Introdução a Programação

Revisão

Quais são as propriedades dos operadores?

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

Para que servem as estruturas de controle? Dê exemplos.

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

Exemplos: *if, else, while, break, for, continue, do-while, switch-case*

Quais são os tipos de estrutura de controle? Dê exemplo de cada tipo.

Desvios Condicionais:
decidem, através de uma condição, se parte do programa será executada ou não.

Exemplos: *if, if else, if else if, switch-case*

Repetições (ou iterações):
permitem a execução de uma ou mais instruções repetidamente até que uma condição de parada seja satisfeita.

Exemplos: *while, do-while, for*

Desvios Incondicionais:
indicam incondicionalmente que instrução será executada em seguida.

Exemplos: *break, continue*

Aula 3

Constante Simbólica

- É o identificador que se associa a um valor constante.
- É definida pela diretiva de pré-processador #define

Formato:

```
#define CONSTANTE valor
```

Constante Simbólica

Exemplos:

```
#define PI 3.14
#define MINIMO 0
#define MAXIMO 100
```

Constante Simbólica

Exemplos:

```
#include<stdio.h>
#define PI 3.14

int main() {

 printf("Valor de PI = %.2f\n" , PI);
 printf("Valor de MINIMO = %f\n" , MINIMO);
 return 0;
}
#define MINIMO 0
```

Constante Simbólica

O uso de constante simbólica em um programa tem dois objetivos principais:

- ➔ Tornar o programa mais legível;
- ➔ Tornar o programa mais fácil de ser modificado.

Casting

Casting (ou conversões) são as transformações que podem ser feitas em tipos de variáveis.

Conversões podem acontecer de várias formas:

- Conversões Implícitas:
  - Conversão de Atribuição;
  - Conversão Aritmética Usual;
- Conversões Explícitas.

Casting - Implícito

Conversão de Atribuição:
O lado **direito** é convertido para o lado **esquerdo** em uma atribuição.

```c
#include<stdio.h>

int main() {

 int a;
 a = 2.4;
 printf("a vale: %d\n" , a);
 printf("\n\n");
 return 0;

}
```

Casting - Implícito

Conversão de Atribuição:
O lado **direito** é convertido para o lado **esquerdo** em uma atribuição.

```c
#include<stdio.h>

int main() {

 int a;
 a = 2.4;
 printf("a vale: %d\n" , a);
 printf("\n\n");
 return 0;

}
```

```
a vale: 2
```

Casting - Implícito

Conversão de Atribuição:
O lado **direito** é convertido para o lado **esquerdo** em uma atribuição.

```c
#include<stdio.h>

int main() {

 char c;
 c = 65;
 printf("c vale: %c\n" , c);
 printf("\n\n");
 return 0;

}
```

Casting - Implícito

Conversão de Atribuição:
O lado **direito** é convertido para o lado **esquerdo** em uma atribuição.

```c
#include<stdio.h>

int main() {

 char c;
 c = 65;
 printf("c vale: %c\n" , c);
 printf("\n\n");
 return 0;

}
```

```
c vale: A
```

Casting - Implícito

➔ **Conversão de Atribuição:**

Acontece quando se usa variáveis e constantes de tipos diferentes numa expressão.

Segue a ordem:

int → [...] → float → double → long double

OBS.: [...] representa outras versões do tipo inteiro, como unsigned ou long

Casting - Implícito

Conversão de Atribuição:

```c
#include<stdio.h>

int main() {
 int a;
 float b, c;
 a = 5;
 b = 2.3
 c = b + a;
 printf("c vale: %d\n\n" , c);
 return 0;
}
```

Casting - Implícito

Conversão de Atribuição:

"a" mesmo sendo inteiro, foi tratado como float na hora da operação

```c
#include<stdio.h>

int main() {
 int a;
 float b, c;
 a = 5;
 b = 2.3;
 c = b + a;
 printf("c vale: %f\n\n" , c);
 return 0;
}
```

```
c vale: 7.300000
```

Casting - Implícito

Riscos de conversão implícita:

```c
#include<stdio.h>

int main() {

 char c;

 c = 168;
 printf("c vale: %c\n\n" , c);
 return 0;

}
```

```
c vale: ?
```

Casting - Implícito

Riscos de conversão implícita:

```
#include<stdio.h>

int main() {

 char c;

 c =936;
 printf("c vale: %c\n\n" , c);
 return 0;

}
```

```
teste2.c:7:5: warning: implicit conversion from 'int' to 'char' changes value
      from 936 to -88 [-Wconstant-conversion]
c = 936;
  - ^~~
1 warning generated.
```

```
c vale: ?
```

Casting - Implícito

Riscos da conversão implícita:

O que aconteceu?

- ➔ Variáveis do tipo *char* ocupa apenas 1 byte;
- ➔ 936 ocupa 2 bytes.

936 - 00000011 10101000
168 - 10101000

Casting - Implícito

Riscos da conversão implícita:

- **Perda da precisão:** conversão de *double* para *float*;
- **Perda por excesso *(overflow)*:** conversão de um *long* para um *int*;
- **Problemas na impressão na tela.**

Casting - Explícito

Coloca-se o tipo entre parênteses para fazer a conversão de uma variável.

```c
#include<stdio.h>

int main() {

 int a = 12;
 float b;
 b = (float) a;
 printf("b vale: %f\n\n" , b);
 return 0;

}
```

Casting - Explícito

Coloca-se o tipo entre parênteses para fazer a conversão de uma variável.

```c
#include<stdio.h>

int main() {

 int a = 12;
 float b;
 b = (float) a;
 printf("b vale: %f\n\n" , b);
 return 0;

}
```

```
b vale: 12.000000
```

Casting - Explícito

➔ **Uso:**
A construção (tipo) é um operador unário.

Exemplo:

```
int a, b;
float c;
a = 5;
b = 2;
c = (float) a/b; //EQUIVALENTE A ((float) a)/b
```

Exercício

1. Escreva um programa que calcule a área do círculo. O valor mínimo do raio deve ser 2 e o máximo 10, ou seja, quando o usuário entrar com um valor fora desse intervalo, o programa deve pedir ao usuário que digite novamente (use constantes simbólicas).

Entrada:	**Saída:**
3 1115	28,26 78,5

$$A = \pi * r^2$$

Exercício

2. Escreva um programa que deve receber 3 números que serão armazenados em variáveis do tipo int, depois divida o primeiro pelo segundo e multiplique o resultado pelo terceiro sem perder nenhuma casa decimal (caso apareça). Após isso, imprima o resultado da operação completa junto dos três inteiros na forma float. Use conversão de tipos.

Entrada:

3 2 3

Saída:

4.500000 3.000000 2.000000 3.000000

Exercício

3. Escreva um programa que recebe um número do tipo float e exibe sua parte real. Após isso, o programa imprime na tela “O número é real” se sua parte real é diferente de 0 ou “O número é inteiro” se sua parte real é 0. Use conversão de tipos.

Entrada:

12.43
2

Saída:

0.430000 o numero eh real
0.000000 o numero eh natural