Aula 03:

Estruturas Condicionais

Prof. Edvard

edvard@unifei.edu.br

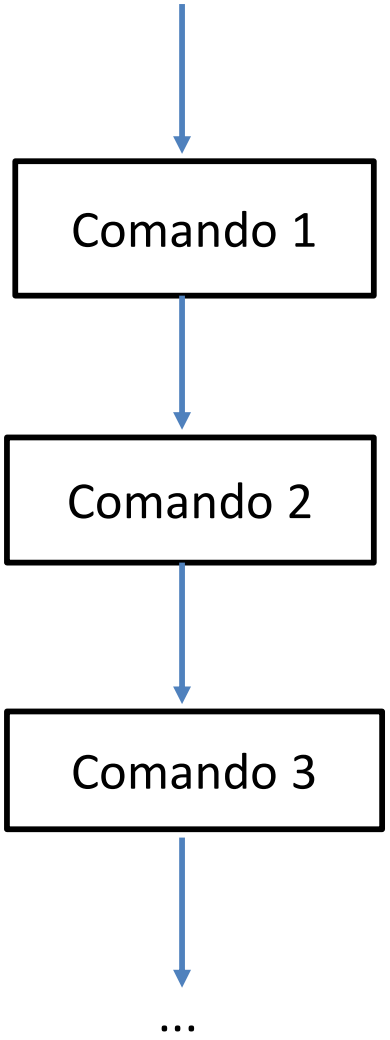
Universidade Federal de Itajubá

Estruturas Condicionais

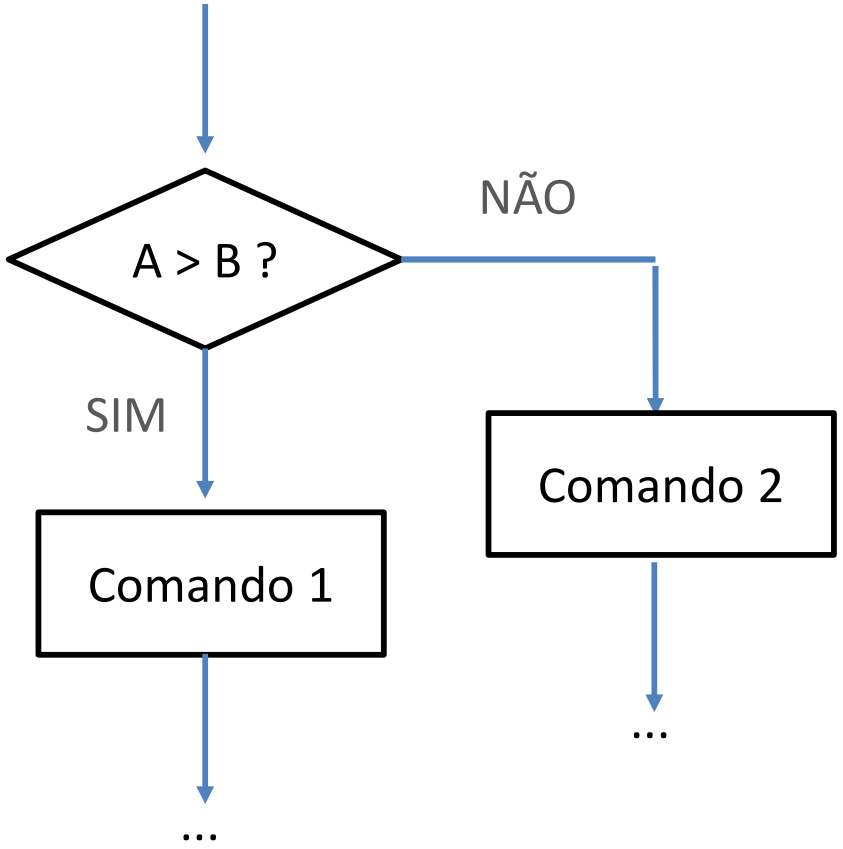
- Os programas que vimos até agora são todos **programas sequenciais**.
  - Neles, todos os comandos escritos são executados, um depois do outro, de acordo com a ordem estabelecida pelo programador.
- No entanto, existem casos onde um determinado comando só deverá ser executado se alguma **condição for satisfeita**.
  - **É preciso realizar o teste**. Se a condição for verdadeira, executamos aquele bloco de código. Caso contrário, tomamos uma atitude diferente ou simplesmente ignoramos aqueles comandos e seguimos com nosso programa.

