INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Passo Fundo

Ciência da
Computação

ALGORITMOS I

Professores:

Adilso Nunes de Souza

Maikon Cismoski dos Santos

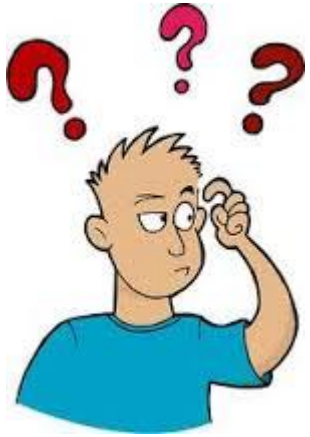
ROTEIRO DA AULA

- Teste de mesa
- Histórico da linguagem C++
- Características
- Etapas de um programa
- Linguagens compiladas e interpretadas
- Tipos de dados
- Comentários
- Constantes
- Regras básicas

TESTE DE MESA

- Um algoritmo só é correto se produz o resultado esperado para qualquer entrada possível.[MEDINA, 2006]

- Como verificar se o algoritmo está correto?



TESTE DE MESA

- É o meio pelo qual podemos acompanhar a execução de um algoritmo, passo a passo, ou instrução a instrução.[MEDINA, 2006]
- Simular manualmente a execução do algoritmo, atentando para as entradas, processamento e saídas de dados, acompanhando o comportamento das variáveis que foram utilizadas.[VILARIM, 2004]

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

TESTE DE MESA

- Conhecido como “método Chinês”, onde o desenvolvedor coloca-se no lugar do computador, acompanhando o que a máquina faria ao encontrar cada instrução.
- Aprender a verificar se o algoritmo leva a um resultado esperado através da simulação de valores.[PIMENTEL, 2005]
- Possibilita compreende a lógica utilizada e encontrar possíveis erros existentes.

TESTE DE MESA

- Como realizar um teste de mesa:
  - Providencie papel e caneta/lápis.
  - Identifique as variáveis envolvidas.
  - Crie uma tabela com linhas e colunas.
  - Inicie a leitura de cada instrução de cima para baixo e da esquerda para a direita, anotando os resultados na tabela.

TESTE DE MESA

- Na primeira linha identifique as variáveis
- Na primeira coluna escreva a linha da instrução
- Nas demais colunas insira os valores das variáveis na linha em questão.

LINHA	A	B	C

TESTE DE MESA

- Se o valor foi lido coloque-o entre parênteses.
- Se o valor foi escrito coloque-o entre chaves.
- Se o valor foi resultado de uma operação somente escreva seu valor.
- Quando uma variável não possui valor em uma determinada linha coloque a interrogação “?”.

TESTE DE MESA

```
1  algoritmo "soma"
2
3  var
4     a, b, c: inteiro
5
6  inicio
7     escreva ("Informe o valor A: ")
8     leia(a)
9     escreva ("Informe o valor B: ")
10    leia(b)
11    c <- a + b
12    escreval("Resultado: ", c)
13 fimalgoritmo
```

LINHA	A	B	C
4	?	?	?
8	(5)	?	?
10		(8)	?
11			13
12			{13}

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

TESTE DE MESA

```
1  algoritmo "troca"
2
3  var
4     a, b: inteiro
5
6  inicio
7     escreva ("Informe o valor A: ")
8     leia(a)
9     escreva ("Informe o valor B: ")
10    leia(b)
11    a <- a
12    b <- a
13    escreval("Mostre: ", a)
14    escreval("Mostre: ", b)
15 fimalgoritmo
```

TESTE DE MESA

LINHA	A	B
4	?	?
8	(2)	?
10		(7)
11	7	
12		7
13	{7}	
14		{7}

TESTE DE MESA

```
1  algoritmo "troca"
2
3  var
4     a, b, aux: inteiro
5
6  inicio
7     escreva ("Informe o valor A: ")
8     leia(a)
9     escreva ("Informe o valor B: ")
10    leia(b)
11    aux <- a
12    a <- b
13    b <- aux
14    escreval("Mostre a: ", a)
15    escreval("Mostre b: ", b)
16 fimalgoritmo
```

TESTE DE MESA

- Existem várias metodologias para fazer os testes de mesa.
- Alguns métodos são simplificados focando apenas nas modificações das variáveis sem detalhes das linhas ou mesmo o histórico dos valores recebidos, porém isso dificulta a reconstituição do passo a passo.
- Testes de mesa são essenciais para algoritmos complexos.

