



A Tecnologia e o Futuro num só Centro



# 0809- Programação em C/C++ - Fundamentos

Luís Reis



ASSOCIAÇÃO PORTUGUESA DAS EMPRESAS  
DO SECTOR ELÉCTRICO E ELECTRONICO

Lisb@20<sup>20</sup>

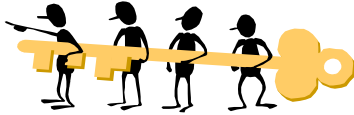
PORTUGAL  
2020



# Conceitos Básicos

- Estrutura básica de um programa
- Variáveis e tipos de dados
- Comandos básicos
- Estruturas de controlo de fluxo

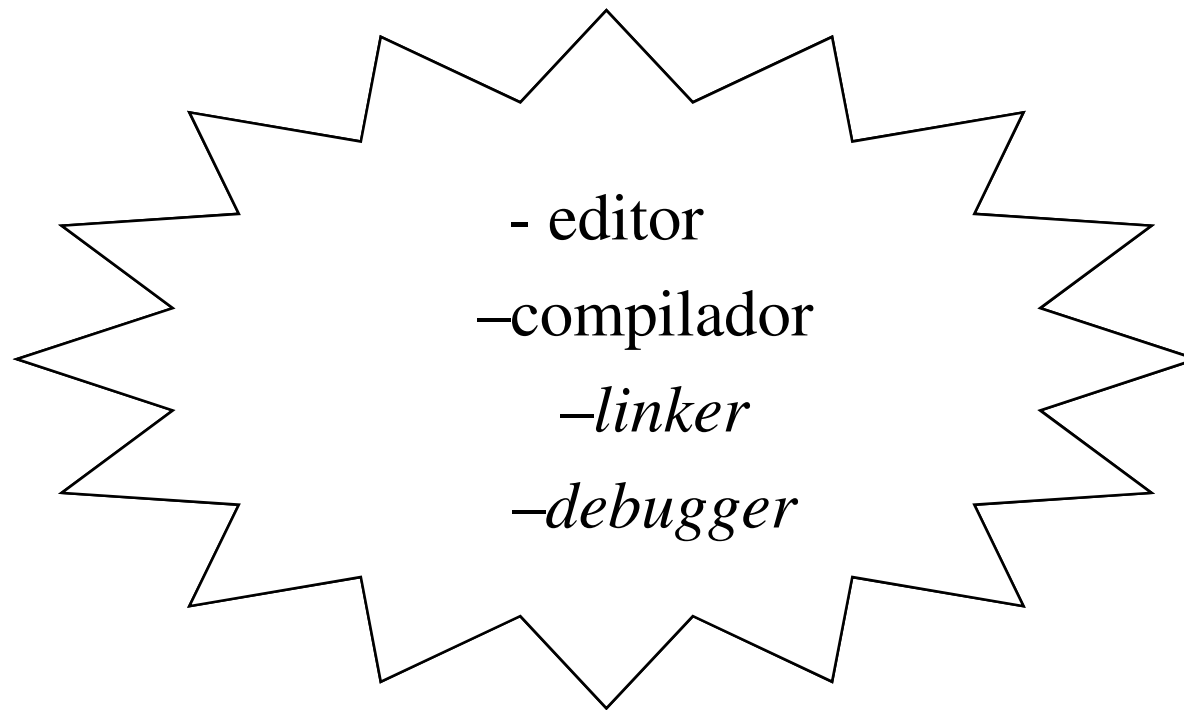
# 1º Programa em C++



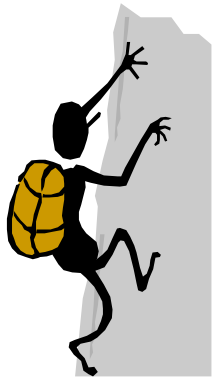
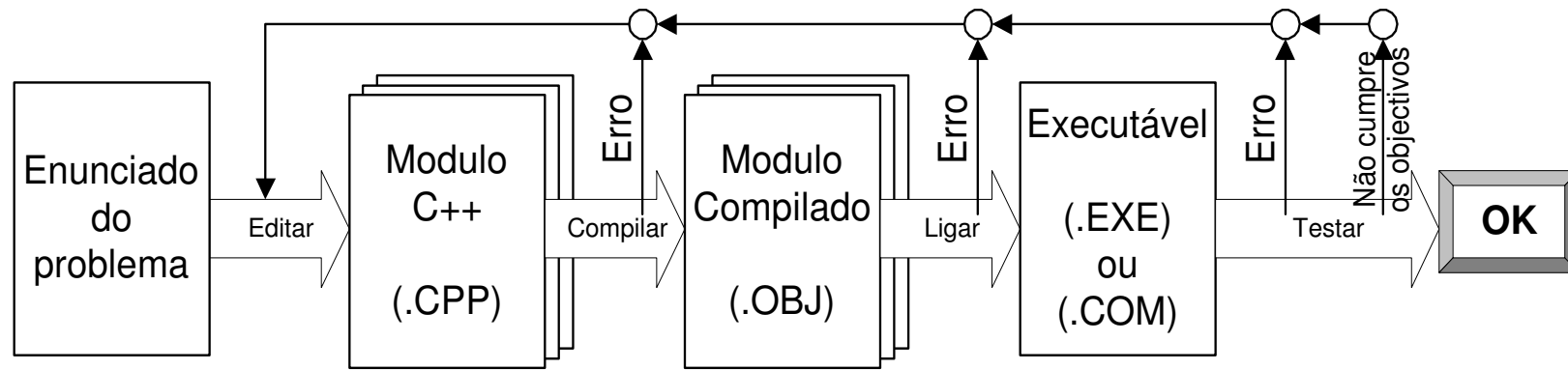
```
#include <iostream>
int main() {
    // Mensagem a mostrar no ecrã.
    cout << "Hello, World!\n";
    return 0;
}
```

- Um programa em C++ é constituído por:
  - Várias funções, das quais uma obrigatoriamente tem que se chamar main.
    - A função main(), como qualquer outra é definida :
      - por um cabeçalho constituído por:
        - tipo de dados que a função devolve.
        - o seu nome.
        - parâmetros formais que recebe.
      - por um corpo (definido entre “{ }”) com :
        - declarações
        - definições
        - instruções
        - comentários
  - Inclusão de ficheiros *header* com:
    - protótipos de funções, macros, instruções, declarações de tipos

# Ambiente de desenvolvimento



# Fases de geração de um programa



Sempre que um programa for constituído por vários módulos, torna-se necessário criar um projecto, indicando quais os módulos que devem ser ligados, na fase de criação do programa final.



# Declarações e definições - variáveis

Em C/C++ qualquer identificador tem que ser declarado antes de ser usado.

