# SSC 0301 – IC para Engenharia Ambiental

## *Comando de seleção – if*

Prof. Márcio Delamaro e Prof. Jorge Luiz e Silva

delamaro@icmc.usp.br; jsilva@icmc.usp.br

# Recordando – Hello world

```
#include <stdio.h>

int main()
{
    printf("Hello world\n");
}
```

# Recordando - E/S

- `scanf(formato, &var1, &var2, ...);`
  - `scanf("%lf", &angulo);`
  - `scanf("%d %d %d", &dia, &mes, &ano);`
- `printf(formato, expr1, expr2, ...);`
  - `printf("O valor lido foi: %d\n", dia);`
  - ```
    printf("O valor do seno de %lf eh %lf\n",
                        x, sin(x));
    ```
    ```

# Recordando – expressões e atribuição

- Operadores aritméticos: $((a + b) * (c - d))/2$
- Operadores relacionais: $a > b$
- Atribuição
  - `k = k + 1;`
  - `y = sin(x) + cos(x);`
  - `j = (sin(x) == cos(x));` // Faz sentido????

# Execução sequencial

- Até agora, todos os comandos que vimos são executados em uma ordem pré-determinado
- Um comando aparece escrito “em baixo do outro”
- A ordem de execução dos comandos é aquela que aparece no texto do programa
- Em alguns casos precisamos de estruturas mais elaboradas para expressar os nossos algoritmos

# Comando de seleção (IF)

- Se esta chovendo, esperar. Senão, caminhar
- Escrever um programa que leia dois números e diga qual deles é maior
  a. Ler dois números em x e y
  b. Se x for maior escrever que o maior é x
  c. Caso contrário escrever que y é o maior

# Comando if – exemplo

```
#include <stdio.h>

int main()
{
int x, y;
    printf("Digite os dois numeros inteiros");
    scanf("%d %d", &x, &y);
    if ( x > y )
        printf("O maior eh: %d", x);
    else
        printf("O maior eh: %d", y);
}
```

# Comando if – exemplo

```
#include <stdio.h>

int main()
{
int x, y, maior;
    printf("Digite os dois numeros inteiros");
    scanf("%d %d", &x, &y);
    if ( x > y )
        maior = x;
    else
        maior = y;
    printf("O maior eh: %d", maior);
}
```

# Estrutura do if

- ```
  if ( expressão )
      comando 1;
  else
      comando 2;
  ```
- ```
  if ( expressão )
  {
      comando 1; comando2; ...
  }
  else
  {
      comando 3; comando 4; ...
  }
  ```

# Comando if – exemplo (2)

```
int main()
{
int x, y, maior, menor;
    printf("Digite os dois numeros inteiros");
    scanf("%d %d", &x, &y);
    if ( x > y )
    {
        maior = x;
        menor = y;
    }
    else
    {
        maior = y;
        menor = x;
    }
    printf("Maior: %d -- Menor: %d", maior, menor);
}
```

# Comandos if aninhados

- Dentro de um comando `if` pode aparecer qualquer comando, inclusive outro `if`
- Ler três números e dizer qual é o maior deles

# if aninhado – exemplo

```
int x, y, z, maior;
    printf("Digite os 3 numeros inteiros");
    scanf("%d %d %d", &x, &y, &z);
    if ( x > y )
    {
        if ( z > x )
            maior = z;
        else
            maior = x;
    }
    else
    {
         if ( z > y )
            maior = z;
         else
            maior = y;
    }
    printf("O maior eh : %d\n", maior);
```

# if aninhado – exemplo

```
int x, y, z, maior;
    printf("Digite os 3 numeros inteiros");
    scanf("%d %d %d", &x, &y, &z);
+---if ( x > y )
|    {
|     +--if ( z > x )
|     |       maior = z;
|     +--else
|     +--     maior = x;
|    }
+---else
|    {
|      +--if ( z > y )
|      |      maior = z;
|      +--else
|      +--    maior = y;
+---}
    printf("O maior eh : %d\n", maior);
```

# Indentação

- A forma como os comandos são organizados na forma de um texto é completamente livre
- Para que se possa entender o código de um programa, existem algumas regras que devem ser seguidas
- Elas não mudam o significado do programa mas servem para embelezá-lo e deixá-lo mais inteligível

# Regras de indentação

- Declaração de função deve ser feita na primeira coluna
- Chaves devem estar na primeira coluna, em linhas sozinhas
- Comandos não devem aparecer na mesma linha das chaves

```
#include <stdio.h>

int main()
{
    // Comandos aqui
}
```

# Regras de indentação

- Declaração de variáveis também deve aparecer na primeira coluna
- Deixe uma linha em branco após as declarações

