Estruturas de Seleção

Rafael Beserra Gomes

UFRN

Material compilado em 13 de dezembro de 2017.

✎ Exercício em sala

Escreva um programa em C que leia do usuário 3 números reais (a, b e c) e escreva na tela as raízes da equação:

$$ax^2 + bx + c = 0$$

Teste o seu programa com as seguintes entradas:

- a = 2, b = 6 e c = 3
- a = 1, b = 4 e c = 4
- a = 1, b = 1 e c = 1

Estruturas de Seleção/Condicionais

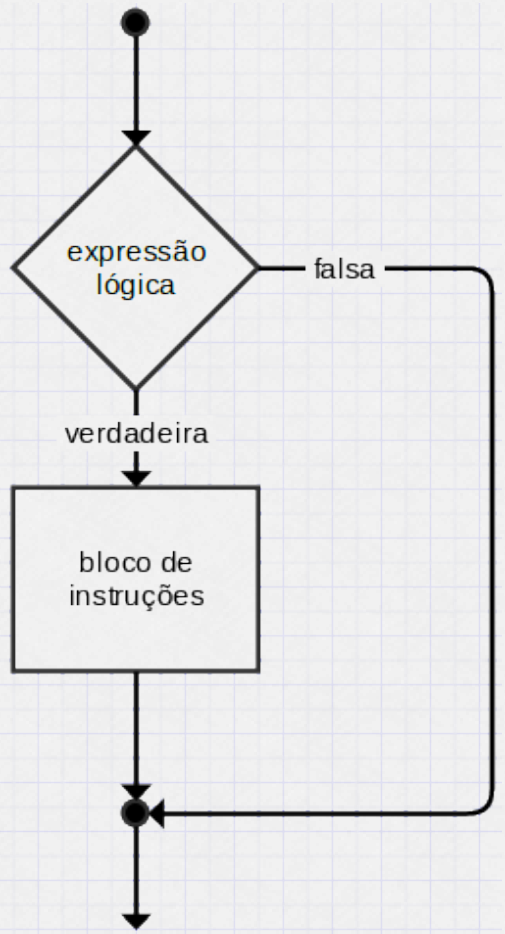
- Até o momento o **fluxo de execução** do programa foi único
- Seja o problema de, dados os coeficientes de uma equação do 2^o grau, escrever na tela suas raízes
- No caso de delta < 0, o programa exibe as raízes como nan (*not a number*)
- No caso de delta = 0, o programa exibe a mesma raiz duas vezes
- É interessante que o programa execute uma ou outra coisa de acordo com **condições**

No programa de extração de raízes da equação de segundo grau:

- **Condição 1**: delta == 0, vamos escrever uma única raiz
- **Condição 2**: delta > 0, vamos escrever as duas raízes
- **Condição 3**: delta < 0, vamos escrever que não há raízes reais

Estrutura de Seleção **if**

```
if (<expressão lógica>) {
␣<instrução 1>
␣<instrução 2>
␣<...>
␣<instrução n>
}
```



- Os espaços ␣representam aqui a indentação:
- As chaves {} definem o bloco de instruções que será executado caso a expressão lógica seja **verdadeira**

Por exemplo:

```c
#include <stdio.h>

int main() {

    int a, b;

    printf("Digite dois numeros: ");
    scanf("%d %d", &a, &b);

    if(a == b) {
        printf("Os dois numeros sao iguais.\n");
    }

    if(a != b) {
        printf("Os dois numeros sao diferentes.\n");
    }

    return 0;
}
```

✎ Exercício em sala

Escreva um programa que leia do usuário um número **real** a e um **inteiro** b. Em seguida, se o número b for diferente de zero, escreva na tela o resultado da divisão $\frac{a}{b}$.

✎ Exercício em sala

Escreva um programa em C que leia do usuário 3 números reais (a, b e c) e escreva na tela as raízes da equação de acordo com a quantidade de raízes reais.