**Declarar** uma entidade, consiste exclusivamente em **anunciar a sua existência**, explicitando-lhe o **nome** e o **tipo**.

No entanto, a maior parte das declarações, são também definições, isto é, definem a entidade que passa a ser designada por esse nome.



```
int y = 10; // variável y do tipo inteiro  
char c = 'k'; // variável c do tipo char, iniciada com o caracter 'k'
```

## Tipos primitivos de dados e sua representação

Tipos fundamentais	Significado
int	inteiro
char	carácter
float	vírgula flutuante ( <i>floating point</i> ) de precisão simples
double	vírgula flutuante de precisão dupla

Qualificadores	Aplica-se a	Representa
short	int	menor dimensão
long	int ; double	maior dimensão
signed	char; int	com sinal
unsigned	char; int	valor sem sinal



A partir dos **tipos de variáveis básicos**, definem-se outros tipos de variáveis ditos **tipos derivados**, tais como *arrays*, **enumerados**, **apontadores**, **referências**, **estrutura e uniões** desses tipos fundamentais (ou básicos).

# Alcance e espaço de memória ocupado

Tipo	Ambiente 16 bits	Ambiente 32 bits
signed int	2 bytes (de -32768 a +32767)	4 bytes (- 2 147 483 648 a + 2 147 483 647)
unsigned int	2 bytes (de 0 a 65535)	4 bytes (de 0 a 4 194 967 295)
float	4 bytes (de 3.4E-38 a 3.4E+38)	4 bytes (de 3.4E-38 a 3.4E+38)
signed char	1 byte (-128 a +127)	1 byte (-128 a +127)
unsigned char	1 byte (de 0 a 255)	1 byte (de 0 a 255)
short ou short int	2 bytes (Idêntico a int)	2 bytes (de -32768 a +32767)
unsigned short ou unsigned short int	2 bytes (Idêntico a unsigned int)	2 bytes (de 0 a 65535)
long ou long int	4 bytes (- 2 147 483 648 a + 2 147 483 647)	4 bytes (- 2 147 483 648 a + 2 147 483 647)
unsigned long int ou unsign long	4 bytes (0 a 4 194 967 295)	4 bytes (0 a 4 194 967 295)
double ou long double	8 bytes (1.7E-308 a 1.7E+308)	8 bytes (1.7E-308 a 1.7E+308)



