Universidade de São Paulo - São Carlos,SP
Instituto de Ciências Matemáticas e de Computação

USP - ICMC - SSC
SSC 0301 - 2o. Semestre 2013

Disciplina de Introdução à Computação para Engenharia Ambiental

Prof. Dr. Fernando Santos Osório
LRM - Laboratório de Robótica Móvel do ICMC / CROB-SC
Email: fosorio icmc. usp. br ou fosorio gmail. com
Página Pessoal: http://www.icmc.usp.br/~fosorio/

Material on-line:
Wiki ICMC - http://wiki.icmc.usp.br/index.php
Wiki SSC0301 - http://wiki.icmc.usp.br/index.php/SSC-301-2013(fosorio)

Aula 03

Linguagem de Programação "C"

Agenda:

- **Programa Seqüencial em "C"**
- **Programas com IF**
  - **> Operadores: Relacionais, Lógicos, Aritméticos, ...**
  - **> Expressões Condicionais**
  - **> Fluxo de Execução**
  - **> Funções da linguagem "C"**

Informações Complementares a Atualizadas:
Consulte REGULARMENTE o material disponível na WIKI
http://wiki.icmc.usp.br/index.php/SSC-301-2013(fosorio)

Programa Seqüencial em "C"

Linguagem "C" : Exemplo de programa seqüencial em "C"

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

int main(int argc, char *argv[])
{
   int Nota1, Nota2;
   double Media;

   printf ("\nCalculo da Media\n");
   printf ("Valor1: ");   scanf("%d",&Nota1);
   printf ("Valor2: ");   scanf ("%d",&Nota2);
   Media = Nota1 + Nota2 /  2;
   printf ("Media = %2.2lf\n", Media);

   system("PAUSE");
   return 0;
}
```

>>> gcc media.c -o media.exe -lm

USP – SSC0301 - Semestre 2013-2
Introdução à Computação – Eng. Ambiental
Prof. Fernando Osório Teoria

Programa Seqüencial em "C"

Linguagem "C" : Exemplo de programa seqüencial em "C"

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

int main(int argc, char *argv[])
{
   int Nota1, Nota2;
   double Media;

   printf ("\nCalculo da Media\n");
   printf ("Valor1: ");   scanf("%d",&Nota1);
   printf ("Valor2: ");   scanf ("%d",&Nota2);
   Media = Nota1 + Nota2 /  2;
   printf ("Media = %2.2lf\n", Media);

   system("PAUSE");
   return 0;
}
```

Jogo dos 3 erros!

>>> gcc media.c -o media.exe -lm

Programa Seqüencial em "C"

Linguagem "C" : Exemplo de programa seqüencial em "C"

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

int main(int argc, char *argv[])
{
   int Nota1, Nota2;
   double Media;

   printf ("\nCalculo da Media\n");
   printf ("Valor1: ");   scanf("%d",&Nota1);
   printf ("Valor2: ");   scanf ("%d",&Nota2);
   Media = (double) ( Nota1 + Nota2 ) /  2.0;
   printf ("Media = %2.2lf\n", Media);

   system("PAUSE");
   return 0;
}
```

>>> gcc media.c -o media.exe -lm

Jogo dos 3 erros!

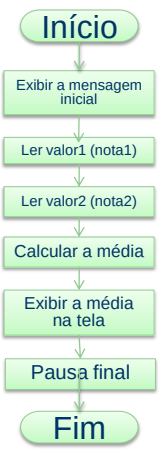
USP – SSC0301 - Semestre 2013-2
Introdução à Computação – Eng. Ambiental
Prof. Fernando Osório Teoria

Programa Seqüencial em "C"

Linguagem "C" : Exemplo de programa seqüencial em "C"

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

int main(int argc, char *argv[])
{
   int Nota1, Nota2;
   double Media;

   printf ("\nCalculo da Media\n");
   printf ("Valor1: ");   scanf("%d",&Nota1);
   printf ("Valor2: ");   scanf ("%d",&Nota2);
   Media = (double) (Nota1 + Nota2 ) /  2.0;
   printf ("Media = %2.2lf\n", Media);

   system("PAUSE");
   return 0;
}
```

