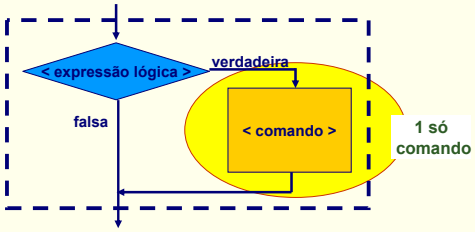
Comando condicional simples

C

```
if ( expressão lógica )
   comando ;
```

Linguagem algorítmica

```
se < expressão lógica >
   < comando >
```



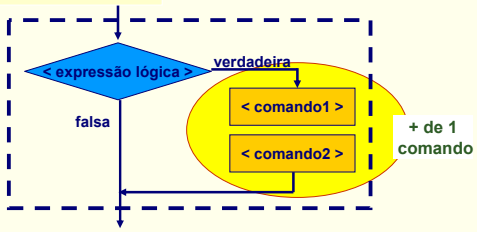
Comando condicional simples

C

```
if ( expressão lógica )
  {
     comando1;
     comandon;
  }
```

Linguagem algorítmica

```
se < expressão lógica >
   < comandoS >
```



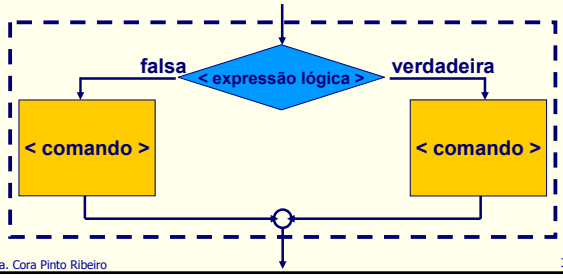
Comando de seleção dupla

C

```
if (expressão lógica)
   comando;
else
   comando;
```

Linguagem algorítmica

```
se < expressão lógica >
   < comando >
senão
   < comando >
```



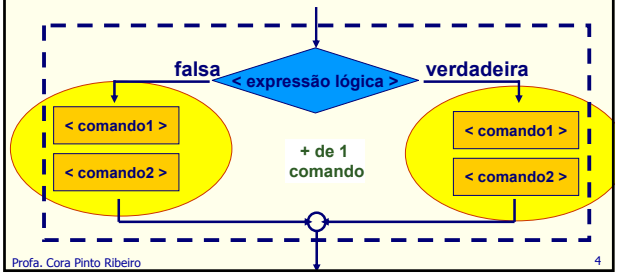
Comando de seleção dupla

C

```
if (expressão lógica)
   comando;
else
   comando;
```

Linguagem algorítmica

```
se < expressão lógica >
   < comando >
senão
   < comando >
```



'If' aninhados

```
if ( expressão lógica )
   comando ;
```

```
if (expressão lógica)
   comando;
else
   comando;
```

comando:
- comando simples
- comando composto (entre chaves!)

Qualquer comando!
Inclusive outro if

Cuidado:
- *else* sempre se refere ao *if* mais próximo!

Solução para modificar: uso de chaves.

'Ifs' encadeados

```
If (< condição >)
  {
     < comando >;
     if  (< condição >)
        {
           < comandos >;
        }
     < comandos >;
  }
```

```
if (< condição 1 >)
   if (< condição 2 >)
      if (< condição 3 >)
         < comando >;
```

```
if (< condição 1 >)
   if (< condição 2 >)
      if (< condição 3 >)
         < comando >;
      else
         <comando>;
   else
      <comando>;
```

'If' encadeados – cuidado !

```
if (< condição 1 >)
   if (< condição 2 >)
      if (< condição 3 >)
         < comando >;
   else // do 2° if
         <comando>;
```

```
if (< condição 1 >)
   if (< condição 2 >)
   {
      if (< condição 3 >)
         < comando >;
      else
         <comando>;
   }
   else
      <comando>;
```

```
if (< condição 1 >)
   if (< condição 2 >)
   {
      if (< condição 3 >)
         < comando >;
   }
   else // do 2° if
         <comando>;
```

ATENÇÃO: ao aninhar-se *ifs*, usar **chaves**, se necessário, para que os *elses* correspondam aos *ifs* corretos ou para documentação.

Exercício

Tendo como dados de entrada o sexo (M ou F) e a altura de uma pessoa (em metros), informe o peso ideal (em kg), sabendo que para homens o peso ideal é obtido por altura x 72,7 - 58 e, para mulheres, por altura x 62,1 - 44,7.

