

2

Bibliotecas e a Diretiva using namespace

BREVE HISTÓRICO

A linguagem C++ foi desenvolvida inicialmente por Bjarne Stroustrup na AT&T, de 1979 a 1983, à partir da linguagem C, tendo como idéia principal a de agregar o conceito de classes, de orientação à objetos, àquela linguagem.

A partir da primeira versão de 1983, a linguagem foi sendo revisada e evoluindo, tornou-se disponível fora da AT&T em 1985, e após um longo processo foi padronizada pela ISO no final de 1997, pelo padrão ISO/IEC 14882. Você pode obter mais informações sobre o desenvolvimento da linguagem na página do próprio autor em [STR 2004].

STL é uma parte do padrão C++, trata-se de uma biblioteca de funções e estruturas de dados que todo compilador C++ deve oferecer, provê as implementações mais comuns com diferentes tipos de dados.

IDE – COMPILADORES

IDE/Compilador é qualquer software que permite a compilação (correção de erros de sintaxe) e execução de códigos fontes de uma determinada linguagem. Os compiladores do projeto GNU (Licença Livre) podem ser obtidos livremente através da Internet. O ambiente BloodShed Dev-C++ roda no Windows e utiliza os compiladores gcc e g++.

Você poderá também utilizar outros IDEs tais como: DevCpp, DevC++, Falcon C++, Codblocks, NetBeans, Eclipse, repl.it e outros.

BLOCO - ESTRUTURA BÁSICA DO C++

Um programa C++ é composto de funções, cada função com seu código digitado dentro de duas chaves opostas o que chamamos de Bloco { }. Todo comando sempre em letra minúscula se finalizando com ponto e vírgula (;) .

Estrutura de um programa C/C++:

```
#include <iostream >  
using namespace std;  
int main (void)  
{  
    comando 1;  
    comando 2;  
    comando 3;  
  
    return (0) ;  
}
```

PRIMEIRAS INSTRUÇÕES EM C++

- função **main()** => o código fonte deverá ser digitado dentro do bloco dessa função principal do C++, sem ela o programa não executa;
- instrução **cout <<** => este comando faz parte da biblioteca `iostream`, serve para enviar/imprimir texto ou variáveis no vídeo;
- instrução **cin >>** => permite a entrada/leitura de variáveis, também da biblioteca `iostream`;
- instrução **system()** => permite a execução de um comandos/programas externos ao código fonte;
- instrução **setlocale(LC_ALL, “Portuguese”)** => permite acentuação gráfica em português.

DIRETIVAS DO PRÉ-PROCESSADOR

São instruções, colocadas geralmente no início do código fonte, que serão chamadas por um programa denominado pré-processador com a finalidade de executar as bibliotecas e códigos incluídos pelas mesmas.

Uma directiva não contém ponto-e-vírgula no final da linha, quando está entre `< >` o arquivo é procurado somente na pasta `include` e quando entre `" "` é procurado primeiro na pasta onde está o executável do código fonte, caso não encontre a biblioteca nesta pasta então irá procurar posteriormente na pasta default `"include"`.

PRINCIPAIS DIRETIVAS

#include <> => serve para incluir os arquivos de bibliotecas.

Ex.: **#include <iostream>** ou **#include "iostream"**

iostream => biblioteca que contém comandos e funções específicas de entrada e saída, necessárias no código fonte, instruções cin e cout entre outras.

using namespace std => A diretiva using evita a digitação da sigla da biblioteca **std**(sigla derivada de STL (Standard template Library), sem essa instrução seria necessário colocar sempre o prefixo **std::** em todos os comandos do código fonte.

