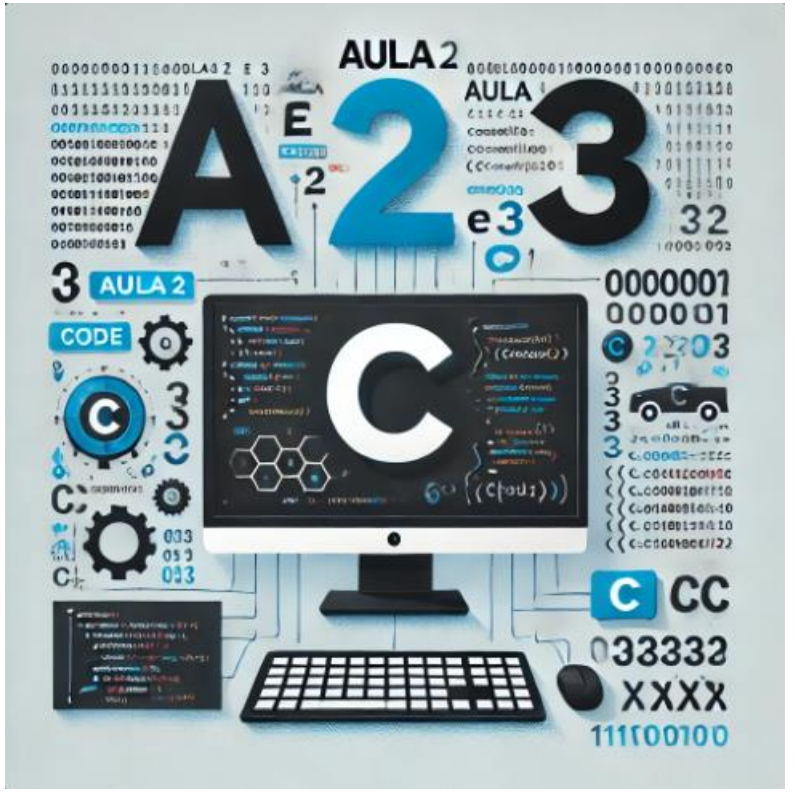
ESCOLA SECUNDÁRIA DE SANTO ANDRÉ

CURSO PROFISSIONAL TÉCNICO DE INFORMÁTICA - SISTEMAS

AULA 2 e 3

Conteúdos Específicos

- Conceito de Função
- Nomenclatura de uma Função
- Estrutura de uma Função
- A instrução Return
- Função Void
- Argumentos de uma Função
- Exemplos práticos



ESCOLA SECUNDÁRIA DE SANTO ANDRÉ

CURSO PROFISSIONAL TÉCNICO DE INFORMÁTICA - SISTEMAS

Sumário da aula 2 e 3

- Conceito de função e diferenças entre subprogramas e funções.
- Tipos de funções:
  - Com retorno de valor.
  - Do tipo void (sem retorno de valor).
- Nomenclatura e características das funções.
- Argumentos de uma função.
- Resolução de exercícios práticos conduzidos.

SUBPROGRAMA vs FUNÇÃO

Diferenças entre SUBPROGRAMA E FUNÇÃO

- Um **subprograma** é **um bloco de código independente**, ou seja fora da **função main()** e que **executa uma tarefa específica** como por exemplo calcular uma soma.

- Uma função é um tipo especifico de subprograma que retorna um valor - **Exemplo 1.**

- Em C os **subprogramas** são **funções** que retornam um valor (return) ou não retornam valor através da função (void) - **Exemplo 2.**

Exemplo 1

```
4  int soma(int a, int b) {
5      return a + b;
6  }
```

Exemplo 2

```
4  void saudacao()
5  {
6      printf("Olá! Bem-vindo!\n");
7  }
```

TIPOS DE SUBPROGRAMAS

Funções com Retorno de valor:

- **São funções** que executam uma tarefa e **retornam um valor** ao programa que as chamou.

- O **tipo de retorno** pode ser um **número inteiro**, **ponto flutuante**, **caracter**, entre outros.

- São usadas quando é necessário calcular ou obter um resultado que será usado posteriormente.

- Para serem executadas têm de ser chamadas.

Soma.cpp

```
1   #include <stdio.h>
2
3   int soma(int a, int b)
4   {
5       return a + b;
6   }
7
8   int main()
9   {
10      int resultado = soma(5, 3);
11      printf("Resultado da soma: %d\n", resultado);
12      return 0;
13  }
14
```

TIPOS DE SUBPROGRAMAS

Funções Void sem Retorno de valor:

- **São funções** que executam uma tarefa, mas **não retornam nenhum valor.**

- São utilizadas para:

  - Exibição de mensagens;
  - Alteração de variáveis;
  - Validações.

[*] Soma.cpp | Void.cpp

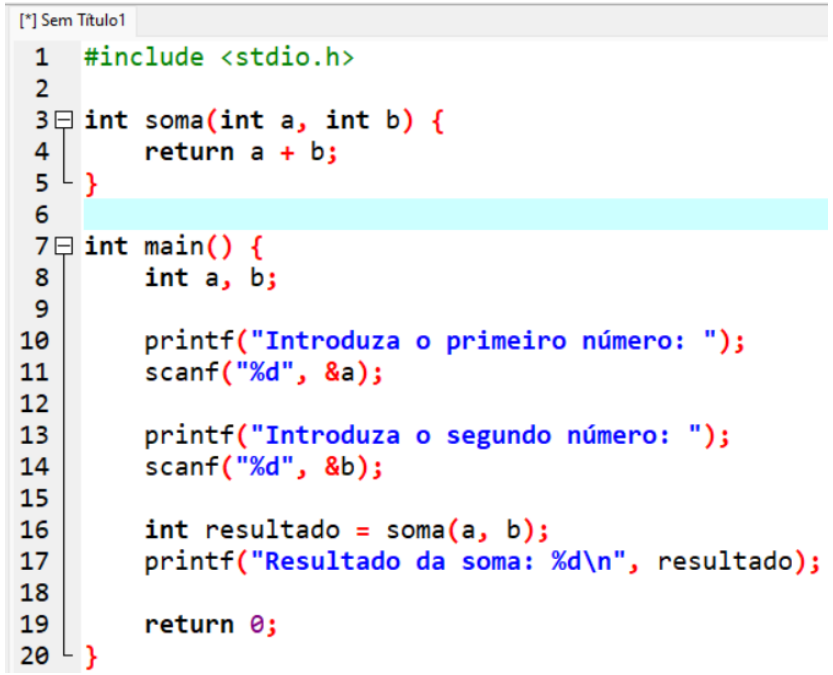
```
1   #include <stdio.h>
2
3   void exibirMensagem()
4   {
5       printf("Ola, mundo!\n");
6   }
7
8   int main()
9   {
10      exibirMensagem();
11      return 0;
12  }
```

RELEMBRANDO

Embora ainda sem saber como escrever uma função, já as temos utilizado ao longo dos nossos programas.

Exemplos:

- main() - Função principal do programa.
- printf(…) – Função de **escrita associada à** biblioteca **stdio.h – (Output).**
- scanf(…) – Função de leitura (**também associada à biblioteca stdio.h**) e armazenamento – **(Input).**



NOMENCLATURA DE FUNÇÕES

Nome de uma função:

- Deve seguir as mesmas regras das variáveis;
- Ser único no programa;
- Fácil leitura.

NOMENCLATURA DE FUNÇÕES

Relembrando as regras:

- O nome de uma variável/função pode ser constituído por letras do alfabeto (minúsculas, maiúsculas), dígitos e ainda pelo carácter underscore.
- O primeiro carácter não pode ser um digito. Terá de ser uma letra ou o carácter underscore.
- É desaconselhável a utilização do underscore para inicio da variável/função.
- Uma variável/função não pode nunca ter um nome que esteja reservado à própria linguagem C como por exemplo: float, if, for etc….

CARACTERISTICAS DE UMA FUNÇÃO

- Cada função precisa de ter um nome único para ser invocada no programa.
- Uma função pode ser chamada a partir de outras funções.
- A função deve realizar **uma única tarefa bem definida**.
- Deve comportar-se como uma "caixa negra": **o importante é o resultado**, não o processo interno.
- O código da função deve ser **o mais genérico possível** para ser reutilizável.
- Pode receber **argumentos** para adaptar-se a diferentes situações.
- Pode **retornar um valor** como **resultado** do seu processamento.

PARÂMETROS E ARGUMENTOS

Características:

- **Argumento:** Variável que serve de cálculo à função.
- **Passagem de Argumentos:** Os argumentos são passados para a função entre parênteses e separados por vírgulas.

  Exemplo : soma(10, 20).

ARGUMENTOS DE UMA FUNÇÃO

O que acontece neste exemplo?

Há retorno de valor?

Ou há impressão de um resultado?

```c
void saudacao(char nome[], int idade)
{
    printf("Olá, %s tens %d anos.\n", nome, idade);
}
int main()
{
    saudacao("João", 25);
    saudacao("Maria", 30);

    return 0;
}
```

ARGUMENTOS DE UMA FUNÇÃO

O que acontece neste exemplo?

Há retorno de valor?

Ou há impressão de um resultado?

```c
# include <stdio.h>

int soma(int a, int b)
{
        return a + b;
}

int main()
{
        printf("%d\n", soma(3,5));
        return 0;
}
```

VARIÁVEIS LOCAIS E GLOBAIS

Variável local:

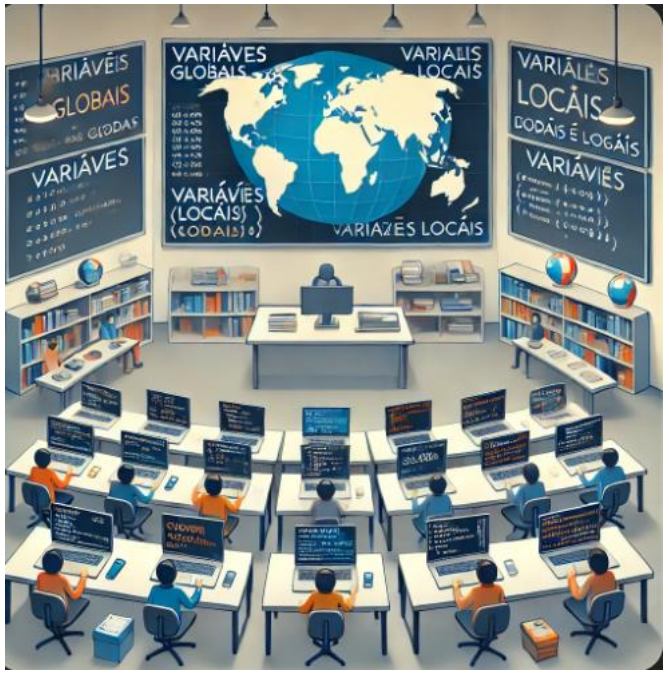
- É definida **dentro da função**.
- A variável é válida e assume o valor atribuído dentro da função.

```c
void funcao() {
    int local = 10; // Variável local
    printf("Variável local: %d\n", local);
}
```

Variável Global:

- É definida **fora da função**.
- A variável é válida e assume valor atribuído em todo o programa.

```c

int global = 20; // Variável global

void funcao() {
    printf("Variável global na função: %d\n", global);
}
```



NOMENCLATURA DE FUNÇÕES

Exercício prático conduzido:

1. Função que permite calcular a soma de dois números inteiros.

2. Função sem retorno que imprime uma mensagem de boas-vindas.

3. Função que permite verificar se um número é par ou ímpar.

1

```
[*] Sem Título1
1
2   int soma(int a, int b)
3   {
4       return a + b;
5   }
```

2

```
1
2   void mensagemBoasVindas()
3   {
4       printf("Bem-vindo ao programa de exemplos de funções!\n");
5   }
```

3

```
[*] Sem Título1
1   void verificaParImpar(int numero)
2   {
3       if (numero % 2 == 0)
4           printf("%d é par.\n", numero);
5       else
6           printf("%d é ímpar.\n", numero);
7
8   }
```

PARÂMETROS E ARGUMENTOS

Exercício explicativo:

- Escreva um programa em C que utilize uma função que receba dois argumentos para calcular a área do retângulo e retornar o resultado.
- Identificação das variáveis.
- Escreve as funções necessárias da biblioteca de funções do C para que o utilizador possa na consola calcular os valores pretendidos.

[*] Sem Título1

```c
#include <stdio.h>

float calculaAreaRetangulo(float largura, float altura)
{
    return largura * altura;
}

int main()
{
    float largura, altura;

    printf("Digite a largura do retângulo: ");
    scanf("%f", &largura);

    printf("Digite a altura do retângulo: ");
    scanf("%f", &altura);


    float area = calculaAreaRetangulo(largura, altura);


    printf("A área do retângulo é: %.2f\n", area);

    return 0;
}
```

FUNCIONAMENTO DE UMA FUNÇÃO

- **Execução sob Invocação:** O código de uma função só é executado quando esta é invocada (chamada) no programa.
- **Suspensão Temporária do Programa Principal:** Quando a função é chamada, a execução do programa principal é "suspensa" temporariamente. A função executa as suas instruções.
- **Retorno ao Ponto de Invocação:** Após a função terminar, o controlo de execução volta ao local no programa onde a função foi invocada.
- **Recebimento de Valores:** A função pode receber valores que influenciam o seu comportamento.
- **Devolução de Resultado:** A função pode retornar ou não um resultado para o programa que a chamou.

VANTAGENS DE UTILIZAR FUNÇÕES

- **Reaproveitamento de Código:** Permite reutilizar código já escrito, seja por si ou por outros programadores.
- **Evita Repetição de Código:** Previne a repetição de blocos de código em várias partes do programa.
- **Facilidade de Alteração:** Ao modificar uma função, a alteração é feita apenas dentro desta, tornando a manutenção mais rápida e eficiente.
- **Organização do Programa:** Mantém os blocos do programa menores e mais organizados, facilitando a compreensão.
- **Leitura Facilitada:** Melhora a legibilidade do código fonte.
- **Separação de Tarefas:** Permite separar o programa em partes independentes, tornando-o mais modular.

CONCEITO DE FUNÇÃO

- Na programação em C, **todos os programas** devem ter a **função main(),** independentemente de outras funções que incluam.

- **A execução de um programa começa sempre na main()**, mesmo que esta não esteja no início do código.

- **Todas as funções só serão executadas se forem chamadas pela main().**

CONCEITO DE FUNÇÃO

- **Uma função é um bloco de código** que pode ser chamado para executar uma tarefa.
- Permite **agrupar instruções** o que facilita a leitura do código.
- Permite **decompor códigos extensos** em pedaços mais pequenos e reutilizáveis.
- Existem funções mais simples e outras mais complexas. As funções podem conter uma instrução ou várias instruções.

O que significa isto?

