Quantas maçãs não caíram na cabeça de Newton antes de ele ter percebido a dica! Robert Frost.

VALORES LÓGICOS

Como visto no Laboratório 3 existem quatro tipos de dados em C (**int**, **char**, **float**, **double**), mas não existe nenhum tipo para representar valores lógicos (Verdadeiro e Falso). Em C o valor lógico FALSO é representado por 0 (ZERO) e tudo o que for diferente de 0 (ZERO) é considerado VERDADE. Os valores lógicos podem ser resultados da avaliação de **operadores relacionais** mostrados na Tabela T1:

Tabela T1

Operador	Exemplo	Significado
==	a == b	igual
>	a > b	maior
>=	a >= b	maior/igual
<	a < b	menor
<=	a <= b	menor/igual
!=	a != b	diferente

PT1: Implemente o programa a seguir e teste para os valores 7 e 5. Interprete o resultado obtido para cada **operador**.

```
1: #include <stdio.h>
2: main()
3: { int x, y;
4:   printf("Digite dois inteiros: ");
5:   scanf("%d%d",&x,&y);
6:   printf("%d == %d : %d \n",x,y,x==y);
7:   printf("%d  > %d : %d \n",x,y,x>y);
8:   printf("%d >= %d : %d \n",x,y,x>=y);
9:   printf("%d  < %d : %d \n",x,y,x<y);
10:  printf("%d <= %d : %d \n",x,y,x<=y);
11: printf("%d  != %d : %d \n",x,y,x!=y);}
```

Um cuidado especial deve ser tomado ao se usar o operador **==**, pois um erro freqüente é se trocar este por **=**. O operador **==** verifica a igualdade de duas expressões ao passo que o operador **=** é uma atribuição.

IF-ELSE

Os **operadores relacionais** têm especial utilidade quando associados à instrução **if-else** e servem para indicar uma **condição**. A sintaxe da instrução **if-else** é:

```
if (condição)
  instrução 1;
else
  instrução 2;
```

Observe que a **condição** do **if** deve estar sempre dentro de parênteses e que as instruções 1 e 2 são seguidas de ponto e virgula (**;**).

PT2: Implemente o programa a seguir e diga o que ele faz.

```
1: # include <stdio.h>
2: main()
3: { int x;
4:   printf("Introduza um No:");
5:   scanf("%d", &x);
6:   if (x >= 0)
7:     printf("Numero positivo \n");
8:   else
9:     printf("Numero negativo \n");
10:}
```

Observe que a condição (**x >= 0**) é avaliada e se for verdadeira, a instrução relativa à impressão da mensagem: "Numero positivo" será executada. **Senão**, a instrução relativa à impressão da mensagem: "Numero Negativo" é que será impressa. Observe que após cada instrução, seja do **if** ou do **else**, é necessário usar ponto e virgula (**;**).

PE1: Baseado no **PT2** implemente um programa que indique se o inteiro lido é zero ou não (Dica: use o **operador lógico** !=). Teste para 7, 0 e -5.

PE2: Reimplemente o **PE1** e use **(x)** no lugar de **(x != 0)**. Observe que em C o valor de uma variável ou constante pode ser aproveitado pelo programador como valor lógico.

PE3: Crie um programa que leia o valor do salário e caso o mesmo seja inferior a 100.000,00 adiciona 1.000,00.

BLOCO DE INSTRUÇÕES

A sintaxe do **if-else** mostrada até agora só permite apenas uma instrução após o **if** ou uma outra após o **else**. Para executar mais de uma instrução, estas devem ser escritas entre **{}** (dentro do **bloco de instruções)**. Após as chaves (**bloco de instruções)** não se usa **;**.

PT3: Use o programa abaixo e diga o que ele faz. Teste para 2 e 1; -6 e 0.

```
1:  #include <stdio.h>
2:  main()
3:  {
4:    int x, y, tmp;
5:    printf("Introduza 2 nos: ");
6:    scanf("%d %d", &x, &y);
7:    if (x>y)
8:    {
9:       tmp = x; // indentação
10:      x = y;     // melhora a
```

```
11:      y = tmp; // legibilidade
12:  }
13:  printf("%d %d \n",x,y);
14: }
```

O que ocorreria se não fosse usado **{}**?

Observe que a **indentação/paragrafação** (deslocamento do texto) facilita a leitura do programa, embora este espaço a mais seja ignorado pelo compilador.