HISTÓRICO

- Dennis M. Ritchie e Ken Thompson, laboratório Bell em 1972
- Evolução da linguagem B de Thompson que evoluiu da linguagem BCPL
- A linguagem C++ é baseada na linguagem C

HISTÓRICO

- C++ representa uma evolução e refinamento de algumas das melhores características das linguagens anteriores

- A linguagem C++ foi criada em 1980 por Bjarne Stroustrup do laboratório Bell

CARACTERÍSTICAS

- Não é um ambiente de programação
- Não é uma linguagem visual
- Linguagem de nível médio
  - Combina elementos das linguagens de alto nível com as funcionalidade da linguagem assembly
  - Relativamente simples
  - Programação estruturada e modular

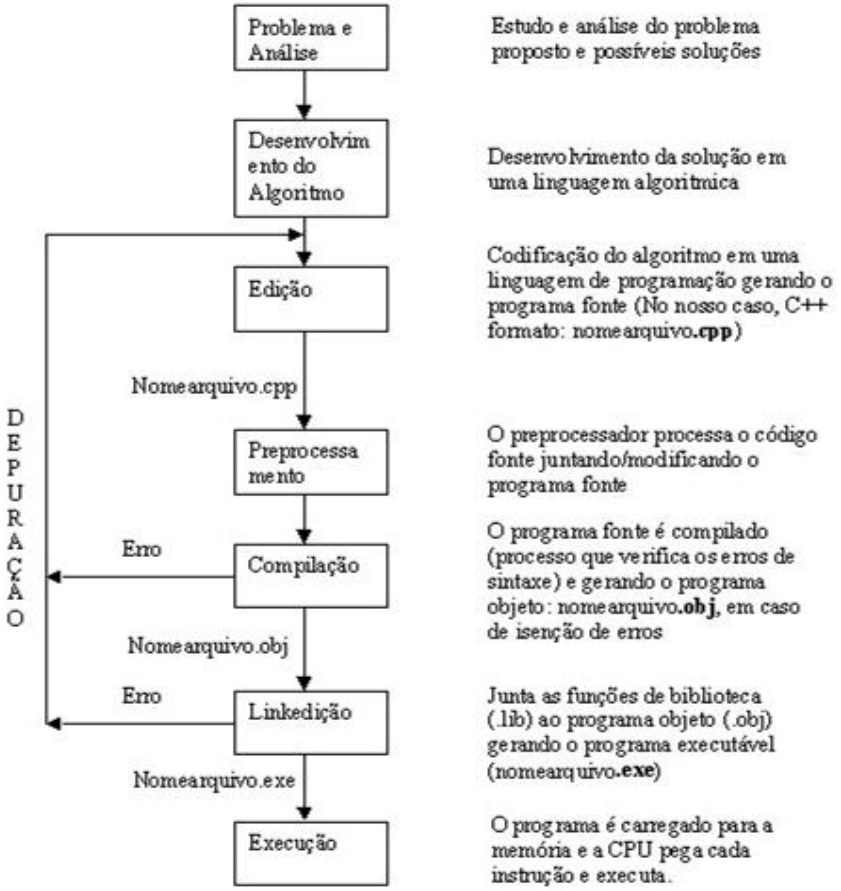
CARACTERÍSTICAS

- Híbrida (procedural e OO)
- Portabilidade
  - Programas em C e C++ podem ser compilados e executados sem alterações em diferentes plataformas
- Eficiência e economia
  - Programas são executados em menos tempo e ocupam pouco espaço de armazenamento

CARACTERÍSTICAS

- Variáveis Case Sensitive
- Variedade de softwares para a linguagem C++
- Coleção de funções e bibliotecas padronizadas nas implementações da linguagem C++