Sintaxe em C:

```
nomeDaFuncao()
{
        < bloco de instruções ; >
}
```

FUNÇÕES COM INSTRUÇÕES

- Uma função com **uma instrução:**

```
int soma(int a, int b)
{
    return = a + b;
}
```

- Uma função com **mais instruções:**

```
int calculo(int a, int b)
{
    int soma;
    soma = a + b;
    soma = soma + 1;
    return soma;
}
```

ESTRUTURA De UMA FUNÇÃO

- Uma função com várias instruções:

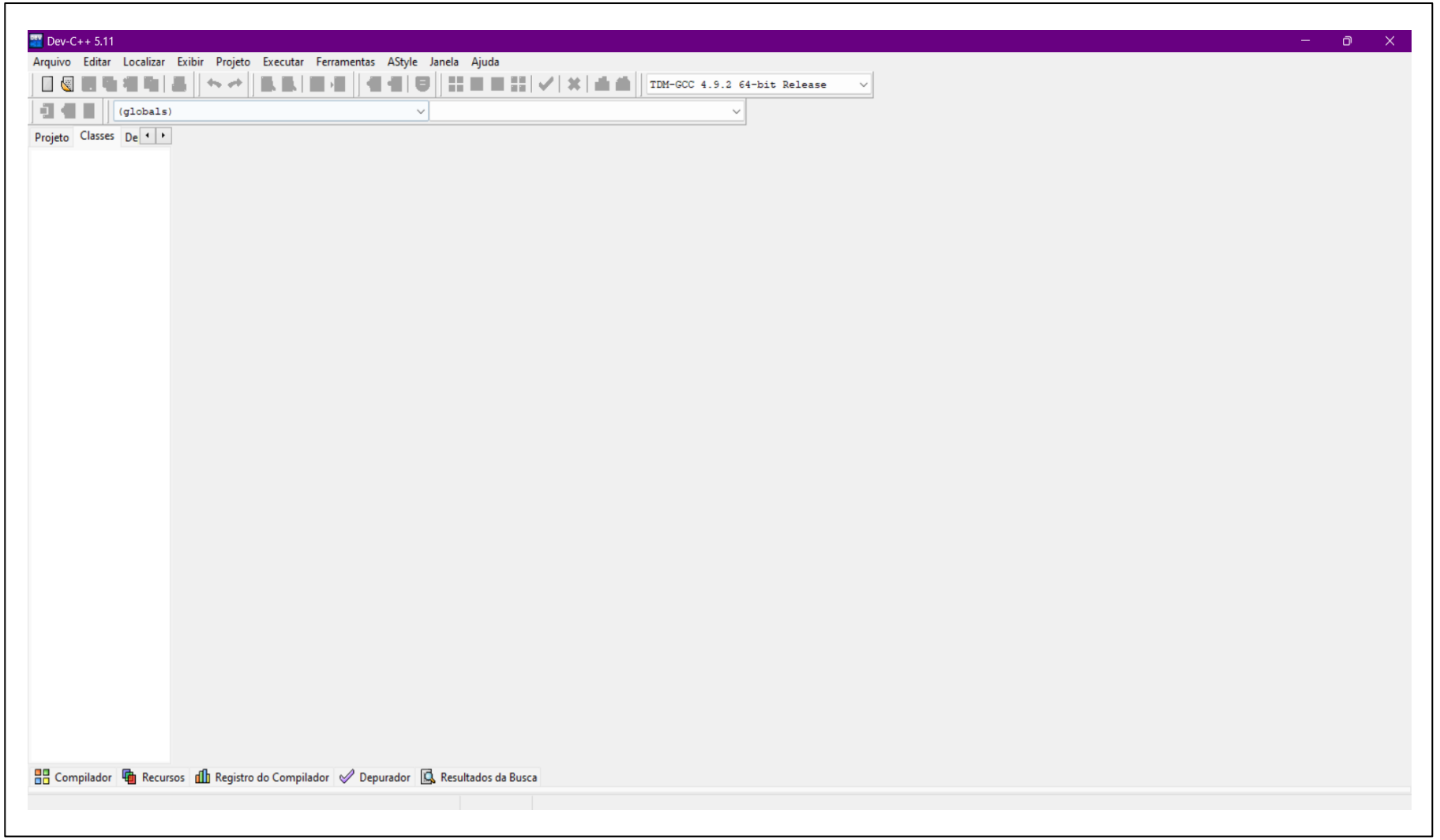
```
int calculo(int a, int b)
{
      int soma;
      soma = a + b;
      soma = soma + 1;
      return soma;
}
```

- Nome da função: calculo
- Cabeçalho da função: int calculo(int a, int b).
- O cabeçalho inclui o tipo de retorno (int), o nome da função (calculo), e os argumentos de entrada (int a, int b).
- As variáveis – São valores que são passados para a função.

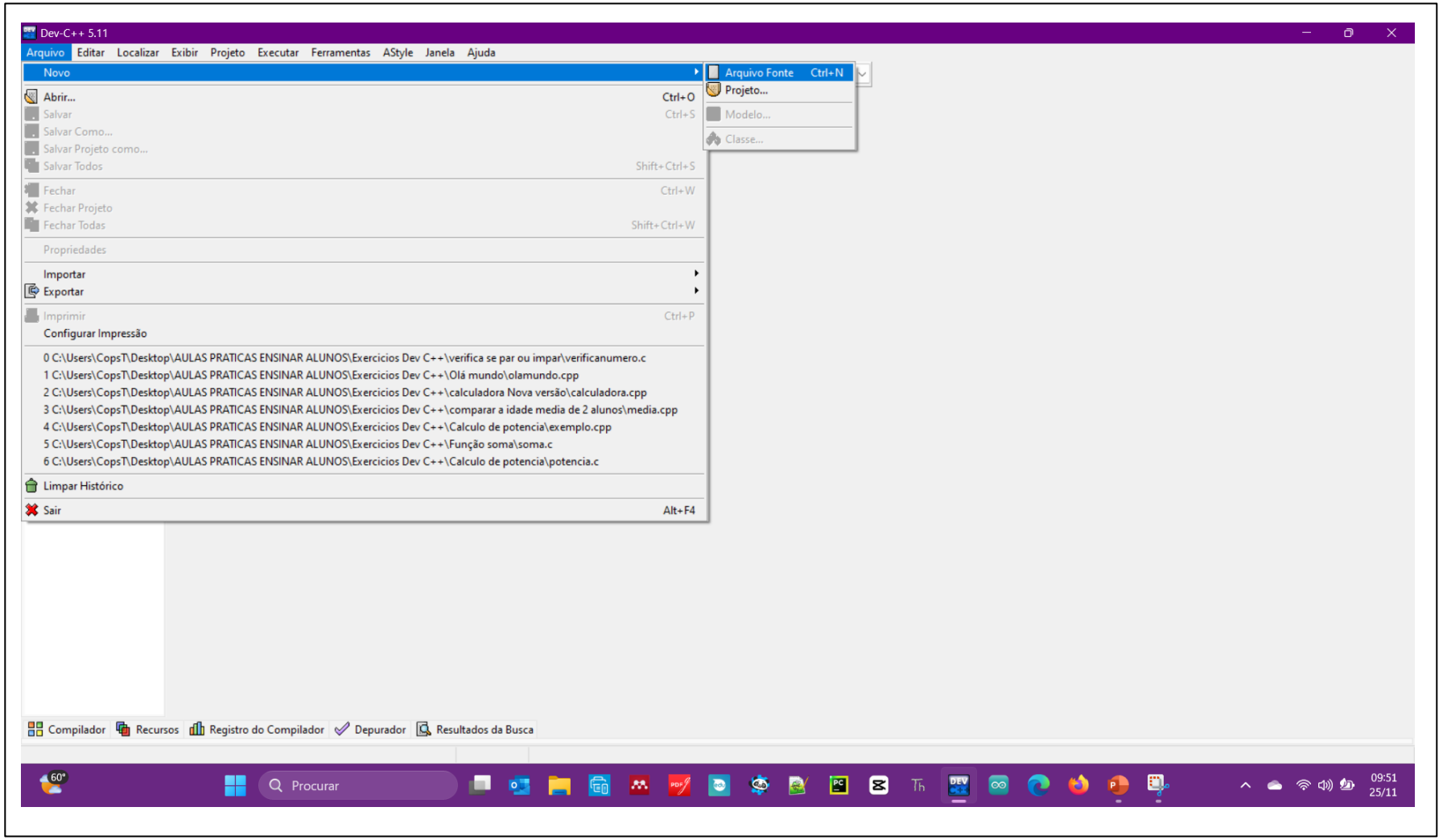
INICIO DO EXERCICIO 1 CONDUZIDO STEP BY STEP

- Vamos criar um programa em C que utilize uma função para calcular a soma de dois números inteiros à escolha do utilizador.

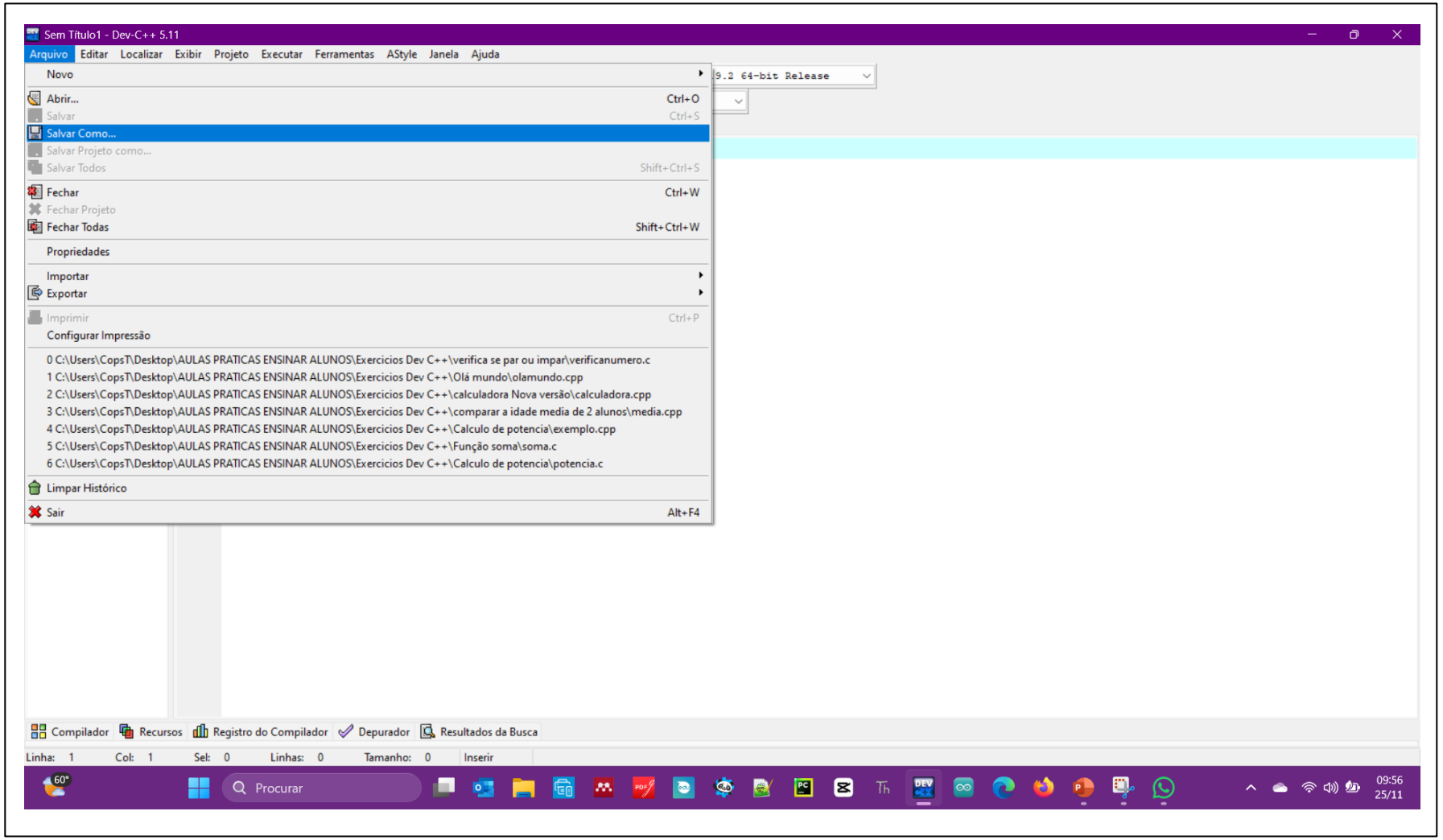
Passo 1 – Abrir o programa Dev C++



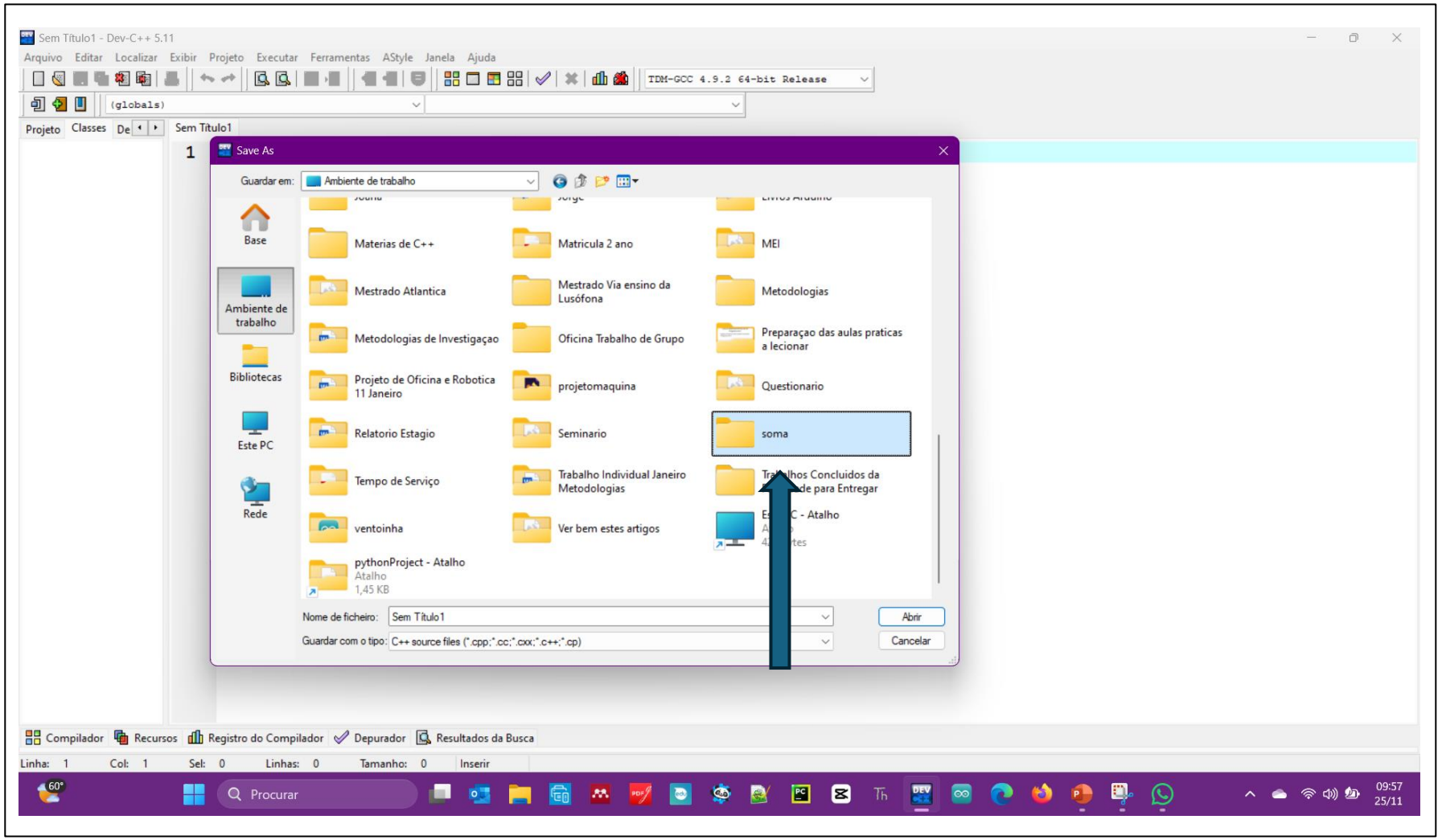
Passo 2 – Vamos a arquivo – Novo – Arquivo Fonte