INSTRUÇÕES ENCADEADAS

Para situações em que deve se avaliar mais de uma condição, os comandos **if-else** podem ser usados de forma **aninhada** (um comando dentro do outro).

PT4: O que faz o programa a seguir?

```
1: #include <stdio.h>
2: main()
3: {
4:  float soldo;
5:  printf("Qual o salário: ");
6:  scanf("%f",&soldo);
7:  if (soldo <= 0)
8:    printf("Salario: Valor Invalido \n");
9:  else
10:   if (soldo > 1000)
11:  printf("Imposto = %.2f\n",soldo*0.10);
12:  else
13:  printf("Imposto = %.2f\n",soldo*0.05);
14: }
```

Como você melhoraria a clareza deste programa utilizando **{}**? E a **indentação** nas linhas 11 e 13?

Observe que tanto o **if** da linha 10, como o **else** da linha 12 só será executado se a condição do **if** da linha 7 (soldo <= 0) for falsa, ou seja, se o valor da variável **soldo** for um valor válido e, portanto, maior que zero.

PE4: Escreva um programa que calcula o salário bruto, o salário líquido e o imposto a pagar de acordo com a regra da Tabela **T2**:

Tabela T2

Salário	Imposto
< 1313	0.0%
>= 1313 e < 2625	15.0%
>= 2625	27.5%

Dica: Use **if-else** para obter a taxa. Use **if**s aninhados para calcular as alíquotas de 15% e 27.5%. Para a taxa de 15% deve-se usar 0.15 ao invés de 15/100. Por quê (a menos do uso de **conversão explícita** ou **cast**)?

PE5: Escreva um programa em C para executar as seguintes ações:

a) Leia um valor para X. Se X < 0 escreva uma mensagem dizendo que o valor é negativo e peça para digitar um valor positivo, imprima o novo valor digitado e passe a usá-lo.
b) Se X está entre 0 e 10, imprima uma mensagem dizendo que X está entre 0 e 10, seguido do valor de X.
c) Se X está entre 10 e 100, imprima "O valor de X está entre 10 e 100. X = . Se X > 100, imprima " O valor de X eh maior que 100, X= " , e emita 7 sinais sonoros.

PE6 As raízes de uma equação quadrática da forma $\mathbf{ax^2 + bx + c = 0}$ são reais se e somente se o discriminante dado por $\mathbf{b^2 - 4ac}$ for maior ou igual a zero. Fazer um programa em C para ler os valores dos coeficientes a, b, c e imprimir o valor do discriminante dizendo se as raízes são reais ou não. Se as raízes forem reais, calcular as raízes.

OPERADORES LÓGICOS

Uma opção para consideração simultânea de duas ou mais condições é utilizar os **operadores lógicos** dados na **Tabela T3**:

Tabela T3

Operador	Significado	Exemplo
&&	AND	x>=1 && x< 19
||	OR	x == 1 || x < 0
!	NOT	! verdade

O funcionamento dos operadores lógicos é fornecido nas **Tabelas T5, T6 e T7**:

Tabela T4

&&		**cond1**	
		Verdade	**Falso**
cond2	**Verdade**	Verdade	Falso
	Falso	Falso	Falso

Tabela T5

||		**cond1**	
		Verdade	**Falso**
cond2	**Verdade**	Verdade	Verdade
	Falso	Verdade	Falso

Tabela T6

!	**Verdade**	**Falso**
	Falso	Verdade

PT5: Crie um programa que aplique 10% de imposto aos solteiros e 8% aos casados.

```
1: #include <stdio.h>
2: main()
3: {
4:   float soldo;
5:   char est_civil;
6:   printf("Soldo: "); scanf("%f",&soldo);
7:   printf("Est.C:"); scanf(" %c",&est_civil);
8:   if (est_civil == 'C' || est_civil == 'c')
9:     printf("Imposto: %.2f\n",soldo*0.09);
10: else
11:   if (est_civil=='S' || est_civil == 's')
12:    printf("Imposto: %.2f\n",soldo*0.10);
13:   else
14:    printf("Estado civil incorreto!!! \n");
15:}
```

Observe que a tag **%c** do scanf da leitura do estado civil deve começar com um espaço!
Observe, novamente, que os comandos da linha 11 até a linha 14 só serão executados se o **else** da linha 10 for utilizado. Para aumentar a clareza deste programa uma opção seria usar as chaves**{}**. Outra observação é que no **if** da linha 8 duas condições são reunidas através do operador lógico || (OR). É verificado se a variável est_civil contém o carectere 'C' ou 'c'. Qualquer uma destas opções resulta na execução do comando de impressão da linha 9.