Estruturas Condicionais

Estrutura Sequencial



Estrutura Condicional

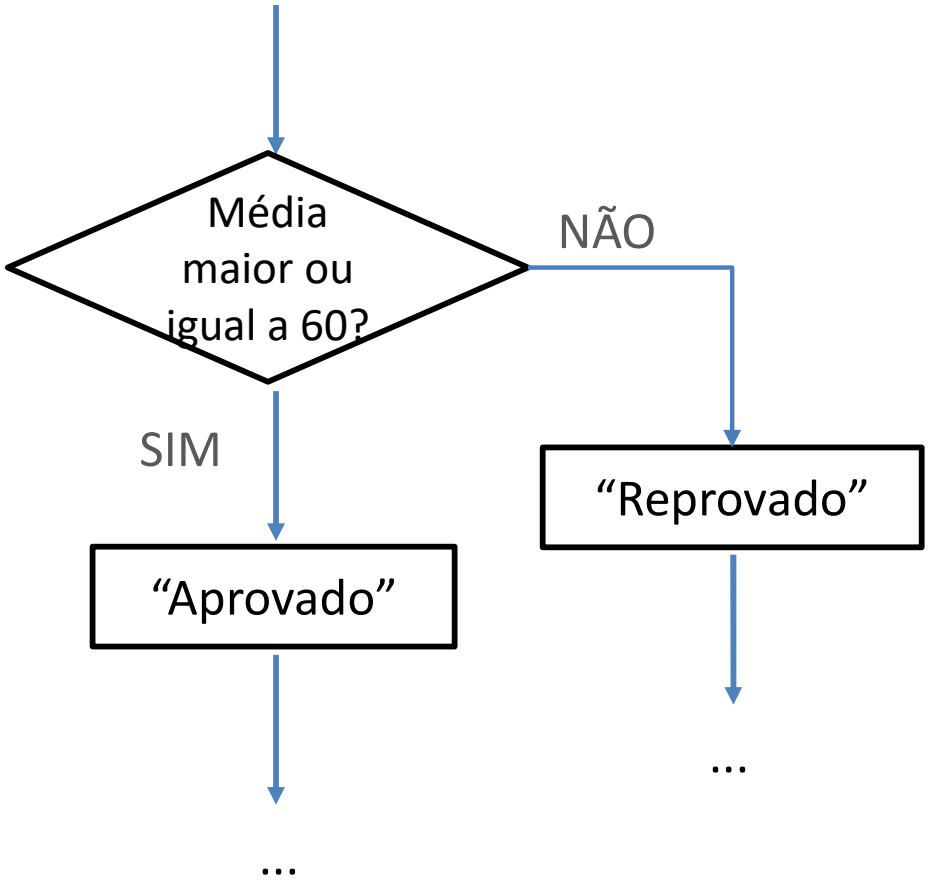


Estruturas Condicionais

- Considerando essa nova visão de que a execução dos comandos pode ter desvios dentro do código do programa e **não seguir sempre a mesma sequência**, vamos retomar um exemplo básico.
  - Dado aquele programa que calculava e imprimia na tela a média de duas notas de um aluno, poderíamos incrementá-lo com uma estrutura condicional.
  - Vocês têm ideia do que poderia ser?

Estruturas Condicionais

- Exatamente! Poderíamos facilitar a vida do usuário, imprimindo também uma informação sobre seu status, baseando-se na média obtida. Podemos dizer se o aluno foi “Aprovado” ou “Reprovado”.
- Vamos ver como fica:



Estrutura Condicional Básica

Se condição **então**
    Faça alguma coisa
Senão
    Faça outra coisa!
Fim se

Em C, sua forma geral é:

```
If (condição)
{
        // Faça alguma coisa
}
else
{
        // Faça outra coisa!
}
```

```c
#include <stdio.h>

int main()
{
    int p1, p2, media;

    printf("Entre com as nota da P1 e da P2: ");
    scanf("%d %d", &p1, &p2);

    media = (p1 + p2) / 2;

    printf("A media final sera %d\n", media);

    if(media >= 60)
    {
        printf("Status: Aprovado\n\n");
    }
    else
    {
        printf("Status: Reprovado\n\n");
    }

    return 0;

}
```

Até aqui:
Estrutura sequencial
Nada de novo

Nova sintaxe!
Estrutura condicional simples if/else

```
Entre com a nota da P1: 75
Entre com a nota da P2: 52
A media final sera 63
Status: Aprovado
```

