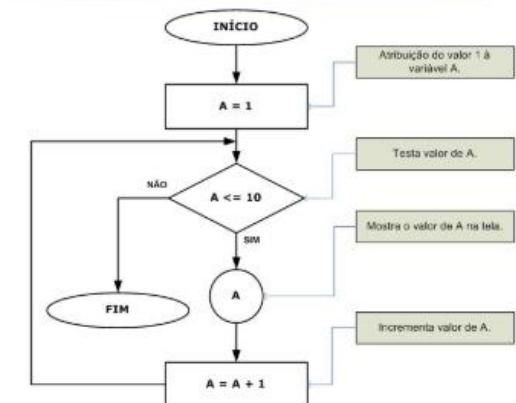
Sequenciais



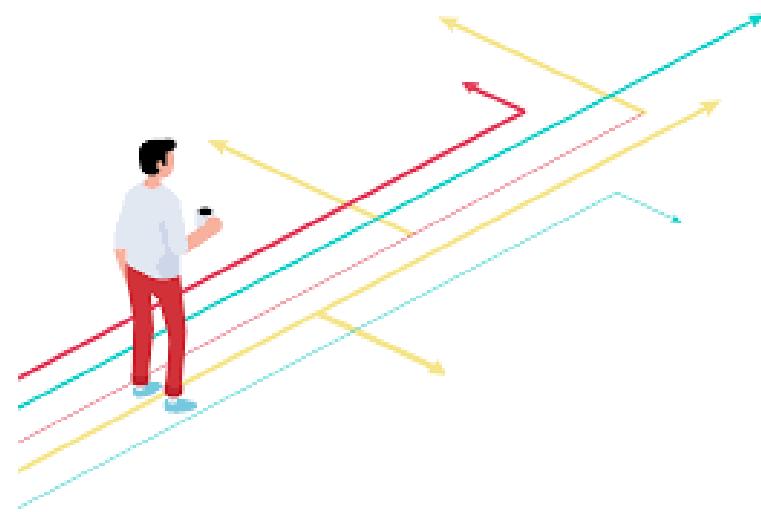
Decisão



Repetição



Estruturas de Decisão



Estruturas de Decisão

- São conhecidas como instruções de salto ou desvio.
- Utilizando – as é possível fazer com que o programa proceda de uma ou outra maneira, de acordo com as decisões lógicas tomadas.
- Principais estruturas: “Se Então”, “Se Então Senão” e “Caso Selecione”

Instrução Se ... Então...Senão

Semântica: a condição é avaliada.

- ✓ **Se o resultado for verdadeiro**, então o comando_composto 1 é executado e ao término de sua execução o fluxo do algoritmo prossegue pela primeira instrução seguinte ao Fim_se.

- ✓ **Se o resultado for falso**, o comando_composto 2 é executado e ao término de sua execução o fluxo do algoritmo prossegue pela primeira instrução seguinte ao Fim_se.