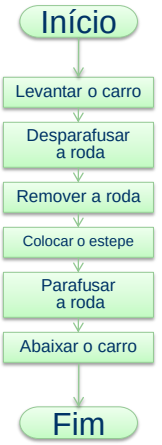
>>> gcc media.c -o media.exe -lm



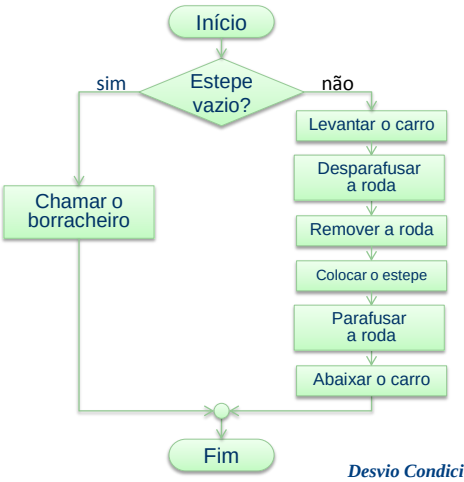
Fluxo de um Programa em "C"

Fluxo de Execução de um Programa

- **Fluxo Seqüencial (sem desvios)**
- **Fluxo com Desvio Condicional (com IF)**



Seqüencial



Desvio Condicional

Fluxo de um Programa em "C"

Fluxo de Execução de um Programa => Fluxo Seqüencial

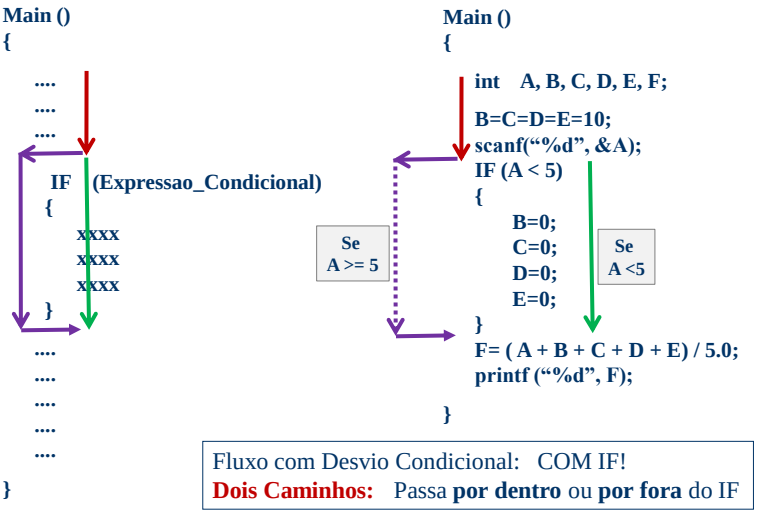
```
Main ()
{

    ....
    ....
    ....
    ....
    ....
    ....
    ....
    ....
    ....

}
```

```
Main ()
{

    int   A, B, C, D, E, F;

    A=1;
    B=2;
    C=3;
    D=4;
    E=5;
    F= A + B + C + D + E;
    printf ("%d", F);

}
```

Fluxo Seqüencial: SEM DESVIOS!

Fluxo de um Programa em "C"

Fluxo de Execução de um Programa => Desvios Condicionais (IF)



Fluxo com Desvio Condicional: COM IF!
Dois Caminhos: Passa **por dentro** ou **por fora** do IF

Comando IF

IF

if (<*expressão*>)
<*comando*>;
else
<*comando*>;