ETAPAS


Problema e Análise
Estudo e análise do problema proposto e possíveis soluções
Desenvolvimento do Algoritmo
Desenvolvimento da solução em uma linguagem algoritmica
Edição
Codificação do algoritmo em uma linguagem de programação gerando o programa fonte (No nosso caso, C++ formato: nomearquivo.cpp)
Nomearquivo.cpp
Preprocessamento
O preprocessador processa o código fonte juntando/modificando o programa fonte
Erro
Compilação
O programa fonte é compilado (processo que verifica os erros de sintaxe) e gerando o programa objeto: nomearquivo.obj, em caso de isenção de erros
Nomearquivo.obj
Erro
Linkedição
Junta as funções de biblioteca (.lib) ao programa objeto (.obj) gerando o programa executável (nomearquivo.exe)
Nomearquivo.exe
Execução
O programa é carregado para a memória e a CPU pega cada instrução e executa.
DEPURAÇÃO

LINGUAGEM

- Uma linguagem de programação é o elo entre a linguagem que os seres humanos entendem e as instruções que uma máquina deve executar para realizar uma determinada ação.

COMPILADORES X INTERPRETADORES

- Um programa é a maneira de se comunicar com um computador e a única linguagem que o computador entende é a linguagem de máquina. Por isso todos os programas que se comunicam com o computador devem estar em linguagem de máquina.
- A forma como os programas são traduzidos para a linguagem de máquina é classificada em duas categorias: interpretados e compilados

INTERPRETADOR

- Lê a primeira instrução do programa, faz a consistência de sua sintaxe e se não houver erro converte-a para linguagem de máquina para depois executá-la
- Segue para a próxima instrução repetindo o processo até a execução da última instrução
- O interpretador precisa estar presente todas as vezes que vamos executar o programa, portanto o trabalho de checagem da sintaxe e tradução é repetitivo

COMPILADOR

- O compilador lê todas as instruções e faz a consistência da sintaxe se não houver erro converte para linguagem de máquina gerando um arquivo com o sufixo .OBJ com as instruções já traduzidas.
- Na sequência são agregadas ao arquivo rotinas em linguagem de máquina que lhe permitirão a execução, isto é feito pelo linkeditor, que além de juntar as rotinas cria o produto final ou arquivo executável.

COMPARATIVOS

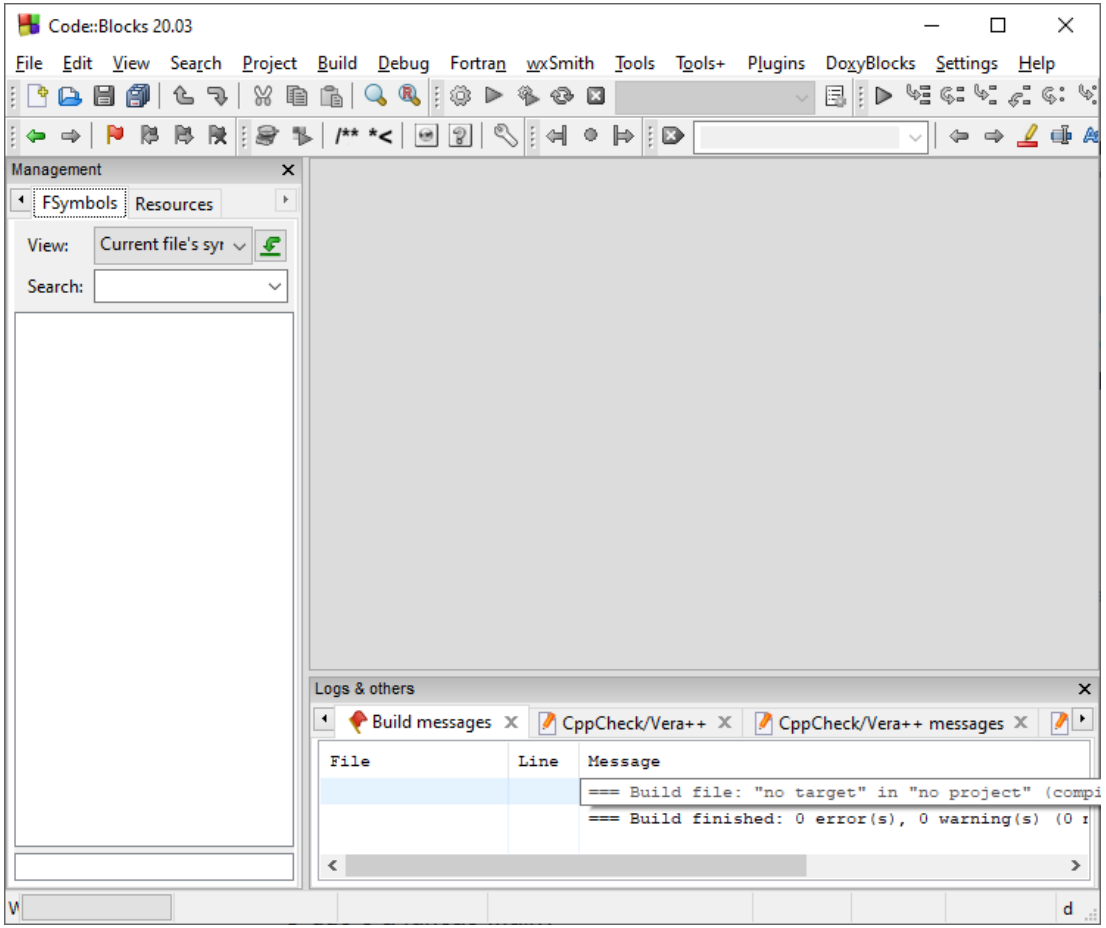
- Velocidade de execução do programa compilado chega a ser 20 vezes mais rápida do que o programa interpretado
- O programa compilado e linkeditado pode ser executado diretamente sobre o sistema operacional não necessitando de nenhum outro software
- Programas .exe não podem ser alterados, o que protege o código fonte