Estruturas Condicionais

- Na linguagem C, o comando **if** pode ser utilizado de duas maneiras diferentes:
  - Quando é necessário **escolher entre dois caminhos** dentro do programa (if/else);
  - Quando uma parte do código só precisa ser executada se alguma **condição for satisfeita** (somente if).
- Portanto, o comando **else** não é utilizado todas as vezes em conjunto com o comando **if**, mas somente quando for necessário definir dois caminhos alternativos. Vejamos um exemplo onde sua utilização não é necessária:

Estruturas Condicionais

- Imagine um programa que imprima o valor absoluto de um número inteiro digitado pelo usuário:

```
int main()
{
    int num;

    printf("Entre com um valor inteiro (positivo ou negativo): ");
    scanf("%d", &num);

    // Caso o número seja negativo, multiplique-o por -1
    if(num < 0)
        num = num*(-1);

    printf("Seu valor aboluto = %d \n\n", num);

    return 0;
}
```

Somente precisamos processar o número caso ele seja negativo.

```
Entre com um valor inteiro (positivo ou negativo): -27
Seu valor aboluto = 27
```

```
Entre com um valor inteiro (positivo ou negativo): 42
Seu valor aboluto = 42
```

Estruturas Condicionais

- Repare que, no exemplo anterior, não utilizei as chaves para demarcar os limites do comando **if**. Falta de atenção? Não!
  - Isso somente é válido quando, dentro do if (ou do else), **um único comando for executado**.

```
if(a > b)
     a = 10;
else
     b = 10;

if(c < 0)
     c = c * 10
```

```
if(c == b)
     a = 10;
else
{
     c = b;
     a = 5;
}
```

```
if(a == b)
     c = a;
     b = c;
else
     c = 10
```

Estruturas Condicionais

- Como falamos anteriormente, alguns comandos em C não são terminados com ponto e vírgula.
  - O operador ponto e vírgula é utilizado para **separar as instruções do programa**. Colocá-lo logo após o comando if faz com que o compilador entenda que o comando if já terminou, e trata o comando seguinte como se estivesse fora do if.
- Qual seria o resultado impresso no código a seguir?

```
int num = 10;

if(num < 0);
    num = num*(-1);

printf("Seu valor aboluto = %d \n\n", num);
```

Exatamente! -10!
Portanto, nunca utilize ponto e vírgula após o comando if.

Estruturas Condicionais

- Portanto, nossa estrutura básica condicional:
  - É formada por um **bloco if**, de comandos que são executados somente se uma condição for satisfeita (**verdadeira**).
  - Caso seja necessário, é completada por um **bloco else**, que contém comandos que serão executadas se, e somente se, a condição testada pelo if não for satisfeita (**falsa**).
- Observações
  - Os dois blocos NUNCA serão executados em sequência. Apenas um deles será executado.
  - Não é possível incluir uma condição no bloco else.
    - Ex: if (a > b) { ... } else (b < a) { ...}
    - Errado!

Estruturas Condicionais

- E como definimos uma condição?
  - Primeiramente, vamos conhecer os **operadores relacionais**. São eles:

Operador	Significado	Exemplo
>	Maior que	x > 5
>=	Maior ou igual a	x >= 10
<	Menor que	x < 1000
<=	Menor ou igual a	X <= 25
==	Igual a	x == 0
!=	Diferente de	x != 0

Operadores relacionais sempre operam sobre **dois valores**, visando realizar uma **comparação** entre eles.

Estruturas Condicionais

- Como resultado, um operador relacional retorna:
  - O valor 1, se a expressão for VERDADEIRA.
  - O valor 0, se a expressão for FALSA.
- Observações:
  - Repare que a ordem é muito importante na escrita de operações de comparação. **Não existem operadores =<, =>, =!**.
  - O **operador <>**, que significa "diferente de" em algumas outra linguagens, como Pascal, **não é válido em C**.
  - O **operador ==** compara valores. **O sinal =** significa uma coisa completamente diferente: atribuição.

Estruturas Condicionais

- E como definimos uma condição?
  - Por condição, entende-se qualquer expressão relacional que resulte em uma expressão do tipo **verdadeiro** ou **falso**.
    - Caso a expressão utilizada como condição retorne **falso** (**zero**), o bloco **else** será executado.
    - Caso contrário (**diferente de zero**), a expressão é considerada **verdadeira** e o bloco if é executado.

```
if(num)
{
    // num é diferente de zero
}
```

Além dos testes habituais, utilizando operadores relacionais, podemos realizar testes como ao lado, **utilizando apenas o nome da variável**. Se for diferente de zero, entra no if. Caso contrário, poderia entrar no else.