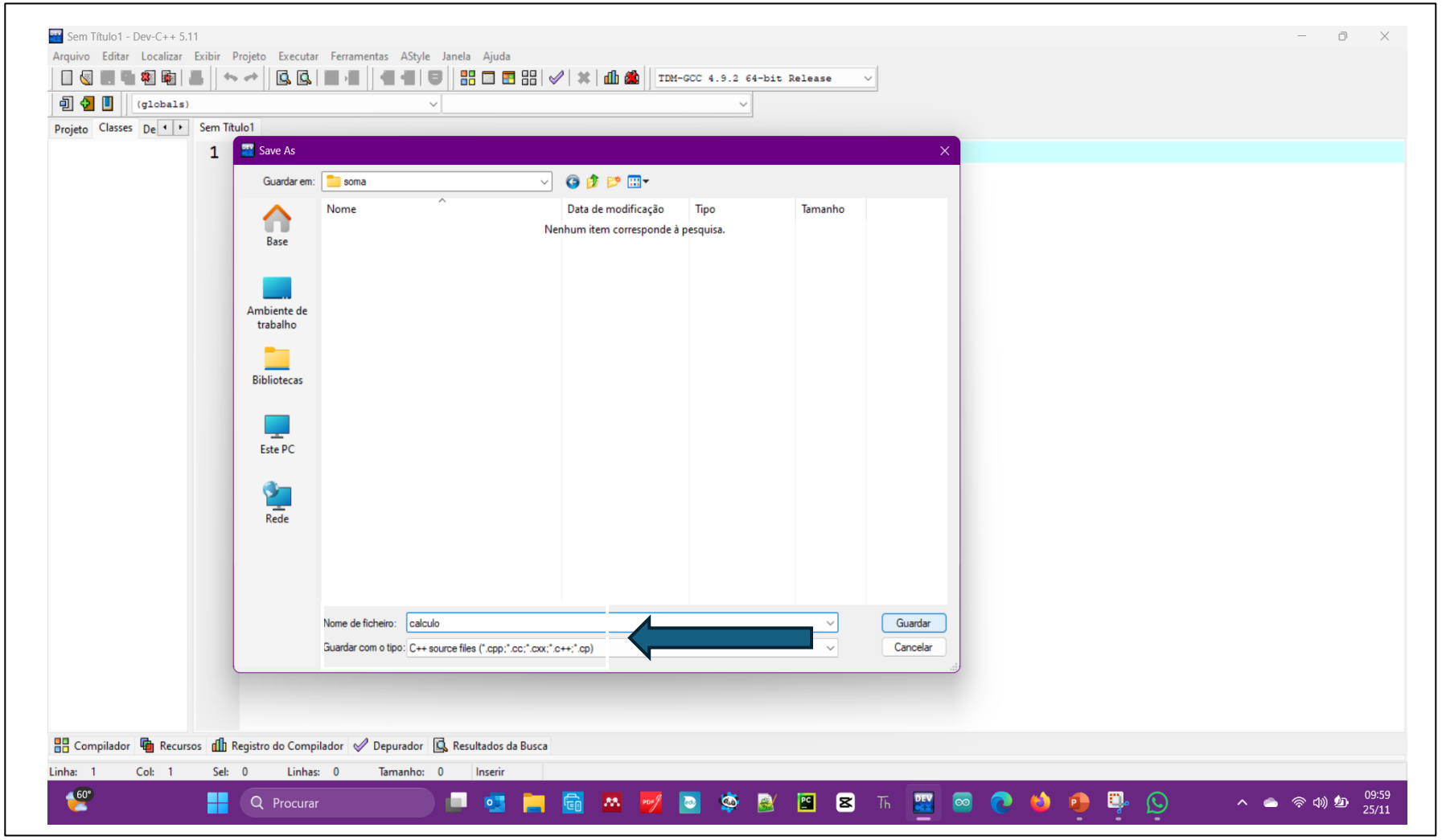
Passo 3 – Vamos agora salvar o ficheiro e vamos a salvar como…



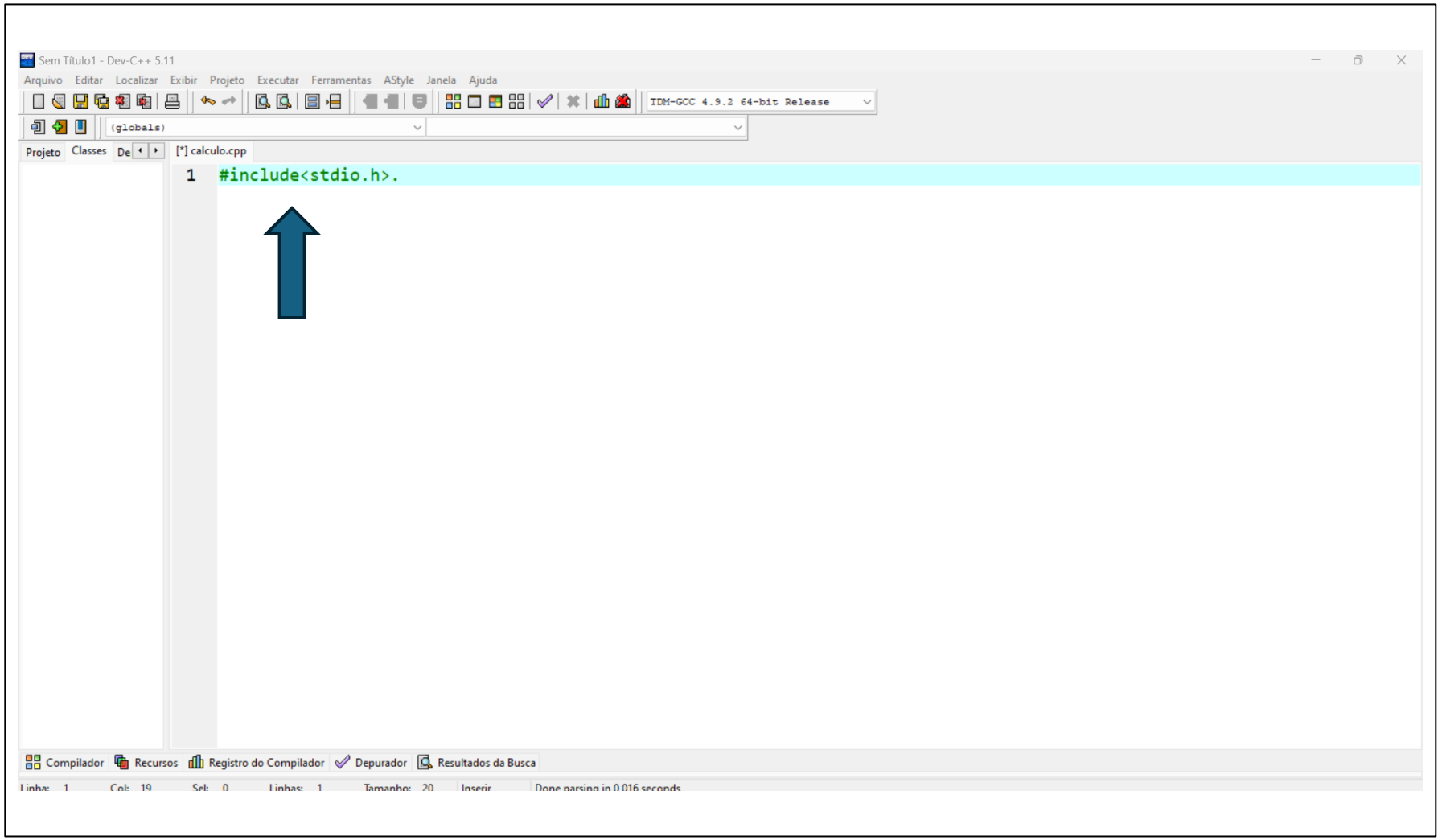
Passo 4 – Vamos agora salvar o ficheiro numa pasta chamada “soma”



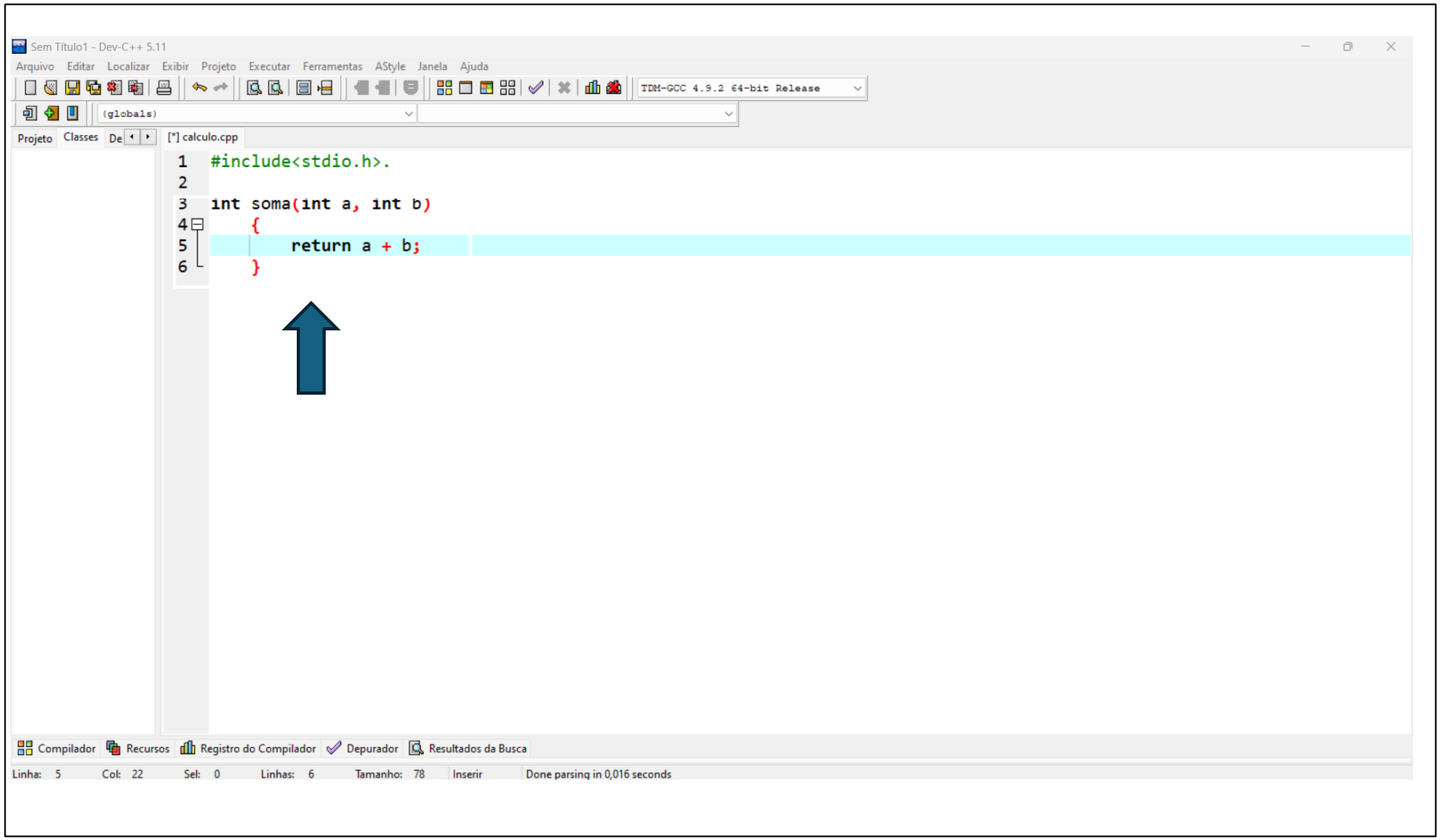
Passo 5 – Vamos agora salvar o ficheiro dentro da pasta soma e dão lhe o nome “calculo”



