FORÇA AÉREA
PORTUGUESA

TESTES E CONDIÇÕES

Controlo de fluxo – IF ELSE

```
if (condição)
    instrução1;
[ else instrução2; ]
```

1. *A condição é avaliada;*
2. *Se o resultado da condição for verdadeiro, executa a instrução 1;*
3. *Se o resultado da condição for falso executa a instrução 2 (caso exista **else**);*

Controlo de fluxo – IF ELSE

```
if (condição)
    instrução1;
[ else instrução2; ]
```

Condição para calcular o módulo de um número inteiro n?

1. *A condição é avaliada;*
2. *Se o resultado da condição for verdadeiro, executa a instrução 1;*
3. *Se o resultado da condição for falso executa a instrução 2 (caso exista **else**);*

Controlo de fluxo – IF ELSE

```
if (condição)
    instrução1;
[ else instrução2; ]
```

```
double modulo(double n){
    if (n > 0) return n;
    else return -n;
}
```

1. *A condição é avaliada;*
2. *Se o resultado da condição for verdadeiro, executa a instrução 1;*
3. *Se o resultado da condição for falso executa a instrução 2 (caso exista **else**);*

Controlo de fluxo – IF ELSE

```
if (condição)
    instrução1;
[ else instrução2; ]
```

```
double modulo(double n){
    if (n > 0) return n;
    else return -n;
}
```

Condição para verificar se um ano é bissexto?

- **São bissextos** todos os anos múltiplos de 400, p.ex: **1600, 2000, 2400, 2800...**
- **São bissextos** todos os múltiplos de 4, exceto se for múltiplo de 100 mas não de 400, p.ex: **1996, 2000, 2004, 2008, 2012, 2016, 2020...**
- **Não são bissextos** todos os demais anos.

1. *A condição é avaliada;*
2. *Se o resultado da condição for verdadeiro, executa a instrução 1;*
3. *Se o resultado da condição for falso executa a instrução 2 (caso exista **else**);*

Controlo de fluxo – IF ELSE

```
if (condição)
    instrução1;
[ else instrução2; ]
```

```
double modulo(double n){
    if (n > 0) return n;
    else return -n;
}
```

```
if ( (ano % 400 == 0) ||
     ( (ano % 4 == 0 ) && ( ano % 100 != 0 ) ) )
    return 1;
else
    return 0;
}
```

1. *A condição é avaliada;*
2. *Se o resultado da condição for verdadeiro, executa a instrução 1;*
3. *Se o resultado da condição for falso executa a instrução 2 (caso exista **else**);*

Operadores de comparação

```
int a = 3;
if (a == 3) printf("a igual a 3\n");
if (a != 3) printf("a diferente de 3\n");
if (a >= 3) printf("a maior ou igual que 3\n");
if (4 >  a) printf("4 maior que a\n");
if (a <= 3) printf("a menor ou igual que 3\n");
if (a <  5) printf("a menor que 5\n");
```

Operadores de comparação

```
int a = 3;
if (a == 3) printf("a igual a 3\n");
if (a != 3) printf("a diferente de 3\n");
if (a >= 3) printf("a maior ou igual que 3\n");
if (4 >  a) printf("4 maior que a\n");
if (a <= 3) printf("a menor ou igual que 3\n");
if (a <  5) printf("a menor que 5\n");
```

```
a igual a 3
a maior ou igual que 3
4 maior que a
a menor ou igual que 3
a menor que 5
```

Operadores de comparação

```
int a = 3;
if (a == 3) printf("a igual a 3\n");
if (a != 3) printf("a diferente de 3\n");
if (a >= 3) printf("a maior ou igual que 3\n");
if (4 >  a) printf("4 maior que a\n");
if (a <= 3) printf("a menor ou igual que 3\n");
if (a <  5) printf("a menor que 5\n");
printf("3==3 -> %d\n", 3==3);
printf("3!=3 -> %d\n", 3!=3);
```

Operadores de comparação

```
int a = 3;
if (a == 3) printf("a igual a 3\n");
if (a != 3) printf("a diferente de 3\n");
if (a >= 3) printf("a maior ou igual que 3\n");
if (4 >  a) printf("4 maior que a\n");
if (a <= 3) printf("a menor ou igual que 3\n");
if (a <  5) printf("a menor que 5\n");
printf("3==3 -> %d\n", 3==3);
printf("3!=3 -> %d\n", 3!=3);
```