Sem a instrução o comando seria digitado assim: **std::cout << "Olá Mundo...!";**

Com instrução, poderia ser digitado assim: **cout << "Olá Mundo";**

TIPOS PRIMITIVOS SUPORTADOS

Cada variável ou constante poderá armazenar um tipo de informação em forma de texto ou de números:

bool – tipo lógico/booleano, exemplo: bool maior = true ou maior = false (1 ou 0).

float – tipo real com 32 bits, exemplo float x= 0.55F, y = 1.22F.

char – têm o tamanho de byte ou seja 8 bits, armazena caracteres ASCII ou números de 8 bits, exemplo char x = 'a'; (caracteres especiais são representados entre aspas simples)

int – armazena números inteiros com 32 bits.

string - não é primitivo, mas serve para armazenar texto a função getline()

OPERADORES ARITMÉTICOS E RELACIONAIS

São os principais símbolos utilizados em fórmulas matemáticas e comparações entre dados:

divisão => / produto => * soma => + subtração => -

raíz cúbica: raíz = cbrt (valor) ex. double x = cbrt (27) resultado 3

raiz quadrada: raiz = sqrt (valor) ex. double x = sqrt (25) resultado 5

potenciação: potência = pow(base, expoente) ex. x = pow(4, 2) resultado 16

Qualquer Raiz: pow(base, expoente) ex. raiz quinta de 2 x = pow(2, 0.2)

comparação entre valores => ==

diferença => !=

! Negação


= uso em fórmulas / atribuição

Comparações compostas && AND

|| OR (pipe)

CÓDIGOS ESPECIAIS

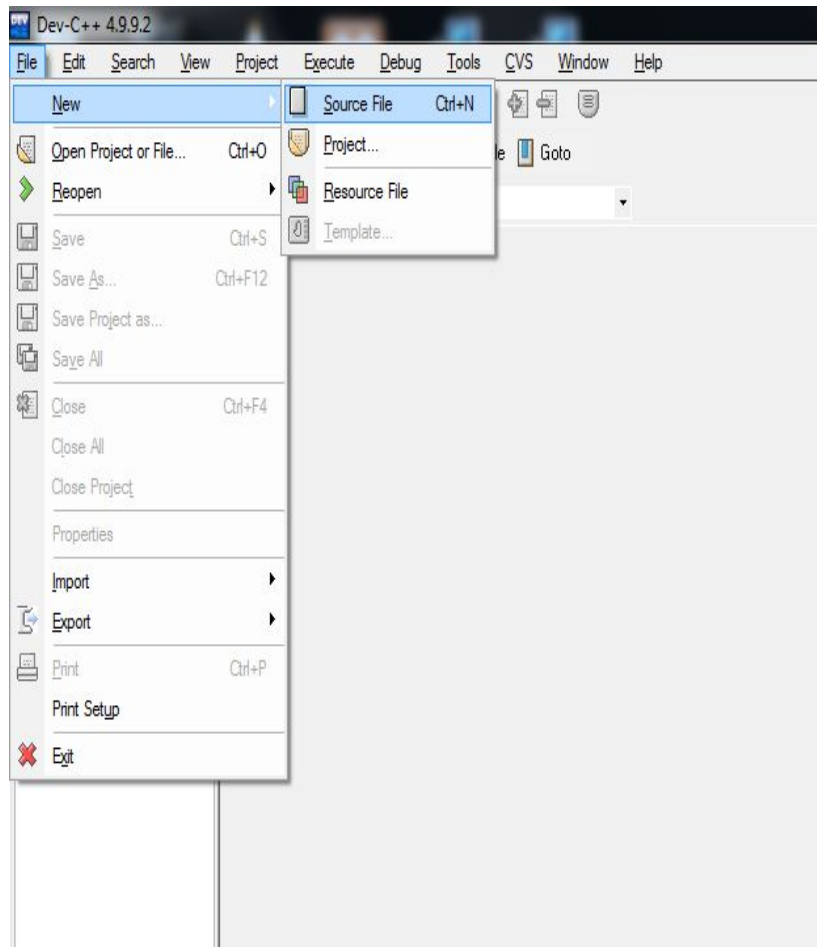
São usados dentro dos textos que serão impressos no vídeo ou impressora quando necessários.