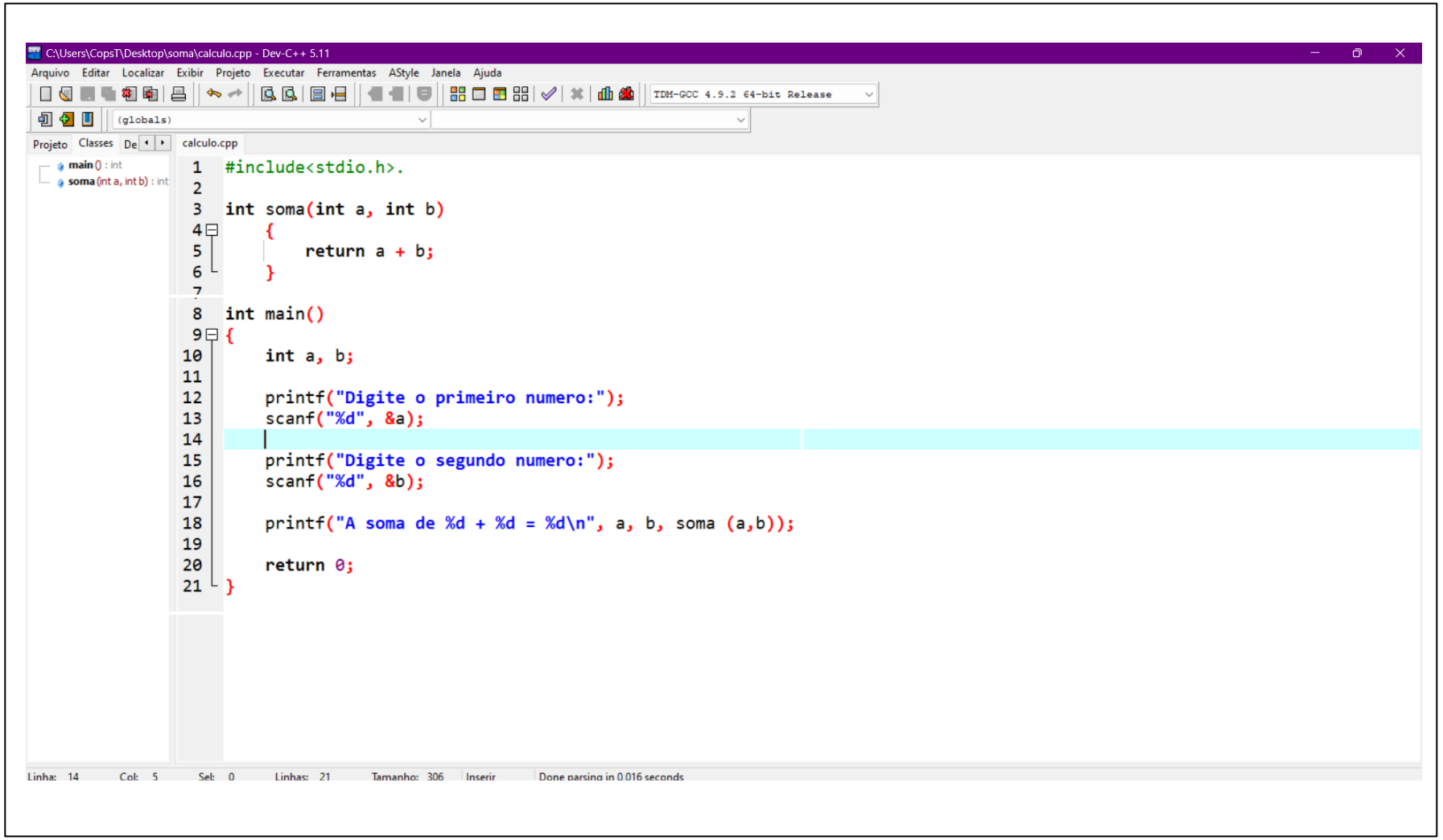
Passo 6 – No topo do programa do Dev C++ linha 1 devemos incluir a biblioteca - #include<stdio.h>



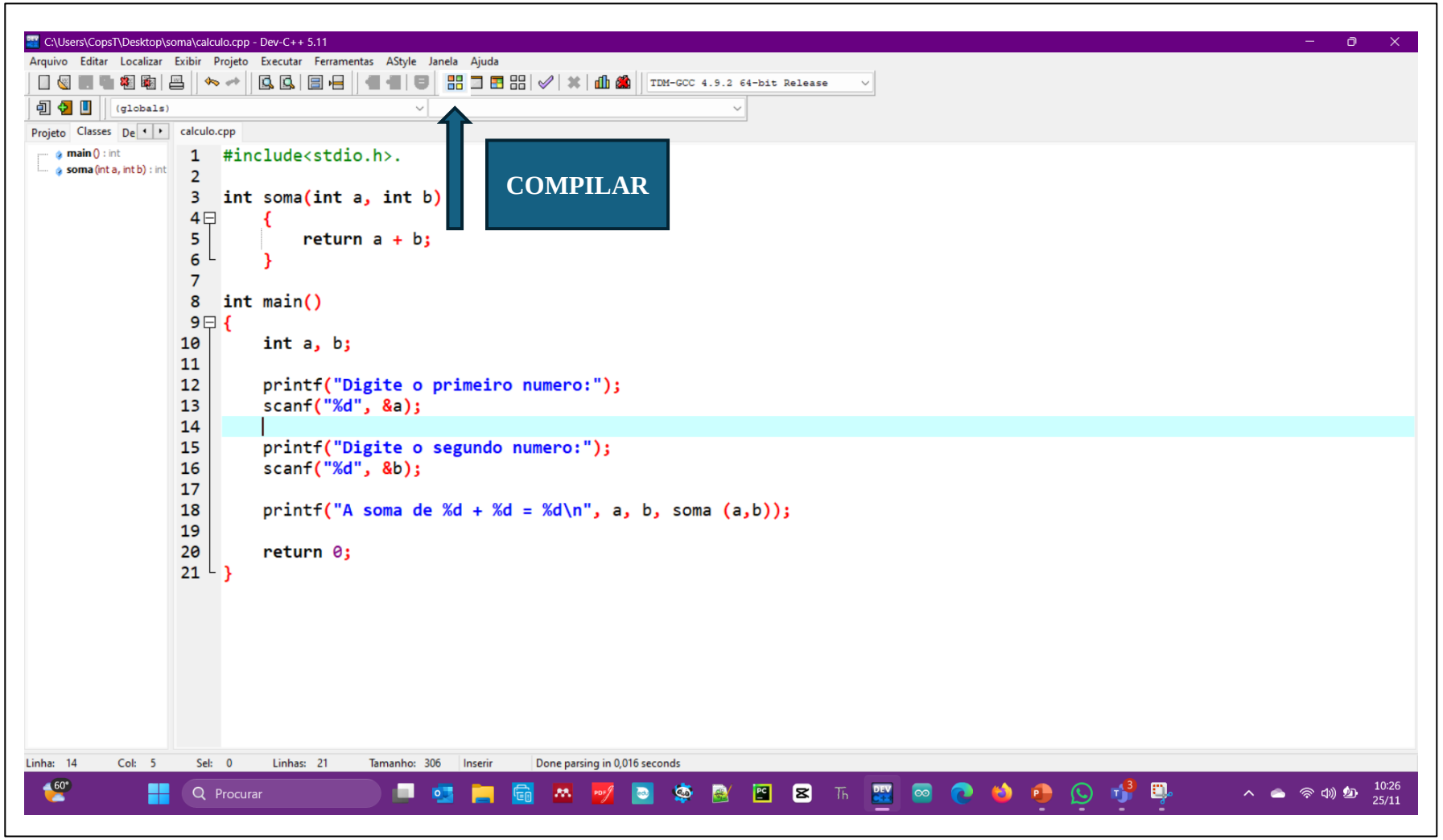
Passo 7 – Vamos declarar a função int soma(int a, int b) antes da função main()



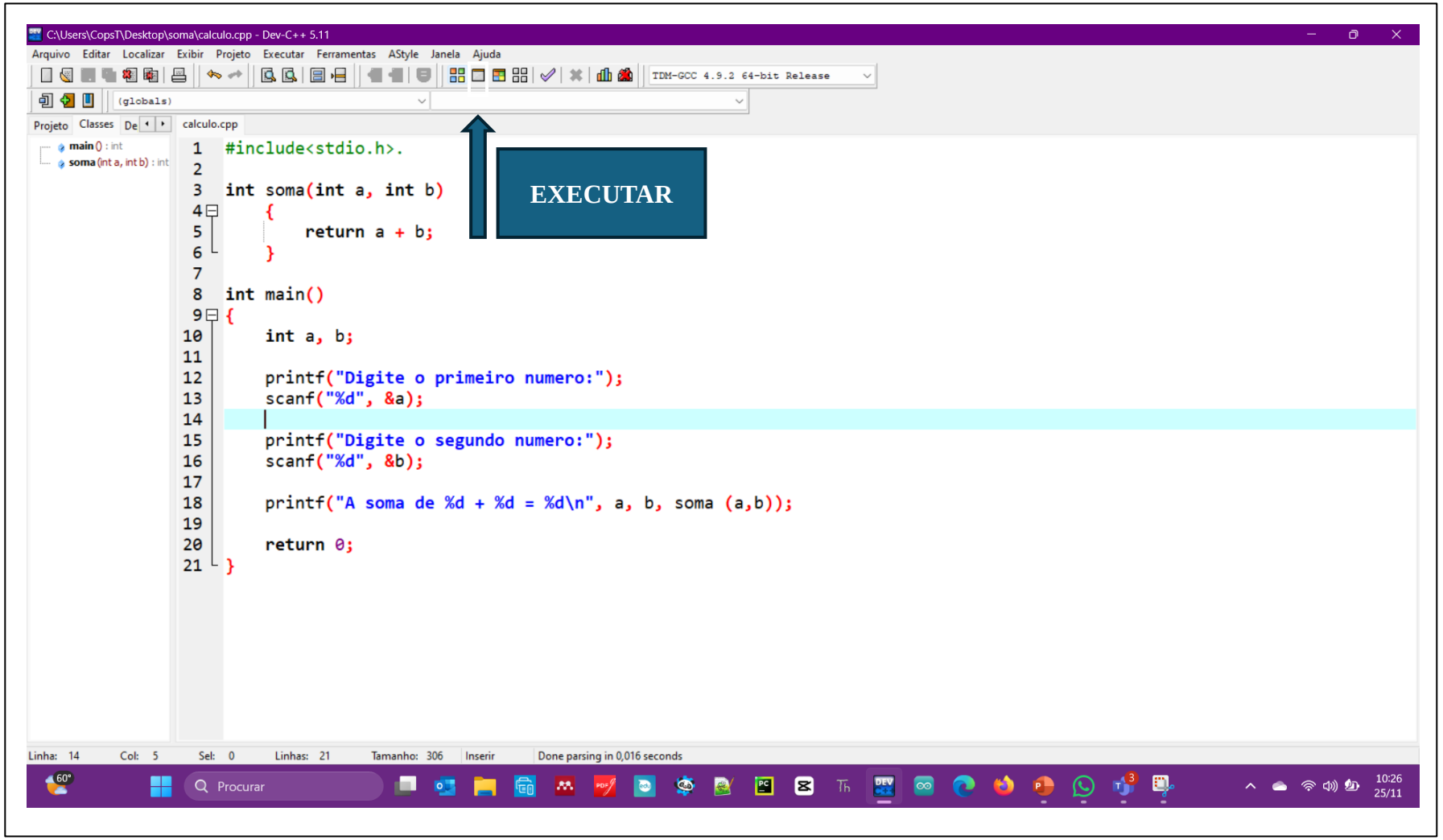
Passo 8 – Vamos implementar a função soma na função main()



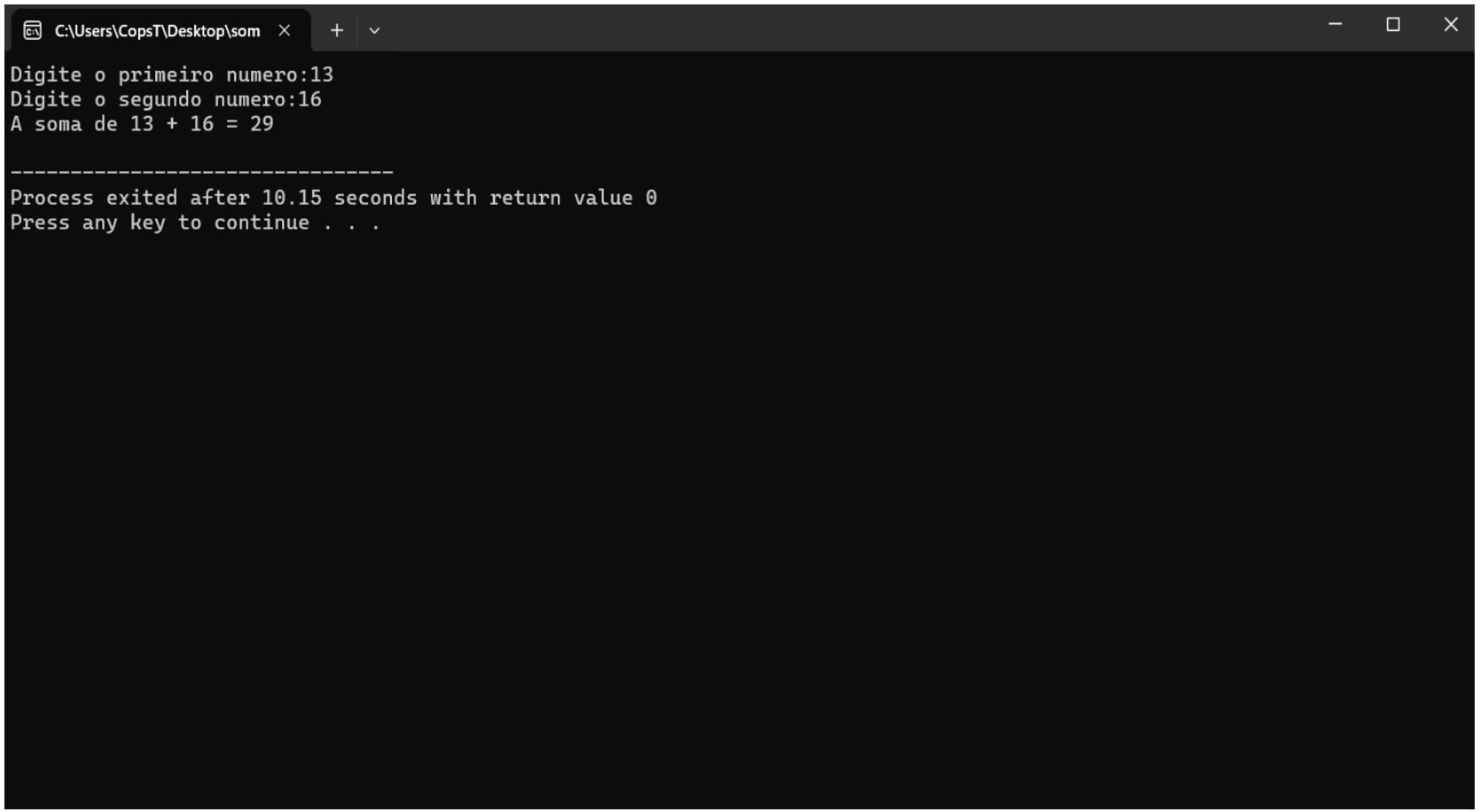
Passo 9 – Agora vamos compilar o código para verificarmos se existem erros