## Apresentação de números no ecrã (*cout*)

```
#include <iostream>
using namespace std;

int main() {
    int int_num = 255;
    float float_num = 99.99895;
    // long int por omissão.
    long big_num = 1250500750.75;
    cout << "12345678901234567890\n";
    cout << int_num << endl;
    cout << big_num << endl;
    cout << float_num << endl;
}
```

Qual será o *output* deste programa



```
12345678901234567890
255
1250500750
99.9989
```

# Operadores e expressões

Como constituintes de um programa, constam também **operadores aritméticos, lógicos e relacionais** ( +, -, \*, /, = ).

Com **variáveis e operadores**, constroem-se **expressões** a que ficam associados valores. Expressões operam sobre variáveis de forma a produzir novos valores.



*Cálculo da área de um círculo*

```
#include <iostream> // Directiva para o pré processador.
Int main() {
    // Declaração (definição e iniciação) de uma constante do tipo float.
    const float PI = 3.14159265;
    int raio; // Declaração (definição) de um inteiro.
    float area; // Declaração (definição) de um float.

    // Instruções de saída e entrada de dados
    cout << "Qual o raio do círculo? ";
    cin >> raio;
    area = PI * raio * raio ; // Aqui têm que existir conversões de tipos.
    cout << "A area e" << area << endl; // Escrita do área no ecrã.
}
```

## Operadores - Precedência e ordem de avaliação (1)

Símbolo	Descrição sumária	Forma de aplicação	Associatividade
:: ::	resolução de alcance refere nome global	nome_classe :: membro :: nome	→
-> [] () () sizeof sizeof	selecção de membro Indexação chamada a função construção de objecto dimensão de objecto dimensão de tipo	<i>apontador -&gt; membro</i> <i>apontador [ exp ]</i> <i>exp ( lista_exp )</i> <i>tipo ( lista_exp )</i> <i>sizeof exp</i> <i>sizeof tipo</i>	→
++ -- ~ ! - + & * new delete delete[] ()	pós ou pré incremento pós ou pré decremento complemento <i>bit a bit</i> negação lógica unário menos unário mais endereço de desreferência criar , alojar destruir , desalojar destruir <i>array</i> <i>cast</i> , conversão de tipo	<i>lvalor ++ ou ++ lvalor</i> <i>lvalor -- ou -- lvalor</i> <i>~ exp</i> <i>! exp</i> <i>- exp</i> <i>+ exp</i> <i>&amp; lvalor</i> <i>* exp</i> <i>new tipo</i> <i>delete apontador</i> <i>delete [ ] apontador</i> <i>( tipo ) exp</i>	←
* / %	multiplicar dividir módulo, resto	<i>exp * exp</i> <i>exp / exp</i> <i>exp % exp</i>	→
+ -	adição , mais subtracção , menos	<i>exp + exp</i> <i>exp - exp</i>	→
<< >>	deslocar esquerda deslocar direita	<i>lvalor &lt;&lt; exp</i> <i>lvalor &gt;&gt; exp</i>	→

## Operadores - Precedência e ordem de avaliação (2)

Símbolo	Descrição sumária	Forma de aplicação	Associatividade
< <= > >=	menor que menor ou igual que maior que maior ou igual que	$exp < exp$ $exp <= exp$ $exp > exp$ $exp >= exp$	→
== !=	Igual Diferente	$exp == exp$ $exp != exp$	→
&	AND <i>bit a bit</i>	$exp \& exp$	→
^	XOR <i>bit a bit</i>	$exp \wedge exp$	→
	OR <i>bit a bit</i>	$exp   exp$	→
&&	AND lógico	$exp \&\& exp$	→
	OR lógico	$exp    exp$	→
? :	Operador condicional	$exp ? exp : exp$	←
= *= /= %= += -= >>= <<= &=  = ^=	afectação simples multiplica e afecta divide e afecta módulo e afecta soma e afecta subtrai e afecta desloca direita e afecta desloca esquerda e afecta AND e afecta <i>bit a bit</i> OR e afecta <i>bit a bit</i> XOR e afecta <i>bit a bit</i>	$lvalor = exp$ $lvalor * = exp$ $lvalor / = exp$ $lvalor \% = exp$ $lvalor + = exp$ $lvalor - = exp$ $lvalor >> = exp$ $lvalor << = exp$ $lvalor \& = exp$ $lvalor   = exp$ $lvalor \wedge = exp$	←
,	vírgula, sequência	$exp , exp$	→