Estruturas Condicionais

- **Ao trabalho:** Par ou Ímpar!
- Escreva um programa que peça para o usuário entrar com um número.
- A seguir, realize um teste para verificar se o número inserido pelo usuário é **par ou ímpar**.
  - Se for par, imprima na tela "Número Par".
  - Caso contrário, imprima "Número Ímpar".

Estruturas Condicionais

- Além disso, é possível combinar várias condições em uma só estrutura condicional, através de operadores lógicos.
  - **&&**, que Significa “e”, e é utilizado sempre que **todas** as condições listadas precisem ser satisfeitas para que o bloco de comandos seja acessado.
  - ||, que significa “ou”, e é utilizado sempre que **apenas uma** das condições listadas precise ser satisfeita para que o bloco de comandos seja acessado.
  - !, que é o operador de negação, e é utilizado sempre que precisamos testar o **inverso** de uma condição.

Estruturas Condicionais

- **Operador E (&&):** A expressão resultante só é verdadeira se ambas as expressões unidas por esse operador também forem.

```
if((temperatura > 36.5) && (temperatura < 37.5))
    printf("Febre Moderada.");

if((x > 0) && (y > 0))
    z = 10 / (x * y);
```

- Repare que cada teste possui duas condições, e somente no caso em que **uma E outra** são verdadeiras, é que o bloco if é executado.
- Cada condição deve ser delimitada pelo **uso de parênteses**, para evitar confusão e facilitar a leitura do código.

Estruturas Condicionais

- **Operador OU (&&):** A expressão é verdadeira se alguma das expressões unidas por esse operador também for verdadeira.

```
if((nota < 0) || (nota > 100))
    printf("Nota invalida. Por favor, insira outra nota.");

if((saldo_conta <= 0) || (cartao_credito >= 5000))
    printf("Cuidado com suas finanças!");
```

- Repare que cada teste possui duas condições, e sempre que **uma OU outra** forem verdadeiras, o bloco if será executado.
- Cada condição também deve ser delimitada pelo **uso de parênteses**, para evitar confusão e facilitar a leitura do código.
- É possível combinar operadores lógicos, como vemos abaixo:

```
if((saldo_conta <= 0) || (cartao_credito >= 1500 && limite_cartao <= 2000))
    printf("Cuidado com suas finanças!");
```

Estruturas Condicionais

- Quando apenas dois caminhos não forem suficientes, uma das opções é **encadear ifs** em uma técnica chamada de "**aninhamento de if**".
- Imagine, por exemplo, uma situação onde é necessário tomar uma medida diferente para:
  - Variável **negativa, v**ariável **nula** e Variável **positiva.**

```
if(saldo_gols > 0)
{
    printf("Saldo positivo: %d gols", saldo_gols);
}
else
{
    if(saldo_gols == 0)
        printf("Saldo de gols igual a zero.");
    else
        printf("Saldo negativo: %d gols", saldo_gols);
}
```

Estruturas Condicionais

- Podemos, naturalmente, inserir comandos if/else dentro de qualquer bloco de comandos, inclusive dentro dos próprios if e else.
  - A sintaxe fica, normalmente, as seguinte maneira:

```
if( expressao1 ) {
  comando1;
}
else {
  if( expressao2 ) {
    comando2;
  } else {
    comando3;
  }
}
```

A segunda condição somente será testada no caso da primeira condição resultar em resultado falso.

Ao final, o programa pode escolher entre três caminhos diferentes: comando 1, comando 2 ou comando 3.

Estruturas Condicionais

- **Ao trabalho:** Quem é maior
- Escreva um programa que peça para o usuário entrar com três números inteiros (a, b e c).
- A seguir, realize testes, utilizando ifs aninhados, para verificar qual dos números é o maior deles.
- Imprima a resposta na tela.

Estruturas Condicionais

- **Ao trabalho:** Fórmula de Bhaskara!
- Escreva um programa que calcule as raízes de uma equação dos segundo grau.
  - Segundo o matemático Bhaskara Akaria, do século XII, a solução para uma equação do tipo **$ax^2 + bx + c = 0$** seria dada por:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},$$