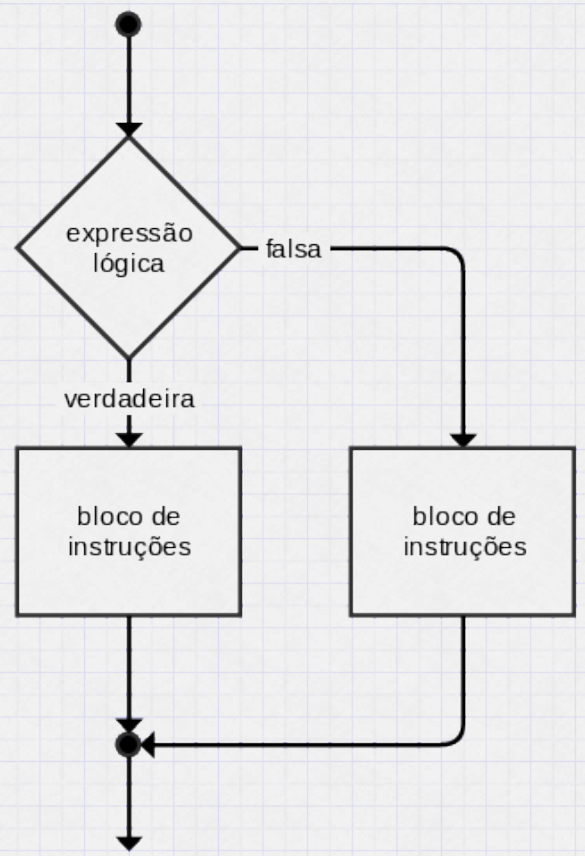
$$ax^2 + bx + c = 0$$

Teste o seu programa com as seguintes entradas:

- a = 2, b = 6 e c = 3
- a = 1, b = 4 e c = 4
- a = 1, b = 1 e c = 1

Estrutura de Seleção **if/else**

```
if (<expressão lógica>) {
␣<instrução 1>
␣<instrução 2>
␣<...>
␣<instrução n>
} else {
␣<instrução 1>
␣<instrução 2>
␣<...>
␣<instrução n>
}
```



- O par de chaves do **if** definem o bloco de instruções que será executado caso a expressão lógica seja **verdadeira**
- O par de chaves do **else** definem o bloco de instruções que será executado caso a expressão lógica seja **falsa**

Por exemplo:

```c
#include <stdio.h>

int main() {

    int a, b;

    printf("Digite dois numeros: ");
    scanf("%d %d", &a, &b);

    if(a == b) {
        printf("Os dois numeros sao iguais.\n");
    } else {
        printf("Os dois numeros sao diferentes.\n");
    }

    return 0;
}
```

✎ Exercício em sala

Escreva um programa em C que leia dois números inteiros **a** e **b** (assuma que o usuário digita $a > 0$ e $b > 0$). Em seguida deve escrever na tela "Um dos números divide o outro" caso um dos números divida o outro ou "Nenhum dos números divide o outro" caso nenhum dos números divida o outro.

Condicionais aninhados

O bloco de instruções pode incluir outros condicionais (**condicionais aninhados**)!

```
1  scanf("%f %f %f", &n1, &n2, &n3);
2  mediaParcial = (4*n1 + 5*n2 + 6*n3)/15
3  if(mediaParcial < 3)
4      printf("Reprovado\n");
5  if(mediaParcial >= 7)
6      printf("Aprovado\n");
7  if(mediaParcial >= 3 and mediaParcial < 7) {
8      scanf("%f", &n4);
9      mediaFinal = (mediaParcial + n4)/2;
10     if(mediaFinal >= 5)
11         printf("Aprovado\n");
12     if(mediaFinal < 5)
13         printf("Reprovado\n");
14 }
```

Utilizando if/else no problema da aprovação do aluno:

```
1  scanf("%f %f %f", &n1, &n2, &n3);
2  mediaParcial = (4*n1 + 5*n2 + 6*n3)/15
3  if(mediaParcial < 3)
4      printf("Reprovado\n");
5  else {
6      if(mediaParcial >= 7)
7          printf("Aprovado\n");
8      else {
9          scanf("%f", &n4);
10         mediaFinal = (mediaParcial + n4)/2;
11         if(mediaFinal >= 5)
12             printf("Aprovado\n");
13         else
14             printf("Reprovado\n");
15     }
16 }
```

✎ Exercício em sala

Escreva um programa que leia o número de gols de um time de futebol A e o número de gols de um time B no tempo normal em uma partida de futebol. Se houver um vencedor, o programa deve informar qual foi o time. Mas, se houver empate, o programa deverá solicitar o número de gols de cada uma das equipes nos pênaltis e, em seguida, qual foi o time vencedor (não há mais possibilidade de empates).