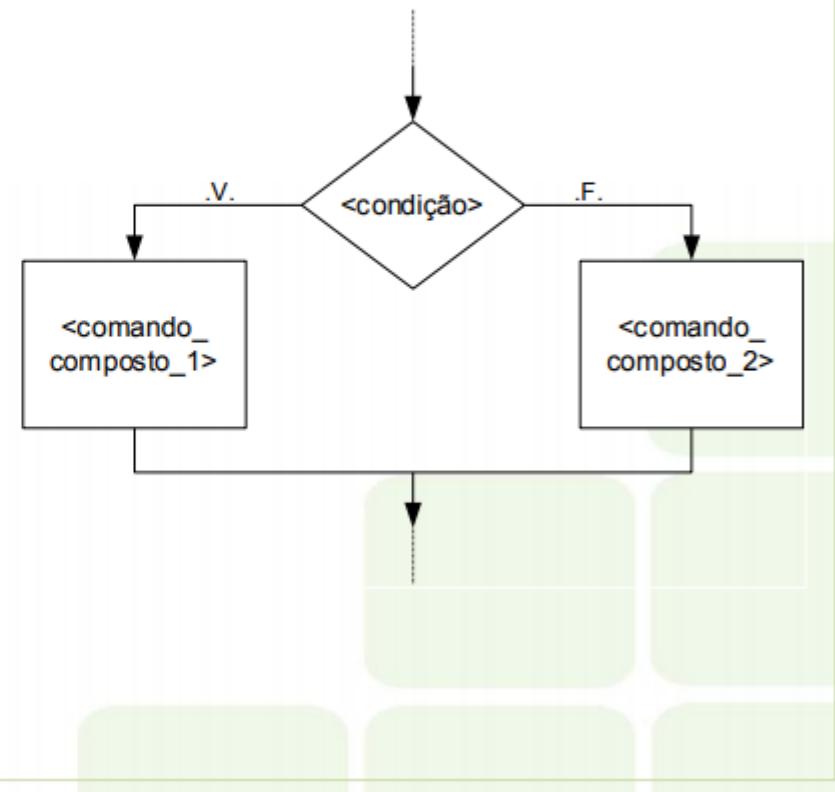
Instrução Se ... Então...Senão

❖ Estruturas de Decisão simples (Tipo Se-Senao)

Estrutura em Pseudocódigo

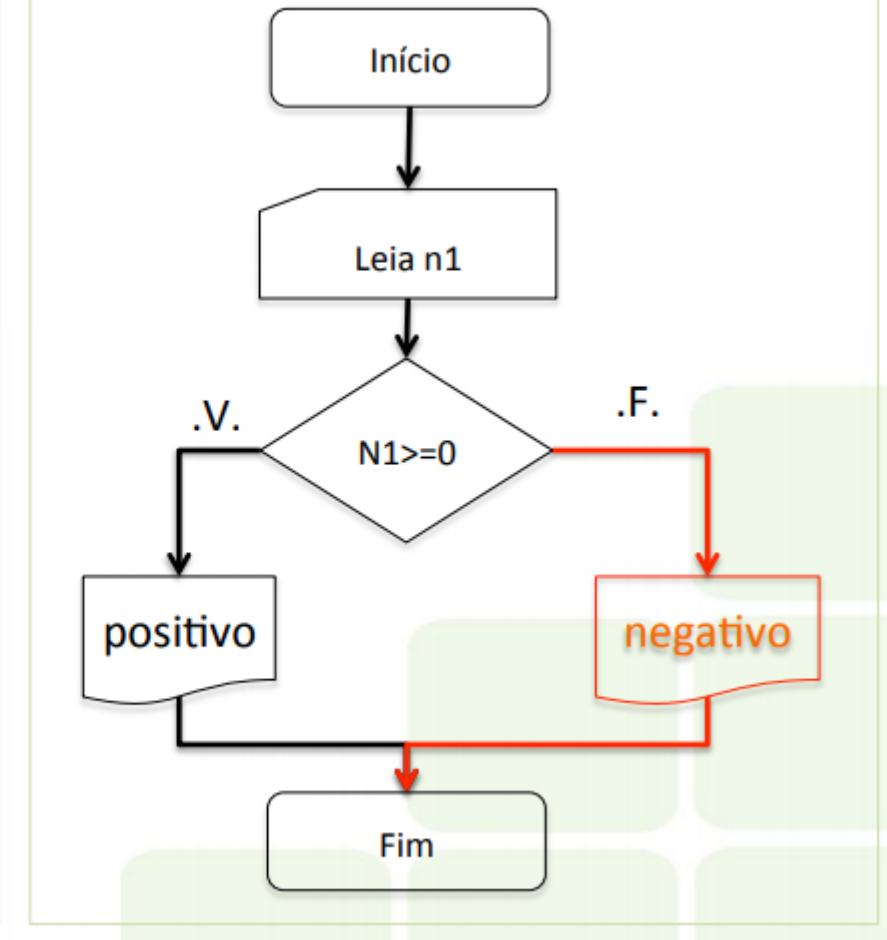
```
Se <condição> Entao
    <comando_composto_1>
Senao
    <comando_composto_2>
Fimse
```

❖ Fluxograma



Instrução Se ... Então...Senão - Exemplo

```
algoritmo "exemplo1"
var n1: inteiro
Inicio
leia(n1)
se n1>=0 entao
    escreval("O numero é positivo.")
senao
    escreva("O numero é negativo.")
Fimse
fimalgoritmo
```



Instrução Se ... Então

- ✓ Se aplica ao caso particular onde o **comando_composto 2 (Falso)** é um conjunto vazio de instruções, então a porção relativa ao Senão pode ser omitida.

