Programação em C++:
Introdução

J. Barbosa

J. Tavares

Conceitos básicos de programação

- **Algoritmo**
  - Conjunto finito de regras sobre as quais se pode dar execução a um dado processo (Knuth73v1)
    - Ex: ordenação de um conjunto, pesquisa numa base de dados.
  - Atributos que deve possuir:
    - Ser finito, inteligível, exequível, caracterizável.
  - Formas de representação :
    - Narrativa, Fluxograma, Pseudo código, Linguagens de programação

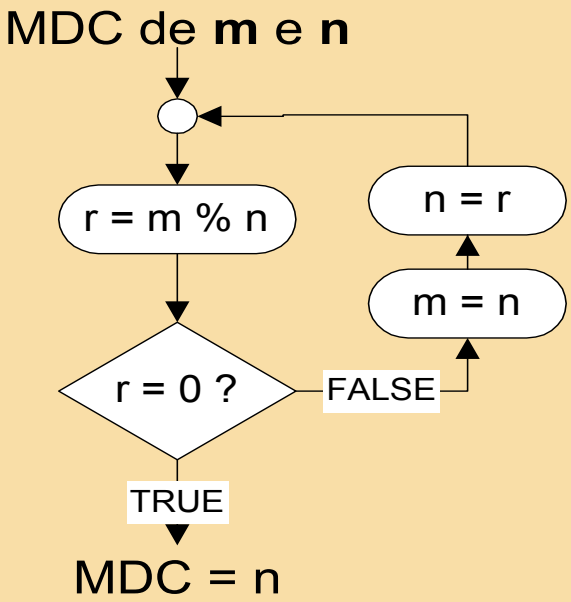
Exemplo: Algoritmo de Euclides

- **Enunciado**:
  - Dados dois inteiros **m** e **n**, encontrar o maior inteiro que os divida a ambos exactamente.

- **Descrição narrativa:**
  - **Algoritmo mdc** (Algoritmo de Euclides)
    - 1- (Encontrar o resto) - Dividir **m** por **n** e afectar r com o resto (**0 <= r < n**)
    - 2 - (O resto é zero?) - Se **r=0**, o algoritmo termina ; **n** é o valor procurado.
    - 3 - (Substituir) - Afectar **m** com **n** e **n** com **r**, voltando ao passo 1.

Exemplo: Algoritmo de Euclides

Descrição em fluxograma



Descrição em linguagem C++

```
int mdc(int m, int n) {
   int r;
   while ( (r= m % n) != 0) {
            m=n;
            n=r;
   }
   return n;
}
```

Programação em C++

- 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 (main).
        - » parâmetros formais que recebe.
      - por um corpo (definido entre "{ }") com :
        - » declarações
        - » definições
        - » instruções
        - » comentários

```
#include <iostream.h>
int main(int argc, char *argv[])
{   // comentário
    char st[] = "Hello world!";

    cout << st << endl;
    return 0;
}
```

  - Inclusão de ficheiros *header* com:
    - protótipos de funções, macros, instruções, declarações de tipos

Programação em C++ : Declarações no programa

```
int mdc(int m, int n) {
   int r;
   while ( (r= m % n) != 0) {
            m=n;
           n=r;
   }
   return n;
}
```

- Em C++ qualquer identificador tem que ser **declarado** antes de ser usado.
- **Declarar** uma entidade, consiste em anunciar a sua existência, explicitando-lhe o nome e o tipo e possivelmente definir o seu valor.

Exemplo:

```
int y;            // variável y do tipo inteiro
char x='A';   /*variável do tipo char,   iniciada com o carácter 'A'.*/
char y=65;    // x == y
```

Programação em C++ : Operadores e expressões

```
int mdc(int m, int n) {
   int r;
   while ( (r= m % n) != 0) {
            m=n;
            n=r;
   }
   return n;
}
```

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

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

Representação de caracteres

- **ASCII** *(American Standard Code for Information Interchange)*: código alfanumérico, porque representa letras e algarismos.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NUL							BEL	BS	HT	LF	VT	FF	CR		
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DEL											ESC				
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	T	_
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	del

Palavras chave

Palavras chave do C++					
asm	auto	break	case	catch	char
class	const	continue	default	delete	do
double	else	enum	extern	float	for
friend	goto	if	inline	int	long
new	operator	private	protected	public	register
return	short	signed	sizeof	static	struct
switch	template	this	throw	try	typedef
union	unsigned	virtual	void	volatile	while

Tipos primitivos de dados

Tipos fundamentais	Significado
int	inteiro
char	carácter
float	vírgula flutuante de precisão simples
double	vírgula flutuante de precisão dupla

Qualificadores	Aplica-se a
short	int
long	int
signed	int; char
unsigned	int; char

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).