Expressão:
- Expressão lógica, relacional, aritmética

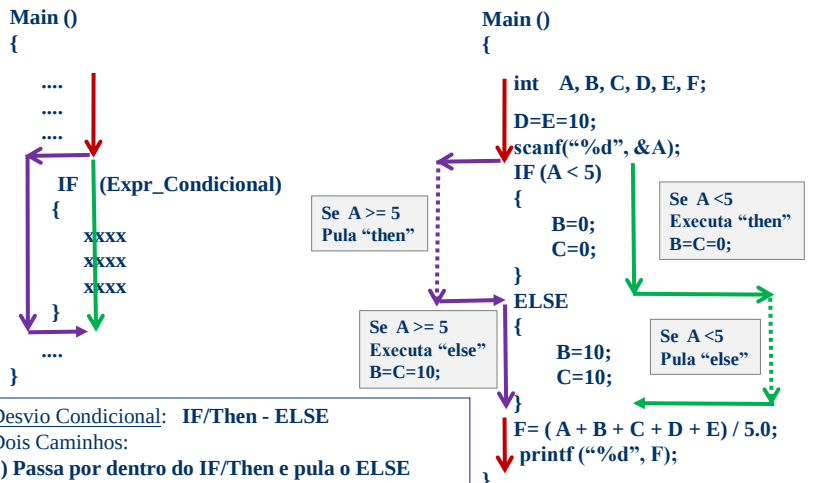
Comando:
- Comando simples ou bloco de comandos
- Bloco de comandos: { ... }
{ comando;comando; ... } ~ comando;

```
if  ( salario > 100.00)
   printf ("Salário maior que R$100,00\n");

if ( salario == 0.00)
  printf ("Este já foi demitido faz tempo...\n");
else
  printf ("Este ainda está sendo pago...\n");
```

CUIDADO: ***if (a == b)*** ***NÃO É O MESMO QUE*** ***if (a = b) !!***

Fluxo de um Programa em "C"

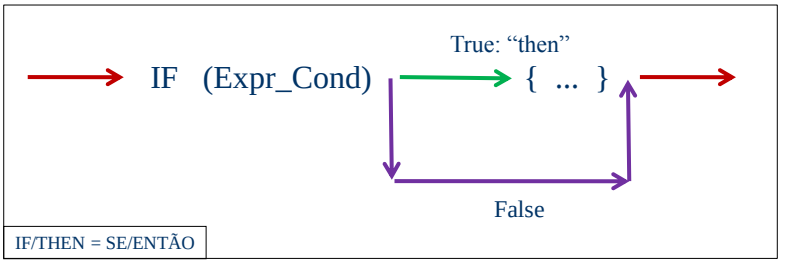
Fluxo de Execução de um Programa => Desvios Condicionais (IF)



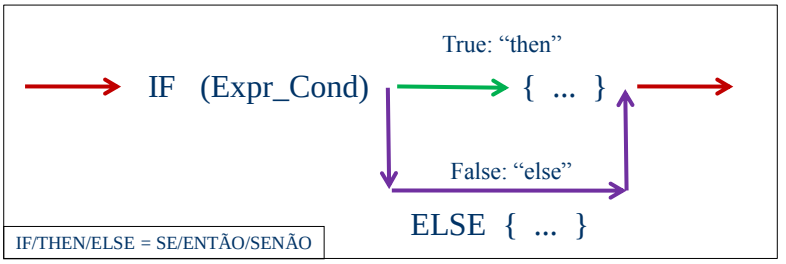
Desvio Condicional: **IF/Then - ELSE**
Dois Caminhos:
1) Passa por dentro do IF/Then e pula o ELSE
2) Pula o "then" do IF e passa por dentro do ELSE

Fluxo de um Programa em "C"

Fluxo de Execução de um Programa => Desvios Condicionais (IF)



