LABORATÓRIO DE PROGRAMAÇÃO

Estrutura de Decisão Simples e Composta em C

Profº. Sérgio Roberto Costa Vieira, M.Sc.
Cursos de Computação
1º. Período

Laboratório de Programação

Roteiro

- **Estrutura de Decisão em C**
  - **Instrução Simples if…**
  - **Instrução Composta if…else**
  - **Exercícios de Fixação**

Laboratório de Programação

Estrutura de Decisão

As estruturas de decisão (condicionais) são utilizadas para tomar uma decisão baseada no resultado da avaliação de uma condição de controle e seleciona uma ou mais ações possíveis (comandos) para serem executados pelo computador.

ESTRUTURA DE DECISÃO SIMPLES

As estruturas de decisão (condicionais) simples tem por finalidade tomar uma decisão e efetuar um desvio no processamento do programa, dependendo da condição.

Sintaxe:

```
if ( condição ) {
        sequência de comandos;
}
```

ESTRUTURA DE DECISÃO SIMPLES

Sendo a condição *Verdadeira*, será executada a instrução que estiver escrita após a instrução if.

Se após a condição, tiver apenas uma instrução a ser executada, não necessita criar um bloco. Caso tenha mais de uma instrução a ser executada após a condição, elas devem estar escritas dentro de um bloco { ... }.

```
if ( condição )
    instrução a ser executada;
```

```
if ( condição ) {
    instrução1 a ser executada;
    instrução2 a ser executada;
    instrução1 a ser executada;
}
```

Laboratório de Programação

Exemplo de Estrutura de Decisão Simples

```
/*Programa que faz a leitura de dois valores numéricos, efetuar a adição e apresentar o seu resultado, caso o valor somado seja maior que 10. */
#include<stdio.h>
#include<conio.h>
int main() {

 int A, B, X;
 printf("\n Informe o valor para a variável A: \n");
 scanf(" %d", &A);
 printf("\n Informe o valor para a variável B: \n");
 scanf(" %d", &B);
 X = A + B;
 if ( X > 10)
    printf("\n O valor da variável X equivale a: %d", X);

 getch();
 return(0);
}
```

Laboratório de Programação

Exemplo de Estrutura de Decisão Simples

```
/*Programa que faz a leitura de dois valores numéricos e independentemente da ordem em que foram informados, eles devem ser impressos na ordem crescente. */
#include<stdio.h>
#include<conio.h>
int main() {
   int A, B, X;
   printf("\n Informe o valor para a variável A: \n");
   scanf(" %d", &A);
   printf("\n Informe o valor para a variável B: \n");
   scanf(" %d", &B);
   if ( A > B) {
       X = A;
       A = B;
       B = X;
    }
    printf("\n Os valores ordenados sao:  %d  e  %d", A, B);
getch();
return(0);  }
```

OPERADORES RELACIONAIS

Nos exemplos anteriores, foi utilizado o sinal de > (maior que) para verificar o estabelecido da variável quanto ao seu valor lógico.

As verificações são efetuadas com a utilização dos chamados operadores relacionais:

Operador	Relação
>	Maior-que
>=	Maior-ou-igual-a
<	Menor-que
<=	Menor-ou-igual-a
==	Igual-a
!=	Diferente de

ESTRUTURA DE DECISÃO COMPOSTA

Agora vamos aprender a usar a instrução **if ... else**.

Sendo a condição ***Verdadeira***, será executada a instrução que estiver posicionada entre as instruções **if e else.**

Sendo a condição ***Falsa***, será executada a instrução que estiver posicionada logo após a instrução **else**.

Sintaxe:

```
if ( condição )
        instrução para condição verdadeira;
else
        instrução para condição falsa;
```

ESTRUTURA DE DECISÃO COMPOSTA

Caso exista mais de uma instrução verdadeira ou falsa para uma determinada condição, elas devem estar inseridas em um bloco { ... }.

```
if ( condição ) {
        instrução1 para condição verdadeira;
        instrução2 para condição verdadeira;
} else {
        instrução1 para condição falsa;
        instrução2 para condição falsa;
}
```

Laboratório de Programação

Exemplo de Estrutura de Decisão Composta

```
/*Programa que faça a leitura de duas variáveis, efetuar a soma, se for maior que 10 , atribuir a soma mais 5, senão atribuir menos 7. */
#include<stdio.h>
#include<conio.h>
int main() {
 int A, B, X;
 printf("\n Informe um valor para a variável A ..... \n");    scanf(" %d", &A);
 printf("\n Informe um valor para a variável B ..... \n");    scanf(" %d", &B);
 X = A + B;
 printf("\n O resultado equivale a: ” ) ;
 if ( X >= 10)
        printf(“%d”, X + 5);
 else
        printf(“%d”, X - 7);
 getch();
 return(0);
}
```

Laboratório de Programação

Exemplo de Estrutura de Decisão Composta

```
/*Programa que faça a leitura das notas de um aluno e calcule a media. */
#include<stdio.h>
#include<conio.h>
int main() {
 float N1, N2, N3, N4, MD;
 printf("\n Entre com a Nota 1 ..... \n");    scanf(" %f", &N1);
 printf("\n Entre com a Nota 2 ..... \n");    scanf(" %f", &N2);
 printf("\n Entre com a Nota 3 ..... \n");    scanf(" %f", &N3);
 printf("\n Entre com a Nota 4 ..... \n");    scanf(" %f", &N4);
 MD = (N1 + N2 + N3 + N4) / 4;
 if ( MD >= 5) {
       printf("\n Aluno Aprovado com Media = ”);
       printf(“%2.2f \n”, MD);
} else {
       printf("\n Aluno Reprovado com Media = ”);
       printf(“%2.2f \n”, MD);
}
 getch();
 return(0); }
```

Laboratório de Programação

Operadores Lógicos

- Operam sobre valores Verdadeiro ou Falso, e são avaliados também como Verdadeiro ou Falso.