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)

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	*apontador -> membro* *apontador [exp]* *exp (lista_exp)* *tipo (lista_exp)* *sizeof exp* *sizeof tipo*	→
++ -- ~ ! - + & * new delete delete[] ()	pós ou pré incremento pós ou pré decremento complemento *bit* a *bit* negação lógica unário menos unário mais endereço de desreferência criar , alojar destruir , desalojar destruir *array* *cast* , conversão de tipo	*lvalor ++ ou ++ lvalor* *lvalor -- ou -- lvalor* *~ exp* *! exp* *- exp* *+ exp* *& lvalor* ** exp* *new tipo* *delete apontador* *delete [ ] apontador* *(tipo) exp*	←
* / %	multiplicar dividir módulo, resto	*exp * exp* *exp / exp* *exp % exp*	→
+ -	adição , mais subtracção , menos	*exp + exp* *exp - exp*	→
<< >>	deslocar esquerda deslocar direita	*lvalor << exp* *lvalor >> exp*	→

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 *bit* a *bit*	*exp* & *exp*	→
^	XOR *bit* a *bit*	*exp* ^ *exp*	→
|	OR *bit* a *bit*	*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 *bit* a *bit* OR e afecta *bit* a *bit* XOR e afecta *bit* a *bit*	*lvalor* = *exp* *lvalor* * = *exp* *lvalor* / = *exp* *lvalor* % = *ex* *lvalor* + = *exp* *lvalor* - = *exp* *lvalor* >> = *exp* *lvalor* <<= *exp* *lvalor* & = *exp* *lvalor* | = *exp* *lvalor* ^ = *exp*	←
,	vírgula, sequência	*exp* , *exp*	→

Exercício

- Programa para verificar o espaço de memória ocupado por cada tipo de variável.

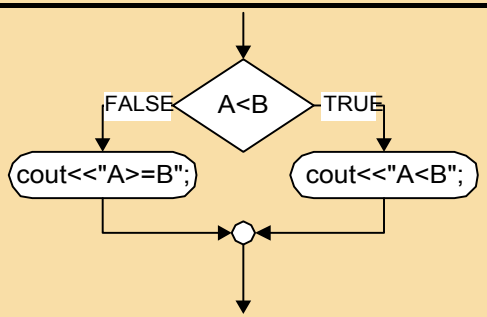
```
#include <iostream.h>

int main()
{
  char ch;
  int num1;
  cout << "Bytes usados para guardar 1 char: "
       << sizeof(ch) << endl;
  cout << "Bytes usados para guardar 1 inteiro: "
       << sizeof(num1) << endl;

  return 0;
}
```

Instruções de controlo de execução

- Decisão binária - *if*
- Operador condicional ternário (?:)
- Decisão múltipla – *switch*
- Repetição condicional - *while, for, do-while*

Decisão binária - *if*

Fluxograma	Em C++

	`if ( A < B )`
	`  cout << "A<B";`
	`else`
	`  cout << "A>=B";`

Exemplo: Determinar se o ano é bissexto

```
#include <iostream.h>
void main() {
   int year;
   cout << "Ano -> "; cin >> year;
   if (year%400 == 0 || year%4 == 0 && year%100 != 0)
         cout << "E'";
   else
         cout << "Nao e'";
   cout << " um ano bissexto." << endl;
}
```

Operador condicional ternário (?:)