Algoritmo PesoIdeal
{Ler sexo e altura e calcular peso ideal}
Entrada: sexo - M ou F - e altura (m);
Saída: peso ideal (kg)
1 - início
2 - ler sexo
3 - ler altura
4 - se sexo = 'm' // pode aceitar maiúscula
4.1 pesoideal ← altura * 72.7 - 58
4.2 senão
4.2.1 pesoideal ← altura * 62.1 - 44.7 // pode consistir sexo errado
5 - imprimir pesoideal
6 - fim


```
//Ler sexo e altura e calcular peso ideal:
#include <stdio.h>
#include <stdlib.h>

int main()
{
float altura, pesoideal;
char sexo;
printf("digite o sexo (M/F):\n");
scanf("%c", &sexo);
printf("digite a altura:\n");
scanf("%f",&altura);

if (sexo=='M' || sexo=='m')
     pesoideal = altura*72.7 - 58;
else if (sexo=='f' || sexo=='F')
          pesoideal = altura*62.1 - 44.7;
      else
          printf("sexo informado inválido\n");
printf("seu peso ideal é %.2f kg\n",pesoideal);
system("pause");
return 0;
}
```

Se digita outro caractere??? Imprime resultado...


```
//Ler sexo e altura e calcular peso ideal:
#include <stdio.h>
#include <stdlib.h>

int main()
{
float alt
char sex
printf("
scanf("%
printf("
scanf("%

if (sexo
     pesoideal = altura*72.7 - 58;
else if (sexo=='f' || sexo=='F')
          pesoideal = altura*62.1 - 44.7;
      else
          printf("sexo informado inválido\n");
printf("seu peso ideal é %.2f kg\n",pesoideal);
system("pause");
return 0;
}
```

```
          else
              {
               printf("sexo informado inválido\n");
               pesoideal = 0;
              }
    if (pesoideal > 0)
        printf("seu peso ideal é %.2f kg\n",pesoideal);
```



```
//Ler sexo e altura e calcular peso ideal:
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h> // para usar funções toupper – maiúsculas – e tolower

int main()
{
float altura,pesoideal;
char sexo;
printf("digite o sexo (M/F):\n");
scanf("%c", &sexo);
printf("digite a altura:\n");
scanf("%f",&altura);
// compara sempre com minúscula:
if ( tolower(sexo)=='m')          // ou toupper(sexo)=='M'
   pesoideal = altura*72.7 - 58;
else if (tolower(sexo) =='f')
       pesoideal = altura*62.1 - 44.7;
          else
          {
           printf("sexo informado inválido\n");
           pesoideal = 0;
          }
if (pesoideal > 0)
   printf("seu peso ideal é %.2f kg\n",pesoideal);
system("pause");
return 0;
}
```



```
//Ler sexo e altura e calcular peso ideal:
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h> // para usar funções toupper – maiúsculas – e tolower

int main()
{
float altura,pesoideal;
char sexo;
printf("digite o sexo (M/F):\n");
scanf("%c", &sexo);
printf("digite a altura:\n");
scanf("%f",&altura);
sexo = toupper(sexo);  // converte para maiúscula ou tolower(sexo) - minúscula
if (sexo=='M')
  pesoideal = altura*72.7 - 58;
  else if (sexo =='F')
      pesoideal = altura*62.1 - 44.7;
         else
         {
          printf("sexo informado inválido\n");
          pesoideal = 0;
         }
if (pesoideal > 0)
   printf("seu peso ideal é %.2f kg\n",pesoideal);
system("pause");
return 0;
}
```

Exercícios propostos

Fazer o algoritmo e programa C para calcular e informar as raízes de uma equação do 2º grau. Os valores das variáveis a, b e c devem ser fornecidos via teclado.

Algoritmo Equação do Segundo Grau
{ informa as raízes de uma equação do segundo grau }
***entradas*: 3 coeficientes da equação**
***saídas*: 2 raízes ou mensagens**
1. **início**
2. ***ler* 3 coeficientes**
3. ***se* primeiro nulo**
3.1 ***escrever* não é equação do segundo grau**
3.2 ***senão***
3.2.1 **calcular o discriminante**
3.2.2 ***se* discriminante < zero**
3.2.2.1 ***escrever* as raízes são imaginárias**
3.2.2.2 ***senão***
3.2.2.2.1 **calcular 2 raízes**
3.2.2.2.2 ***escrever* as raízes calculadas**
4. ***fim***

1. início
2. *ler* 3 coeficientes
3. *se* primeiro nulo
3.1 *escrever* não é equação do segundo
3.2 *senão*
3.2.1 calcular o discriminante
3.2.2 *se* discriminante < zero
3.2.2.1 *escrever* as raízes são imaginária
3.2.2.2 *senão*
3.2.2.2.1 calcular 2 raízes
3.2.2.2.2 *escrever* as raízes calculadas
4. *fim*