Semântica:

- ✓ **Quando a condição for verdadeira**, o comando_composto 1 é executado e após o seu término, o fluxo de execução prossegue pela próxima instrução após o Fim_se.
- ✓ **Quando a condição for falsa**, o fluxo de execução prossegue normalmente pela primeira instrução após o Fim_se.

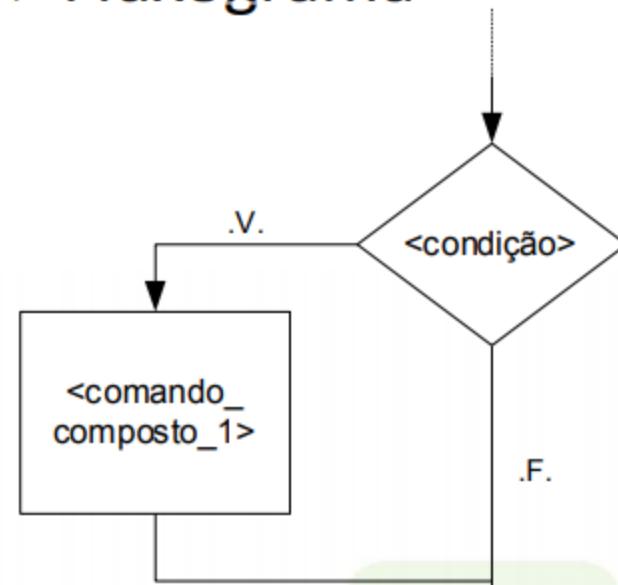
Instrução Se ... Então

❖ Caso particular do SE

Estrutura em Pseudocódigo

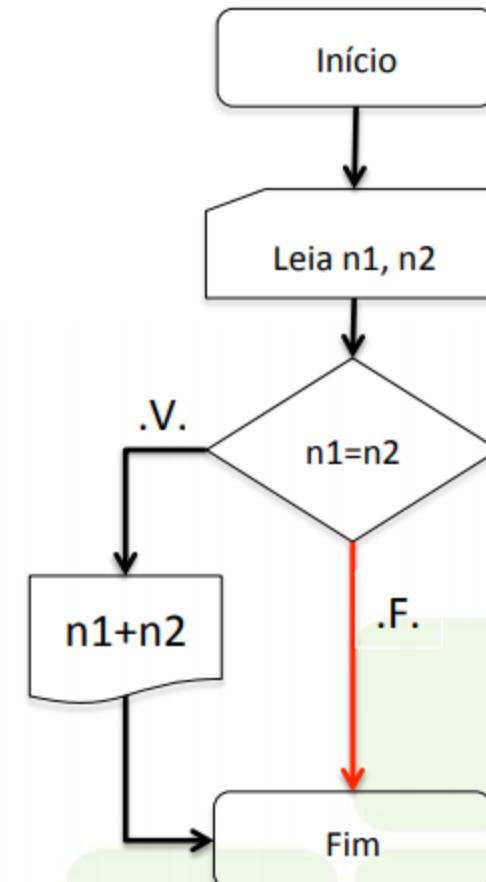
```
se <condição> entao
    <comando_composto_1>
fimse
```

❖ Fluxograma



Instrução Se ... Então

```
algoritmo "exemplo 2"
var n1, n2: inteiro
Início
leia(n1)
leia(n2)
se (n1=n2) entao
    escreval("A soma dos numeros é: ",
n1+n2)
fimse
finalgoritmo
```



InSTRUÇÃO Caso... Selecione

- ✓ Estruturas de Decisão do Tipo Caso/Selecione, também é conhecida como **Múltipla Escolha**.

- ✓ Na estrutura de decisão do tipo **Escolha** pode haver uma ou mais condições a serem testadas e um comando composto diferente associado a cada uma destas.