CÓDIGOS ESPECIAIS	SIGNIFICADO	Utiliza-se o \ (Barra Invertida)
\n	Nova linha (CR+LF)	
\t	Tab	
\b	Retrocesso	
\f	Salta página de formulário	
\a	Beep — Toca o alto-falante	
\r	CR — Cursor para o início da linha	
\\	\ — Barra invertida	
\0	Null — Zero	
\'	' — Aspa simples	
\"	" — Aspa dupla	
\xdd	Representação hexadecimal	

**Passos para criar um
código fonte C++ no site
repl.it e no DevC++**

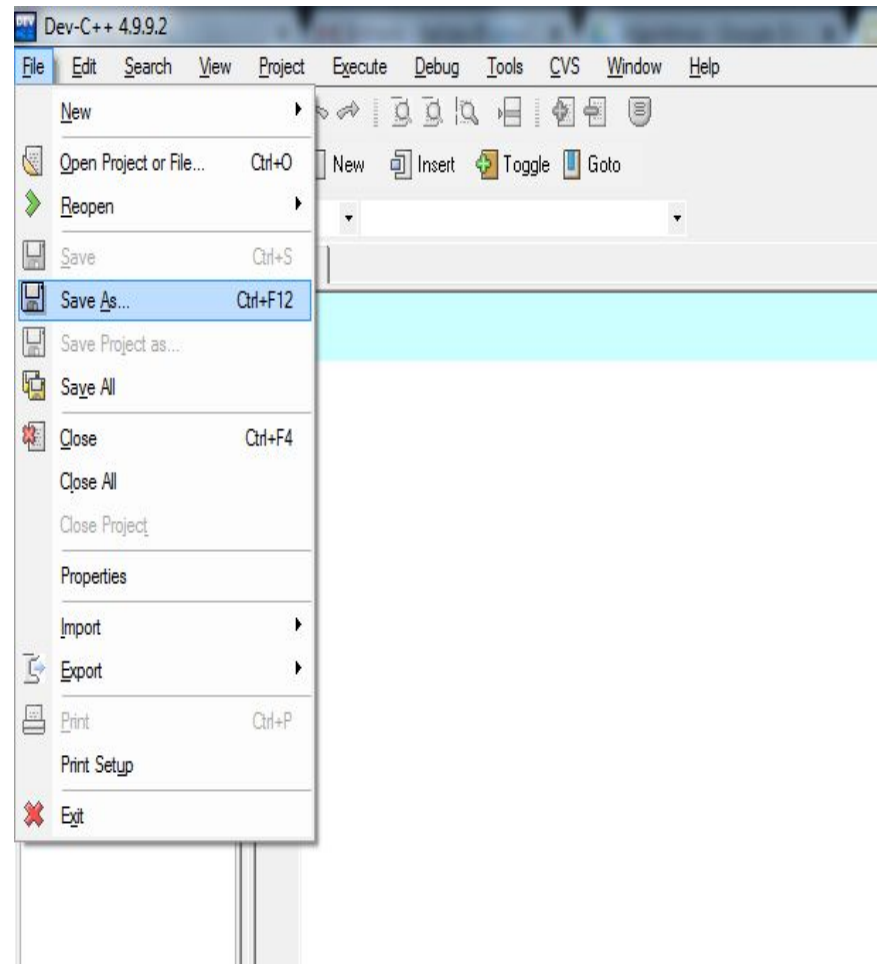
Criando um código fonte C++ no Dev Cpp

- 1) No Dev Cpp, você deverá acessar o menu, selecionar novo e executar a opção Arquivo Fonte:



