Algoritmos e Programação

Aula 14

Estrutura Condicional em C

IF-ELSE

SWITCH-CASE

Conteúdo

- Expressões Lógicas
- IF-ELSE
- SWITCH-CASE

Expressões Lógicas

- São expressões que retornam apenas dois valores possíveis:
  - **Verdadeiro**
  - **Falso**

- Em C, temos a seguinte convenção para representar *verdadeiro* e *falso:*
  - $0 = falso$
  - Qualquer $x \neq 0 = verdadeiro$

- Normalmente escolhemos o valor 1 para verdadeiro.

Expressões Lógicas

- São expressões que retornam apenas dois valores possíveis:
  - **Verdadeiro**
  - **Falso**

- Em C, temos a seguinte convenção para representar *verdadeiro* e *falso:*
  - $0 = \boldsymbol{falso}$
  - Qualquer $x \neq 0 = \boldsymbol{verdadeiro}$

- Normalmente escolhemos o valor 1 para verdadeiro.

Operadores relacionais

- Operadores relacionais realizam **comparações**, e retornam valores lógicos.

Operações:

==	**Igual**
!=	**Diferente**
<	**Menor que**
<=	**Menor ou igual a**
>	**Maior que**
>=	**Menor ou igual a**

Prioridade:

Operadores aritméticos	Prioridade
<, <=, >, >=	Maior
==, !=	Menor

Operadores relacionais - exemplo

Exemplo:

```
#include <stdio.h>
int main() {
    int x = 1, y = 0, a, b, c, d, e, f;
    a = x == 4;
    b = 12 <= 13;
    c = 12 > -20;
    d = y != x;
    e = (13 < 22 == 0);
    f = 23 - 23 != -1 < 0;
    printf("a=%d b=%d c=%d\n", a, b, c);
    printf("d=%d e=%d f=%d\n", d, e, f);
}
```

Saída: **a=0 b=1 c=1**

d=1 e=0 f=1

Operadores Lógicos

- Podemos combinar valores verdade utilizando operações lógicas.

Operador C	Conectivo Lógico
!	NÃO
&&	E
||	OU

Tabela verdade:

A	B	!A	!B	A && B	A || B
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

Prioridade	Operadores
Maior	!
	operadores aritméticos
	operadores relacionais
	&&
Menor	||

Operadores Lógicos – exemplo

Exemplo:

```
#include <stdio.h>
int main() {
    int x, y, z;
    x = !(13 < 14) || (35 == (49 - 14));
    y = x && (43 - 20 < 23);
    z = !(!x && (y || ((2 - 2) && !0)));
    printf("x=%d\n", x);
    printf("y=%d\n", y);
    printf("z=%d\n", z);
}
```

Saída:

```
x=1
y=0
z=1
```

Conteúdo

- Expressões Lógicas
- Estruturas de seleção
  - IF-ELSE
  - SWITCH-CASE

Seleção simples – SE-ENTÃO

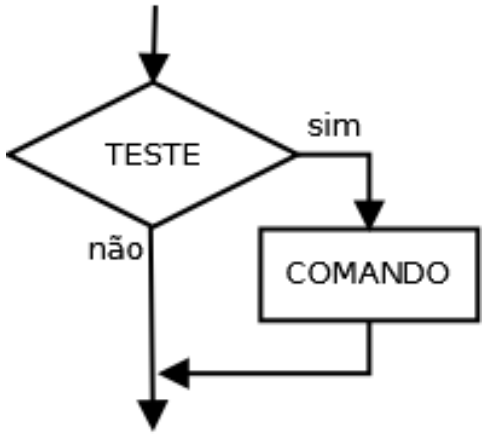
O condicional mais simples em C é a seleção simples, do tipo SE-ENTÃO;

Sintaxe:

```
if (TESTE)
    COMANDO;
```

O **TESTE** é uma **expressão qualquer**, **interpretada** como **valor lógico.**

Corresponde ao seguinte fluxograma:



Seleção simples – SE-ENTÃO: exemplo

```
#include <stdio.h>
#include <math.h>
int main() {
    float f;
    printf("Digite um numero: ");
    scanf("%f", &f);
    if(f < 0)
        printf("Atencao: O numero eh negativo\n");

    printf("raiz do numero %f = %f\n", f, sqrt(f));
}
```