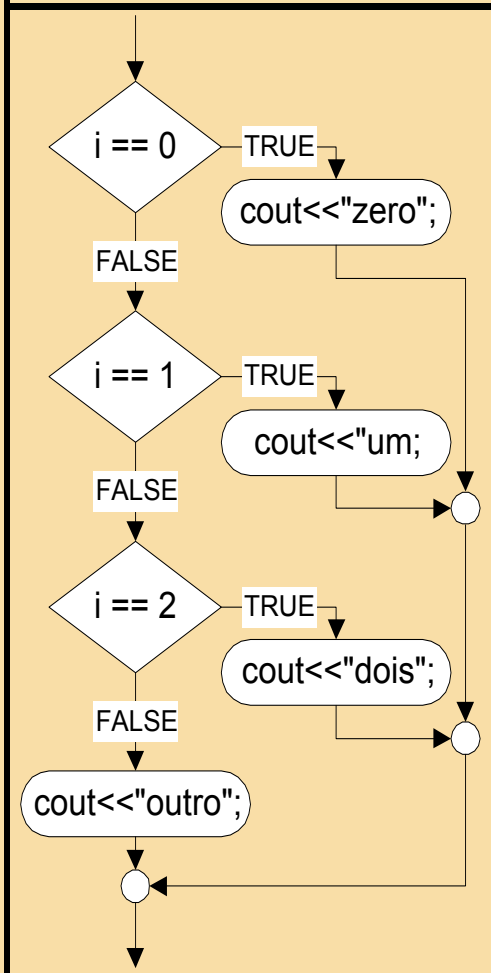
O operador condicional ternário é uma forma compactada de exprimir uma acção condicional **if-else;**

```
modulo_n = (n<0) ? -n: n;
```

```
if (n<0)
   modulo_n=-n;
else
   modulo_n=n;
```

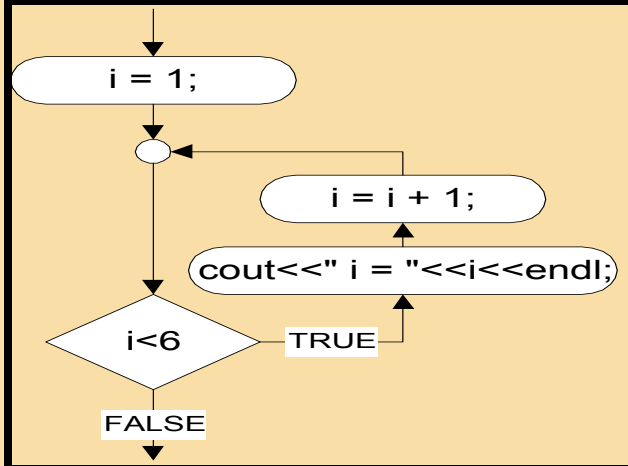
Decisão múltipla – *switch*

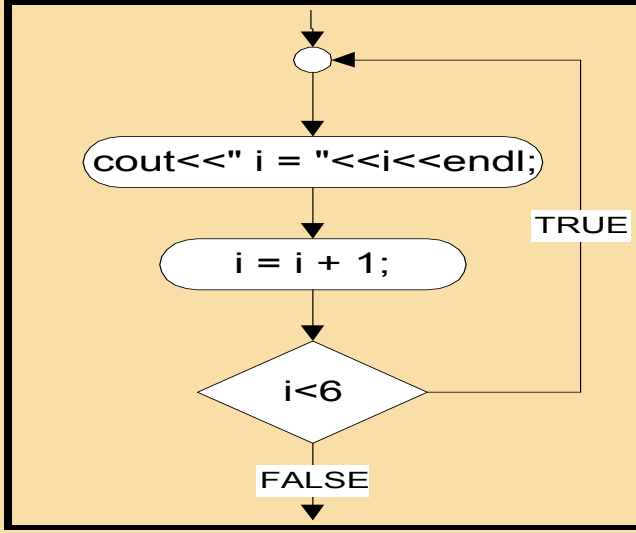
Fluxograma	Em C++

	`switch( i ) {`
	`case 0: cout<<"zero"; break;`
	`case 1: cout<<"um"; break;`
	`case 2: cout<<"dois"; break;`
	`default: cout<<"outro";`
	`}`