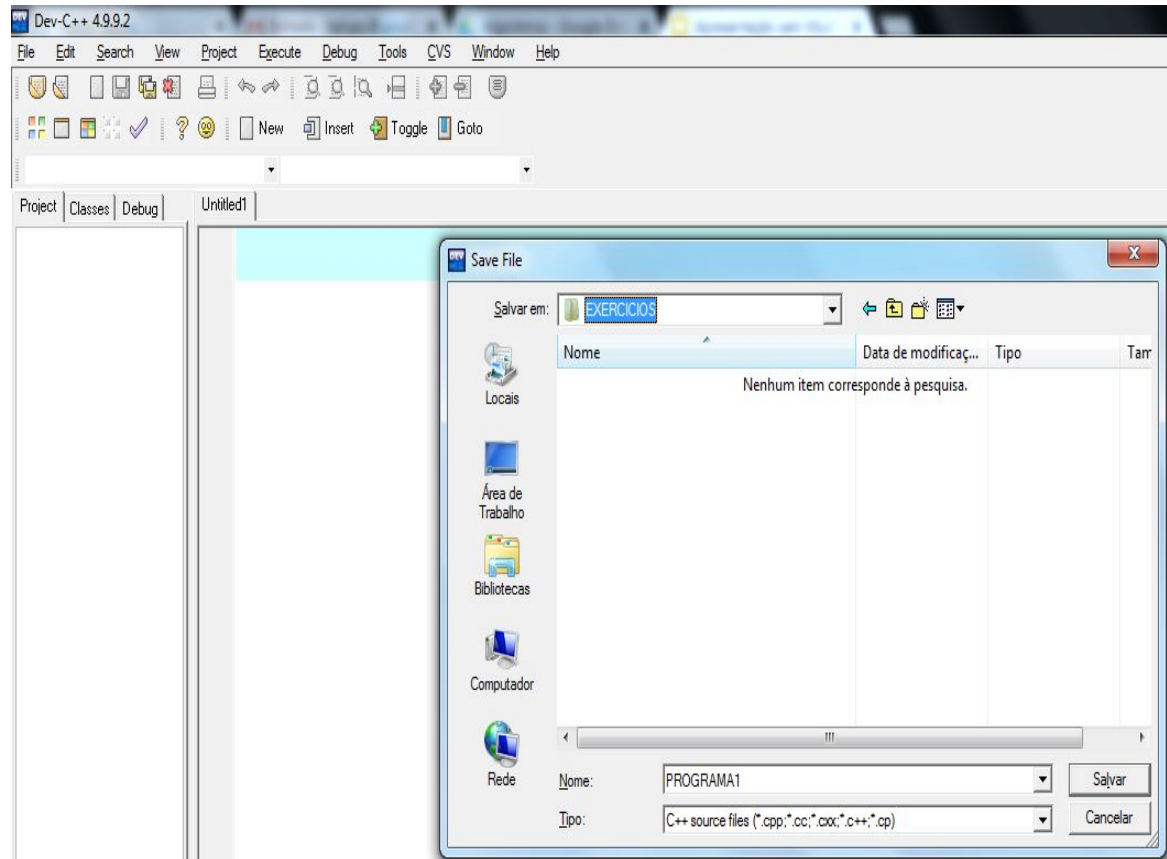
Salvando o código fonte no Devcpp

2) Selecionar a opção do menu Novo, salvar como:



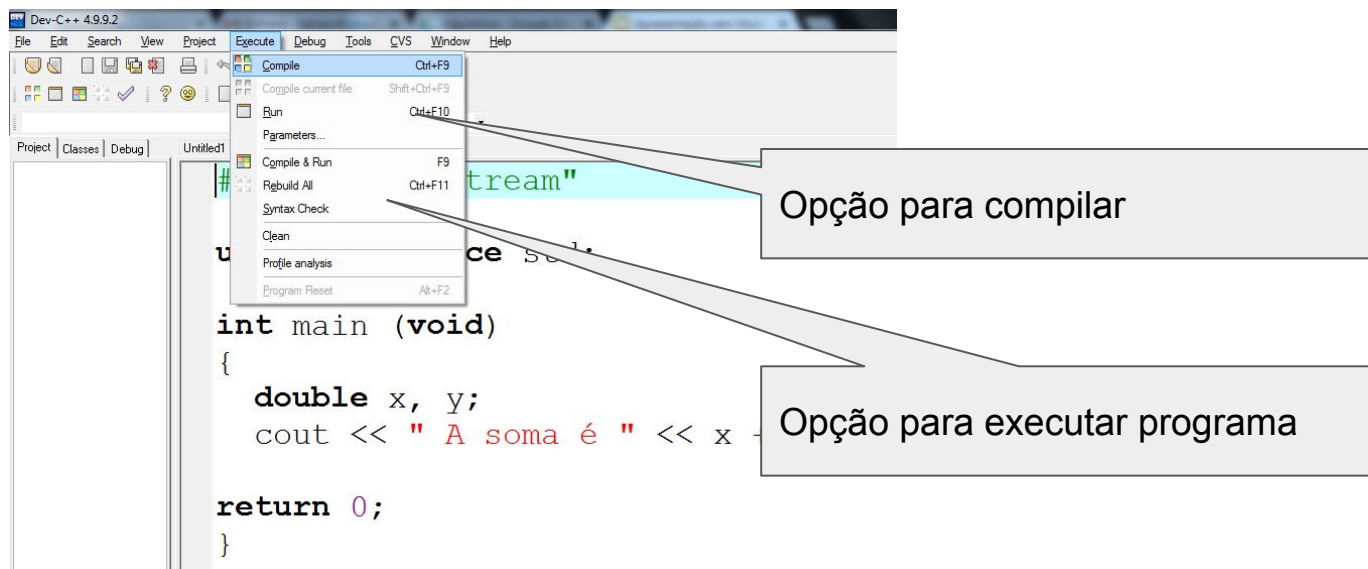
Salvando o código fonte no Devcpp

3) Escolher uma pasta qualquer, por exemplo, a área de trabalho, criar a pasta EXERCÍCIOS, selecionar a pasta EXERCÍCIOS e digitar um nome qualquer para o programa e selecionar o botão salvar:



Digitando e compilando o código fonte no Devcpp

4) Através do menu executar, escolha a opção **compilar (compile)** para verificar os erros, quando não haver mais erros, através do menu executar escolha a opção **executar (run)**:



APRENDIZAGEM: Programa 1

```
#include "iostream" // os comandos cout << e cin >>
#include "math.h" // operadores e funções matemáticas
#include "cstdlib" // comandos system("cls") e
system("pause")
using namespace std; // inibe o uso de std:: antes de cada
comando

int main()
{ system("cls"); // apaga a tela preta
  setlocale(LC_ALL, "Portuguese"); // configura idioma

  // O comando cout imprime mensagens na tela e \n pula linhas
  cout << "\nOlá, tudo bem?\n\nPara aprender a programar, você";
  cout << "\ndeverá estudar em casa também.\n\n\n";
  system("pause"); // faz uma pausa no processamento
}
```



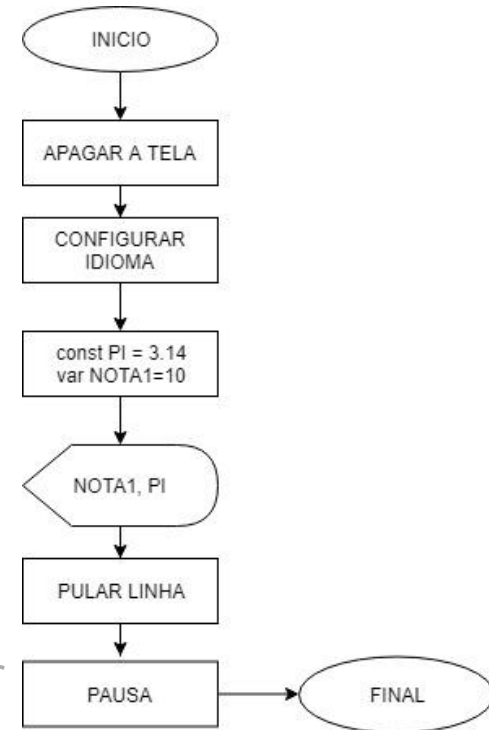
APRENDIZAGEM: Programa 2

```
#include "iostream"  
#include "math.h"  
#include "cstdlib"  
using namespace std;
```

```
int main() { system("cls");  
    setlocale(LC_ALL, "Portuguese");
```

```
double const pi= 3.14; // cria uma constante imutável  
double nota1 = 10; // cria uma variável e atribui o valor
```

```
cout << "\nO valor da nota 1 é " << nota1; // mostra  
a nota1  
cout << "\nMostra o valor do pi = " << pi ; // mostra a  
nota3  
cout << endl; // manda o cursor para linha de baixo  
system("pause"); }
```