Algoritmo Equação do Segundo Grau
{ informa as raízes de uma equação do segu
entradas: 3 coeficientes da equação – a,b,c
saídas: 2 raízes ou mensagens – r1,r2
1. **início**
2. ***ler* a,b,c**
3. ***se* a = 0**
3.1 ***escrever* não é equação do segundo grau**
3.2 ***senão***
3.2.1 **discriminante ← $b^2 - 4ac$** // como escrever em C?
3.2.2 ***se* discriminante < zero**
3.2.2.1 ***escrever* as raízes são imaginárias**
3.2.2.2 ***senão***
3.2.2.2.1 **r1 ← (-b + √discriminante) / 2a**
3.2.2.2.2 **r2 ← (-b - √discriminante) / 2a**
3.2.2.2.3 ***escrever* r1, r2**
4. ***fim***


```
/* Programa Equacao de Segundo Grau */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>  // para usar função sqrt
int main()
{
  float a,b,c; //coeficientes d
  float disc; // discriminante
  float r1,r2;  // raízes
  // Obtém valor dos coeficie
  printf("digite A:");
  scanf("%f",&a);
  printf("digite B:");
  scanf("%f",&b);
  printf("digite C:");
  scanf("%f",&c);
  // Cálculo das raízes:
  if (a == 0 )
     {
      printf("não é equação de segundo grau\n");
      if (b!=0)
          printf("raiz é %f",-c/b);
      else
          printf("não possui solução\n");
     }
  else
     {
      disc = b*b - 4*a*c;
      printf("disc : %f\n",disc);
      if (disc < 0)
          printf("raizes imaginárias\n");
      else
          {
           r1 = (-b + sqrt(disc))/(2*a);
           r2 = (-b - sqrt(disc))/(2*a);
           printf("raiz 1 : %f\n",r1);
           printf("raiz 2 : %f\n",r2);
          }
     }
  system("pause");
  return 0;
} // Final do programa
```


Exercício

Tendo como dados de entrada 3 valores inteiros e positivos, verificar se estes valores podem ser os lados de 1 triângulo. Em caso positivo, identificar o triângulo formado (eqüilátero, isósceles ou escaleno).
Saída: é a informação referente ao triângulo!

Objetivos: verificar se 3 valores inteiros podem formar 1 triângulo e, em caso positivo, se o triângulo formado é eqüilátero, isósceles ou escaleno.

Requisitos:
- Como verificar se 3 valores inteiros podem formar 1 triângulo?
- O que é eqüilátero, isósceles ou escaleno?

Exercício

Presupostos:
- **Nenhum lado de 1 triângulo pode ser maior ou igual a soma dos outros dois lados.**
- **Escaleno: 3 lados diferentes**
- **Isósceles: apenas 2 lados iguais**
- **Eqüilátero: 3 lados iguais**
- **Nenhum lado pode ser <= zero.**

Estratégia:
- **Identificar seqüência correta e eficaz.**

Exercícios: **Tendo como dados de entrada 3 valores inteiros e positivos, verificar se estes valores podem ser os lados de 1 triângulo. Em caso positivo, identificar o triângulo formado (eqüilátero, isósceles ou escaleno).**

Algoritmo EhTriangulo
{Ler valores e.....}
Entrada: a,b,c {3 valores inteiros, maiores que 0}
Saída: mensagem
1- Início
2- Ler a,b,c
3- Se (a > 0) e (b > 0) e (c > 0) {valores válidos}
3.1 se (a >= b + c) ou (b >= a + c) ou (c >= a + b) {não é triângulo}
3.1.1 escreve ' Não forma triângulo'
3.1.2 senão {é triângulo}
3.1.2.1 se (a=b) e (b=c) {identifica equilátero}
3.1.2.1.1 escreve 'Triângulo Equilátero'
3.1.2.1.2 senão
3.1.2.1.2.1 se (a=b) ou (b=c) ou (a=c)
3.1.2.1.2.1.1 escreve 'Triângulo Isósceles'
3.1.2.1.2.1.2 senão escreve 'Triângulo Escaleno'
3.2 senão escreve 'Numeros fornecidos não são inteiros positivos'
4 - Fim

```c
/* verificar se 3 valores fornecidos são inteiros positivos e se podem
formar 1 triângulo, identificando o tipo de triângulo formado */
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int a,b,c;
    // obtem valores:
    printf("digite A:");
    scanf("%d",&a);
    printf("digite B:");
    scanf("%d",&b);
    printf("digite C:");
    scanf("%d",&c);

// continua no próximo slide
```



```c
// continuação:
    if (a > 0 && b > 0 && c > 0) //valores informados válidos
    {  // verifica se é triângulo:
        if ( (a >= b + c) || (b >= c + a) || (c >= b + a)) // se 1 lado maior ou igual, erro
            printf(" Valores informados não formam triângulo!");
        else            // se chega aqui, é triângulo!
            if ( a==b && b==c)  // identifica equilátero
                printf(" Valores informados formam um triângulo equilátero.");
            else
                if (a == b || b == c || c == a) // identifica isósceles
                    printf(" Valores informados formam um triângulo isósceles.");
                else    // sobrou triângulo escaleno
                    printf(" Valores informados formam um triângulo escaleno.");
    }
    else            // valores informados inválidos
        printf("\nForneca 3 valores inteiros e positivos!");
    printf("\n");
    system("pause");
    return 0;
} // fim do programa
```

Operador && - operador lógico AND

Operador & - operador lógico AND que atua bit à bit, aplicável a variáveis char e int.

Neste caso, usar && !