Passo 10 – Agora vamos executar o nosso código



Passo 11 – Programa a correr na consola

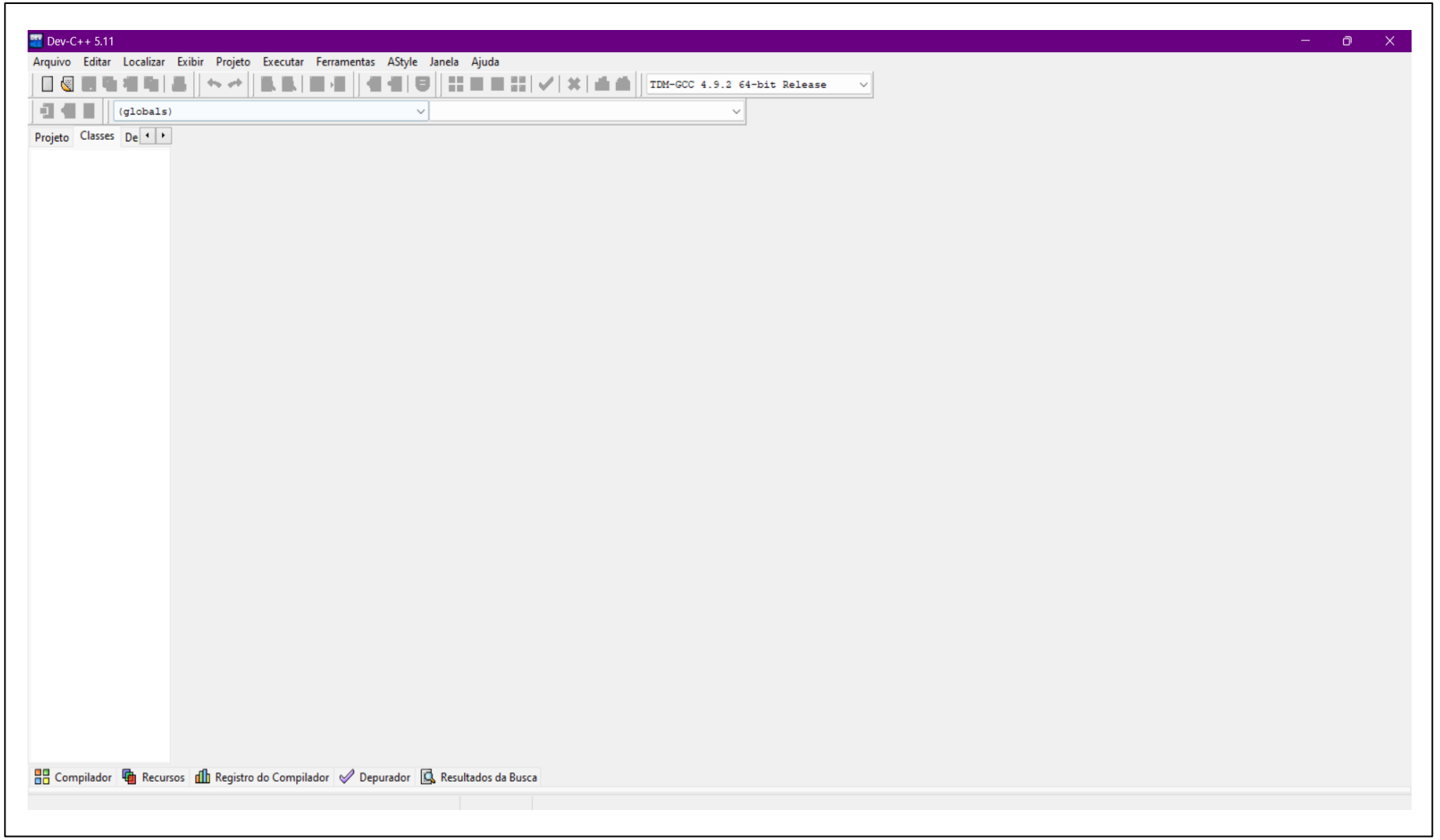


FIM DO EXERCICIO 1 CONDUZIDO STEP BY STEP

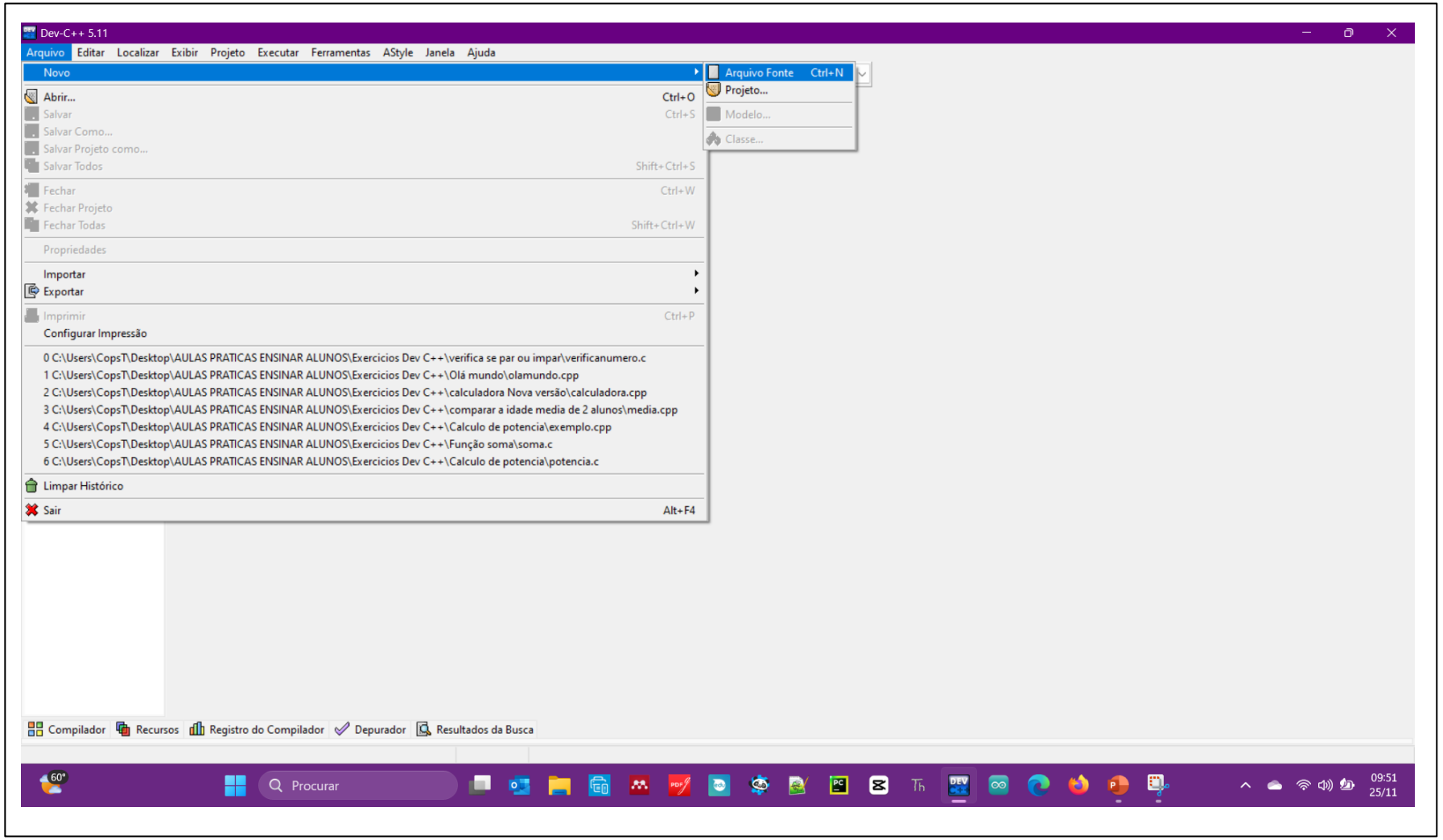
INICIO DO EXERCICIO 2 CONDUZIDO STEP BY STEP

- Desenvolva um programa que calcule a média de três notas de um aluno. Crie uma função com o nome calculaMedia() e que retorne o valor da média. O programa deve exibir a média no ecrã

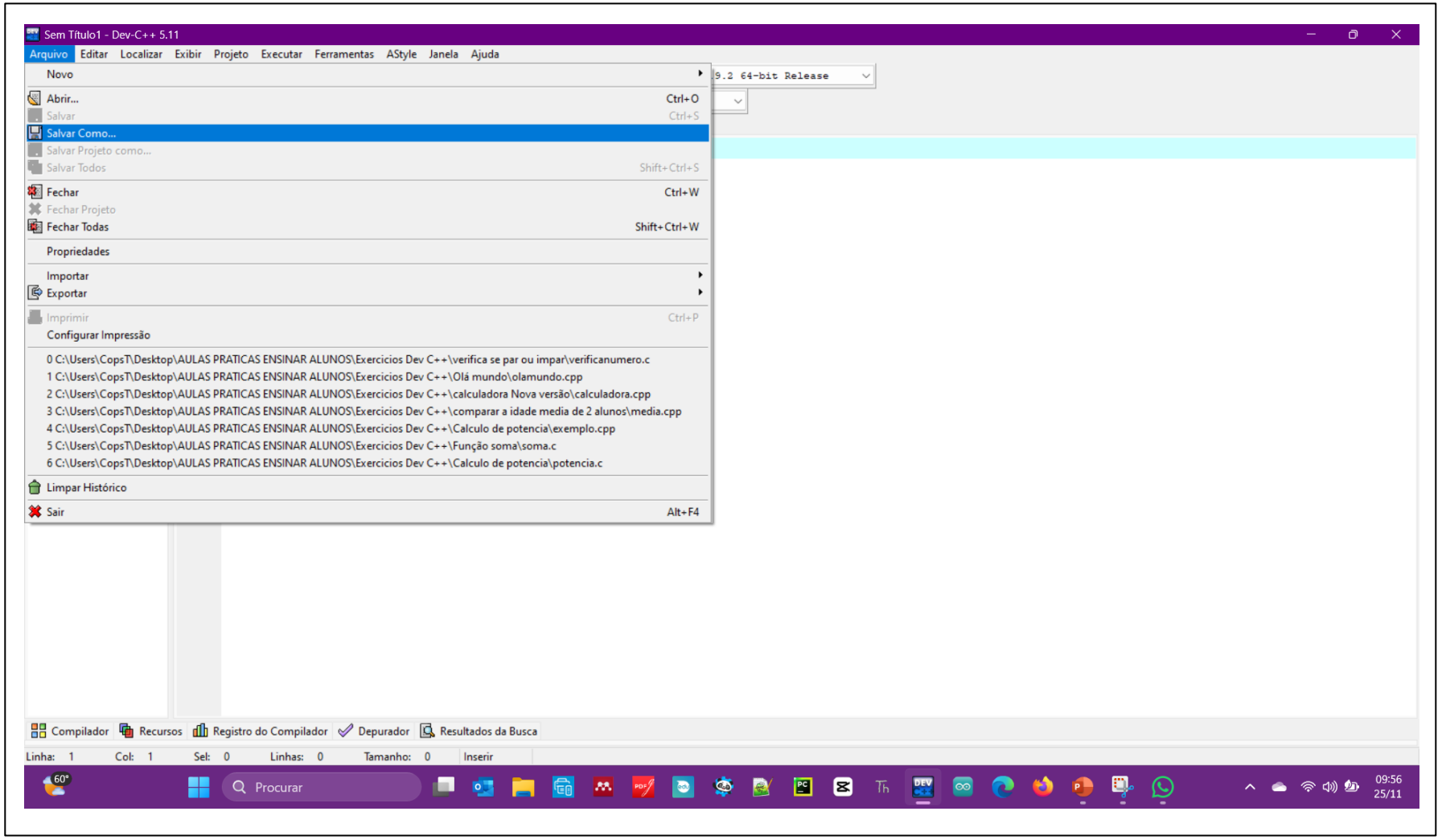
Passo 1 – Abrir o program Dev C++



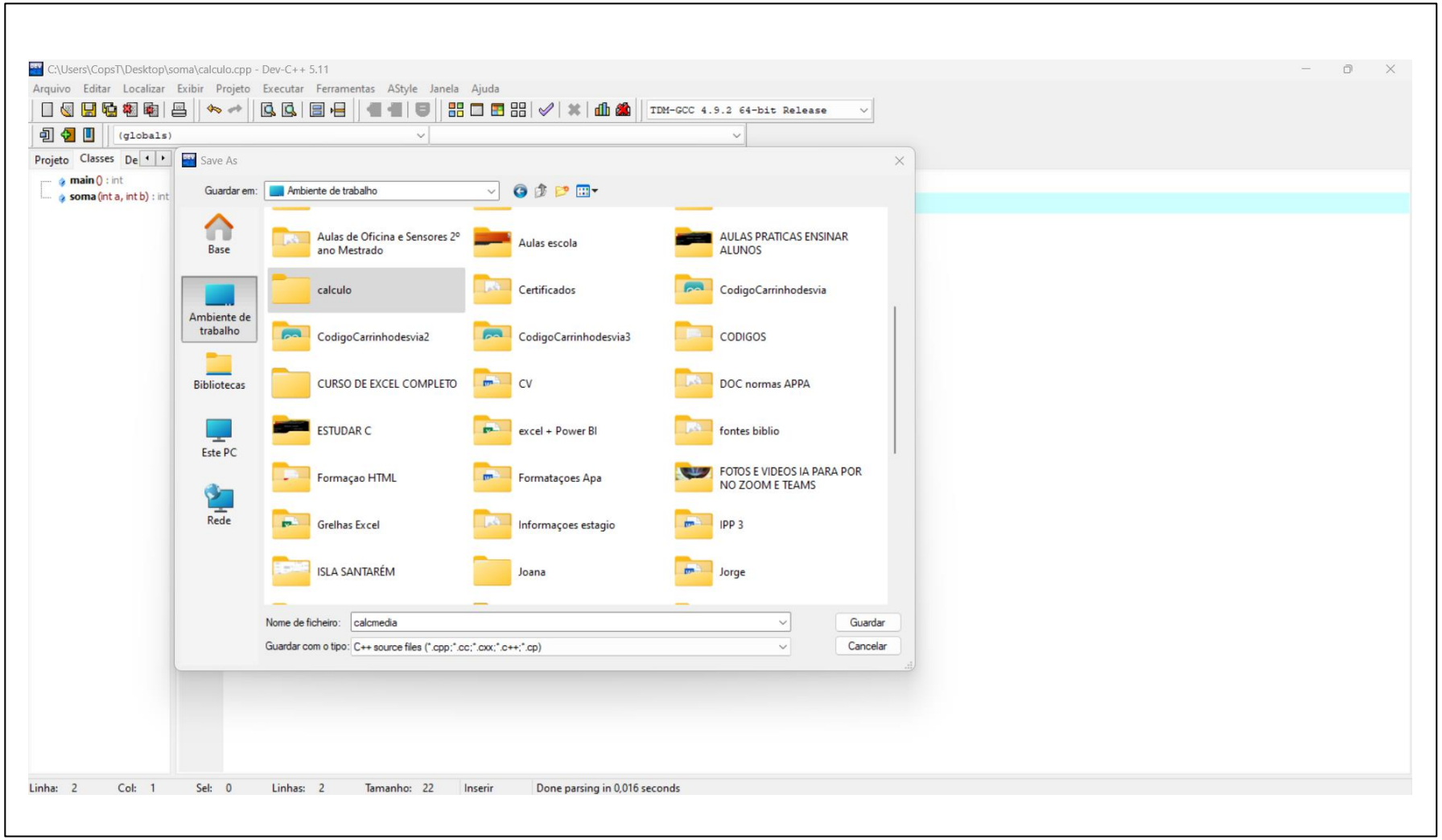
Passo 2 – Vamos a arquivo – Novo – Arquivo Fonte



