Introdução a Programação

Revisão

O que é um algoritmo?

É uma sequência ordenada e sem ambiguidades de comandos que levam à execução de uma tarefa ou à solução de um problema.

Quais os tipos primitivos de C e como declaramos variáveis?

char, int, long, unsigned, unsigned long, short, float e double

tipo nome_da_variável

OBS: Pode ser atribuído um valor na variável ao declararmos, este valor deve ser compatível com seu tipo

Quais funções usamos para entrada e saída, respectivamente?

Entrada: scanf (string_de_controle, argumentos)

Saída: printf (string_de_controle, argumentos)

Qual a diferença entre operadores lógicos e operadores aritméticos?

Operadores lógicos são usados para avaliar se uma expressão é verdadeira ou falsa

Operadores aritméticos são usados para o cálculo de um valor

Aula 2

Propriedades dos Operadores

- Aridade
- Resultado
- Precedência
- Associatividade
- *Efeito Colateral
- *Ordem de Avaliação
- *Curto Circuito

Propriedade dos operadores

- **Aridade**:

Indica a quantidade de operandos sobre os quais o operador atua.

```
//Operador "não boleano" (NOT)
S = !boolean;

//Operador aritmético "soma"
R = a + b;

//Operador condicional (ternário)
E = (condição)? (exp1) : (exp2);
```

Propriedade dos operadores

- **Resultado:**

É o valor da aplicação da operação sobre os operandos.

```
//Operador "não boleano" (NOT)
S = !boolean;

//Operador aritmético "soma"
R = a + b;

//Operador condicional (ternário)
E = (condição)? (exp1) : (exp2);
```

Propriedade dos operadores

- **Precedência:**

É a ordem em que um operador é aplicado em relação a outros operadores.

```
//Operador "não boleano" (NOT)
S = !boolean;

//Operador aritmético "soma"
R = a + b;

//Operador condicional (ternário)
E = (condição)? (exp1) : (exp2);
```

Propriedade dos operadores

- Associatividade:

É a ordem de execução de operadores que têm a mesma precedência.

```
int main(void){
 int E, D, a = 2, b = 1, c = 2;
 E = a/b*c;
 D = a*b/c;

 printf("\n\n\n\n\n\t\tE = %d\t\tD = %d\n\n\n\n\n", E,
 D);
 return 0;
}
```

Propriedade dos operadores

- Efeito Colateral:

É propriedade que alguns operadores têm de alterar o valor de seus operandos.

- Ordem da avaliação:

Indica qual dos operandos será avaliado primeiro durante a aplicação do operador.

Propriedade dos operadores

- Curto Circuito:

O operador não avalia todos os operandos pois o resultado já é conhecido.

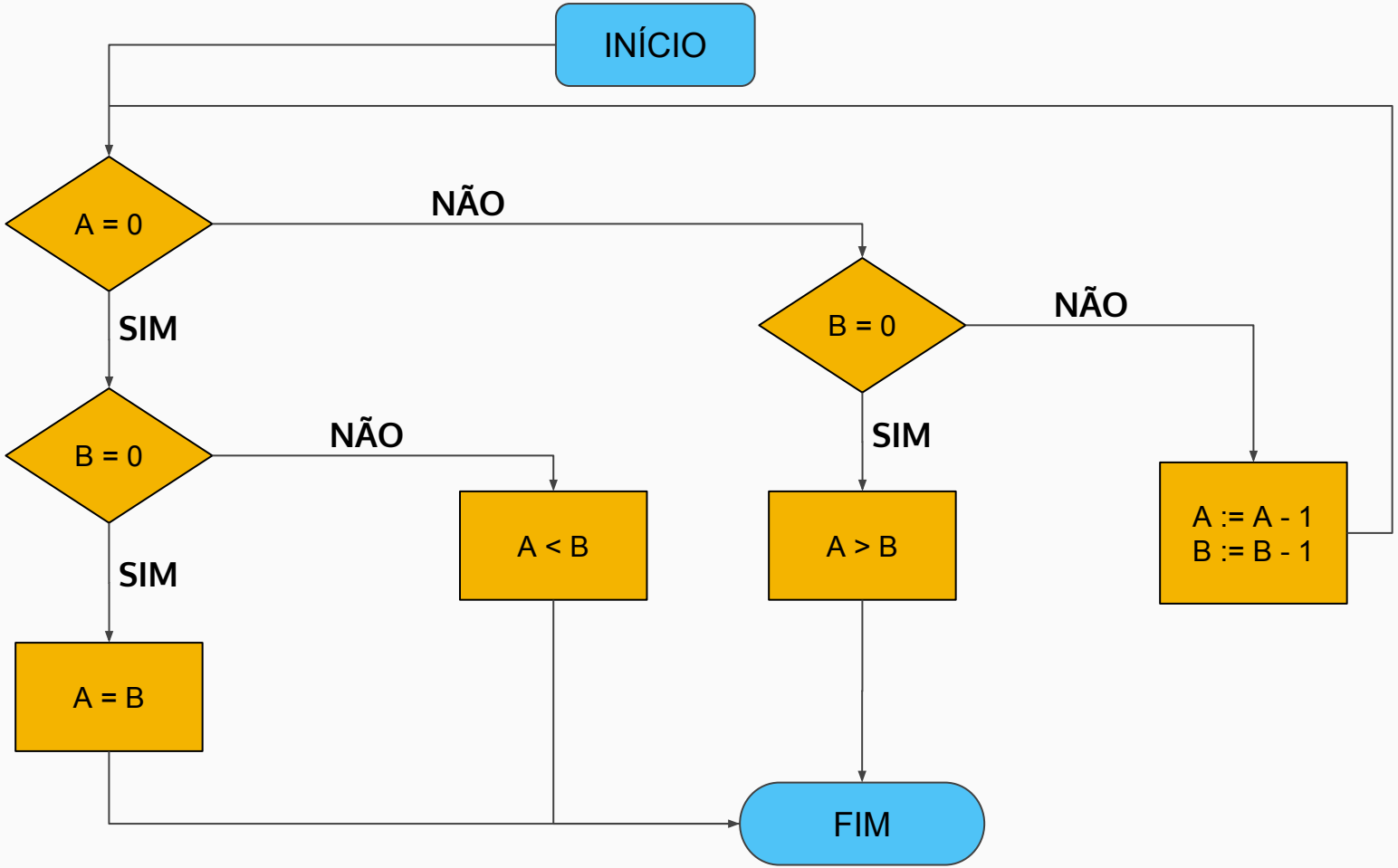
```
//Operador E(AND)
esquerda && direita;

//Operador OU (OR)
esquerda || direita;

//Operador "atribuição" e "incremento sufixal"
A = y++;
```