PE7: Reescreva o **PT5** usando **getche** e **toupper**, sabendo que a sintaxe destas funções é: `c = getche()` tal que um caractere do teclado será lido e `c = toupper(c)` tal que letras minúsculas contidas em c são transformadas em maiúsculas. Observe que a condição do **if** da linha 8 do **PT5** pode ser simplificada. Use <conio.h> e <ctype.h>.

A **Tabela T7** fornece a ordem de precedência (ordem de prioridade de execução) dos **operadores lógicos e relacionais**:

Tabela T7

Ordem	Operador
1	< <= > >=
2	== !=
3	&&
4	||
5	?:

Portanto a seguinte expressão:
`if (x != 10 || y > 1 && y < 10)` equivale à:
`if ( (x != 10) || ( (y > 1) && (y < 10) ) )`.

PE7 Preparar um programa em C para ler os comprimentos dos três lados de um triângulo (S_1, S_2, e S_3) e determinar que tipo de triângulo tem-se, com a base nos seguintes casos. Sejam A o maior dos lados de S_1, S_2, e S_3 e B e C os outros dois lados. Então:

- Se $A \geq B + C$ nenhum triângulo é formado.
- Se $A < B + C$ e se
  1) $A^2 = B^2 + C^2$ um triângulo retângulo é formado,
  2) $A^2 > B^2 + C^2$ um triângulo obtusângulo é formado
  3) $A^2 < B^2 + C^2$ um triângulo acutângulo é formado.

OPERADOR CONDICIONAL =?:

O **operador condicional** é semelhante ao comando **if-else**. Sua sintaxe é:

```
(condição) ? (Verdade = ação 1):(Falso = ação 2)
```

PT6: Criar um programa que calcula os aumentos de salário tal que se o mesmo for > 1000, então salário deve ser aumentado 5%, senão aumentar em 7%.

```
1: # include <stdio.h>
2: main()
3: {
4:   float s;
5:   printf("Qual o soldo: ");
6:   scanf("%f",&s);
7:   s = (s> 1000) ? (s*1.05) : (s*1.07);
8:   printf("Novo soldo: %.2f\n",s);
9: }
```

O operador **?** devolve o resultado de s*1.05 se s > 1000, senão devolve s*1.07.

Embora semelhante ao **if-else**, o operador **?** só é capaz de fornecer um resultado ao passo que o **if-else** pode indicar vários comandos a serem executados (através do uso de **{}**).

SWITCH

Quando é necessário escolher uma dentre múltiplas opções, o comando **switch** pode ser mais apropriado que o uso de instruções **if-else** encadeadas. A sintaxe do switch é dada por:

```
switch (expressão)
{
  case constante1: intruções1;
  case constante2: instruções2;
```

```
  ...
  case constante_n: instruções_n;
  [default: instruções;]
}
```

A palavra expressão indica qualquer avaliação cujo resultado seja um valor numérico dos tipos **char**, **int** ou **long**. O **switch** compara o resultado da expressão com o valor de cada constante que segue a cada um dos **case**. Se o valor não for igual a nenhuma das constantes apresentadas pelos **case**, então, são executadas as instruções que seguem o **default**. Mas, a utilização do comando **default** é opcional. Além disso, cada **case** do **switch** só pode avaliar uma única constante do tipo **char**, **int** ou **long**. Uma observação importante é que após um comando **case** ser selecionado, todos os **cases** subseqüentes e suas instruções serão executados. Verifique esta afirmativa a partir da execução do **PT7**.

PT7: Verifique o problema do seguinte programa:

```
1: # include <stdio.h>
2: main()
3: {
4:   char est;
5:   printf("Qual o estado Civil: ");
6:   scanf(" %c",&est); // ou est = getchar();
7:   switch(est)
8:   {
9:      case 'C': printf("Casado \n");
10:     case 'S': printf("Solteiro \n");
11:     case 'D': printf("Divorciado \n");
12:     case 'V': printf("Viúvo \n");
13:    default: printf("Estado Incorreto \n");
   }
}
```

Use as letras 'C', 'D' e 'X'. Observe que, por exemplo, no caso da letra 'C' os demais comandos, associados aos **case** subseqüentes, são executados.