InSTRUÇÃO CASO... SELECIONE

❖ Estruturas de Decisão do Tipo Escolha

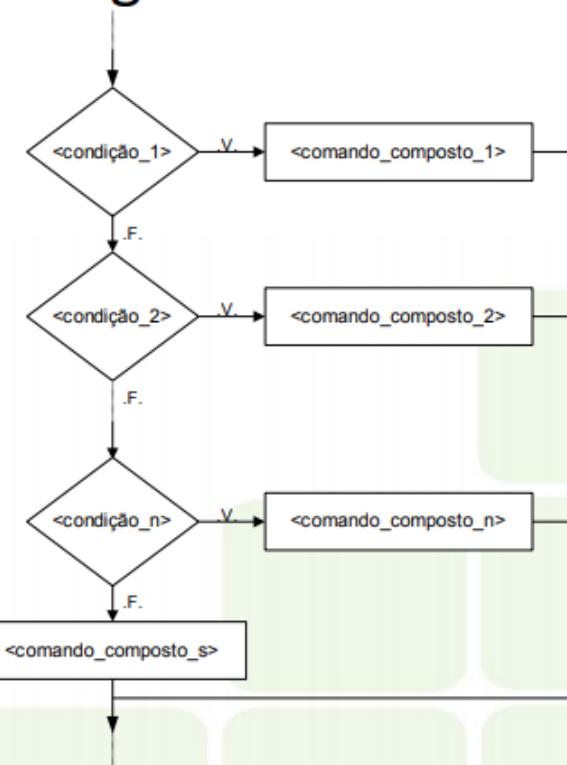
Estrutura em Pseudocódigo

Escolha

```
Caso <condição_1>
    <comando_composto_1>
Caso <condição_2>
    <comando_composto_2>
Caso <condição_n>
    <comando_composto_n>
Senao
    <comando_composto_s>
```

Fimescolha

❖ Fluxograma



Instrução Caso... Selecione

Algoritmo exemplo3

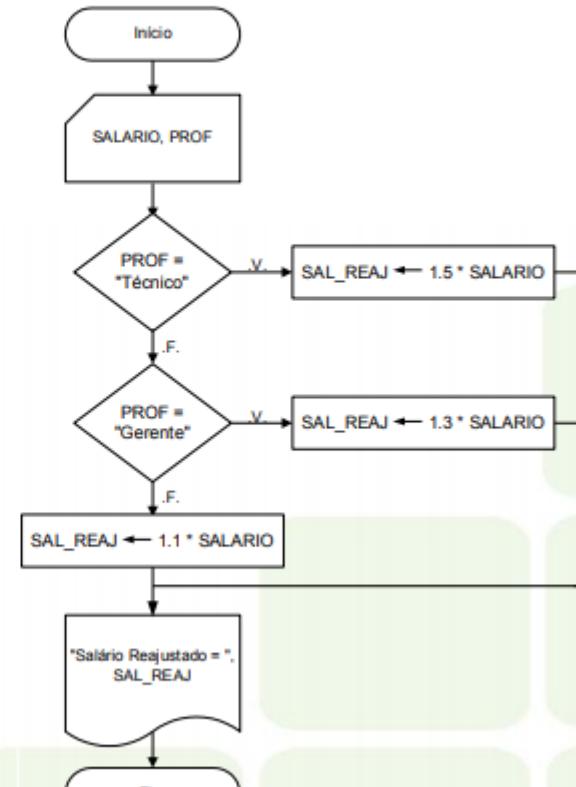
```

Var SALARIO, SAL_REAJ : real
PROF : literal

Início
  Leia SALARIO, PROF
  Escolha PROF
  Caso "Técnico"
    SAL_REAJ  $\leftarrow$  1.5 * SALARIO
  Caso "Gerente"
    SAL_REAJ  $\leftarrow$  1.3 * SALARIO
  outrocaso
    SAL_REAJ  $\leftarrow$  1.1 * SALARIO
  Fimescolha
  Escreva ("Salário Reajustado= ", SAL_REAJ)
Fim.

```

❖ Fluxograma



Síntese

- ✓ Uma estrutura sequencial é aquela em que os comandos vão sendo executados numa sequência pré-estabelecida, um após o outro.
- ✓ Um comando composto é um conjunto de um ou mais comandos simples, sejam eles instruções primitivas ou construções.
- ✓ As estruturas de decisão permitem escolher qual o caminho a ser seguido num algoritmo em função de uma ou mais condições. A construção **SE** utiliza apenas uma condição, ao passo que a construção **ESCOLHA** utiliza uma ou mais condições.