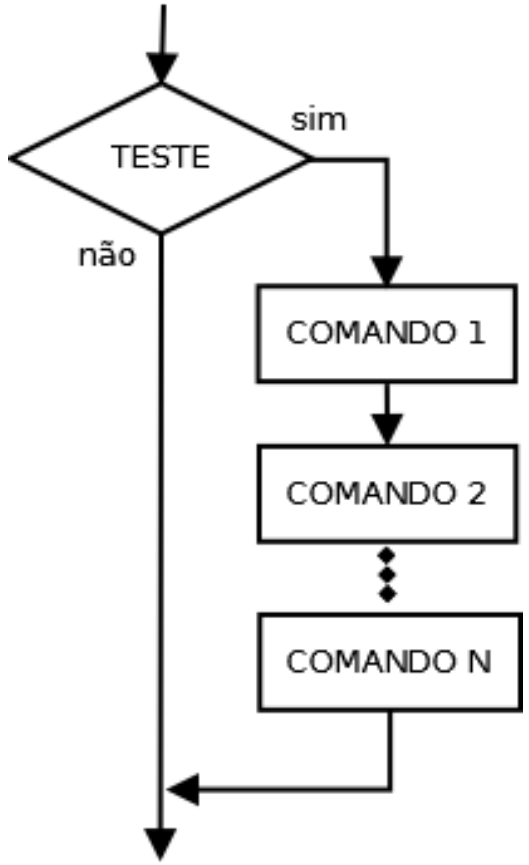
Entrada	Saída
-25	Atencao: O numero eh negativo raiz do numero -25.000000 = -1.#IND00
100	raiz do numero 100.000000 = 10.000000

Seleção Simples: Blocos de comandos

Note que a sintaxe do SE-ENTÃO determina a execução condicional de **um comando** somente.

E se quisermos executar uma série de comandos?

```
if (TESTE) ???
     COMANDO-1;
     COMANDO-2;
     ...
     COMANDO-N;
```

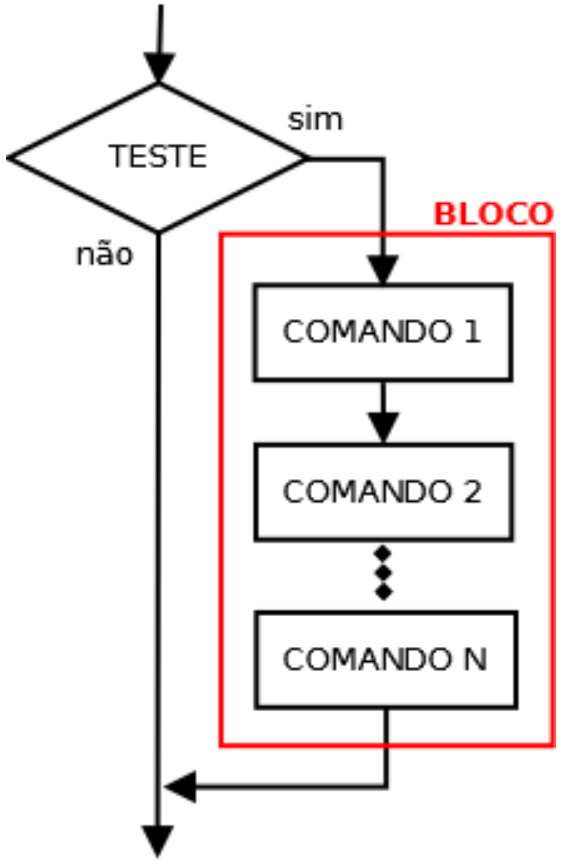


Seleção Simples: Blocos de comandos

Note que a sintaxe do SE-ENTÃO determina a execução condicional de **um comando** somente.

E se quisermos executar uma série de comandos?

```
if (TESTE) {
     COMANDO-1;
     COMANDO-2;
     ...
     COMANDO-N;
}
```



Usamos '{' e '}' para delimitar um **bloco de comando**.