# Atividade 1

- Crie um programa que receba dois números inteiros do teclado, guarde os mesmos em duas variáveis diferentes, some os dois números e imprima o resultado no ecrã

Output: "O resultado de N1 + N2 = resultado"

- Crie um programa que permita calcular a área de um quadrado (LXL);
- Crie um programa que receba do teclado um valor de temperatura em graus Celsius e converta para Fahrenheit, imprimindo o resultado da conversão no ecrã.
  - Fórmula de conversão -  $F = 1,8C + 32$

## Atividade 2

- Crie um programa que calcule o IMC baseado na seguinte fórmula:

$$\text{IMC} = \text{peso} / (\text{altura} \times \text{altura})$$

Nota: receber os dados do teclado

- Crie um programa que receba dois valores guardados nas variáveis “num1” e “num2”. Seguidamente troque os valores contidos nas variáveis  $\text{num1} \rightarrow \text{num2}$  e  $\text{num2} \rightarrow \text{num1}$ .
  - Identifique o potencial problema;
  - Desenvolva uma estratégia para resolver esse problema.

# Constantes

- Por definição são valores não alteráveis, ao contrário das “variáveis”;
- Existem constantes que podem ser declaradas pelo programador, podendo ser definidas através de uma diretiva `#define` ou através da palavra-chave `const`

- Exemplo:

```
#include<iostream>
#define PI 3.14159265
int main()
{
    int raio = 3;
    double areaCirc;
    //PI poderia ter sido definido PI como
    //const float PI = 3.14159265
    areaCirc = raio*raio*PI ;

    ...
}
```

# Funções de Entrada e Saída Formatada



# Representação no ecrã (saída formatada)

## Função *printf()*

- Sintaxe: `printf(<“%especificador”>, <variável>).`
  - Ex: `printf(“O número é %d”, num);`
  - `printf(“Olá Mundo!”);`

<i><b>especificador</b></i>	<b>Saída</b>	<b>Exemplo</b>
d	Inteiro com sinal	392
u	Inteiro sem sinal	7235
x	Inteiro hexadecimal sem sinal	7fa
X	Inteiro hexadecimal sem sinal (maiúscula)	7FA
f	Ponto flutuante decimal (float, double)	392,65
c	Caracter	‘A’
s	Sequência de caracteres – “string”	“Nome”
p	Endereço do ponteiro	b8000000

# Representação no ecrã (saída formatada)

## Função *printf()*

- Caracteres especiais

- Existem caracteres especiais que não podem ser expressos sem ser no código fonte do programa, como por exemplo, nova linha ('`\n`') ou tabulação ('`\t`').
- Todos são precedidos pela barra "`\`". A tabela 1.3 seguinte apresenta a lista dos caracteres especiais.
- Usados dentro das "" da função, p.e:

```
printf("\tOla \n"); // Produzirá uma tabulação antes e um  
                    // quebra de linha depois do texto.
```

Caracter	Significado
<code>\n</code>	nova linha
<code>\r</code>	cursor para 1ª coluna
<code>\t</code>	tabulação
<code>\b</code>	<i>backspace</i>
<code>\'</code>	plica simples
<code>\"</code>	aspas simples

# Entrada de dados (entrada formatada)

## Função *scanf()*

- Sintaxe: `scanf(<especificador(s)>, variável(eis)).`

<i>especificador</i>	Saída	Exemplo
d	Inteiro decimal assinado	392
u	Inteiro decimal sem sinal	7235
x	Inteiro hexadecimal	7fa
f	Ponto flutuante decimal	392,65
c	Caracter	'A'
s	Sequência de caracteres - "string"	"Nome"
p	Endereço do ponteiro	b8000000

- Exemplo 1 (leitura de inteiros):

```
scanf("%d", &num); // Necessário "&" antes da variável.
```

- Exemplo 2 (leitura de cadeias de caracteres – "strings"):

```
char nome[20];  
scanf("%s", nome); // Não é necessário "&" nas "strings"  
printf("O nome é: %s", z nome);
```

# Atividade

Utilização de funções de entrada e saída formatada

- Escreva um programa em C que converta quilômetros por hora em milhas por hora.  $1\text{km} = 0,621371192$  milhas.