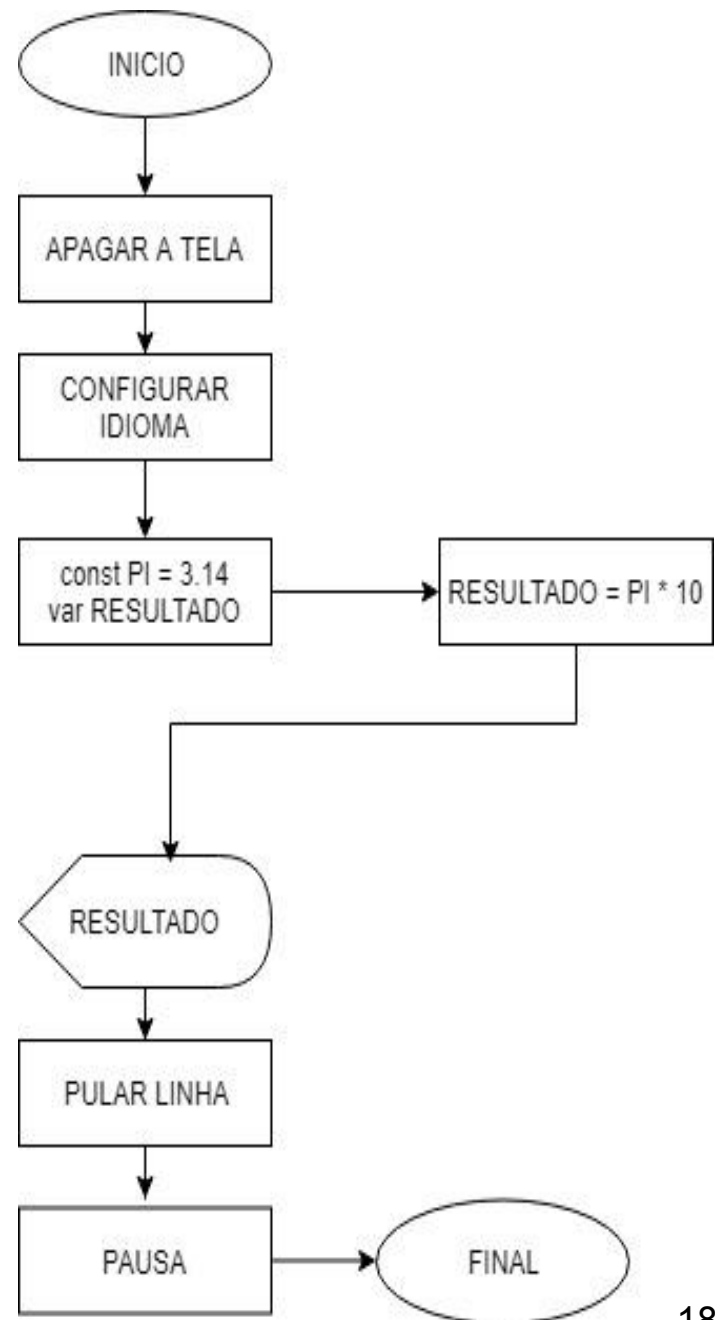
APRENDIZAGEM: Programa 3

```
#include "iostream"
#include "math.h"
#include "cstdlib"
#define pi 3.14 // cria constante pi com #define
using namespace std;
```

```
int main() {
    system("cls");
    setlocale(LC_ALL, "Portuguese");
```

```
// double const pi = 3.14;
double resultado; // cria a variável resultado
resultado = pi * 10; // faz o cálculo
cout << "\nO resultado será :" << resultado << endl;
```

```
system("pause"); }
```