Blocos de comando: exemplo

```
#include <stdio.h>
#include <math.h>
int main() {
    float f;
    printf("Digite um numero: ");
    scanf("%f", &f);
    if(f < 0) {
        printf("Atencao: Numero digitado negativo.\n");
        printf("Convertendo %f para positivo.\n", f);
     f = f * -1;
    }
    printf("raiz do numero %f = %f\n", f, sqrt(f));
}
```

Entrada	Saída
-25	Atencao: Numero digitado negativo. Convertendo -25.000000 para positivo. raiz do numero 25.000000 = 5.000000
100	raiz do numero 100.000000 = 10.000000

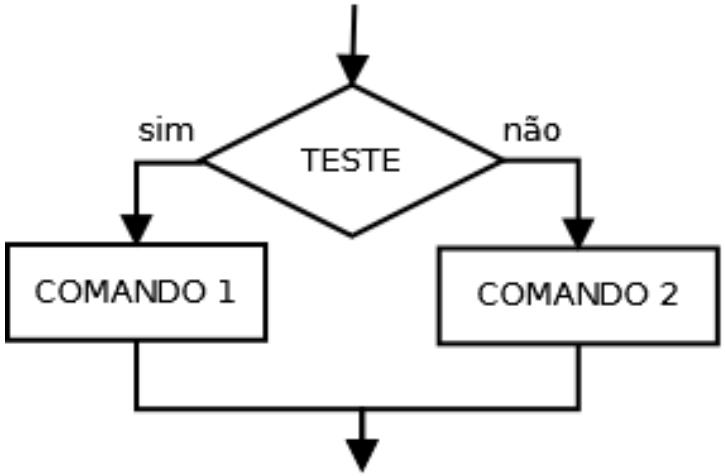
Seleção Composta – SE-ENTÃO-SENÃO

Construções do tipo SE-ENTÃO-SENÃO usam a cláusula else após um if para indicar o que fazer caso o teste resulte falso

Sintaxe:

```
if (TESTE)
     COMANDO-1;
else
     COMANDO-2;
```

Corresponde ao seguinte bloco de comando:



E como fazemos para executar uma série de comandos?

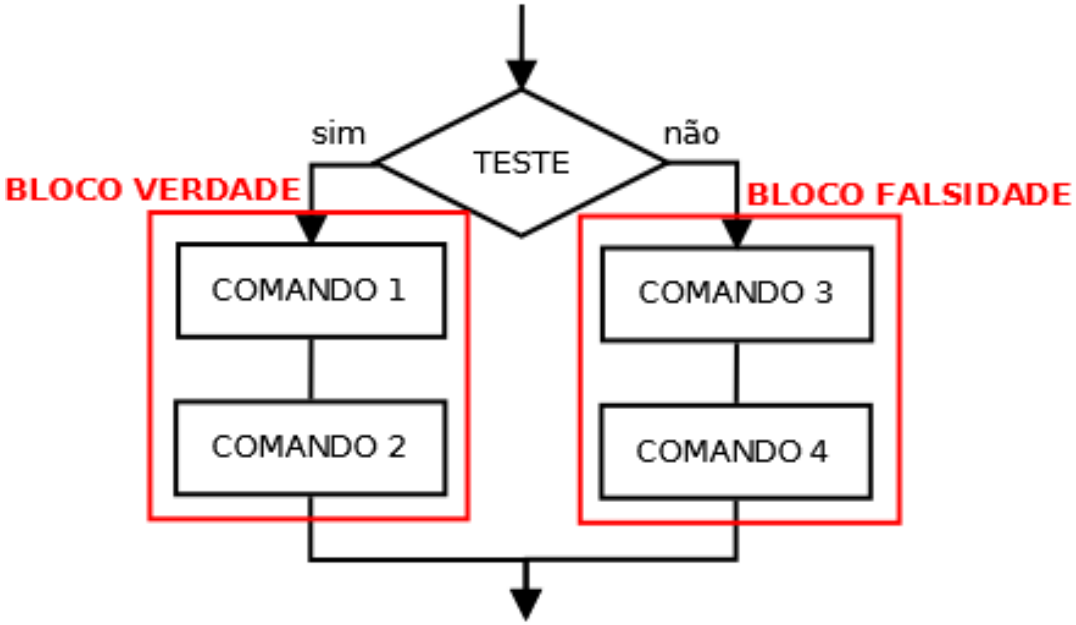
Seleção Composta – SE-ENTÃO-SENÃO

Construções do tipo SE-ENTÃO-SENÃO usam a cláusula else após um if para indicar o que fazer caso o teste resulte falso

Sintaxe:

```
if (TESTE){
    COMANDO-1;
    COMANDO-2;
} else {
    COMANDO-3;
    COMANDO-4;
}
```