  - O que resulta em três diferentes tipos de respostas:
    - Duas raízes distintas (delta > 0)
    - Apenas uma raiz (delta == 0)
    - Solução impossível (sem números complexos! Delta < 0)

```c
#include <math.h>

int main()
{
    int a, b, c, delta;
    float x1, x2;

    printf("Entre com os coeficientes da equacao do
    scanf("%d %d %d", &a, &b, &c);

    delta = b*b - 4*a*c;

    if(delta < 0)
    {
        printf("Nao ha resposta.");
    }
    else
    {
        if(delta == 0)
        {
            x1 = (float)(-b/(2*a));
            printf("A raiz eh unica e igual a %.2f. \n\n", x1);
        }
        else
        {
            x1 = (float)((-b - sqrt(delta))/(2*a));
            x2 = (float)((-b + sqrt(delta))/(2*a));
            printf("As raizes sao %.2f e %.2f. \n\n", x1, x2);
        }
    }

    return 0;
}
```

Operações de *casting*, ou conversão de tipos, são necessárias quando utilizamos operandos de um tipo e precisamos obter um resultado de um tipo diferente. Como no exemplo ao lado.

Por isso, utilizamos (float), que transforma o resultado inteiro em número com ponto flutuante.

Estruturas Condicionais

- Além dos comandos if e else, a linguagem C possui um comando de seleção múltipla chamado **switch**.
  - É indicado para casos onde é necessário testar uma variável em relação a diversos valores pré-estabelecidos.
  - A comparação é feita somente para **números** e **caracteres**.
  - É mais limitado que o if/else, uma vez que a **única** relação que o switch verifica é a de igualdade.

Estruturas Condicionais

- A sua sintaxe básica é:

```
switch( variavel )
{
   case 1:
      comando1;
      break;
   case 2:
      comando2;
      break;
   default:
      comando3;
      break;
}
```

Testa valor da variável.

Se for igual a 1, executa “case 1”

Se for igual a 2, executa “case 2”

Se não se encaixar em nenhum dos cases anteriores, executa o bloco “default”

```c
#include <stdio.h>

int main()
{
    int tipo_combustivel;
    float quantidade;

    printf("Entre com a quantidade de combustivel em litros: ");
    scanf("%f", &quantidade);

    printf("Entre com o tipo (0 - alcool, 1 - gasolina, 2 - diesel): ");
    scanf("%d", &tipo_combustivel);

    switch(tipo_combustivel)
    {
        case 0:
            printf("Total gasto: R$ %.2f", quantidade*2.20);
            break;
        case 1:
            printf("Total gasto: R$ %.2f", quantidade*2.95);
            break;
        case 2:
            printf("Total gasto: R$ %.2f", quantidade*2.38);
            break;
        default:
            printf("Combustivel nao identificado");
            break;
    }

    return 0;
}
```

Estruturas Condicionais

- Para que serve o **break**?
  - Quando o valor associado a um comando **case** é igual ao valor da variável do **switch**, a respectiva sequência de comandos é executada até encontrar um comando **break**, que interrompe o switch imediatamente.
  - Caso o comando **break** não exista, a sequência de comandos do **case seguinte** também será executada, e assim por diante.
  - Pode gerar muita confusão e erro! Não esqueça dos breaks!

Desafio

URI Online Judge | 1397

Jogo do Maior

Por Lucas Hermann Negri, UDESC Brasil

Timelimit: 1

Og gosta muito de brincar com seus filhos. Seu jogo preferido é o *jogo do maior*, de autoria própria. Este passatempo (no tempo das cavernas se tinha muito tempo disponível para jogos) é jogado em dupla, Og e um dos seus filhos. O jogo procede da seguinte forma: os dois participantes escolhem um número de rodadas e, a cada rodada, cada participante diz um número de 0 até 10 em voz alta, sendo que o participante que falar o número mais alto ganha um ponto (em caso de empate, ninguém ganha o ponto). No final das rodadas, os pontos são contabilizados e o participante com o maior número de pontos ganha.

Og e seus filhos gostam muito do jogo, mas se perdem na contagem dos pontos. Você conseguirá ajudar Og a verificar a pontuação de uma lista de jogos?

Entrada

A entrada é composta por vários casos de teste (partidas). Cada caso é iniciado com um inteiro **N** (de 0 até 10) representando o número de rodadas da partida, sendo que o valor 0 representa o final da entrada e não deve ser processado. Cada uma das próximas **N** linhas contém dois inteiros, **A** e **B**, onde **A** é o número escolhido pelo primeiro jogador e **B** é o número escolhido pelo segundo jogador ($0 \leq$ **A**, **B** ≤ 10).

Saída

A saída deve ser composta por uma linha por caso de teste, contendo o número de pontos de cada jogador, separados por um espaço.

Exemplo de Entrada	Exemplo de Saída
3 5 3 8 2 5 6 2 5 5 0 0 0	2 1 0 0

Maratona de Programacao UDESC, Seletiva Iracema 2013.