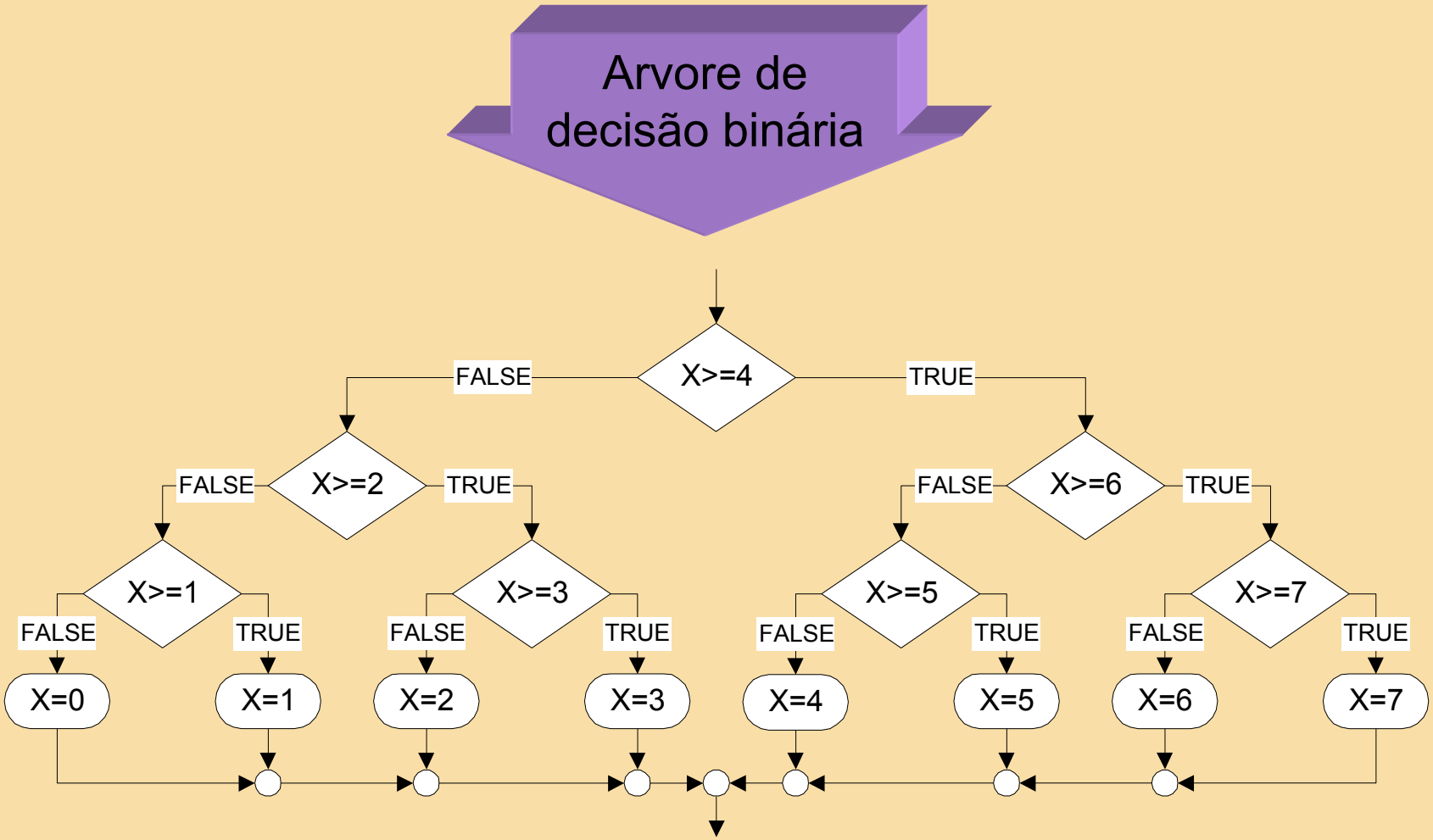
Repetição condicional - *while, for, do-while*

Fluxograma	Em C++

	`i = 1;`
	`while ( i < 6 ) {`
	`  cout << "i=" << i << endl;`
	`  i = i + 1 ;`
	`}`
	
	`for ( i=1 ; i<6 ; i=i+1 )`
	`  cout << "i=" << i << endl;`

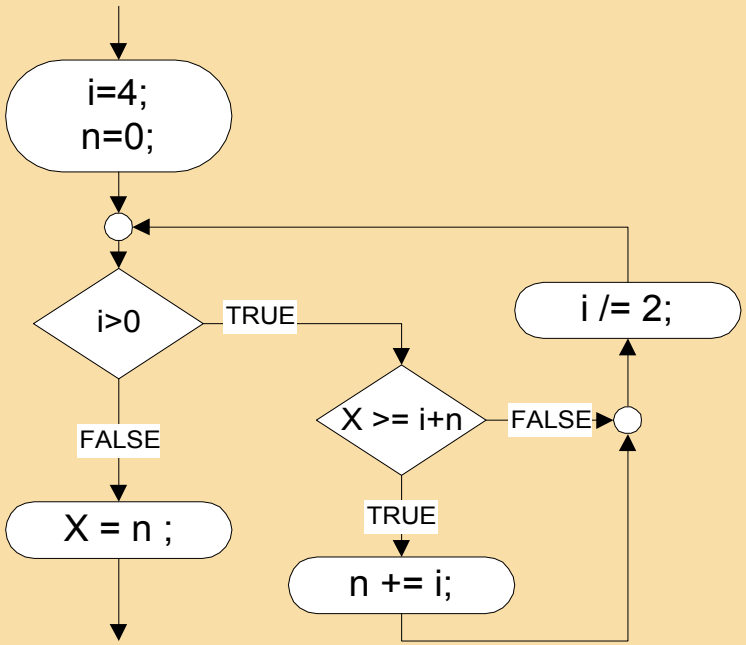
	`do {`
	`  cout << "i=" << i << endl;`
	`  i = i + 1 ;`
	`} while ( i < 6 );`