IF/THEN = SE/ENTÃO



IF/THEN/ELSE = SE/ENTÃO/SENÃO

Expressões Condicionais

Expressões Condicionais: Lógicas, Relacionais, Expressões Gerais

1. **Operadores Lógicos:**
   - **And (&&), Or (||) , Not (!)**
2. **Operadores Relacionais**
   - **Comparação: >, >=, <, <=, !=, ==**
3. **Operadores e Expressões em "C":**
   - **Operadores de Atribuição,**
   - **Operadores Aritméticos,**
   - **Operadores Lógicos e Relacionais,**
   - **Operadores Bitwise (bit a bit),**
   - **Operadores de Assinalamento (var op= expr)**
   - **Pré-Pos Incremento-Decremento (var++, --var),**
   - **Operadores de Endereço**

Expressões Condicionais

Valores Lógicos:

Valores Lógicos = { Verdadeiro; Falso }
{ True = Não Zero; False = Zero }

TODA variável pode ser interpretada como um valor lógico!!

```
char letra='a';          /* Portanto letra é True */
int  qtde=0, idade=10;   /* Portanto qtde é False e idade é True */
```

Operadores Lógicos:

- **Operam sobre valores lógicos, números, caracteres, ... qualquer tipo de dado, resultando sempre em um valores lógico (True se diferente de Zero ou False)**
- **Operam sobre expressões, resultando valores lógicos de True ou False.**

OPERADORES: AND, OR, NOT

- **&&** Operação AND (duplo '&' comercial)
- **||** Operação OR (dupla '|' barra vertical)
- **!** Operador de negação NOT - "operador unário" (exclamação)

Expressões Condicionais

Operadores Lógicos:

AND (&&) => Só resulta verdadeiro se AMBOS operandos forem verdadeiros
OR (||) => Resulta em verdadeiro se PELO MENOS UM operando for True
NOT (!) => Inversão (negação) do valor lógico (True vira False, False vira True)

- Operam sobre expressões, resultando em valores lógicos:
Valor Lógico False (zero) ou Valor Lógico True (diferente de zero, ou seja, !0)

- Possuem a característica de "short circuit", ou seja, sua execução é curta e só é executada até o ponto necessário. Exemplos de "short circuit" :

```
( A == B ) && ( B == C )     /*  Se A != B para de avaliar a expressão  */
( A == B ) || ( B == C )     /*  Se A == B para de avaliar a expressão */
```

Caso típico:

```
if (i<= LIMIT) && list[i] != 0) ... ;   /* Não acessa list[i] quando i já passou o LIMIT */
```


Expressões Condicionais

Operadores Relacionais:

- Operam sobre valores e expressões, sejam estes aritméticos ou lógicos, resultando sempre em um valores lógico (True ou False / 0 ou 1)

OPERADORES:

Maior, Maior ou Igual, Menor, Menor ou Igual, Igual, Diferente

>	Operador "Maior que".	Exemplo: A > 10 (testa se A é maior que 10)
>=	Operador "Maior ou igual a".	Exemplo: A >= B
<	Operador "Menor que".	Exemplo: A < B
<=	Operador "Menor ou igual a".	Exemplo: A <= B
==	Operador "Igual a".	Exemplo: A == B
!=	Operador "Diferente de".	Exemplo: A != B

Cuidado!

- Não existem os operadores relacionais : "=<","=>" e "<>".
- Não confunda JAMAIS a atribuição ("=") com a comparação ("==").
- Use e ABUSE dos parênteses nas expressões em "C".

Expressões Condicionais

Operadores Lógicos e Relacionais

AND	OR	NOT
True AND True => True	True OR True => True	NOT True => False
True AND False => False	True OR False => True	
False AND True => False	False OR True => True	NOT False =>True
False AND False => False	False OR False => False	

AND => &&
OR => ||
NOT => !