Teste ao programa :

Input: quilômetros por hora: 15

Output: 9,32 milhas por hora

- Escreva um programa em C que leia horas e minutos como entrada e calcule o número total de minutos;

Teste ao programa

Inputs:

Horas de entrada: 5

Minutos de entrada: 37

Output:

Total: 337 minutos.

# Strings

- Conceitos básicos
- Funções de manipulação de Strings

# Strings – cadeia de caracteres

- É materializado como um *vetor* (cadeia) de caracteres;
- Tipo de dados complexo (estrutura de dados) que deriva do tipo de dados primitivo “char”.
- Dispõe de diversas funções de manipulação incluídas na biblioteca “string.h/cstring”.
- Declaração e definição:  
    char nome[6]; // declara o array  
    char nome[] = {'M','a','n','u','e','l'}; //declara e define os elementos  
    ou  
    char nome[] = “Manuel”; //declara e define os elementos  
    ou  
    char nome[20] = “Manuel”; //declara e define os elementos

# Ler strings em C/C++

```
#include <stdio.h> // ou <iostream> em C++ -> using namespace std
int main()
{
    char nome[20]; // Declaração de uma cadeia de caracteres, tam. 20
    printf("Insira o nome: ");
    scanf("%s", nome); // Leitura de uma string %s
    printf("O seu nome é %s.", nome);
    return 0;
}
```

# Funções de Entrada/Saída de texto

C - <stdio.h>/ C++ - <iostream>

- `getc(stdin)` – Recebe um caracter – Ex: `char cr = getc(stdin);`
- `gets(variável)` – Recebe uma “string” (cadeia de caracteres)
  - Notas sobre esta função:
    - Esta função não é segura! Como não é possível especificar o número máximo de caracteres a serem lidos, é possível ler caracteres além do tamanho da string passada como parâmetro, causando uma falha de segurança conhecida como *buffer overflow*, ou seja, os caracteres passados além do limite da variável serão guardados numa região da memória a seguir ao final do limite, podendo criar uma série de efeitos indesejáveis. Nesse caso é recomendado o uso da função `fgets`.
- `fgets (variável, tamanho, FILE apontador para ficheiro/stdin);`
  - tamanho – tamanho máximo de leitura;
  - FILE – Apontador para um ficheiro, quando se pretende ler de um ficheiro.
  - stdin – Leitura do teclado;
- `putc(stdout)` – Imprime um caracter – Ex: `putc(cr, stdout);`
- `puts(variável)` – Imprime uma “string”



# Let strings em C – Função gets/fgets

## Exemplo

- Usando *gets()*

```
#include <stdio.h> // Em C
int main()
{
    char nome[20];
    printf("Insira o nome: ");
    gets(nome); //não limita entrada
    printf("Nome: ");
    puts(nome); // mostra string
    return 0;
}
```

- Usando *fgets()*

```
#include <stdio.h> // Em C
int main()
{
    char nome[20];
    printf("Insira o nome: ");
    fgets(nome, sizeof(nome), stdin);
    printf("Nome: ");
    puts(nome); // mostra string
    return 0;
}
```

# Manipulação de strings – funções

## <string.h>/<cstring>

- **strlen ()** - calcula o comprimento de uma string
  - Exemplo: `printf("tamanho da string nome = %d \n",strlen(nome));`
- **strcpy ()** - copia uma string para outra
  - Exemplo: `strcpy(str2, str1); // Copia "str1" para "str2"`
- **strcat ()** - concatena (junta) duas strings
  - Exemplo: `strcat(str1, str2); // guarda o resultado em str1`
- **strcmp ()** - compara duas strings
  - Exemplo: `result = strcmp(str1, str2); // compara as duas strings`

Valor de retorno

Observações

0	se as strings são iguais
> 0	se o primeiro caractere em str1 é maior (em ASCII) do que o de str2.
< 0	se o primeiro caractere em str1 é menor (em ASCII) do que o de str2.