Corresponde ao seguinte bloco de comando:



SE-ENTÃO-SENÃO - Exemplo

```c
#include <stdio.h>
#include <math.h>
int main() {
    float f;
    printf("Digite um numero: ");
    scanf("%f", &f);
    if(f >= 0) {
        printf("numero positivo!\n");
        printf("raiz do numero %f = %f\n", sqrt(f));
    } else {
        printf("numero negativo!\n", f);
        printf("%f nao possui raiz real\n", f);
    }
}
```

Entrada	Saída
-5	numero negativo! -5.000000 nao possui raiz real
100	numero positivo! raiz do numero 100.000000 = 10.000000

Seleção encadeada

- Assim como as construções SE-ENTÃO-SENÃO em pseudocódigo, podemos **aninhar estruturas** de **seleção** em C.
- Da mesma maneira que em pseudocódigo, possuímos seleções encadeadas homogêneas e heterogêneas:
  - **Heterogêneas**: não é possível prever o padrão de comportamento.
  - **Homogêneas**: é possível prever o padrão de comportamento.
    - **se-então-se**: quando depois de cada **então** ocorre outro **se**
    - **se-senão-se**: quando depois de cada **senão** ocorre outro **se**

Seleção encadeada - Exemplo

```c
#include <stdio.h>
int main() {
    float a,b,c;
    printf("Digite os valores de a, b e c: ");
    scanf("%f %f %f", &a, &b, &c);
    if(a > b) {
        if(a > c)
            printf("O maior valor eh %f\n", a);
        else
            printf("O maior valor eh %f\n", c);
    } else {
        if(b > c)
            printf("O maior valor eh %f\n", b);
        else
            printf("O maior valor eh %f\n", c);
    }
}
```

Entrada	Saída
1 3 2	O maior valor eh 3.000000

Seleção Encadeada Heterogênea

- **Não possui** um **padrão lógico** de construção em uma estrutura de **seleção encadeada**.

Exemplo:

- Dados três valores A,B,C, verificar se eles podem ser comprimentos dos lados de um triângulo equilátero, isósceles ou escaleno. Informar se não compuserem nenhum triângulo.
  - Dados de entrada: três lados de um suposto triângulo (A,B,C).
  - Dados de Saída – mensagens: não compõe triângulo, triângulo equilátero, triângulo isósceles, triângulo escaleno.
  - Observações:
    - Triângulo é uma figura geométrica fechada de três lados, onde cada lado é menor do que a soma dos outros dois.
    - Triângulo equilátero possui os três lados iguais.
    - Triângulo isósceles possui os 2 lados iguais.
    - Triângulo escaleno possui os três lados diferentes.

Seleção Encadeada Heterogênea: exemplo

- Tabela de decisão:

É triângulo?	É equilátero?	É isósceles?	É escaleno?	Ações
V	V	-	-	"Equilátero"
V	F	V	-	"Isósceles"
V	F	F	V	"Escaleno"
F	-	-	-	"Não é triângulo"

- Expressões lógicas:
  - É triângulo: (A < B + C) **&&** (B < A + C) **&&** (C < A + B)
  - É equilátero: (A == B) **&&** (B == C)
  - É isósceles: (A == B) **||** (A == C) **||** (B == C)
  - É escaleno: (A != B) **&&** (A != C) **&&** (B != C)

Seleção Encadeada Heterogênea

```c
#include <stdio.h>
int main() {
    float a, b, c;
    printf("Digite os 3 lados do triângulo: ");
    scanf("%f %f %f", &a, &b, &c);
    if((a < b + c) && (b < a + c) && (c < a + b)) {
        /* Estes valores formam um triângulo!*/
        /* testar se eh equilatero*/
        if((a == b) && (b == c))
            printf("Triangulo retangulo!\n");
        else {
            /* testar se eh isoscles*/
            if((a == b) || (a == c) || (b == c))
                printf("Triangulo isosceles!\n");
            else
                printf("Triangulo escaleno!\n");
        }
    } else {
        printf("Estes valores não formam um triângulo!\n");
    }
}
```

Seleção Encadeada Homogênea

- **if – then – if**
  - No exercício anterior vimos que um triângulo é uma figura fechada onde cada lado deve ser menor que a soma dos outros dois, ou seja, as condições seguintes devem ser satisfeitas: (A < B + C), (B < A + C) e (C < A + B).
  - Após cada **then** existe outro **if**, não existem **elses.**

```
if(A < B + C) {
    if(B < A + C) {
        if(C < A + B) {
            /* A,B e C formam um
            triângulo */
        }
    }
}
```