Operador	Relação
&&	E (AND)
||	OU (OR)
!	NÃO (NOT)

- E (AND) será Verdade se os dois operandos forem Verdade.
- OU (OR), se algum dos dois forem Verdade.
- NÃO (NOT), se o operando for Falso.

Exemplo:

"A && B" (Depende de A e B)

"!FALSE" (Verdade)

Laboratório de Programação

Operadores Lógicos

- **&& (E lógico): retorna verdadeiro se ambos os operandos são verdadeiros e falso nos demais casos.**

  Exemplo: if(a>2 && b<3).

- **|| (OU lógico): retorna verdadeiro se um ou ambos os operandos são verdadeiros e falso se ambos são falsos.**

  Exemplo: if(a>1 || b<2).

- **!(NÃO lógico): usada com apenas um operando. Retorna verdadeiro se o operando é falso e vice-versa.**

  Exemplo: if(!var).

Laboratório de Programação

Exemplo do Operador && (and)

```
/*Programa exemplo do operador &&*/
#include<stdio.h>
#include<conio.h>
int main() {
 int NUMERO;

 printf("\n Entre com um numero entre 0 e 9 \n");
 scanf(" %d", &NUMERO);

 if ( NUMERO >= 0 && NUMERO <= 9)
       printf(“Valor Válido”);
 else
       printf(“Valor Inválido”);
 getch();
 return(0);
}
```

Laboratório de Programação

Exemplo do Operador || (or)

```
/*Programa exemplo do operador || (OU)      */
#include<stdio.h>
#include<conio.h>
int main() {
 int CODIGO;
 printf("\n Entre com o código de acesso:  \n");
 scanf(" %d", &CODIGO);
 if ( CODIGO == 1 || CODIGO == 2 || CODIGO == 3) {
       if (CODIGO == 1)
            printf(“UM \n”);
        if (CODIGO == 2)
            printf(“DOIS \n”);
        if (CODIGO == 3)
            printf(“TRÊS \n”);
 } else
       printf(“Código Inválido \n”);
 getch();
 return(0);  }
```

Laboratório de Programação

Exemplo do Operador ! (NÃO)

```
/*Programa exemplo do operador ! (NOT)*/
#include<stdio.h>
#include<conio.h>
int main() {
 int VALOR;
 printf("\n Entre com um valor inteiro positivo:  \n");
 scanf(" %d", &VALOR);
 if ( ! (VALOR >= 0) )
       printf(“Valor Inválido \n”);
 else
       printf(“Valor válido, você informou %d  \n”,  VALOR );
 getch();
 return(0);
}
```

Laboratório de Programação

Arquivo de Cabeçalho ISO 646

Cabeçalho ISO 646

À medida que uma linguagem de programação evolui, ela recebe novos recursos. Assim, a linguagem C "ganhou" o arquivo de cabeçalho ISO 646 que permite usar palavras nos operadores lógicos.

No lugar de:

&& usar and

|| usar or

! usar not

Laboratório de Programação

Exemplo do Cabeçalho ISO 646

```
/*Programa exemplo do cabeçalho ISO 646*/
#include<stdio.h>
#include<conio.h>
#include<iso646.h>
int main() {
 int NUMERO;

 printf("\n Entre com um numero entre 0 e 9 \n");
 scanf(" %d", &NUMERO);

 if ( NUMERO >= 0 and NUMERO <= 9)
       printf(“Valor na faixa de 0 a 9”);
 else
       printf(“Valor fora da faixa de 0 a 9”);
 getch();
 return(0); }
```

Laboratório de Programação

Exemplo do Cabeçalho ISO 646

```
/*Programa exemplo do cabeçalho ISO 646 mostrando Divisibilidade*/
#include<stdio.h>
#include<conio.h>
#include<iso646.h>
int main() {
 int N, R4, R5;
 printf("\n Entre com um valor inteiro natural \n");
 scanf(" %d", &N);
 R4 = N – 4 * (N / 4);
 R5 = N – 5 * (N / 5);
 if ( R4 == 0 and R5 == 0)
       printf(“%d \n”,  N);
 else
       printf(“Valor nao é divisivel por 4 e 5 \n ”);
 getch();
 return(0); }
```

Laboratório de Programação

Exercícios de Fixação

1 – Faça um algoritmo para ler um número e imprimir se ele é “PAR” ou “ÍMPAR”.

2 – Faça um algoritmo para ler um número e se ele for maior do que 20, então imprimir a metade do número, senão imprimir o seu quadrado.

3 – Faça um algoritmo para ler um número e imprimir uma das mensagens: é multiplo de 3 ou não é multiplo de 3.

Laboratório de Programação

Exercícios de Fixação

4 – Faça um algoritmo que leia dois números e efetue a adição. Caso o valor somado seja maior que 20, este deverá ser apresentado somando-se a ele mais 8; caso o valor somado seja menor ou igual a 20, este deverá ser apresentado com a diferença de -5.

5 – A prefeitura de Manaus abriu uma linha de crédito para os funcionários estatutários. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto. Fazer um algoritmo que permita entrar com o salário bruto e o valor da prestação e informar se o empréstimo pode ou não ser concedido.

LABORATÓRIO DE PROGRAMAÇÃO

Estrutura de Decisão Simples e Composta em C

Profº. Sérgio Roberto Costa Vieira, M.Sc.
Cursos de Computação
1º. Período