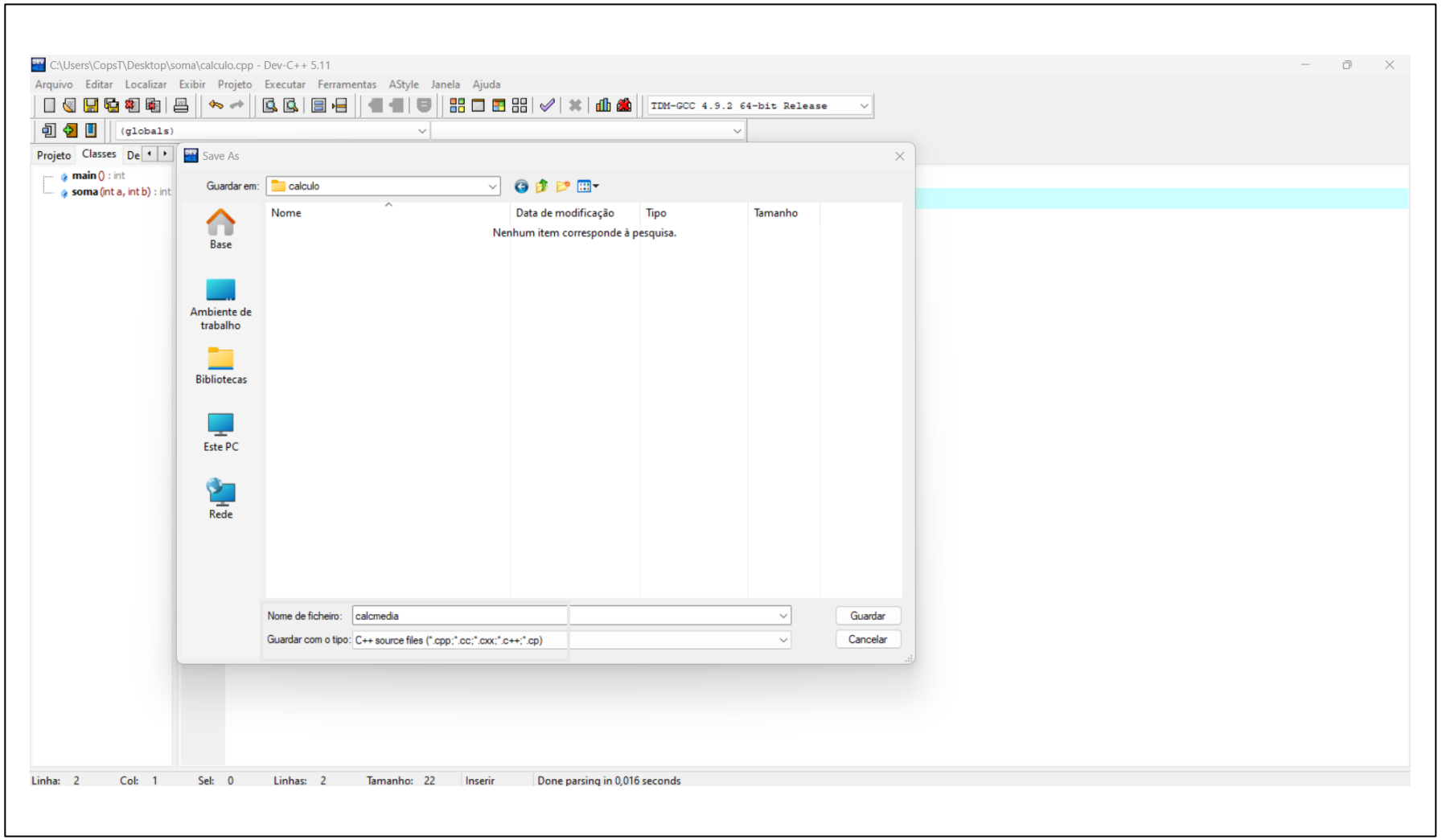
Passo 3 – Vamos agora salvar o ficheiro em salvar como...



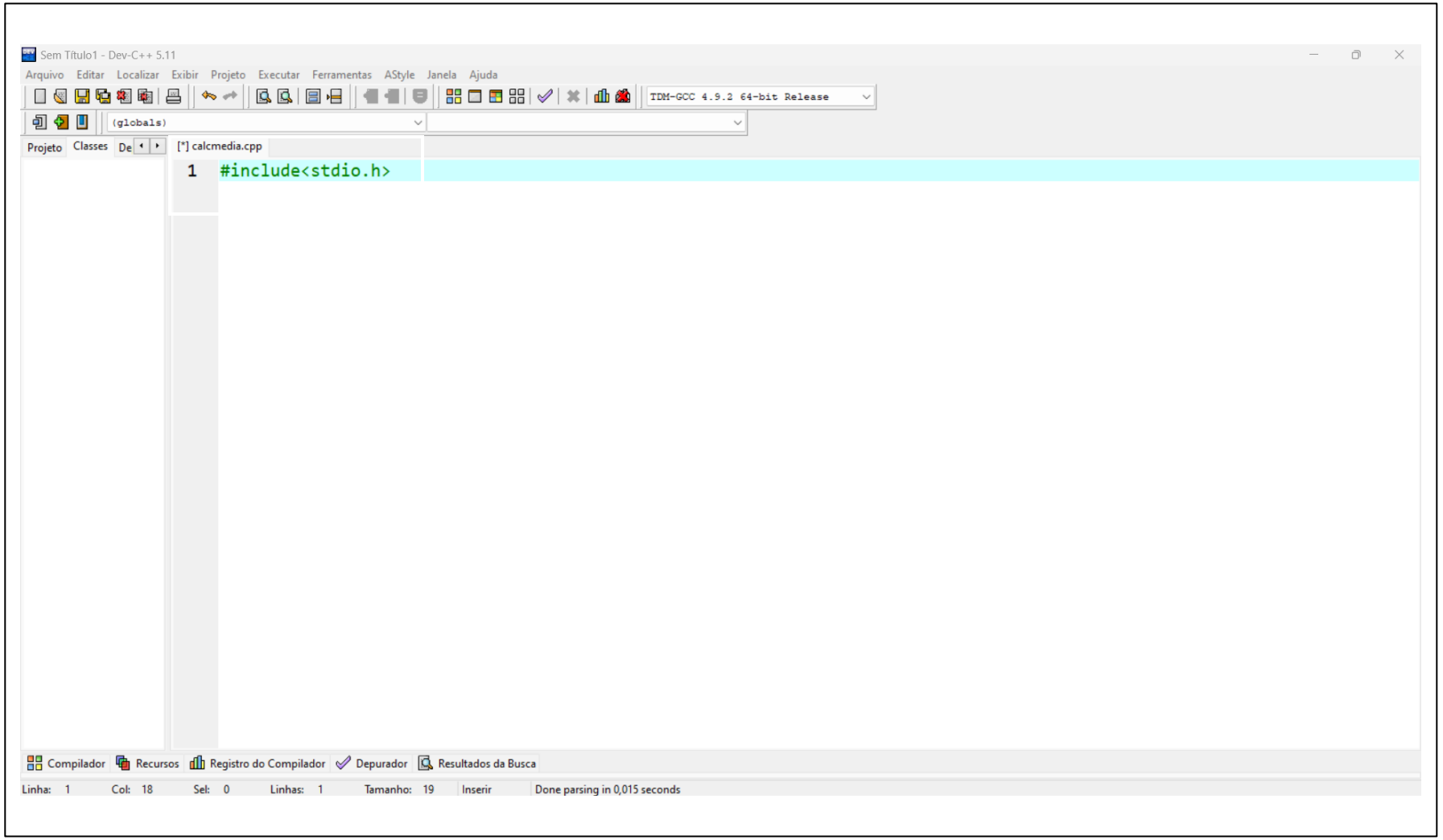
Passo 4 – Vamos agora salvar o ficheiro numa pasta chamada “calcmedia”



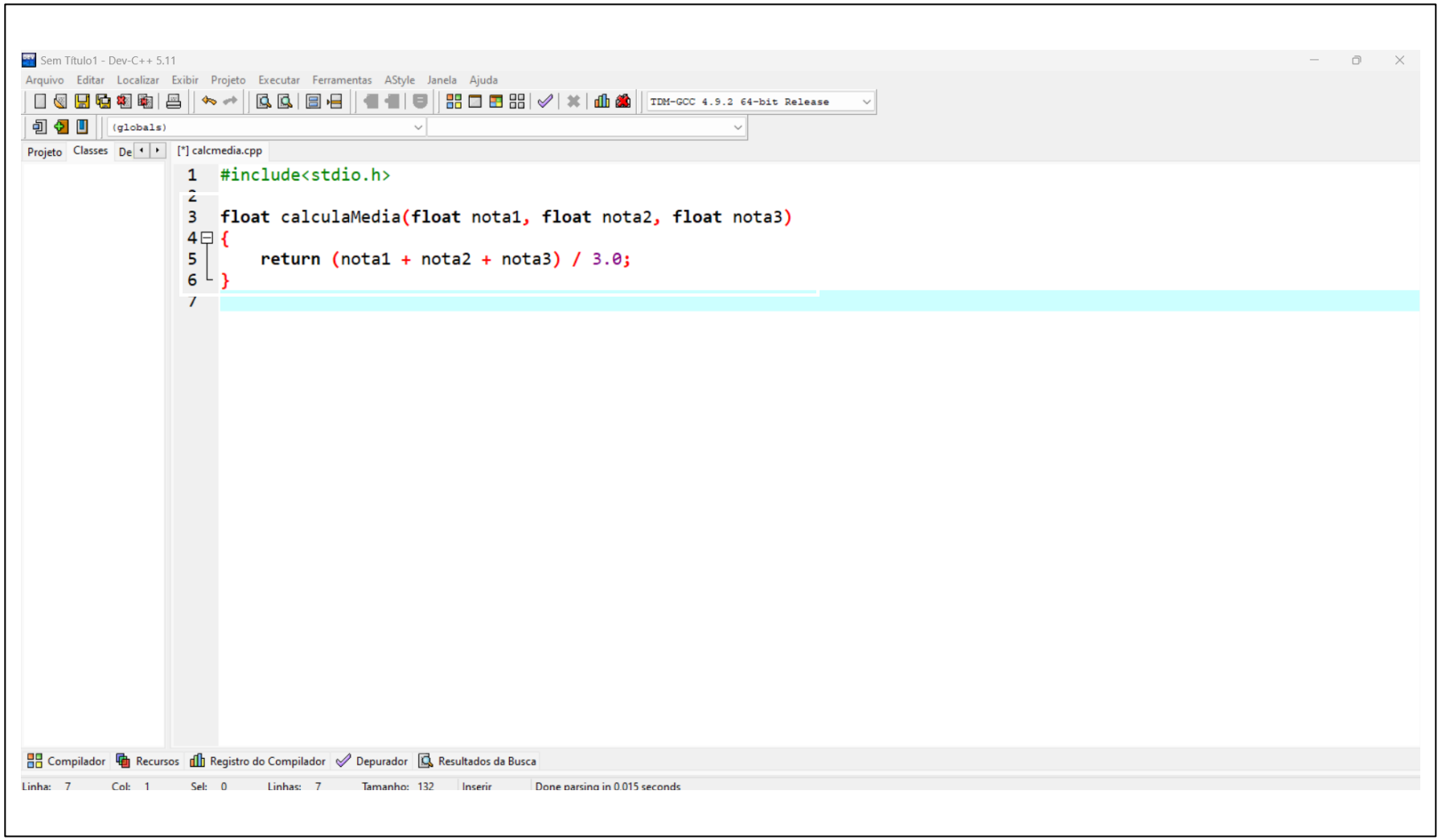
Passo 5 – Vamos guardar o ficheiro com o nome “calcmedia”



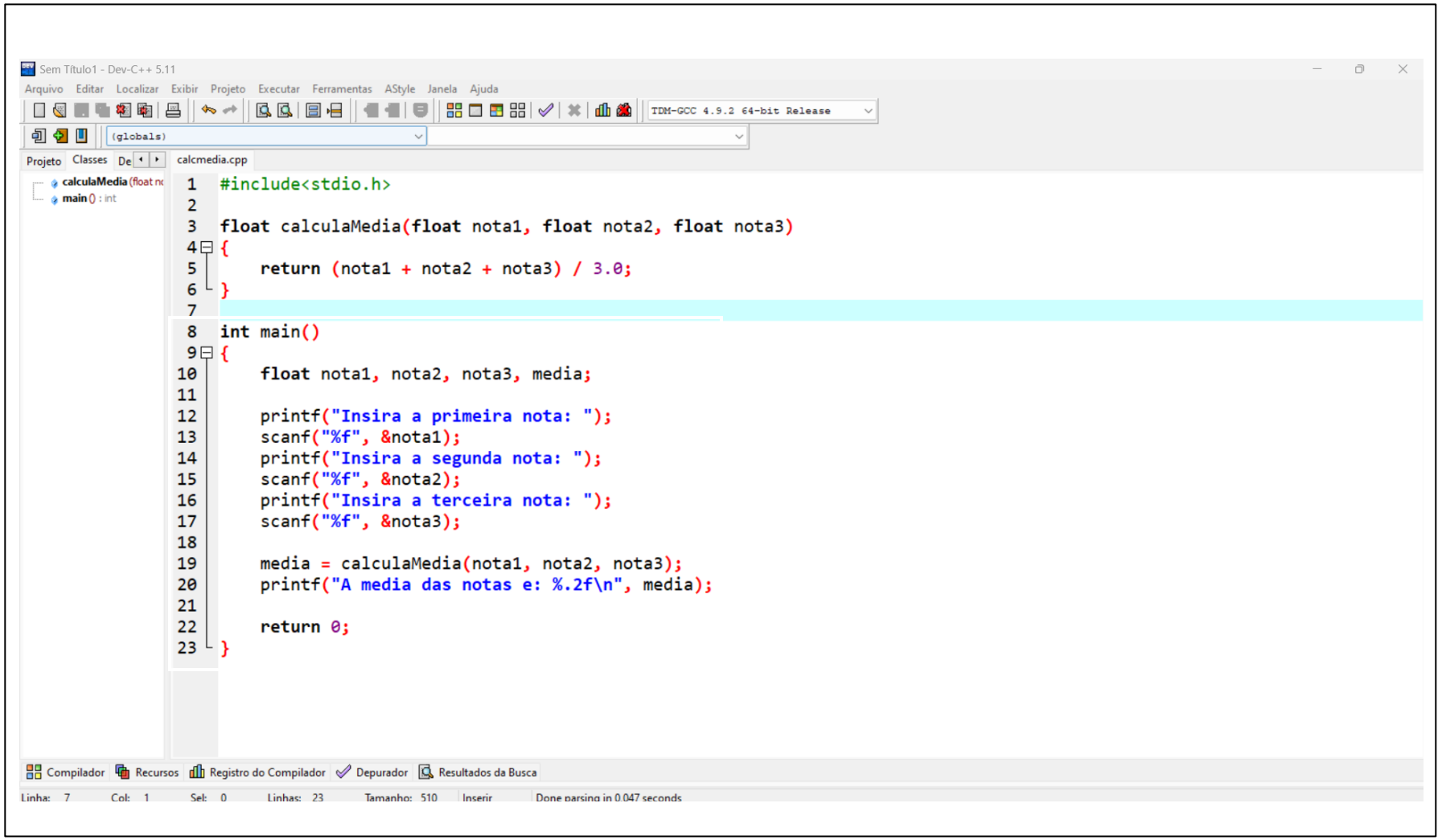
Passo 6 – No topo do programa do Dev C++ linha 1 devemos incluir a biblioteca - #include<stdio.h>