Estes operadores devolvem sempre **0 ou 1**

```
a igual a 3
a maior ou igual que 3
4 maior que a
a menor ou igual que 3
a menor que 5
3==3 -> 1
3!=3 -> 0
```

Valores verdadeiro / falso

```
printf("3==3 -> %d\n", 3==3);
printf("3!=3 -> %d\n", 3!=3);
if (0) printf("condicao falsa\n");
if (42) printf("condicao verdadeira\n");
```

Valores verdadeiro / falso

```
printf("3==3 -> %d\n", 3==3);
printf("3!=3 -> %d\n", 3!=3);
if (0) printf("condicao falsa\n");
if (42) printf("condicao verdadeira\n");
```

```
3==3 -> 1
3!=3 -> 0
condicao verdadeira
```

if else encadeados

```
printf("introduza um inteiro:");
scanf("%d", &b);
if (b == 1)
    printf("Alan");
else
    printf("numero invalido");
```

if else encadeados

```
printf("introduza um inteiro:");
scanf("%d", &b);
if (b == 1)
    printf("Alan");
else if (b == 2)
    printf("Mathison");
else
    printf("numero invalido");
```

if else encadeados

```
printf("introduza um inteiro:");
scanf("%d", &b);
if (b == 1)
    printf("Alan");
else if (b == 2)
    printf("Mathison");
else if (b == 3)
    printf("Turing");
else
    printf("numero invalido");
```

Escreva um programa que indique ao utilizador se o valor introduzido é um valor positivo, negativo ou o valor zero.

Operadores lógicos

NOT

!cond1 = ?	
True	False
False	True

Operadores lógicos

AND

cond1 **&&** cond2 = ?		
True	True	True
False	False	False
False	True	False
True	False	False

NOT

!cond1 = ?	
True	False
False	True

Operadores lógicos

AND

cond1 **&&** cond2 = ?		
True	True	True
False	False	False
False	True	False
True	False	False

OR

cond1 **||** cond2 = ?		
True	True	True
False	False	False
False	True	True
True	False	True

NOT

!cond1 = ?	
True	False
False	True

Operadores lógicos

cond1 **&&** cond2 = ?		
True	True	True
False	False	False
False	True	False
True	False	False

cond1 **||** cond2 = ?		
True	True	True
False	False	False
False	True	True
True	False	True

!cond1 = ?	
True	False
False	True

```
printf("1 && 1 -> %d\n", 1 && 1);
printf("1 && 0 -> %d\n", 1 && 0);
printf("1 || 0 -> %d\n", 1 || 0);
printf("0 || 0 -> %d\n", 0 || 0);
printf("!0-> %d\n", !0);
printf("!1-> %d\n", !1);
```

Operadores lógicos

cond1 **&&** cond2 = ?		
True	True	True
False	False	False
False	True	False
True	False	False

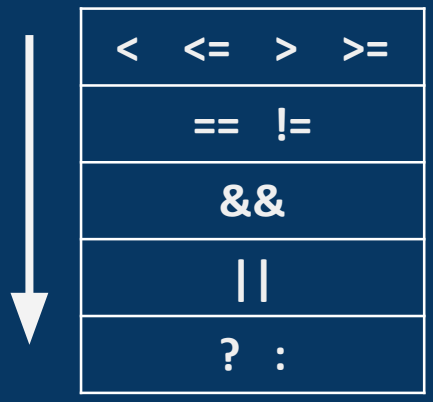
cond1 **||** cond2 = ?		
True	True	True
False	False	False
False	True	True
True	False	True

!cond1 = ?	
True	False
False	True

```
printf("1 && 1 -> %d\n", 1 && 1);
printf("1 && 0 -> %d\n", 1 && 0);
printf("1 || 0 -> %d\n", 1 || 0);
printf("0 || 0 -> %d\n", 0 || 0);
printf("!0-> %d\n", !0);
printf("!1-> %d\n", !1);
```

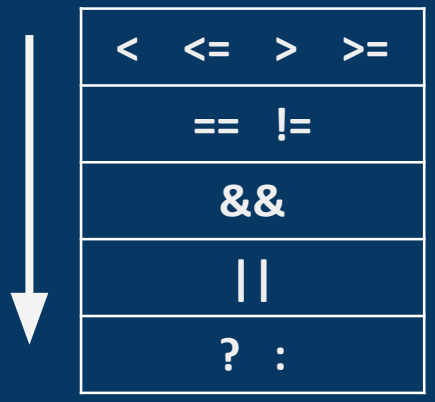
```
1 && 1 -> 1
1 && 0 -> 0
1 || 0 -> 1
0 || 0 -> 0
!0-> 1
!1-> 0
```