EQUAL => = =
DIFFERENT => ! =

> , <
>= , <=

Exemplos de Expressões Lógicas:

```
Main()
{
   char  VL, A, B, C;
   int   X, Y;

   IF ( ! ( ( A && B ) || C ) )     { ... }
   IF ( X != Y )                    { ... }
   IF ( (( X+3.7)/2.0) > 6.0 )      { ... }
   IF ( (X >= 0.0) && (X <= 10.0) ) { ... }
   IF (!((X < 0.0) || (X > 10.0) )  { ... }
   IF ( (X+Y) != A )                { ... }
}
```


USP – SSC0301 - Semestre 2013-2
Introdução à Computação – Eng. Ambiental
Prof. Fernando Osório Teoria

Expressões Condicionais

Operadores Lógicos e Relacionais

AND	OR	NOT
True AND True => True	True OR True => True	NOT True => False
True AND False => False	True OR False => True	
False AND True => False	False OR True => True	NOT False =>True
False AND False => False	False OR False => False	

AND => &&
OR => ||
NOT => !

EQUAL => = =
DIFFERENT => ! =

> , <
>= , <=

Exemplos de Expressões Lógicas:

```
Main()
{
   char  VL, A, B, C;
   int   X, Y;

   VL = ! ( ( A && B ) || C );
   VL = X != Y;
   VL = ( ( X + 3.7 ) / 2.0 ) > 6.0;
   VL = ( X >= 0.0 ) && ( X <= 10.0 );
   VL = ! ( ( X < 0.0 ) || ( X > 10.0 ) );
   VL = ( X + Y ) != A;
}
```

Expressões Condicionais

Operadores Lógicos e Relacionais:

Exercícios...

1. **Faça uma expressão que teste se uma pessoa está habilitada a dirigir, ou seja, retorna verdadeiro para maiores de 18 anos com habilitação;**
2. **Faça uma expressão que teste se a idade de uma pessoa está entre 18 e 60 anos (apto a trabalhar);**
3. **Faça uma expressão que teste se a cotação do dolar ultrapassou a banda cambial prevista entre [1.5 .. 2.0];**
4. **Qual o resultado das expressões abaixo:**

   A) X=0; Y = 2 * X + 3;
   Resultado de Y && X;

   B) X=1; Y=0; Z=1;
   Resultado de ! ((X && Y) || Z)

   C) X =1; Y=1; Z=0;
   Resultado de (X && Y) + (X && Z);

Fluxo com Desvio Condicional

Fluxo de Execução de um Programa => Desvios Condicionais (IF)

Exercício:

Faça um programa adequado para o cálculo da nota final e da situação final de um aluno de uma disciplina avaliada da seguinte forma (aprovado/reprovado)

Requisitos para Aprovação:

Freqüência mínima: 70%

Média Final da Avaliação (MF)
Pesos: 34% Prova Teórica (PT) + 33% Prova Prática (PP) + 33% Trabalhos Práticos (TP)

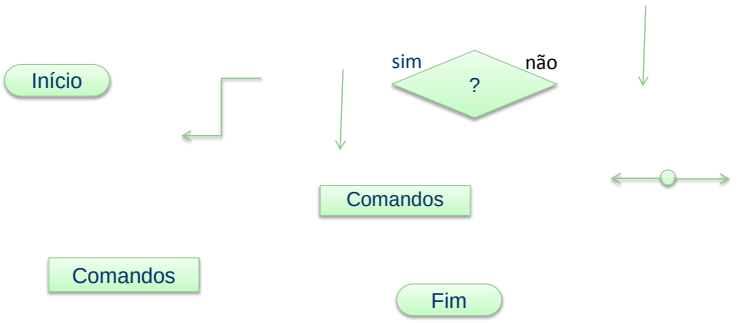
```
MF:  Se PT >= 5.0 e PP >= 5.0 e  TP >= 5.0
       Então  MF = 0.34 PT + 0.33 PP + 0.33 TP
       Senão  MF = Min{PT,PP,MF}

   Se MF >= 5.0
     Então  "Aprovado"
     Senão  Se MF >= 3.0
              Então "Recuperação"
              Senão "Reprovado"
```

Fluxo com Desvio Condicional

Exercício:

Faça um programa adequado para o cálculo da nota final e da situação final de um aluno de uma disciplina avaliada como descrito previamente (aprovado/reprovado)

Você pode iniciar a solução deste problema projetando o desenho do Algoritmo ...