```
#include <stdio.h>

int main()
{
int i, j, k;
double x, y, z;

    // comandos aqui
}
```

# Regras de indentação – comandos

- Os comandos devem estar afastados algumas posições à direita das chaves
- Todos devem iniciar na mesma coluna

```
#include <stdio.h>

int main()
{
    printf(".... ", ...);
      ...
    scanf("....", ...);
}
```

# Regras de indentação – if

- Chaves do `if` devem ficar em linhas separadas
- Comandos dentro do `if` devem ficar mais à direita

```
int main()
{
    printf(".... ", ...);
    if ( a > b )
    {
        maior = b;
    }
    else
        maior = b;
    scanf("....", ...);
}
```

# Regras de indentação – if

- Mesmo vale para `if` aninhado

```
int main()
{
    printf(".... ", ...);
    if ( a > b )
    {
        if ( a > c )
        {
            maior = ac
        }
        else
        {
           maior = c;
        }
    }
    else ...
}
```

# If sem else

- A parte correspondente ao `else` pode não estar presente, ou seja, ela é opcional
- Corresponde a um comando “se acontecer tal condição execute tal comando”
- Se essa condição não acontecer, nada deve ser feito
- Exemplo: se o valor do ângulo for negativo, ajustá-lo para um valor positivo

# If sem else – exemplo

```
#include <stdio.h>
#include <math.h>
int main()
{
double angulo;
int sinal, voltas;
   printf("Digite o valor do angulo ==> ");
   scanf("%lf", &angulo);
   // quantas voltas da no circulo
   voltas = (int) (angulo / (2 * M_PI));
   // valor alem da ultima volta
   angulo = angulo - (voltas * 2 * M_PI);
   sinal = 1;
   if ( angulo < 0 )
   { // -sin(x) é igual sin(-x)
      sinal = -1;
      angulo = -angulo;
   }
   printf("O valor do seno eh %lf\n", sinal * sin(ang
}
```

# Coersão de tipos

- Às vezes é necessário transformas um valor de um tipo para outros
- No exemplo anterior, de `double` para `int`
- Isso é feito colocando o tipo para o qual se deseja transformar, entre parênteses, antes do valor que se deseja transformar
  - voltas = (int) (angulo / (2 * M_PI));
  - ang_rad = ( (float) ang_grau ) * M_PI / 180 )

# Exercício

- Faça um programa que compute as soluções da equação $Ax^2 + Bx + C = 0$. Os valores de A, B e C devem ser fornecidos pelo usuário. Se o valor de A for 0, então o programa deve emitir uma mensagem de erro dizendo que a equação não é do segundo grau. Se o vafor de $\Delta$ for negativo, o programa deve emitir uma outra mensagem de erro, dizendo que não existem raizes reais para a equação.

# Solução – 2o. grau

```
#include <stdio.h>
#include <math.h>
int main()
{
double delta, x1, x2;
double A, B, C;
   printf("Resolve a equacao Ax**2 + Bx + C = 0\n");
   printf("Entre com o valor de A ==> ");
   scanf("%lf", &A);
   printf("Entre com o valor de B ==> ");
   scanf("%lf", &B);
   printf("Entre com o valor de C ==> ");
   scanf("%lf", &C);
```

# Solução – 2o. grau

```
if ( A == 0 )
{
    printf("Essa não é uma equacao do segundo grau\n");
}
else
{
    delta = B * B - (4 * A * C);
    if ( delta < 0 )
    {
        printf("Essa equacao nao tem raiz real\n");
    }
    else
    {
        x1 = -B + sqrt(delta) / (2 * A) ;
        x2 = -B - sqrt(delta) / (2 * A) ;
        printf("Solucoes: %lf e %lf\n", x1, x2);
    }
}
```

# Para facilitar

- O comando `return` pode ser usado para terminar a execução da função `main`
- Isso faz com que o programa termine
- Cuidado, se estiver usando o “pause” para visualizar o resultado
- Ele deve ser usado em cada um dos comandos `return`
- O `return` deve ter um valor que é retornado para o S.O
  - 0 - ok
  - outro valor - erro

# Uso do return

```
if ( A == 0 )
{
    printf("Essa não é uma equacao do segundo grau\n");
    return 0;
}
delta = B * B - (4 * A * C);
if ( delta < 0 )
{
     printf("Essa equacao nao tem raiz real\n");
     return 0;
}
x1 = -B + sqrt(delta) / (2 * A) ;
x2 = -B - sqrt(delta) / (2 * A) ;
printf("Solucoes: %lf e %lf\n", x1, x2);
```

# Uso do return