Precedências




```
printf("1 > 1 == 0 -> %d\n", 1 > 1 == 0);
printf("1 > 1 == 1 -> %d\n", 1 > 1 == 1);
b = 1 > 1 == 1 || 2 < 3 && 5 > 4;
c = 1 > 1 == 1 || 3 < 3 && 5 > 4;
printf("1 > 1 == 1 || 2 < 3 && 5 > 4 -> %d\n", b);
printf("1 > 1 == 1 || 3 < 3 && 5 > 4 -> %d\n", c);
```

Operadores lógicos




```
1 > 1 == 0 -> 1
1 > 1 == 1 -> 0
1 > 1 == 1 || 2 < 3 && 5 > 4 -> 1
1 > 1 == 1 || 3 < 3 && 5 > 4 -> 0
```

```c
printf("1 > 1 == 0 -> %d\n", 1 > 1 == 0);
printf("1 > 1 == 1 -> %d\n", 1 > 1 == 1);
b = 1 > 1 == 1 || 2 < 3 && 5 > 4;
c = 1 > 1 == 1 || 3 < 3 && 5 > 4;
printf("1 > 1 == 1 || 2 < 3 && 5 > 4 -> %d\n", b);
printf("1 > 1 == 1 || 3 < 3 && 5 > 4 -> %d\n", c);
```

Legibilidade é importante!

Não escrevam artefactos destes nos vossos programas...

```
1 > 1 == 1 || 2 < 3 && 5 > 4
```

```
warning: suggest parentheses around comparison in operand of '==' [-Wparentheses]
warning: suggest parentheses around comparison in operand of '==' [-Wparentheses]
warning: suggest parentheses around comparison in operand of '==' [-Wparentheses]
warning: suggest parentheses around '&&' within '||' [-Wparentheses]
warning: suggest parentheses around comparison in operand of '==' [-Wparentheses]
warning: suggest parentheses around '&&' within '||' [-Wparentheses]
```

Operador condicional ? :

```
<condição> ? <expressão1> : <expressão2>
```

```
double modulo(double n){
    if (n > 0) return n;
    else return -n;
}
```

```
double modulo(double n){
    return (condicao) ? expr1 : expr2;
}
```

Operador condicional ? :

```
<condição> ? <expressão1> : <expressão2>
```

```
double modulo(double n){
    if (n > 0) return n;
    else return -n;
}
```

```
double modulo(double n){
    return (n > 0) ? n : -n;
}
```

Escreva um programa que peça um inteiro entre 1 e o número de nomes que tem, e escreve no terminal o nome correspondente a esse número.

Se o inteiro for **<1** ou **>num_max**, escreve **número inválido**.

```
introduza um inteiro: 4
Alan
```

```
introduza um inteiro: 50
numero invalido
```

switch

Útil quando numa tomada de decisão em que o número de possibilidades é elevado (> 2);

```
switch (expr)
{
    case  valor1:
        intr1;
    case  valor2:
        instr2;
         ...
    case valorN:
        instrN;
    default:
        intr_por_defeito;
}
```

switch

```
printf("introduza um inteiro:");
scanf("%d", &b);
if (b == 1)
    printf("Alan");
else if (b == 2)
    printf("Mathison");
else if (b == 3)
    printf("Turing");
else
    printf("numero invalido");
```

```
switch (expr)
{
    case  valor1:
        intr1;
    case  valor2:
        instr2;
          ...
    case valorN:
        instrN;
    default:
        intr_por_defeito;
}
```

```c
printf("introduza um inteiro:");
scanf("%d", &b);
if (b == 1)
    printf("Alan");
else if (b == 2)
    printf("Mathison");
else if (b == 3)
    printf("Turing");
else
    printf("numero invalido");
```

switch

O que é que o **break** faz?

```c
printf("introduza um inteiro:");
scanf("%d", &b);
switch(b){
    case 1:
        printf("Alan"); break;
    case 2:
        printf("Mathison"); break;
    case 3:
        printf("Turing"); break;
    default:
        printf("numero invalido");
}
```

switch

O que é que o `break` faz?

Permite parar execução dentro de um `switch`.

```
printf("introduza um inteiro:");
scanf("%d", &b);
switch(b){
    case 1:
        printf("Alan"); break;
    case 2:
        printf("Mathison"); break;
    case 3:
        printf("Turing"); break;
    default:
        printf("numero invalido");
}
```