FERRAMENTAS

- Muitos dos compiladores C++ estão disponíveis em Ambientes de Desenvolvimento Integrados (IDEs), esses ambientes facilitam e agilizam o processo de desenvolvimento de programas, pois são compostos por um editor, depurador, compilador e ainda permitem a execução do arquivo final.
- Existem vários softwares com esta finalidade, entre eles: Dev-C++, Visual C++, CodeBlocks, Netbeans, etc...

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

CodeBlocks



CodeBlocks

- Para desenvolver nossos programas, durante nossas aulas, utilizaremos o CodeBlocks por ser de fácil instalação, não necessitar de muitos recursos de hardware, ser opensource e disponibilizar as ferramentas necessárias.
- Download:

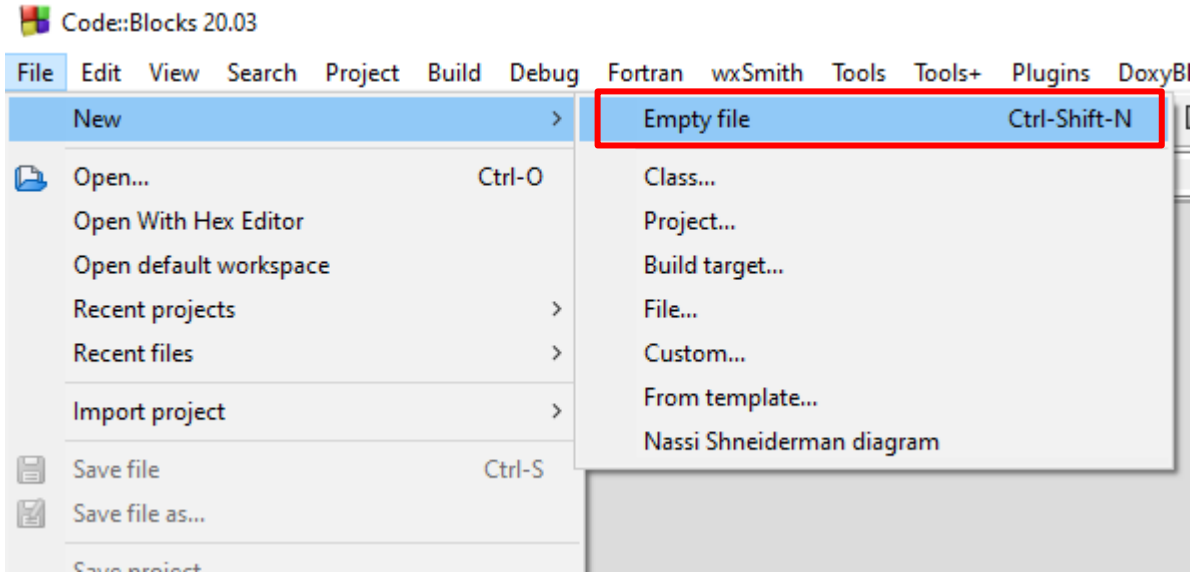
  http://www.codeblocks.org

CodeBlocks

- Principais comandos utilizados:
  - Ctrl+S = Salvar
  - Ctrl+F9 = Compilar
  - Ctrl+F10 = Executar
  - F9 = Compilar e Executar

CodeBlocks

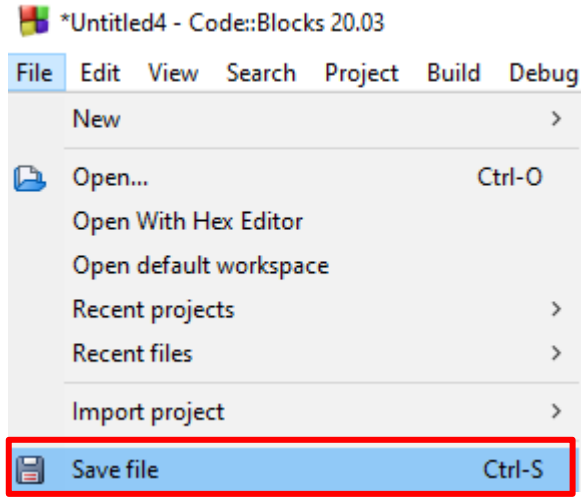
- Criar novo arquivo:

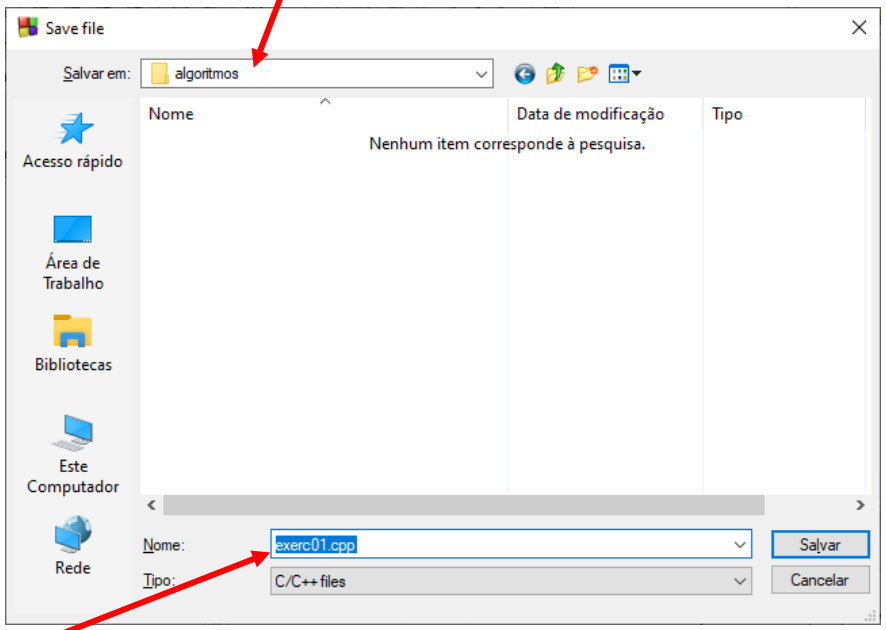


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

CodeBlocks

- Salvar arquivo:



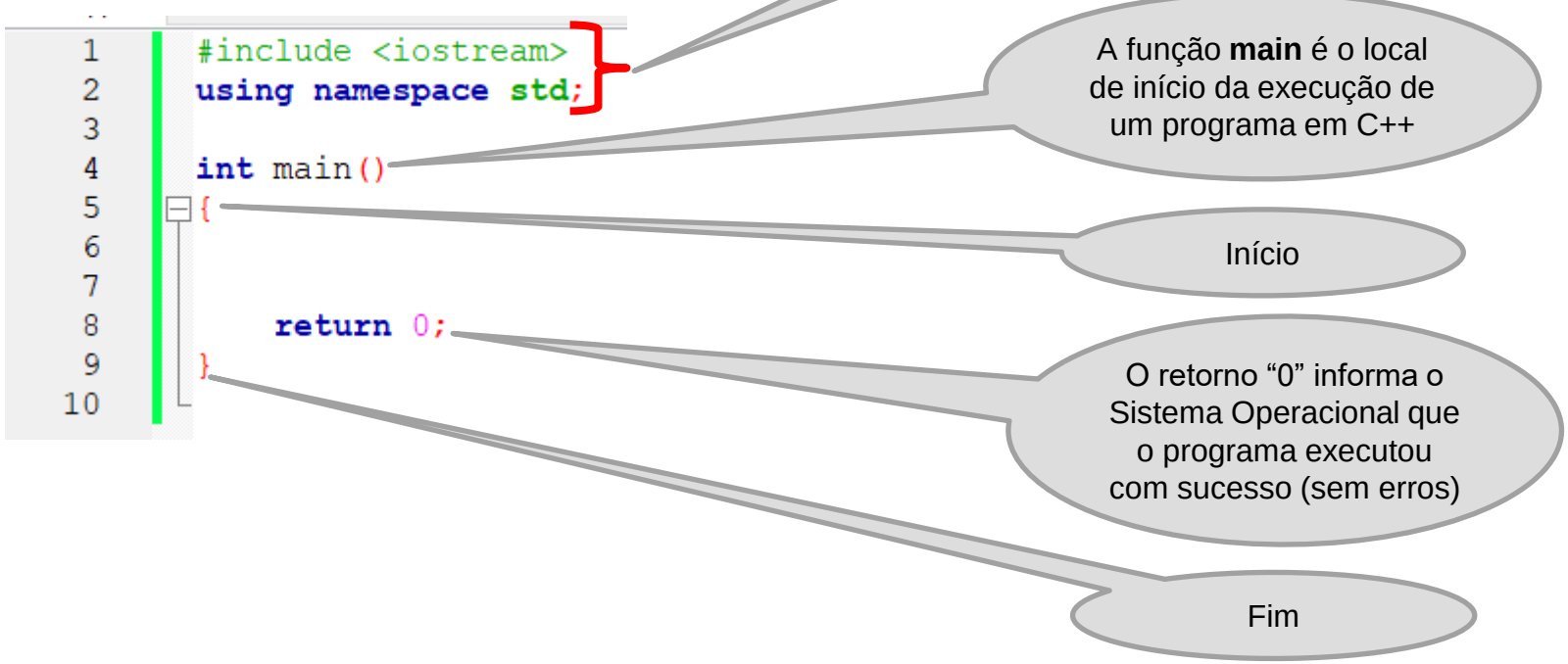


Defina um diretório para salvar os algoritmos

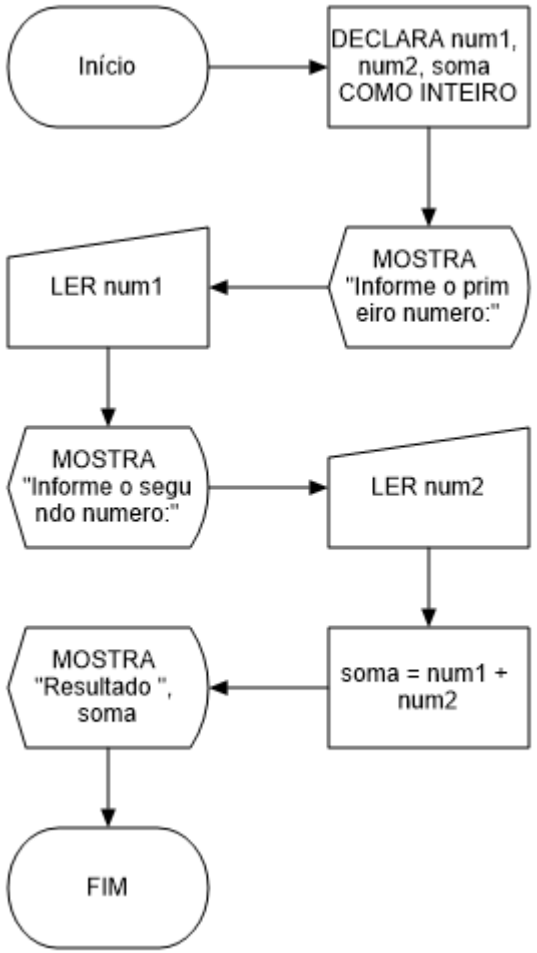
Escreva o nome do arquivo. A extensão do arquivo deve ser **.CPP.**