Exemplo 1:

```
Digite o numero de gols do time A: 3
Digite o numero de gols do time B: 2
Vencedor: time A
```

Exemplo 2:

```
Digite o numero de gols do time A: 3
Digite o numero de gols do time B: 3
Digite o numero de gols do time A nos penaltis: 3
Digite o numero de gols do time B nos penaltis: 5
Vencedor: time B
```

Depuração de código

Depuração de código

Consiste na análise do código-fonte para detecção de problemas.

- Mesmo com o programa compilando, o programa pode produzir resultados inesperados
- Assumir que não há erro em trechos de códigos pode dificultar o encontro do erro
- Vejamos algumas formas de depuração

Formas de depuração

Leitura do código

- muitos problemas são ocasionados por erros de lógica

```c
#include <stdio.h>

int main() {

    int a, b;

    printf("Digite dois numeros: ");
    scanf("%d %d", &a, &b);

    //trocando os valores das duas variaveis
    a = b;
    b = a;

    return 0;
}
```

- muitas vezes esses erros não são fáceis de encontrar

Teste de mesa

- consiste em analisar passo a passo a execução do seu programa para casos de testes, possivelmente usando papel para fazer anotações

```
1 int i, j, k;
2 scanf("%d %d %d", &i, &j, &k);
3 ++i = k;
4 j += 1;
5 k += ++i + j++;
6 //qual o valor de i, j e k?
```

- o teste de mesa deve ser o mais comum para a depuração de códigos nesta disciplina

Mensagens de debug

- acompanhar o comportamento do programa utilizando mensagens escritas na tela

```c
#include <stdio.h>

int main() {

    float a, b, c, delta;

    printf("Digite os coeficientes de ax2 + bx + c = 0\n");
    scanf("%f %f %f", &a, &b, &c);

    delta = b*b - 4*a*b;
    printf("delta = %d\n", delta); //msg para debug

    printf("Uma raiz eh: %f\n", (-b+sqrt(delta))/(2*a));

    return 0;
}
```

- fácil de testar em programas menores, lento/inviável para programas maiores

Assert

- certificar-se que certas condições estão satisfeitas

```c
#include <assert.h>
#include <stdio.h>

int main() {

    int n;

    n = 1;

    //em tese n deve chegar aqui com um valor diferente de 0
    //nao eh um teste para validar n
    //e sim um teste para garantir que n eh diferente de zero
    assert(n != 0);

    printf("10 dividido por n: %f\n", 10.0/n);

    return 0;
}
```

Ferramentas de Debug

- programas que auxiliam a depuração do código (exemplo: gdb)
- **Passo 1:** compile incluindo -g ou -ggdb
- **Passo 2:** execute o gdb: *gdb arquivoexecutavel*
- **Algumas instruções do GDB:**
  - r: começa a executar o programa
  - n: execute a próxima linha de código
  - br *x*: cria um breakpoint na linha *x*
  - c: executa até o próximo breakpoint
  - l: lista o código
  - q: sai do GDB
- algumas IDEs incluem ferramentas mais intuitivas para depuração

Estruturas de Seleção

Nota adicional

- programas que auxiliam a depuração do código (exemplo: gdb)
- **Passo 1:** compile incluindo -g ou -ggdb
- **Passo 2:** execute o gdb: *gdb arquivoexecutavel*
- **Algumas instruções do GDB:**
  - r: começa a executar o programa
  - n: execute a próxima linha de código
  - br *x*: cria um breakpoint na linha *x*
  - c: executa até o próximo breakpoint
  - l: lista o código
  - q: sai do GDB
- algumas IDEs incluem ferramentas mais intuitivas para depuração

Além do if/else há uma estrutura de seleção em C denominada switch e um operador ternário de seleção. O primeiro evita o uso de vários condicionais em certas situações e o segundo abrevia um condicional if/else. Se for possível, serão vistos em momento oportuno.