Tabela de decisão:

A<B+C	B<A+C	C<A+B	Ação
V	V	V	É triângulo

- É equivalente a:

```
if((a < b + c) && (b < a + c) && (c < a + b))
{
    /* Estes valores formam um triângulo!*/
}
```

Seleção Encadeada Homogênea

- **if – else – if:** Supomos que uma variável X possa assumir apenas quatro valores, V1, V2, V3, V4, e que exista um comando (ou bloco) diferente que será executado para cada valor de X

```
if(X == V1)
    <C1>;
if(X == V2)
    <C2>;
if(X == V3)
    <C3>;
if(X == V4)
    <C4>;
```

X=V1	X=V2	X=V3	X=V4	Ação
V	F	F	F	C1
F	V	F	F	C2
F	F	V	F	C3
F	F	F	V	C4

Seleção Encadeada Homogênea

- **if – else – if:** Supomos que uma variável X possa assumir apenas quatro valores, V1, V2, V3, V4, e que exista um comando (ou bloco) diferente que será executado para cada valor de X

```
if(X == V1)
     <C1>;
if(X == V2)
     <C2>;
if(X == V3)
     <C3>;
if(X == V4)
     <C4>;
```

```
if(X == V1)
     <C1>;
else {
     if(X == V2)
          <C2>;
     else {
          if(X == V3)
               <C3>;
          else {
               if(X == V4)
                    <C4>;
          }
     }
}
```

X=V1	X=V2	X=V3	X=V4	Ação
V	F	F	F	C1
F	V	F	F	C2
F	F	V	F	C3
F	F	F	V	C4

X=V1	X=V2	X=V3	X=V4	Ação
V	-	-	-	C1
F	V	-	-	C2
F	F	V	-	C3
F	F	F	V	C4

Seleção Encadeada Homogênea

- **if – else – if:** Supomos que uma variável X possa assumir apenas quatro valores, V1, V2, V3, V4, e que exista um comando (ou bloco) diferente que será executado para cada valor de X

```
if(X == V1)
    <C1>;
else {
    if(X == V2)
        <C2>;
    else {
        if(X == V3)
            <C3>;
        else {
            if(X == V4)
                <C4>;
            else
                <C5>;
        }
    }
}
```

```
    if(X == V1)
        <C1>;
    else if(X == V2)
        <C2>;
    else if(X == V3)
        <C3>;
    else if(X == V4)
        <C4>;
    else
        <C5>;
```

X=V1	X=V2	X=V3	X=V4	Ação
V	-	-	-	C1
F	V	-	-	C2
F	F	V	-	C3
F	F	F	V	C4
F	F	F	F	C5

Seleção Encadeada Homogênea: Exemplo

1. Construa um algoritmo que, tendo como dados de entrada o preço de um produto e seu código de origem, mostre o preço junto de sua procedência. Caso o código não seja nenhum dos especificados, o produto deve ser considerado importado. Siga a tabela de códigos a seguir:

Código de Origem	Procedência
1	Norte
2 ou 3	Nordeste
4, ou de 10 até 20	Sul
5, ou de 25 até 30	Sudeste
6, 7, 8 ou 9	Centro-Oeste

Seleção Encadeada Homogênea: Exemplo

```c
#include <stdio.h>
int main() {
    float preco;
    int cod;
    printf("Digite o preco e o codigo do produto: ");
    scanf("%f %d", &preco, &cod);
    if(cod == 1)
        printf("preco = %f, produto do Norte\n", preco);
    else if(cod == 2 || cod == 3)
        printf("preco = %f, produto do Nordeste\n", preco);
    else if(cod == 4 || cod >= 10 && cod <= 20)
        printf("preco = %f, produto do Sul\n", preco);
    else if((cod == 5) || (cod >= 25 && cod <= 30))
        printf("preco = %f, produto do Sudeste\n", preco);
    else if(cod >= 6 && cod <= 9)
        printf("preco = %f, produto do Centro-Oeste\n", preco);
    else
        printf("preco = %f, produto importado\n", preco);
}
```

Conteúdo

- Expressões Lógicas
- Estruturas de seleção
  - IF-ELSE
  - SWITCH-CASE

Testes encadeados

É muito comum que tenhamos que realizar uma série de testes sobre um determinado dado.