Colocando em Prática



Exercícios: Estrutura de Decisão

- ✓ Elaborar um algoritmo para somar dois números inteiros e mostrar o valor da soma na tela. Caso a soma dos números seja maior que 10 mostrar uma mensagem informativa na tela. Caso a soma dos números seja menor que 10, enviar uma mensagem informando também quando a soma é menor ou igual a 10.
- ✓ Construa um algoritmo em que o usuário vai digitar um número e o programa vai retornar o dia da semana equivalente ao número.
- ✓ **OBS: Na próxima aula, os algoritmos construídos serão desenvolvidos na linguagem C.**

Introdução a Liguagem C



Material de apoio

- Download Dev C++
 - <http://www.baixaki.com.br/download/dev-c-.htm>
- Video Aulas – Curso Completo de Algoritmos e Programação em C no Dev C++.
 - https://www.youtube.com/watch?v=joiwqS2wd1A&list=PL9PzDKD_B1nNpjxJ9kKF EWtN7Uzk6RpFa

Linguagem C

- ✓ Criada em 1972, por Dennis Ritchie;
- ✓ Centro de Pesquisas da Bell Laboratories;
- ✓ Para utilização no S.O. UNIX;
- ✓ C é uma linguagem de propósito geral;
- ✓ Em 1989 o **Instituto Norte-Americano de Padrões (ANSI)** padronizou a linguagem C.

Linguagem C

- ✓ Case sensitive;
- ✓ Tipos de dados primitivos: caractere, inteiro e real;
- ✓ Possui estruturas de controle de fluxo;
- ✓ Operadores aritméticos, lógicos, relacionais e condicional;
- ✓ Todo programa tem uma função principal chamada **main()**;
- ✓ Todo linha de instrução em um programa é finalizada com um “;”;

Palavras-reservadas

Palavras chaves em C (padrão ANSI)

auto	Double	int	Struct
break	Else	long	Switch
case	Enum	register	typedef
char	Extern	return	union
const	Float	short	unsigned
continue	For	signed	void
default	Goto	sizeof	volatile
do	If	static	while

Definição de Variáveis

- ✓ Devem ser declaradas no **início** do programa ou do sub bloco;
- ✓ Podem ser classificadas como **Locais** ou **Globais**.

– Locais

- Declaradas dentro de funções;
- Utilizada apenas dentro do escopo da função;
- O escopo de uma função é determinado por abre-chaves “{“ e termina em fecha-chaves “}”;
- **Só existem** no momento que sua função está em execução.

– Globais

- Declaradas **fora** de todas as funções;
- Podem ser **acessadas** de qualquer parte do programa;
- **Existem durante toda a execução** do programa.

Nomes de Variáveis

- ✓ Deve conter um ou mais caracteres;
- ✓ O primeiro caractere **sempre** deve ser uma **letra**;
- ✓ Os caracteres **subseqüentes** podem ser **letras, números** ou “_”;
- ✓ Não pode ser igual às **palavras-chaves**;
- ✓ Não pode ter o **mesmo nome de funções**;

Correto	Incorreto
Soma1	1soma
soma	soma!
area_triangulo	area...triangulo

Obs: as variáveis “**soma**” e “**Soma**” são **distintas**

Declarando variáveis

Sintaxe

- ✓ <Tipo de dados> Nome_variável;

Ex:

```
char nome;  
int idade;  
int total;
```

Atribuindo valor

- ✓ Nome_da_variavel = expressão;

Ex:

```
nome = 'Joao';  
idade = 18;  
total = 10 + 20;
```

Operadores aritméticos

Operador Binário	Descrição
=	Atribuição
+	Soma
-	Subtração
/	Divisão
%	Modulo (resto da divisão)

Operadores aritméticos Unários e Binários

✓ **Unários** (+, -, ++, --) agem sobre uma variável apenas, modificando ou não o seu valor, e retornam o valor final da variável.

- $a = -b;$
- $a++;$ (ou seja) $a = a+1;$
- $a--;$ (ou seja) $a = a-1;$

Obs: operador “-” como troca de sinal é um operador unário que não altera a variável sobre a qual é aplicado, pois ele retorna o valor da variável multiplicado por -1.

✓ **Binários** (+, -, *, /, %) usam duas variáveis e retornam um terceiro valor, sem modificar as variáveis originais.