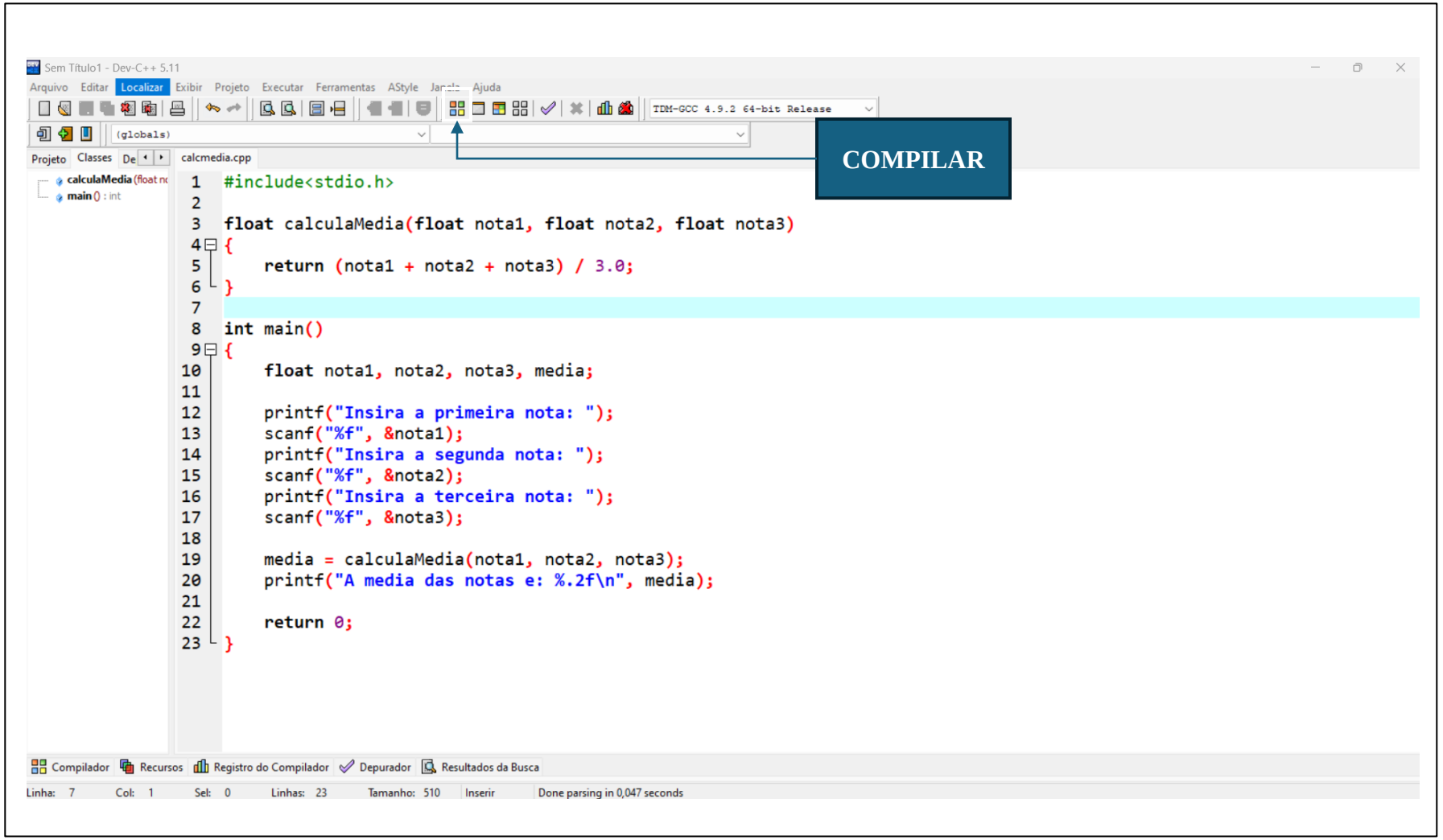
Passo 7 – Vamos declarar a função float calculaMedia(float nota1, float nota2, float nota3) antes da função main()



Passo 8 – Vamos agora escrever o resto do programa utilizando a função main()

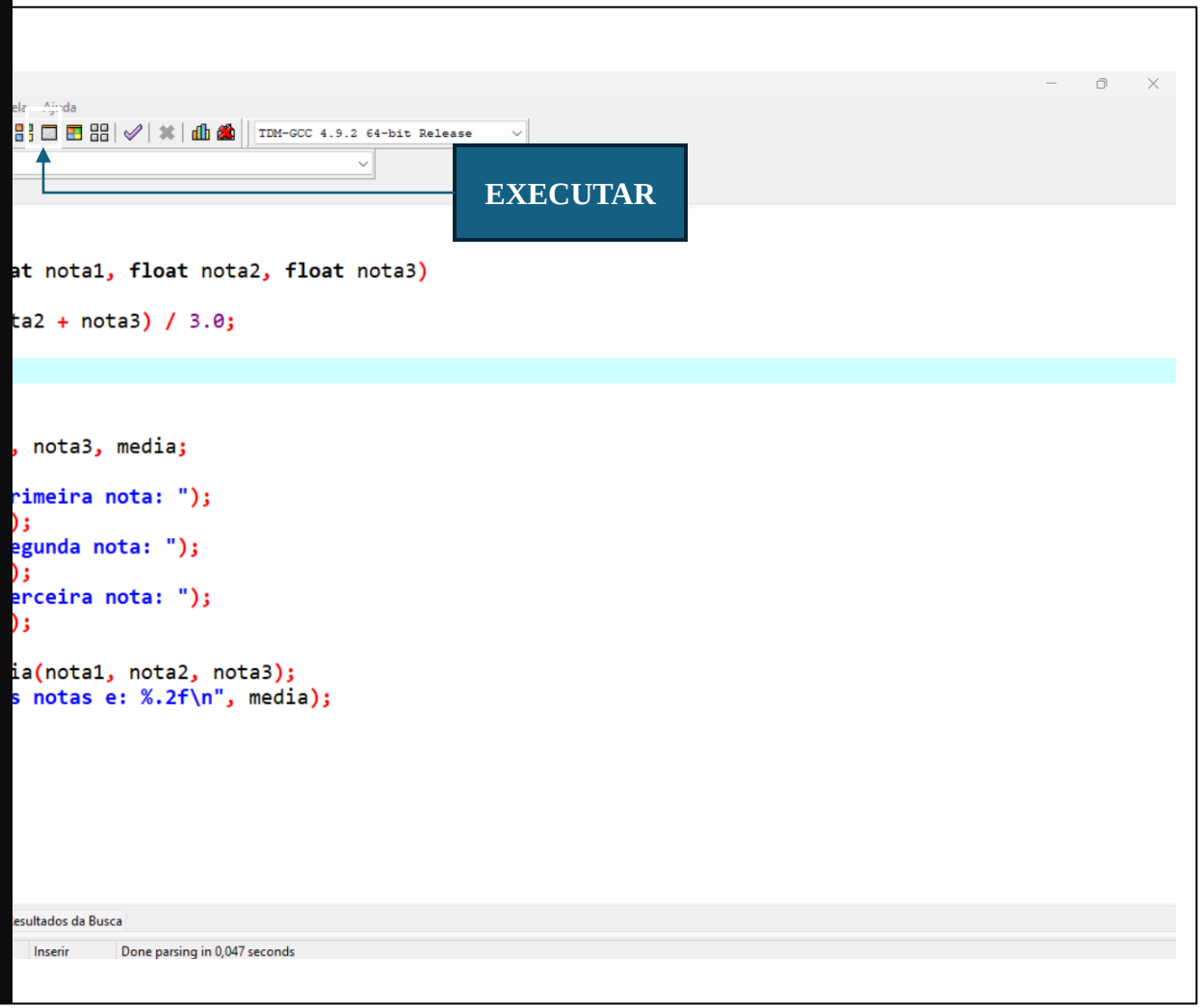


Passo 9 – Vamos agora compilar o programa para verificarmos se existem erros no código

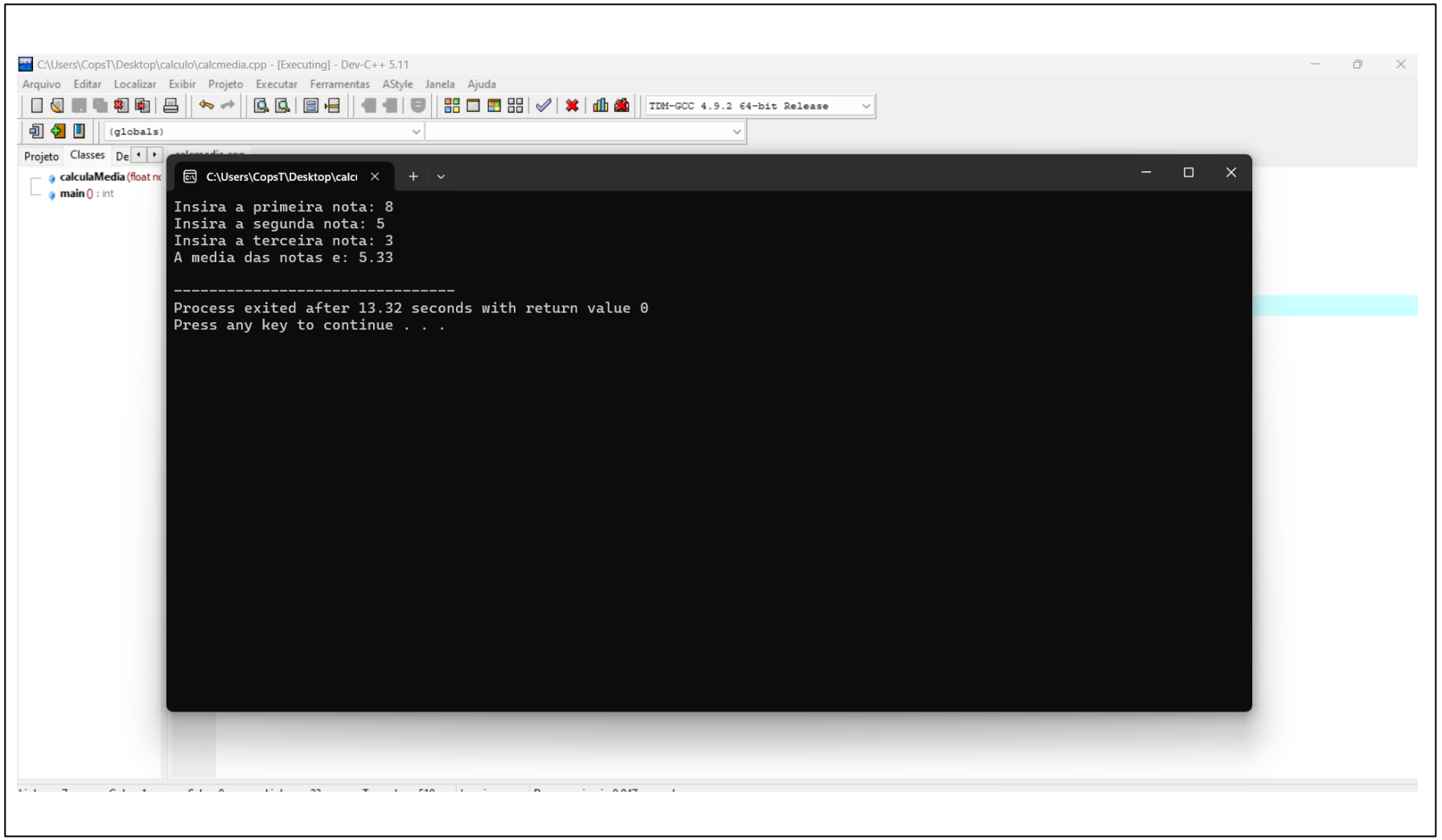


Passo 10 – Vamos agora executar o programa e ver o resultado

```c
#include <stdio.h>

float calcularMedia(float a, float b, float c);

int main() {
    float nota1, nota2, nota3;

    printf("Digite as três notas separadas por espaço: ");
    scanf("%f %f %f", &nota1, &nota2, &nota3);

    printf("A média das notas é: %.2f\n", calcularMedia(nota1, nota2, nota3));

    return 0;
}

float calcularMedia(float a, float b, float c) {
    return (a + b + c) / 3;
}
```