Por exemplo: considere um programa que recebe um caractere informando o conceito alcançado por um aluno em uma determinada avaliação. O programa deve informar qual nota o aluno obteve na prova, de acordo com a seguinte tabela:

CONCEITO	NOTA
A	>= 9 e <= 10
B	>= 8 e < 9
C	>= 7 e < 8
D	< 7

```c
#include <stdio.h>
int main() {
    char conceito;
    printf("Digite conceito (A,B,C ou D): ");
    scanf("%c", &conceito);
    /* Testa e imprime resposta */
    if(conceito == 'A')
        printf("Nota: >= 9 e <= 10\n");
    else {
        if(conceito == 'B')
            printf("Nota: >= 8 e < 9\n");
        else{
            if(conceito == 'C')
                printf("Nota: >= 7 e < 8\n");
            else{
                if(conceito == 'D')
                    printf("Nota: < 7\n");
                else
                    printf("Conceito invalido");
            }
        }
    }
}
```

Como podemos deixar o código mais legível?

```c
#include <stdio.h>
int main() {
    char conceito;
    printf("Digite conceito (A,B,C ou D): ");
    scanf("%c", &conceito);
    /* Testa e imprime resposta */
    if(conceito == 'A')
        printf("Nota: >= 9 e <= 10\n");
    else if(conceito == 'B')
        printf("Nota: >= 8 e < 9\n");
    else if(conceito == 'C')
        printf("Nota: >= 7 e < 8\n");
    else if(conceito == 'D')
        printf("Nota: < 7\n");
    else
        printf("Conceito invalido");
}
```

Usando *else-if* a lógica fica mais clara!

Testes encadeados

Há alguma maneira de deixar este tipo de teste **mais claro**?

O **programa** deve **reagir** de maneira **diferente** de acordo com o **conteúdo** da variável ***op***.

O comando **switch-case** permite uma fácil **escrita** de **testes encadeados** sobre o valor de uma determinada variável ***int*** ou ***char***.

Comando switch-case

Sintaxe:

```
switch (VARIAVEL) {
case VALOR1:
   COMANDOS-1;
   break;
case VALOR2:
   COMANDOS-2;
   break;
...
default:
      COMANDOS-N;
}
```

VARIAVEL deve ser ***int*** ou ***char***.

Cada expressão **case VALOR:** faz uma comparação de valor, e executa os respectivos comandos se a **VARIAVEL** possuir o mesmo **VALOR**. Se não, testa o próximo case.

break; é usado para sair do switch-case. Se não constar no final dos comandos, o próximo case é testado.

default: (opcional) é ativado se nenhum teste anterior for válido.

O comando **switch-case** *somente* opera dados em um **conjunto discreto**. Você não pode utilizar switch-case com um dado que varia no conjunto dos números reais... Se fosse assim, quantos cases teríamos? Um para cada número real?

```c
#include <stdio.h>
int main() {
    char conceito;
    printf("Digite conceito (A,B,C ou D):
");
    scanf("%c", &conceito);
    /* Testa e imprime resposta */
    switch(conceito){
    case 'A':
        printf("Nota: >= 9 e <= 10\n");
        break;
    case 'B':
        printf("Nota: >= 8 e < 9\n");
        break;
    case 'C':
        printf("Nota: >= 7 e < 8\n");
        break;
    case 'D':
        printf("Nota: < 7\n");
        break;
    default:
        printf("Conceito invalido");
    }
}
```

O mesmo programa com o switch-case.

A leitura eh facilitada.

A desvantagem é que o comando switch-case funciona apenas para conjuntos discretos.

Exercícios

1. Escreva um programa em C que lê dois números e escreve “um deles é par”, caso algum número seja divisível por 2. Caso contrário escreve “nenhum deles é par”.
2. Escreva um programa em C que leia um número e testa se ele pertence ao seguinte intervalo numérico:

$$(-6,13] \cup [42,1024) \cup (2048,\infty)$$

3. Escreva um programa em C que leia os coeficientes A, B e C de uma equação de segundo grau da forma $Ax^2 + Bx + C$ e escreva suas raízes reais. Se A for zero ou se a equação não tiver raízes reais, o programa deve escrever uma mensagem de erro explicando a situação.