Estruturas de controle

Usadas para gerar desvios e repetições de certas instruções em um programa.

Exemplos.:

if, else, while, break, for, continue, do-while, switch-case


INÍCIO
A = 0
NÃO
SIM
B = 0
NÃO
SIM
A = B
B = 0
NÃO
SIM
A < B
A > B
A := A - 1
B := B - 1
FIM

Tipos Estruturas de Controle

- Desvios Condicionais:

Decidem, através de uma condição, se parte do programa será executada ou não.

Tipos Estruturas de Controle

- Desvios Condicionais:

if

```
if (condição){

  (bloco de instrução)

}
```

Tipos Estruturas de Controle

- Desvios Condicionais:

if else

```
if (condição){
  (bloco de instrução 1)

}
else {
  (bloco de instrução 2)

}
```

Tipos Estruturas de Controle

- Desvios Condicionais:

if else if

```
if (condição 1){
  (bloco de instrução 1)
}
else if (condição 2){
  (bloco de instrução 2)
}
else if (condição 3){
  (bloco de instrução 3)
}
```

Tipos Estruturas de Controle

- Desvios Condicionais:

switch-case

```
switch (expressão){
 case (expressao constante 1):
  intrução 1;
  break;
 case (expressao constante 2):
  intrução 2;
  break;
 default:
  instrução default;
  break;
}
```

Tipos Estruturas de Controle

- Repetições (ou Iterações):

Permitem a execução de uma ou mais instruções repetidamente até que uma condição de parada seja satisfeita.

Tipos Estruturas de Controle

- Repetições (ou Iterações):

while

```
//Laço de repetição "enquanto"
while (condição){

  (bloco de instrução)


}
```

Tipos Estruturas de Controle

- Repetições (ou Iterações):

while

```
//Laço de repetição "enquanto"
while (condição){

  (bloco de instrução)


}
```

Tipos Estruturas de Controle

- Repetições (ou Iterações):

for

```
//Laço de repetição for
for (inicialização; condição; incremento){

  (bloco de instrução)


}
```

Tipos Estruturas de Controle

- Repetições (ou Iterações):

do-while

```
//Laço de repetição do - while
do {

  (bloco de instrução)


}while (condição)
```

Tipos Estruturas de Controle

- Repetições (ou Iterações):

Loops infinitos

```
while (1){
  (bloco de instrução)
}

for (;;){
  (bloco de instrução)
}
```

Tipos Estruturas de Controle

- Desvios incondicionais:

Indicam incondicionalmente qual instrução será executada em seguida.

Tipos Estruturas de Controle

- Desvios incondicionais:

break

```
int x = 0, y = 0;

while (1){ //loop infinito
 if(x >= y)
  break; //desvio incondicional
 x++;
 y--;
}
```

Tipos Estruturas de Controle

- Desvios incondicionais:

continue

```
int cont = 0;

while (cont < 5){
 cont++;
 if(cont == 3)
  continue;
 printf("\nExecucao num %d do corpo do laco", cont);

}
```

Exercícios

Escreva um programa que lê dois números inteiros e exibe o maior deles.

Entrada	Saída
10 15	15
4 5	5

Escreva um programa que calcula e mostra a soma dos 100 primeiros números inteiros a partir do 1.

$$1 + 2 + 3 + \ldots + 99 + 100$$

Saída

5050

Escreva um programa que recebe um conjunto de valores inteiros, calcula e exibe o maior valor inserido. A entrada de dados deve parar quando for digitado o valor 0.

Entrada	Saída
10 15 400 5 20 0	400
4 3 2 0	4

Escreva um programa que verifica e mostra os números entre 0 e 1000 (inclusive) que, quando divididos por 11, produzam resto igual a 5.

Saída

5 16 27 38 49 60 71 82 ... 973 984 995