EXEMPLOS

- Estrutura básica de um programa em C++:



EXEMPLOS




```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num1, num2, soma;
7
8      cout << "Informe primeiro numero: ";
9      cin >> num1;
10
11     cout << "Informe o segundo numero: ";
12     cin >> num2;
13
14     soma = num1 + num2;
15
16     cout << "Resultado: " << soma << endl;
17
18     return 0;
19 }
```

EXEMPLOS

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num1, num2, soma;
7
8      cout << "Informe primeiro numero: ";
9      cin >> num1;
10
11     cout << "Informe o segundo numero: ";
12     cin >> num2;
13
14     soma = num1 + num2;
15
16     cout << "Resultado: " << soma << endl;
17
18     return 0;
19 }
```

Declaração de variáveis do tipo inteiro.

Mostra uma cadeia de caracteres na tela do computador

Ler um valor para a variável

Expressão e operador aritmético

Mostra o valor da variável

Quebra de linha

Instrução

- Possui uma instrução por linha;
- Utiliza “;” como finalizador de instrução.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num1, num2, soma;
7
8      cout << "Informe primeiro numero: ";
9      cin >> num1;
10
11     cout << "Informe o segundo numero: ";
12     cin >> num2;
13
14     soma = num1 + num2;
15
16     cout << "Resultado: " << soma << endl;
17
18     return 0;
19 }
```

Variável

- Sintaxe para declarar uma variável na linguagem C++:

  - ```
    int x, y;
    float n1, n2, res;
    char nome[100];
    int mat[5][5];
    bool sinal;
    ```
    -de-variáveis>

  - <tipo-de-dado> <variável = valor inicial>

    ```
    int i, cont = 0;
    float salario = 1212.00;
    ```
    ```

COMANDOS DE ENTRADA

- Comando de entrada de dados linguagem C++:

  cin >> <lista-de-variáveis>
- Este comando recebe os valores digitados pelos usuários, atribuindo-os às variáveis cujos nomes estão em <lista-de-variáveis> respeitando a ordem especificada nesta lista.
- Exemplo:

  cin >> x >> y;

COMANDOS DE SAÍDA

- Comando de Saída de dados:
  - ➢ cout << <lista-de-expressões>
- Este comando escreve o conteúdo de cada uma das expressões que compõem a <lista-de-expressões>.
- Exemplo:

```
cout << "Resultado: " << soma;
```

COMANDOS DE SAÍDA

- A quebra de linha pode ser inserida com o uso da palavra reservada ***endl*** ou pelos caracteres "\n".
- Exemplo:

```
cout << "Resultado: " << soma << endl;
cout << "Resultado: " << soma << "\n";
cout << "\nResultado da soma:" << soma << endl;
```

- Exemplo:

```
cout << "\nResultado: " << soma << "\n\n";
cout << endl << "Resultado: " << soma << endl << endl;
```

COMANDOS DE SAÍDA

- Para mostrar caracteres com acentos e cedilha, chame a função *setlocale* no início do código, conforme o exemplo abaixo:

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       setlocale(LC_ALL, "Portuguese");
7
8       int num;
9
10      cout << "Infome um número inteiro: ";
11      cin >> num;
12
13      cout << "O número informado é: " << num << endl;
14
15      return 0;
16  }
17
```

COMENTÁRIOS

- Podem aparecer em qualquer lugar do programa e são desconsiderados pelo compilador. A linguagem C aceita os seguintes tipos de comentários:

  /*comentário de bloco*/

  //comentário de linha

- Não é permitido utilizar um comentário dentro do outro.