USP – SSC0301 - Semestre 2013-2
Introdução à Computação – Eng. Ambiental
Prof. Fernando Osório Teoria

Fluxo com Desvio Condicional

Exercício:

Faça um programa adequado para o cálculo da nota final e da situação final de um aluno de uma disciplina avaliada como descrito previamente (aprovado/reprovado)

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

int Freq;
double PT,PP,TP;
double MinNotas, MFinal;

int main(int argc, char *argv[])
{
  printf("\n");
  printf("Media Teoria: ");
  scanf ("%lf",&PT);
  printf("Media Pratica: ");
  scanf ("%lf",&PP);
  printf("Media Trabalhos: ");
  scanf ("%lf",&TP);
  printf("Frequencia (0-100): ");
  scanf ("%d",&Freq);

  if (Freq < 70)
      printf("\nREPROVADO FREQ.!\n");
  else
  {
     if ((PT >=5.0) && (PP >= 5.0) && (TP >= 5.0))
          MFinal = 0.34*PT + 0.33*PP + 0.33*TP;
     else
     {
          if ((PT < PP) && (PT < TP))
              MinNotas = PT;
          else
              if (PP < TP)
                  MinNotas = PP;
              else
                  MinNotas = TP;
          MFinal = MinNotas;
     }
     if (MFinal >= 5.0)
         printf("\nAPROVADO=%.2lf!\n",MFinal);
     else
         printf("\nREPROVADO=%.2lf!\n",MFinal);

  }  /* fim else */

  printf("\n");

  system("PAUSE");
  return 0;
} /* fim programa */
```

Expressões "C" e Expressões Condicionais

Operadores de Atribuição: =
Operadores Aritméticos : + - * / %
Operadores Lógicos : && || !
Operadores Relacionais : > >= < <= != ==

Operadores Bit-a-Bit (Bitwise):

- Operam sobre valores (char, int), realizando operações bit a bit.
  Não podem ser aplicados sobre valores reais (float, double).

OPERADORES:

- & ➡ Bit AND
- | ➡ Bit OR
- ^ ➡ Bit XOR - Exclusive or (ou exclusivo)
- << ➡ Shift left
- >> ➡ Shift right
- ~ ➡ Bit NOT (complemento)

Obs.: x << n - irá rotar n vezes à esquerda

Cuidado! NÃO confunda o operador bitwise & com o operador lógico &&

Expressões "C" e Expressões Condicionais

Operador de Atribuição:

"=" ➡ Atribui um valor ou resultado de uma expressão contida a sua direita para a variável especificada a sua esquerda. Exemplos:

```
A = 10;   NS = 1.2 * Salario;   B = C * VALOR + GETVAL(X);
A = B = C = 1;   /* Aceita associação sucessiva de valores */
```

Operadores Aritméticos:

Operam sobre números e expressões, resultando valores numéricos

- "+" ➡ soma
- "-" ➡ subtração
- "*" ➡ multiplicação
- "/" ➡ divisão
- "%" ➡ módulo da divisão (resto da divisão inteira)
- "-" ➡ sinal negativo (operador unário)

Expressões "C" e Expressões Condicionais

Operadores de Assinalamento:

- Realizam duas operações de uma vez: operação + atribuição.

SINTAXE:

```
VAR = VAR OP EXPR;      /* Expressão usual. Exemplo:  A = A + B; */
VAR OP= EXPR;           /* Expressão 2-em-1. Exemplo:  A += B;  */
```

Onde podemos substituir OP acima, por um dos seguintes operadores :

+ → Soma	Exemplo:	X += 3.0
- → Subtração	Exemplo:	A -= X;
* → Multiplicação	Exemplo:	A += (1.2 * Salario);
/ → Divisão	Exemplo:	X /= 3.0
% → Modulo da divisão	Exemplo:	Valor %= 10;
>> → Shift right (bitwise)	Exemplo:	VByte >>= 1;
<< → Shift left (bitwise)	Exemplo:	VByte <<= 2;
& → And (bitwise)	Exemplo:	VByte &&= 0x0F;
^ → Xor - Exclusive or (bitwise)	Exemplo:	VByte ^= 0x01;
| → Or (bitwise)	Exemplo:	VByte |= 0x0F;

Expressões "C" e Expressões Condicionais

Operadores de Pré/Pós Incremento/Decremento:

- Realizam operações de incremento (+1) ou decremento (-1) de modo pré-operado (antes de usar a variável) ou pós-operado (depois de usar) .

SINTAXE:

```
++VAR;      /* Pré-Operado - Incremento. Equivale a VAR=VAR+1; */
--VAR;      /* Pré-Operado - Decremento. Equivale a VAR=VAR-1; */
VAR++;      /* Pós-Operado - Incremento. Equivale a VAR=VAR+1; */
VAR--;      /* Pós-Operado - Decremento. Equivale a VAR=VAR-1; */
```

O efeito deste operador (pré/pós) é importante dentro de uma expressão

Exemplos:

```
A = A + 1; X = A;   → X = ++A;
A = A - 1;  X = A;  → X = --A;
X = A ; A = A + 1   → X = A++
X = A ; A = A - 1   → X = A--
X = (++A) - (A--);      /* Qual o resultado? X e A */
```

Expressões "C" e Expressões Condicionais

Operadores de Endereço:

- Realizam operações com endereços de memória.

SINTAXE:

& ➡ Endereço de uma variável. Exemplo:

```
int var;
int *x;
x = &var;
```

*** ➡ Conteúdo do endereço especificado. Exemplo:**

```
var = *x;
```


Expressões "C" e Expressões Condicionais

Expressões Condicionais:

As expressões condicionais se apresentam da seguinte forma :

```
EXPR1 ? EXPR2 : EXPR3
```

Esta expressão é equivalente a :

```
SE EXPR1          Onde : Expr1 -> Condição de teste
ENTAO EXPR2              Expr2/Expr3 -> Valor retornado
SENAO EXPR3
```

Exemplo :