Operadores de Atribuição =, +=, -=, *=, /=, %=

Instrução normal	Instrução reduzida
var = var + expr;	Var += expr;
var = var - expr;	Var -= expr;
var = var / expr;	Var /= expr;
var = var * expr;	Var *= expr;

✓ Exemplos:

- a = 5;
- a += 5; (*ou seja*) a = (a + 5);
- a -= 5; (*ou seja*) a = (a - 5);

Comentários

- ✓ `//` Meu comentário em uma linha
- ✓ `/*` Meu comentário através de um bloco de texto que pode estar em *n* linhas `*/`

Tipos Primitivos

Caractere

- ✓ Definido pela palavra reservada **char**;
- ✓ Ocupa 8 bits (1 byte)
- ✓ Faixa de valores: -128 à 127
- ✓ Exemplo:
 - **char** letra;
 - letra = 'A';

Tipos Primitivos

Inteiro

- ✓ Definido pela palavra reservada **int**;
- ✓ Ocupa 16 bits (2 bytes)
- ✓ Faixa de valores: -32768 à 32767
- ✓ Exemplo:
 - `int num;`
 - `num = -73;`

Tipos Primitivos

Ponto flutuante

- ✓ Definido pela palavra reservada **float**
- ✓ Ocupa 4 bytes
- ✓ Exemplo:
 - **float** a,b,c=2.34;

Ponto flutuante de precisão dupla

- ✓ Definido pela palavra reservada **double**
- ✓ Ocupa 8 bytes
- ✓ Exemplo:
 - **double** x=2.38, y=3.1415;

Tipos de dados - padrão ANSI

Tipo	Tamanho aproximado em bits	Faixa mínima
char	8	-127 a 127
unsigned char	8	0 a 255
signed char	8	-127 a 127
int	16	-32.767 a 32.767
unsigned int	16	0 a 65.535
signed int	16	O mesmo que int
short int	16	O mesmo que int
unsigned short int	16	0 a 65.535
signed short int	16	O mesmo que short int
long int	32	-2.147.483.647 a 2.147.483.647
signed long int	32	O mesmo que long int
unsigned long int	32	0 a 4.294.967.295
float	32	Seis dígitos de precisão
double	64	Dez dígitos de precisão
long double	80	Dez dígitos de precisão

Estrutura básica de um programa em C

```
void main()    // Primeira função a ser executada
{
    /*Aqui ficaria as declarações locais e as instruções
     do programa */
}
```

Operadores

Operadores Relacionais

Operador	Ação
>	maior que
\geq	maior ou igual a
<	menor que
\leq	menor ou igual
\equiv	igual a
\neq	diferente de



IMPORTANTE!

Operadores Lógicos

Operador	Ação
&&	E
 	OU
!	Não

		Não p	Não q	p E q	p OU q
p	q	! p	! q	p && q	p q
falso	falso	verdadeiro	verdadeiro	falso	falso
falso	verdadeiro	verdadeiro	falso	falso	verdadeiro
verdadeiro	falso	falso	verdadeiro	falso	verdadeiro
verdadeiro	verdadeiro	falso	falso	verdadeiro	verdadeiro

Hierarquia dos operadores Relacionais e Lógicos

- Hierarquia ou Precedência – prioridade com que os operadores são executados pelo compilador;
- Operadores com mesmo nível hierárquico são executados da esquerda para a direita;
- Podem ser alterada utilizando “(” “)”.

Hierarquia	Operação
1	!
2	>, \geq , <, \leq
3	\equiv , \neq
4	$\&\&$
5	\parallel

Funções de Entrada e Saída

Funções de Entrada e Saída Formatada

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

io → input/output

Forma geral:

- `printf(string_de_controle, <lista_de_argumentos>);`

- Indica as variáveis com suas respectivas posições através dos códigos de formato “%” mostrado a seguir.

Funções de Entrada e Saída - **scanf()**

scanf() - Leitura de dados;

Sintaxe:

scanf(string_de_controle, lista_de_argumentos);

- ✓ **string_de_controle** → descrição de todas as variáveis que serão lidas, com informações de seus tipos e da ordem em que serão lidas;
- ✓ **lista_de_argumentos** → lista com os identificadores das variáveis que serão lidas.