APRENDIZAGEM: Programa 4

```
#include "iostream"
#include "math.h"
#include "cstdlib"
using namespace std;

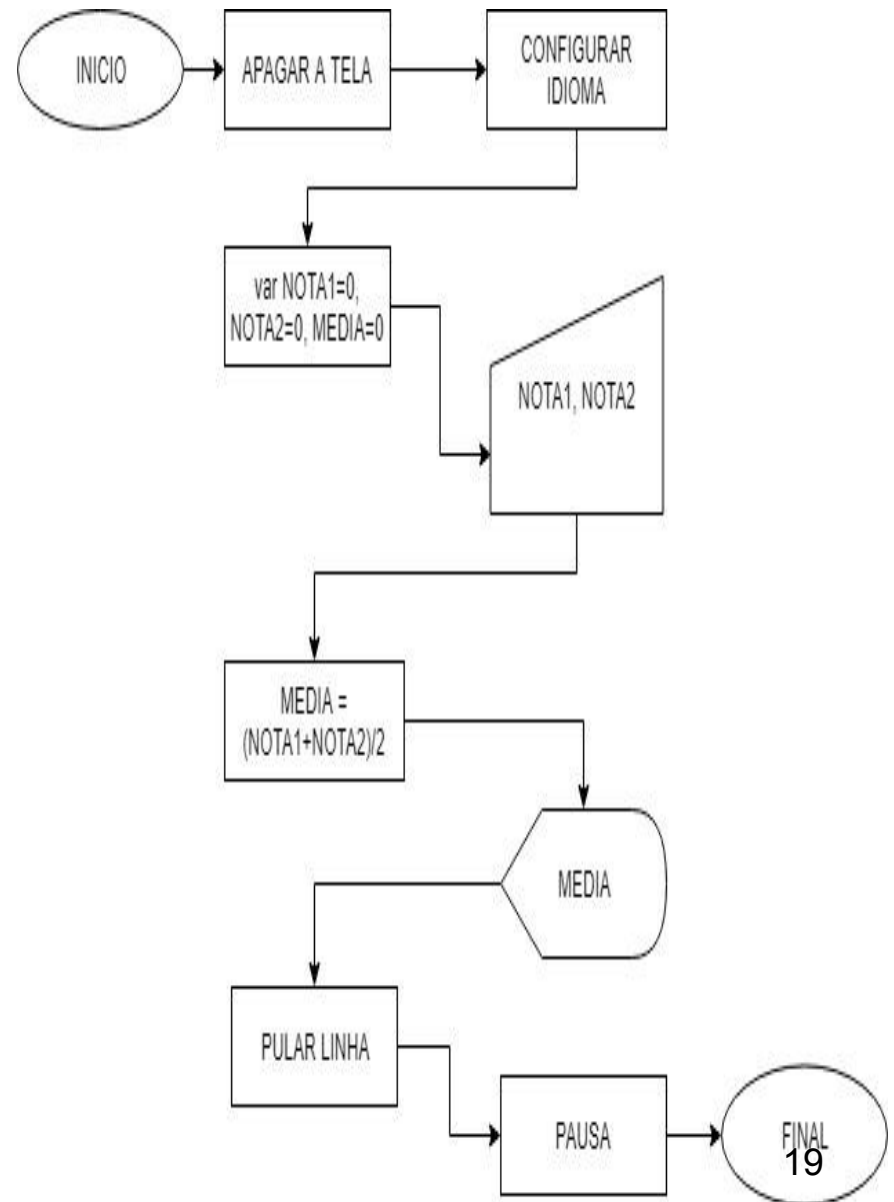
int main()    { system("cls");
               setlocale(LC_ALL, "Portuguese");

               double nota1=0, nota2=0, media=0;

               cout << "\nDigite a nota 1";
               cin >> nota1;

               cout << "\nDigite as nota 2";
               cin >> nota2;

               media = (nota1 + nota2)/2;
               cout << "\nO resultado será :" << media;
               cout << endl << endl;
               system("pause"); }
```



Lista 2 - EXERCÍCIOS DE FIXAÇÃO

- a) Fazer o código fonte dos enunciados A, C, D, E e H do exercício (7) das páginas 25 e 26 do livro: Estudo Dirigido de Algoritmos Manzano.

NOTA: Os códigos deverão ser feitos no computador, impressos, recortados e colados no caderno contendo o número da aula e a data da lista.