# Atividade

- Escreva um programa em C que leia um nome próprio, sobrenome e ano de nascimento e exiba os nomes e o ano um após o outro sequencialmente:

*Resultado esperado :*

Digite o seu primeiro nome: Ana

Digite o seu sobrenome: Silva

Digite o seu ano de nascimento: 1982

Ana Silva 1982

- Guarde numa única variável do tipo “string” o resultado do ponto anterior.

# Instruções de controlo de fluxo

Instruções condicionais

Estruturas de repetição

# Instrução condicional *if...else*

Fluxograma	Em C++
<pre>graph TD; Entry(( )) --&gt; Cond{A &lt; B}; Cond -- TRUE --&gt; TrueBox[cout &lt;&lt; "A &lt; B";]; Cond -- FALSE --&gt; FalseBox[cout &lt;&lt; "A &gt;= B";]; TrueBox --&gt; Merge(( )); FalseBox --&gt; Merge; Merge --&gt; Exit(( ))</pre>	<pre>if ( A &lt; B )     cout &lt;&lt; "A &lt; B"; else     cout &lt;&lt; "A &gt;= B";</pre>

# Exemplo 1

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

```
void main( )
```

```
{
```

```
    int x, y;
```

```
    x = 15;
```

```
    y = 18;
```

```
    if (x > y )
```

```
    {
```

```
        printf("x é maior que y");
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("y é maior que x");
```