```
#define  IMIN(A,B)  ((A<B)?A:B)
B = ((X == Y)?X:Y);
```

Expressões “C” e Expressões Condicionais

Operadores da Linguagem "C"

Op.	Função	Exemplo "C"	Exemplo PASCAL
-	Menos unário	A = -B	A := -B
+	Mais unário	A = +B	A := +B
!	Negação Lógica	! FLAG	not FLAG
~	Bitwise NOT	A = ~B	A := not B
&	Endereço de	A = &B	A := ADDR(B)
*	Referência a ptr	A = *ptr	A := ptr^
sizeof	Tamanho de var	A = sizeof(b)	A := sizeof(b)
++	Incremento	++A ou A++	A := succ(A)
--	Decremento	--A ou A--	A := pred(A)
*	Multiplicação	A = B * C	A := B*C
/	Divisão inteira	A = B / C	A := B div C
/	Divisão real	A = B / C	A := B / C
%	Módulo da divisão	A = B % C	A := B mod C
+	Soma	A = B + C	A := B + C
-	Subtração	A = B - C	A := B - C
>>	Shift Right	A = B >> N	A := B shr N
<<	Shift Left	A = B << N	A := B shl N
>	Maior que	A > B	A > B
>=	Maior ou igual a	A >= B	A >= B
<	Menor que	A < B	A < B
<=	Menor ou igual a	A <= B	A <= B
==	Igual a	A == B	A = B
!=	Diferente de	A != B	A <> B
&	Bitwise AND	A = B & C	A := B and C
|	Bitwise OR	A = B | C	A := B or C
^	Bitwise XOR	A = B ^ C	A := B xor C
&&	Logical AND	flg1 && flg2	flg1 and flg2
||	Logical OR	flg1 || flg2	flg1 or flg2
=	Assinalamento	A = B	A := B
OP=	Assinalamento	A OP= B	A := A OP B

Expressões “C” e Expressões Condicionais

PRECEDÊNCIA DE OPERADORES:

Em uma expressão existe uma "ordem" de execução...

Exemplo:
*** e / precedem + e - em expr. aritméticas**

Solução:
Use e ABUSE dos parênteses!

Operators (grouped by precedence)

struct member operator	*name*.*member*
struct member through pointer	*pointer*->*member*
increment, decrement	++, --
plus, minus, logical not, bitwise not	+, -, !, ~
indirection via pointer, address of object	**pointer*, &*name*
cast expression to type	(*type*) *expr*
size of an object	sizeof
multiply, divide, modulus (remainder)	*, /, %
add, subtract	+, -
left, right shift [bit ops]	<<, >>
relational comparisons	>, >=, <, <=
equality comparisons	==, !=
and [bit op]	&
exclusive or [bit op]	^
or (inclusive) [bit op]	|
logical and	&&
logical or	||
conditional expression	$expr_1$? $expr_2$: $expr_3$
assignment operators	+=, -=, *=, ...
expression evaluation separator	,

Unary operators, conditional expression and assignment operators group right to left; all others group left to right.

PRECEDÊNCIA DE OPERADORES:

Em uma expressão existe uma "ordem" de execução...

Exemplo:
*** e / precedem + e - em expr. aritméticas**

Solução:
Use e ABUSE dos parênteses!

Nível Prec.	Operadores
15 (+ Alta)	() Parêntese [] Colchetes -> Ponteiro . Ponto
14	! Negação ~ NOT * Ponteiro & Endereço ++ Incr. -- Decr. (TYPE) Conversão - Menos (Unário) sizeof Tamanho
13	* Multiplic. / Divisão % Módulo
12	+ (Soma) - (Subtração)
11	>> Shift Left << Shift Right
10	< Menor <= Menor ou Igual > Maior >= Maior ou Igual
9	== Igual != Diferente
8	& (Bitwise AND)
7	^ (Bitwise XOR)
6	| (Bitwise OR)
5	&& AND Lógico
4	|| OR Lógico
3	?: Condicional
2	= Atribuição += -= *= /= %= |= ^= &= <<= >>= Atribuição Combinada
1	, Vírgula

Exercícios – Desvios Condicionais

EXERCÍCIOS:

1) **Faça um programa que leia 2 notas de um aluno, verifique se as notas são válidas e exiba na tela a média destas notas. Uma nota válida deve ser obrigatoriamente um valor entre 0.0 e 10.0, onde caso a nota não possua um valor válido, este fato deve ser informado ao usuário e o programa termina.**

2) **Faça um programa que leia duas notas de um aluno (nota 1 e nota 2) fornecidas pelo usuário que irá entrar as notas digitando pelo teclado. Usando estas 2 notas, calcule a média simples do aluno, e depois mostre na tela o resultado da média calculada. Se o aluno teve nota superior a 5.0, indique que ele foi "Aprovado", se o aluno teve nota entre 3.0 e 5.0 indique que ele está em "Recuperação", e se o aluno teve nota entre 0.0 e 3.0 indique que ele está "Reprovado".**

INFORMAÇÕES SOBRE A DISCIPLINA

USP - Universidade de São Paulo - São Carlos, SP
ICMC - Instituto de Ciências Matemáticas e de Computação
SSC - Departamento de Sistemas de Computação

Prof. Fernando Santos OSÓRIO
Web institucional: http://www.icmc.usp.br/
Página pessoal: http://www.icmc.usp.br/~fosorio/
Página do Grupo de Pesquisa: http://www.lrm.icmc.usp.br/
E-mail: fosorio [at] icmc. usp. br ou fosorio [at] gmail. com

Disciplina de Introdução a Computação – Eng. Ambiental
WIKI - http://wiki.icmc.usp.br/index.php/SSC-301-2013(fosorio)
> Programa, Material de Aulas, Critérios de Avaliação,
> Trabalhos Práticos, Datas das Provas, Notas