TIPOS DE DADOS

- Toda variável deve ser declarada e ao ser declarada deve-se atribuir um tipo. O tipo determina como valores de dados são representados, que valores pode expressar, e que tipo de operações se pode executar com estas variáveis. Os principais tipos aceitos pela linguagem são:
  - char = caracter;
  - int = números inteiros;
  - float = ponto flutuante em precisão simples;
  - double = ponto flutuante em dupla precisão;

CONSTANTE

- Tudo que é invariável.
- Valores que permanecem sem alteração durante toda a vida do programa.
- EX:

```
#define PI 3.1415
const float teste=12.5;
```

OPERADORES

- Atribuição
  - Sinal de igual =
- Aritméticos
  - Multiplicação *
  - Divisão /
  - Adição +
  - Subtração –
  - Resto da divisão inteira %

RETORNO DA FUNÇÃO MAIN

- A função *main* retorna um inteiro para informar o Sistema Operacional sobre o fim da execução do programa:
  - Retornar 0 para informar que o programa terminou com sucesso (não ocorreram erros durante a execução).
  - Retornar um número diferente de zero para informar que o programa terminou de maneira excepcional (erros ocorreram durante a execução).

RETORNO DA FUNÇÃO MAIN

- Tradicionalmente, usam-se as constantes EXIT_SUCCESS (que vale 0) ou EXIT_FAILURE (que vale 1) como valor retornado pela função *main*. Exemplo:

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6
7
8       return EXIT_SUCCESS; //EXIT_SUCCESS vale 0
9   }
10
```

FÓRMULA DO SUCESSO

- Não existe receita pronta
- Programar é dedicação, persistência, errar, acertar, mas o mais importante é não desistir
- Conhecimento dos comandos da linguagem

REGRAS BÁSICAS

- Usar SEMPRE comentários em seu código fonte;
- Usar INDENTAÇÃO de seus fontes. (padrão min 3 espaços); O padrão de indentação deve ser o mesmo em todo o programa e deve ser inserido na digitação do programa e não após ele já estar pronto;
- Nunca tentar redefinir palavras reservadas de uma linguagem;

REGRAS BÁSICAS

- Em expressões matemáticas, não existem {} (Chaves), [] (Colchetes). Todos os níveis de precedência devem ser construídos com () (Parênteses);
- Todas as regras matemáticas são respeitadas pelo computador;
- O 0 (zero) é totalmente diferente de nada -> 0 <> Nada;

REGRAS BÁSICAS

- Devemos sempre que possível aproveitar ao máximo as linhas de código e as variáveis já existentes;
- Uma linha de comando termina sempre com um ponto-e-vírgula (;)
- Podemos ter dois comandos ou mais em uma mesma linha, desde que os mesmos estejam separados por ponto-e-vírgula (;)

REGRAS BÁSICAS

- Recomenda-se quando na codificação dos algoritmos no computador o uso de letras minúsculas (facilidade de leitura), reservando as maiúsculas para algum tipo de destaque ou para constantes.
- As regras para nomes de identificadores (nomes de variáveis, nomes de funções) válidos não devem possuir espaços em branco. Recomenda-se o uso de letras ao inicio dos identificadores.

REGRAS BÁSICAS

- Os nomes de quaisquer identificadores (variáveis, nome de funções) não podem em hipótese nenhuma ser repetidos.
- Após o término de cada área do algoritmo ou após cada função é aconselhável deixarmos uma linha em branco para uma melhor organização.
- **O computador detecta somente erros de sintaxe, nunca erros de lógica.**

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

REFERÊNCIAS

- MEDINA, Marco, FERTIG, Cristina. Algoritmos e programação: teoria e prática. Novatec Editora. São Paulo, 2006
- VILARIN, Gilvan. Algoritmos Programação para Iniciantes. Editora Ciência Moderna. Rio de Janeiro, 2004.
- MORAES, Paulo Sérgio. Curso Básico de Lógica de Programação. Centro de Computação – Unicamp, 2000.
- STEINMETZ, Ernesto H. R.; FONTES, Roberto Duarte Cartilha Lógica de Programação. Editora IFB, Brasília - DF, 2013.