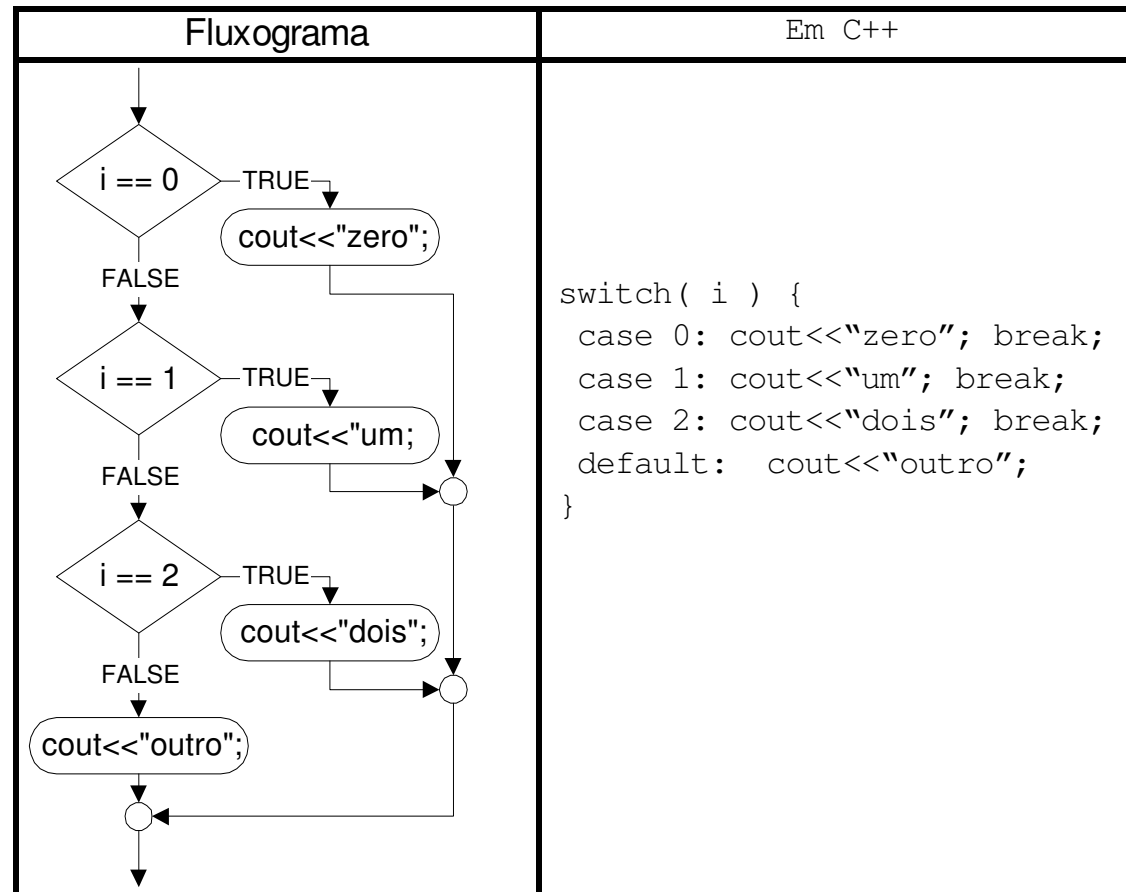
```
    }
```

```
}
```

## Atividade 2

1. Crie um programa que receba dois valores inteiros em duas variáveis distintas e que compare esses valores recebidos.
  - Imprima o maior dos dois valores;
  - Altere o programa anterior de modo a receber e comparar 3 inteiros, verificando qual o maior deles;
2. Dado um número inteiro verifique se o mesmo é par e imprima o resultado.
  - Nota: O resto da divisão inteira por dois sendo zero, comprova a paridade do número. ->  $\text{resto} = \text{num} \% 2$ ;
3. Dadas 3 notas de alunos, crie a sua média e imprima reprovado ou aprovado.

# Instrução condicional *switch...case*





# Instrução condicional

## *switch...case* – Estrutura

```
int op;
cout << "Escolha a opção\n";
cin >> op;
switch (op)
{
case 1:
    comando1; // executa estes comandos se op for igual a 1
    comando2;
    break;
case 2:
    comando1; // executa estes comandos se op for igual a 2
    comando2;
    break;
default:
    comando1; // executa estes comandos se
    comando1; // op for diferente de qualquer um dos valores (1 e 2)
}
```

# Instrução condicional

## *switch...case* – Exemplo

```
printf ("Digite um valor de 1 a 7: ");  
scanf("%d", &valor);  
switch ( valor )  
{  
case 1 :  
printf ("Domingo\n");  
break;  
case 2 :  
printf ("Segunda\n");  
break;  
case 3 :  
printf ("Terça\n");  
break;
```

→

→

```
case 4 :  
printf ("Quarta\n");  
break;  
case 5 :  
printf ("Quinta\n");  
break;  
case 6 :  
printf ("Sexta\n");  
break;  
case 7 :  
printf ("Sabado\n");  
break;  
default :  
printf ("Valor invalido!\n");  
}
```

## Exercício 4 – Switch...Case

1. Elabore um programa que leia dois valores e a operação que se deseja executar (Operações: soma, subtração, divisão, multiplicação). Execute a operação desejada e imprima o resultado.
2. Uma loja atribui 10% de desconto para funcionários e 5% de desconto para clientes frequentes. Desenvolva um programa que calcule o valor total a ser pago por uma pessoa.
  - O programa deverá ler o valor total da compra efetuada e um código que identifique se o comprador é um cliente comum (1), funcionário (2) ou frequente (3).

# Instruções de repetição (ciclos)

*while, for, do-while*

Fluxograma	Em C++
<pre>graph TD; Start(( )) --&gt; Init(i = 1;); Init --&gt; Junction(( )); Junction --&gt; Cond{i &lt; 6}; Cond -- TRUE --&gt; Print( cout &lt;&lt; "i = " &lt;&lt; i &lt;&lt; endl; ); Print --&gt; Inc(i = i + 1; ); Inc --&gt; Junction; Cond -- FALSE --&gt; Exit(( ))</pre>	<pre>i = 1; while ( i &lt; 6 ) {     cout &lt;&lt; "i=" &lt;&lt; i &lt;&lt; endl;     i = i + 1 ; }</pre> <pre>for ( i=1 ; i&lt;6 ; i=i+1 )     cout &lt;&lt; "i=" &lt;&lt; i &lt;&lt; endl;</pre>
<pre>graph TD; Start(( )) --&gt; Junction(( )); Junction --&gt; Print( cout &lt;&lt; "i = " &lt;&lt; i &lt;&lt; endl; ); Print --&gt; Inc(i = i + 1; ); Inc --&gt; Cond{i &lt; 6}; Cond -- TRUE --&gt; Junction; Cond -- FALSE --&gt; Exit(( ))</pre>	<pre>do {     cout &lt;&lt; "i=" &lt;&lt; i &lt;&lt; endl;     i = i + 1 ; } while ( i &lt; 6 );</pre>