Importante: colocar antes de cada variável da lista_de_argumentos o caractere ‘&’

Exemplo:

char letra; //Declarando a variável “letra”

scanf(“%c”, &letra); //Lendo dados digitados pelo usuário

Funções de Entrada e Saída

- ✓ Tabela de códigos de tipos de dados

Código	Formato
%c	Um caracter (char)
%d	Um número inteiro decimal (int)
%f	Ponto flutuante decimal
%s	String

Exemplos:

```
char letra;  
float nota;  
int quantDeFilhos;  
scanf("%c", &letra);  
scanf("%f", &nota);  
scanf("%d", &quantDeFilhos);
```

Estruturas de Controle de Fluxo

Estruturas de Controle de Fluxo - if

Instrução condicional

Sintaxe

```
if(<condição>){  
    <instrução>  
}
```

Estruturas de Controle de Fluxo – if ... else

Instrução condicional

Sintaxe

```
if(<condição>)
{
    <instrução 1>
    ...
    <instrução N>
} else {
    <instrução A>
    ...
    <instrução Z>
}
```

Estruturas de Controle de Fluxo – if ... else

Exemplo

```
#include <stdio.h>

int main ()
{
    char letra = 'a';
    if (letra == 'a'){
        printf("Letra digitada eh: 'A'");
    }else{
        printf("Letra digitada DESCONHECIDA!");
    }
    getchar();
    return(0);
}
```

Estruturas de Controle de Fluxo – if ... else if ... else

```
if (condição_1){  
    instrução 1;  
}else if (condição_2){  
    instrução 2;  
}else if (condição_3){  
    instrução 3;  
}else if (condição_n){  
    instrução n;  
}  
else{  
    instrução padrão;  
}
```

```
int varNumero;  
printf ("Digite um numero: ");  
scanf ("%d", &varNumero);  
  
if (varNumero > 10){  
    printf ("Numero MAIOR que 10");  
}else if (varNumero == 10){  
    printf ("Voce digitou: 10");  
}else if (varNumero < 10){  
    printf ("Numero MENOR que 10");  
}
```

//substituir o último else if faz sentido?

```
else {  
    printf ("Numero MENOR que 10");  
}
```

Estruturas de Controle de Fluxo – if ... else if ... else

Os códigos funcionam? Existe diferença?

```
int varNumero;  
printf ("Digite um numero: ");  
scanf ("%d", &varNumero);  
  
if (varNumero > 10){  
    printf ("Numero MAIOR que 10");  
}else if (varNumero == 10){  
    printf ("Voce digitou: 10");  
}else if (varNumero < 10){  
    printf ("Numero MENOR que 10");  
}
```

```
int varNumero;  
printf ("Digite um numero: ");  
scanf ("%d", &varNumero);  
  
if (varNumero > 10){  
    printf ("Numero MAIOR que 10");  
}  
if (varNumero == 10){  
    printf ("Voce digitou: 10");  
}  
if (varNumero < 10){  
    printf ("Numero MENOR que 10");  
}
```

Estruturas de Controle de Fluxo – ifs aninhados

```
float nota;  
nota = 9.5;  
if (nota >= 7){  
    printf("Aprovado.");  
    if (nota >= 9){  
        printf("Aprovado com  
louvor!!!");  
    }  
}else{  
    printf("Aluno REPROVADO!");  
}
```

Estruturas de Controle de Fluxo – operador ?

if (condição){

instrução 1;

}else{

instrução 2;

}

✓ É equivalente a:

condição?instrução1:instrução2;

Estruturas de Controle de Fluxo – switch

- ✓ Próprio para se **testar** uma **variável** em relação a **valores pré-estabelecidos**.
- ✓ Testa o conteúdo da variável e **executa a instrução** correspondente ao case;
- ✓ **break**, faz com que o switch seja **interrompido**;
- ✓ **default** é opcional;
- ✓ Não aceita expressões.

```
switch (variável)
{
    case constante_1:
        instrução 1;
        break;
    case constante_2:
        instrução 2;
        break;
    ...
    default
        instrução_padrão;
}
```

Estruturas de Controle de Fluxo – **switch**

```
switch (varNumero)
{
    case 9:
        printf ("O numero e igual a 9.");
        break;
    case 10:
        printf ("O numero e igual a 10.");
        break;
    default:
        printf ("O numero nao e nem 9 nem 10.");
}
```

Colocando em Prática



Exercícios: Estrutura de Decisão

- ✓ Crie um programa para somar dois números inteiros e mostrar o valor da soma na tela. Caso a soma dos números seja maior que 10 mostrar uma mensagem informativa na tela. Caso a soma dos números seja menor que 10, enviar uma mensagem informando também quando a soma é menor ou igual a 10.

- ✓ Crie um programa em que o usuário vai digitar um número e o programa vai retornar o dia da semana equivalente ao número.