Exemplo: Algoritmo de aproximações sucessivas
Quantas tentativas são necessárias para adivinhar um nº entre 0 e 7?



Algoritmo de aproximações sucessivas

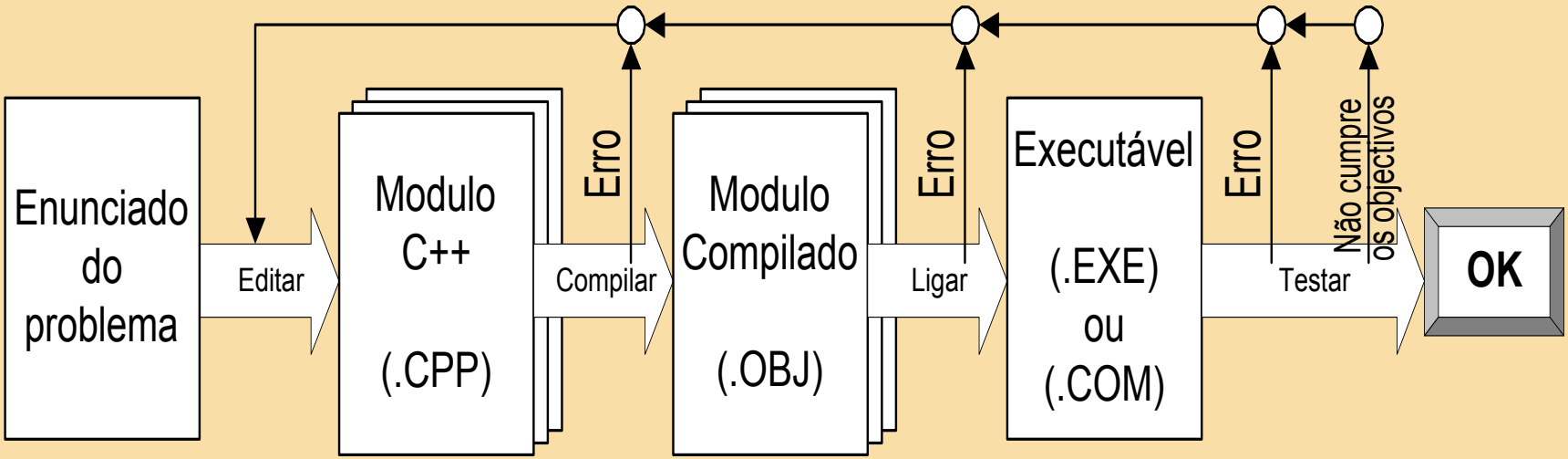
Fluxograma



Programa em C++

```
#include <iostream.h>
#include <ctype.h>
void main() {
  cout << "Pense num numero de 0 a 7 "
          " e prima uma tecla.";
  cin.get();
  // Adivinhar o número.
  int number = 0;
  for(int i = 4 ; i > 0; i /= 2) {
    char ch;
    cout << "\nE' maior ou igual que "
         << (i + number) << " (S/N)?";
    cin >> ch;
    if (toupper(ch) == 'S')
      number += i;
  }
  cout <<"\nO numero que pensou e' "
       << number <<'.' << endl;
}
```

Programação em C++: criação do programa executável



Exercícios

- Escreva um programa para resolver equações quadráticas:

$ax^2+bx+c=0$

- O ganho de tensão de um amplificador é dado por

$\mathbf{v} = [23/(23^2 + (0.5f)^2)^{1/2}]^n$

onde f é a frequência de funcionamento em Hertz e n o número de etapas do amplificador.

Escreva um programa que calcule **v** em função dos valores **f** e **n.**