```
if ( A == 0 )
{
    printf("Essa não é uma equacao do segundo grau\n");
    system("pause");
    return 0;
}
delta = B * B - (4 * A * C);
if ( delta < 0 )
{
    printf("Essa equacao nao tem raiz real\n");
    system("pause");
    return 0;
}
x1 = -B + sqrt(delta) / (2 * A) ;
x2 = -B - sqrt(delta) / (2 * A) ;
printf("Solucoes: %lf e %lf\n", x1, x2);
```

# Operadores lógicos

- São feitos para “combinar” expressões lógicas
- Operador `&&` (AND)
  - Resultado 1 somente se os dois operandos são diferentes de 0
  - Caso contrário, resultado é 0
  - `if ( a > b && b > c ) maior = a;`
- Operador `||` (OR)
  - Resultado 0 somente se os dois operandos são 0
  - Caso contrário, resultado é 1
  - ```
    if ( mes < 1 || mes > 12 )
        printf("Mes inválido\n");
    ```
- Operador `!` (NOT)
  - O resultado é 1 se operando é 0.
  - Caso contrário é 0
  - ```
    if (! ( mes < 1 || mes > 12 ) )
        printf("Mes é válido\n");
    ```

# Exercício

- Faça um programa que leia 2 notas de um aluno, verifique se as notas são válidas e exiba na tela Se o aluno foi aprovado, reprovado, ou se ficou de REC. . Uma nota válida deve ser obrigatoriamente um valor entre 0.0 e 10.0. Caso a nota não possua um valor válido, este fato deve ser informado ao usuário e o programa termina. Escreva duas versões desse programa. Uma usando o operador `&&` para verificar se uma nota é válida ou não e outra usando o operador `||`.

# Solução (1)

```
int main()
{
double nota1, nota2, media;
   printf("Digite o valor da nota 1 ==> ");
   scanf("%lf", &nota1);
   if ( nota1 < 0.0 || nota1 > 10.0 )
   {
       printf("Nota nao eh valida\n");
       return 0;
   }
   printf("Digite o valor da nota 2 ==> ");
   scanf("%lf", &nota2);
   if ( nota2 < 0.0 || nota2 > 10.0 )
   {
       printf("Nota nao eh valida\n");
       return 0;
   }
```

# Solução (1)

```
    media = (nota1 + nota2) / 2.0;
    if ( media >= 5.0 )
    {
       printf("Aluno aprovado\n");
    }
    else
    {
       if ( media >= 3 )
           printf("Aluno de REC\n");
       else
           printf("Aluno reprovado\n");
    }
    return 0;
}
```

# Solução (2)

```
int main()
{
double nota1, nota2, media;
   printf("Digite o valor da nota 1 ==> ");
   scanf("%lf", &nota1);
   if ( ! (nota1 >= 0.0 && nota1 <= 10.0 ) )
   {
       printf("Nota nao eh valida\n");
       return 0;
   }
   printf("Digite o valor da nota 2 ==> ");
   scanf("%lf", &nota2);
   if ( ! (nota2 >= 0.0 && nota2 <= 10.0 ) )
   {
       printf("Nota nao eh valida\n");
       return 0;
   }
```

# Mais exercício

- Uma empresa decide dar um aumento aos seus funcionários de acordo com uma tabela que considera o salário atual e o tempo de serviço de cada funcionário. Os funcionários com menor salário terão um aumento proporcionalmente maior do que os funcionários com um salário maior, e conforme o tempo de serviço na empresa, cada funcionário irá receber um bônus adicional de salário.
- Faça um programa que leia: (1) o valor do salário atual do funcionário; (2) o tempo de serviço deste funcionário na empresa (nro. de anos de trabalho na empresa). Use as tabelas abaixo para calcular o salário reajustado deste funcionário e imprima o valor do salário final reajustado, ou uma mensagem caso o funcionário não tenha direito a nenhum aumento.

# Mais exercício

| Sal atual | Reajuste | Tempo | Bonus |
|---|---|---|---|
| Até 500,00 | 25% | menos 1 ano | Nada |
| Até 1000,00 | 20% | 1 a 3 anos | 100,00 |
| Até 1500,00 | 15% | 4 a 6 anos | 200,00 |
| Até 2000,00 | 10% | 7 a 10 anos | 300,00 |
| Mais de 2000,00 | Nada | Mais de 10 anos | 500,00 |

# Encadeamento de comandos `if`

```
if ( salario < 200 )
{
   // comandos aqui
}
else
if ( salario < 400 )
{
   // comandos aqui
}
if ( salario < 600 )
{
   // comandos aqui
}
```