BREAK

Para interromper a execução de um **switch,** continuando o programa na instrução seguinte ao mesmo, existe o comando **break**.

PT8: Corrija o **PT7** usando **break**.
Substituir as linhas **9** até **13** do **PT7** por:

```
9:   case 'C': printf("Casado \n"); break;
10: case 'S': printf("Solteiro \n"); break;
11: case 'D': printf("Divorciado \n"); break;
12: case 'V': printf("Viúvo \n") break;
13: default: printf("Estado Incorreto \n");
```

Observe que o último **case** ou o **default** do **switch** não precisa do comando **break**, pois não existem demais instruções do **switch**.

PE8: Modifique o programa **PT8** de modo que o programa aceite maiúsculas ou minúsculas sem usar a função toupper();
Dica: use o seguinte trecho de código como referência:

```
case 'c':
case 'C': printf("Casado \n"); break;
```

PT9: O Programa a seguir lê uma operação binária entre dois inteiros e apresente o resultado dessa operação entre os dois inteiros.

```
# include <stdio.h>
main()
{
int num1, num2, res = 0;
char op;
printf("Escreva uma expressão: ");
scanf("%d %c %d", &num1, &op, &num2);
switch(op)
{
  case '+': res = num1 + num 2; break;
  case '-': res = num1-num2; break;
  case '*':
  case 'x':
  case 'X': res = num1*num2; break;
  case '/':
  case '\\':
 case ':': res = num1/num2; break;
}
printf("%d %c %d = %d\n", num1,op,num2, res);
}
```

Note que como \ é um caractere especial, então, deve ser escrito duas vezes com \\.

PE9: Escreva um programa cujas entradas são o salário e o sexo (F/M) e a saída é o salário líquido (descontada uma taxa de imposto de 10% para mulheres e 15% para homens). Utilize o comando **switch** para determinar o valor da variável associada ao imposto e a dica do **PE7** para a questão das letras maiúsculas e minúsculas.

PROGRAMAS BÁSICOS

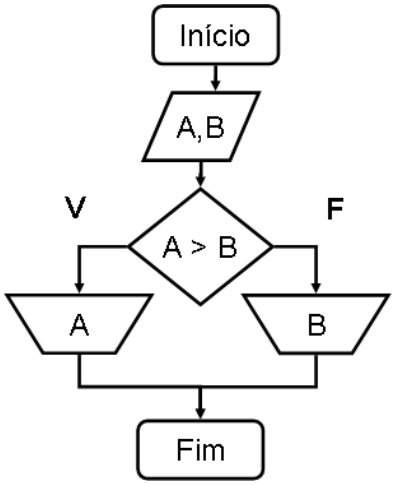
PB1: Escreva um programa que leia o número de horas e imprima o seu correspondente em minutos (indicado pelo usuário com a letra m) ou em segundos (indicado pelo usuário com a letra s). Use comandos **if-else** aninhados.

PB2: Escreva um programa que verifica se um ano é bissexto ou não. Dica: A definição de ano bissexto é que:
(i) Se ele não é divisível por 100, mas é por 4, então é bissexto.
(ii) Se ele é divisível por 100 e por 400, então, é bissexto.
(iii) Os demais anos não são bissextos.

PB3: Escreva um programa que indica o número de dias existentes em um mês (fevereiro = 28 dias), usando apenas **if-else**.

PB4: Reimplemente o **PB3**, mas usando **switch**.

PB5: Implemente o fluxograma descrito a seguir em que é realizada a leitura de dois números e é impresso o maior deles.



PB6: Reimplemente o **PB5**, mas utilizando o operador ternário **?** e **:**.

PB7: Crie um programa para ler 3 números inteiros e imprimi-los em ordem crescente.

PROGRAMAS AVANÇADOS

PA1: Reimplemente o **PB1** usando **switch**.

PA2: Reimplemente o **PA1** sem usar **break**.

PA3: Reimplemente o **PB4** usando **switch** , mas sem usar o comando **break**.

PA4: Escreva cada fluxograma associado aos **PA**s anteriores (Dica: você pode usar a ferramenta da barra Autoformas do Powerpoint para criar figuras como a do **PB5**).