Passo 10 – Vamos agora executar e ver a nossa consola

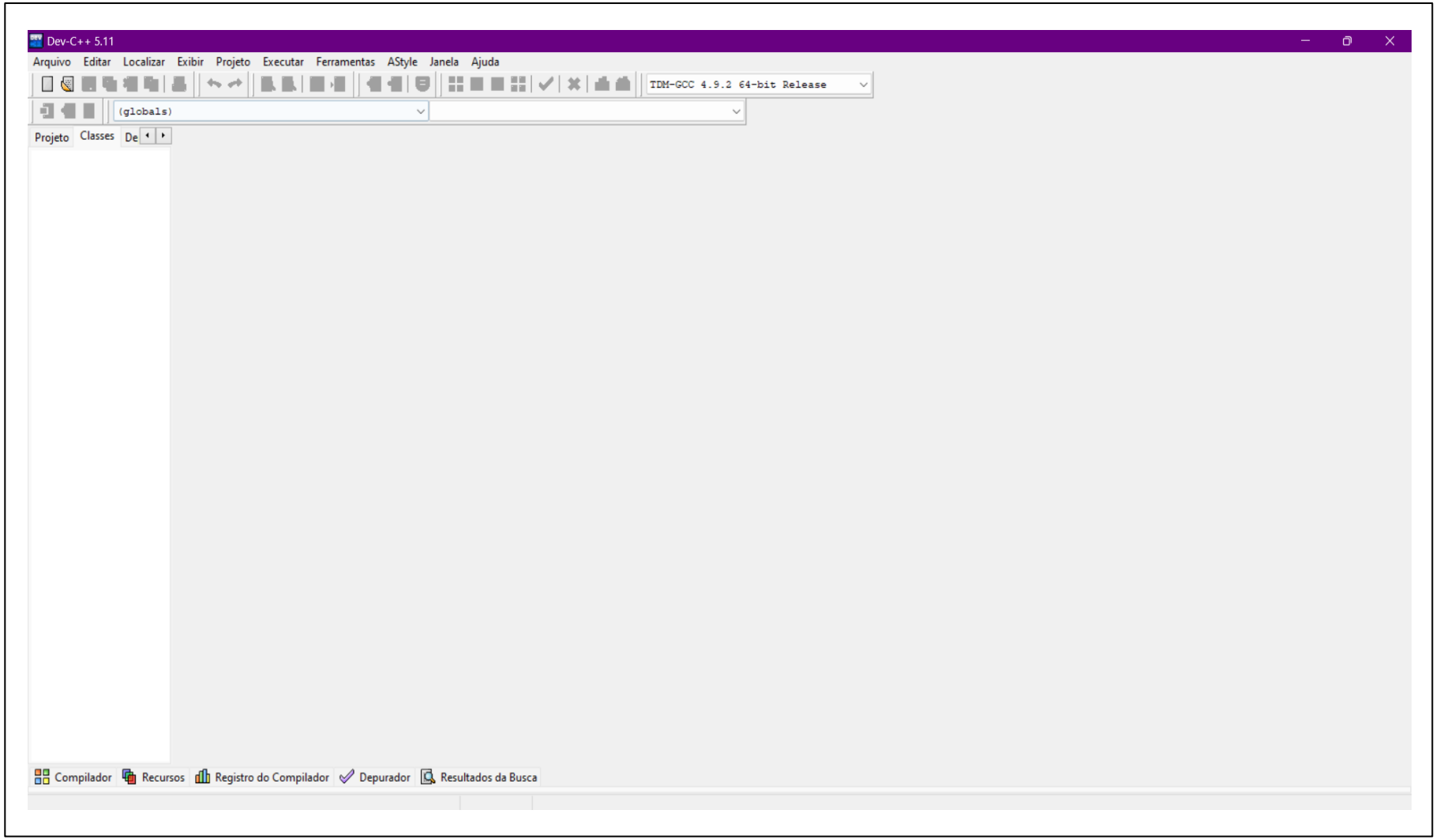


FIM DO EXERCICIO 2 CONDUZIDO STEP BY STEP

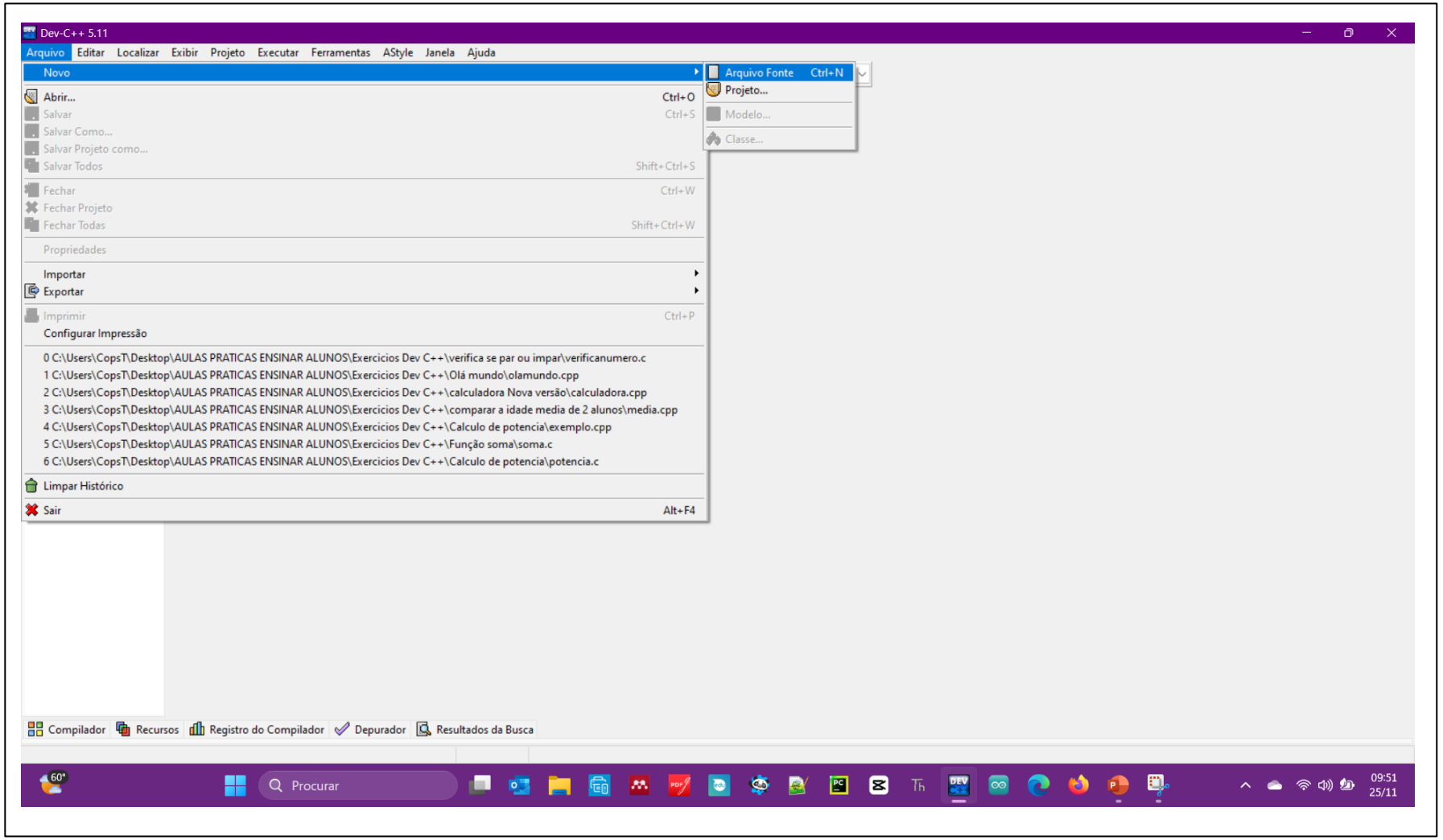
EXERCICIO 3 CONDUZIDO STEP BY STEP

- Desenvolva um programa que calcule a média de três ponderações diferentes sendo que o trabalho vale (25%), a ficha de trabalho (15%) e um teste (60%).
- Crie uma função com o nome calculaMedia() e que retorne o valor da média. O programa deve exibir a média no ecrã.

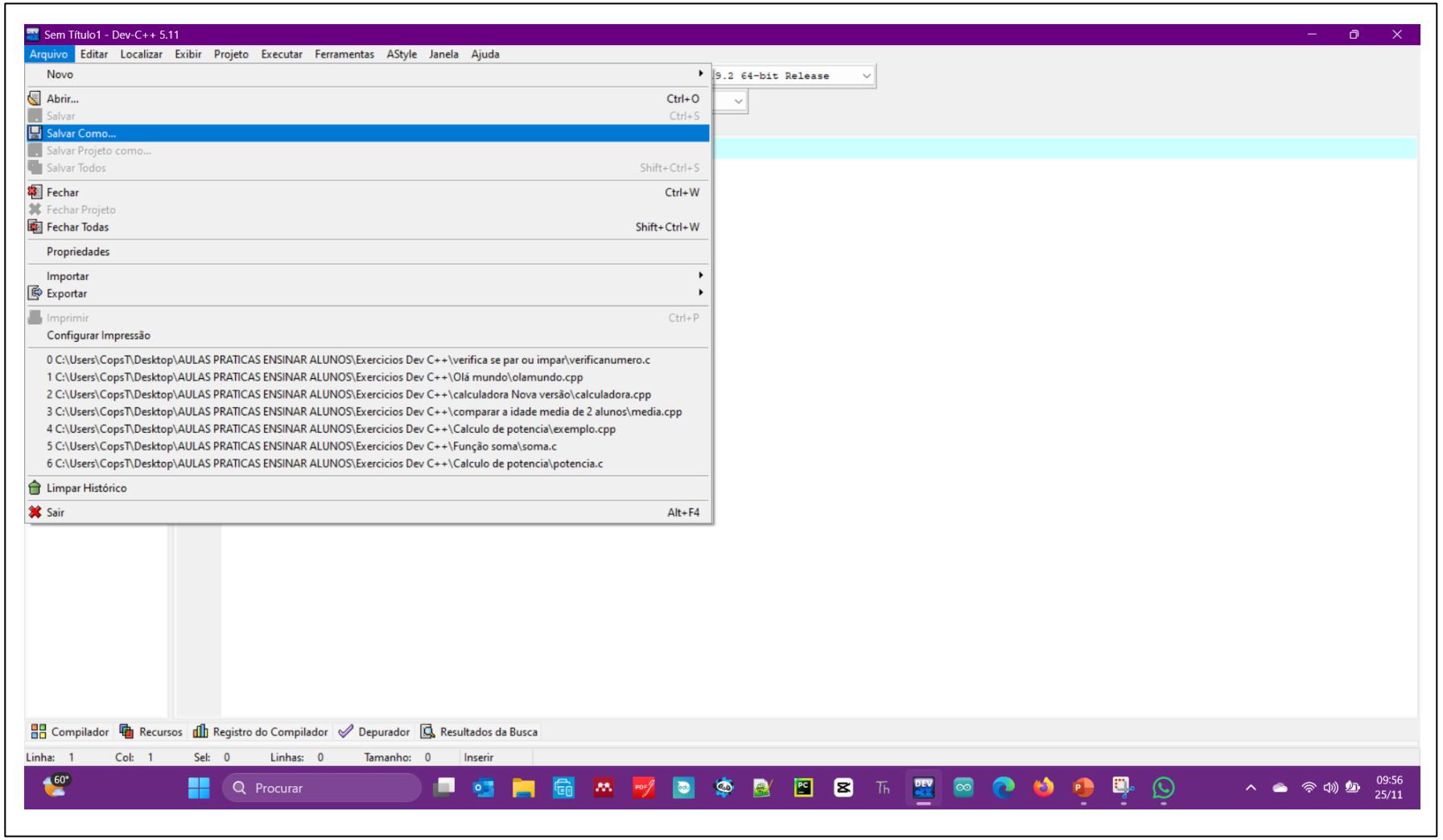
Passo 1 – Abrir o program Dev C++



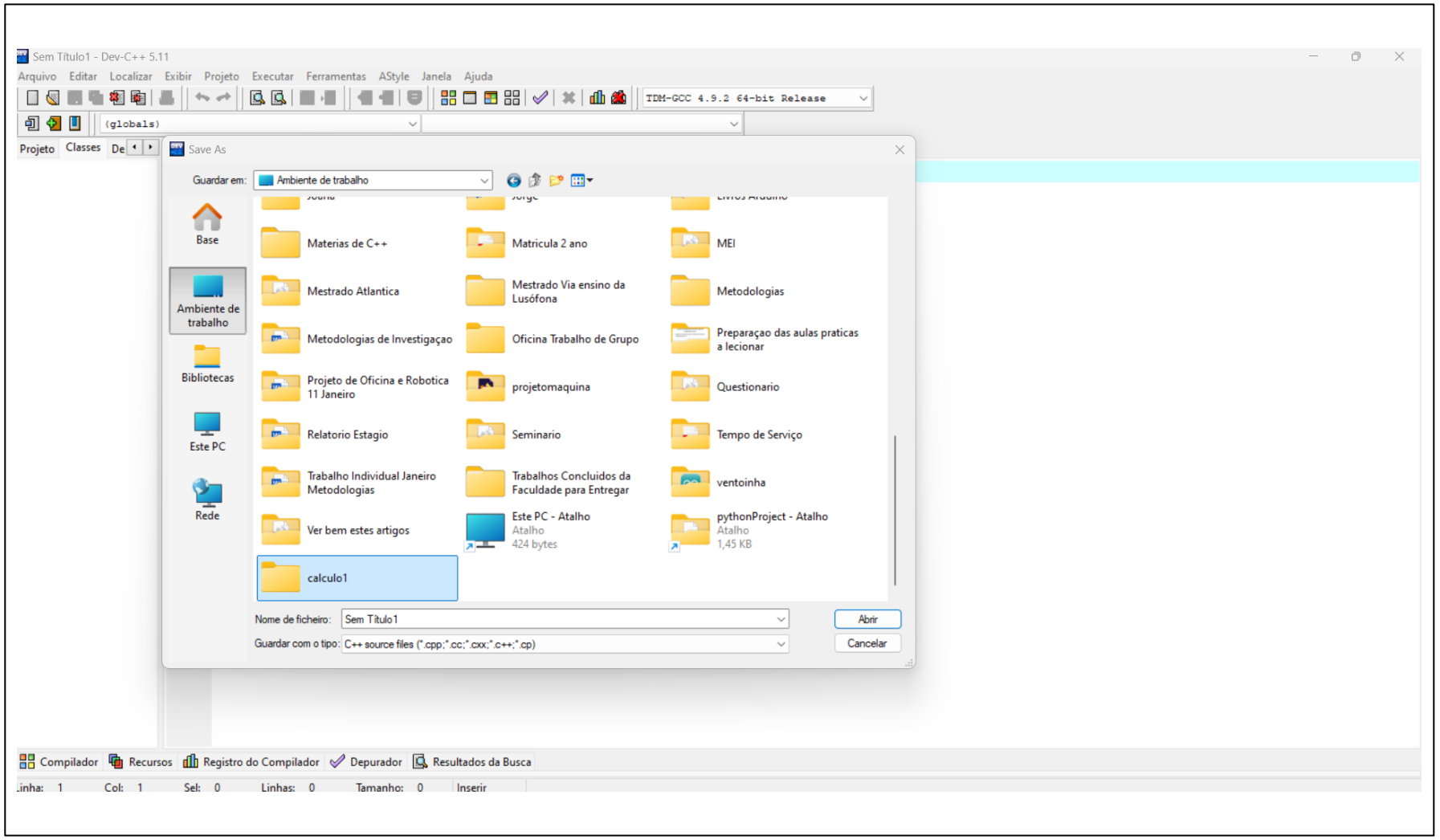
Passo 2 – Vamos a arquivo – Novo – Arquivo Fonte



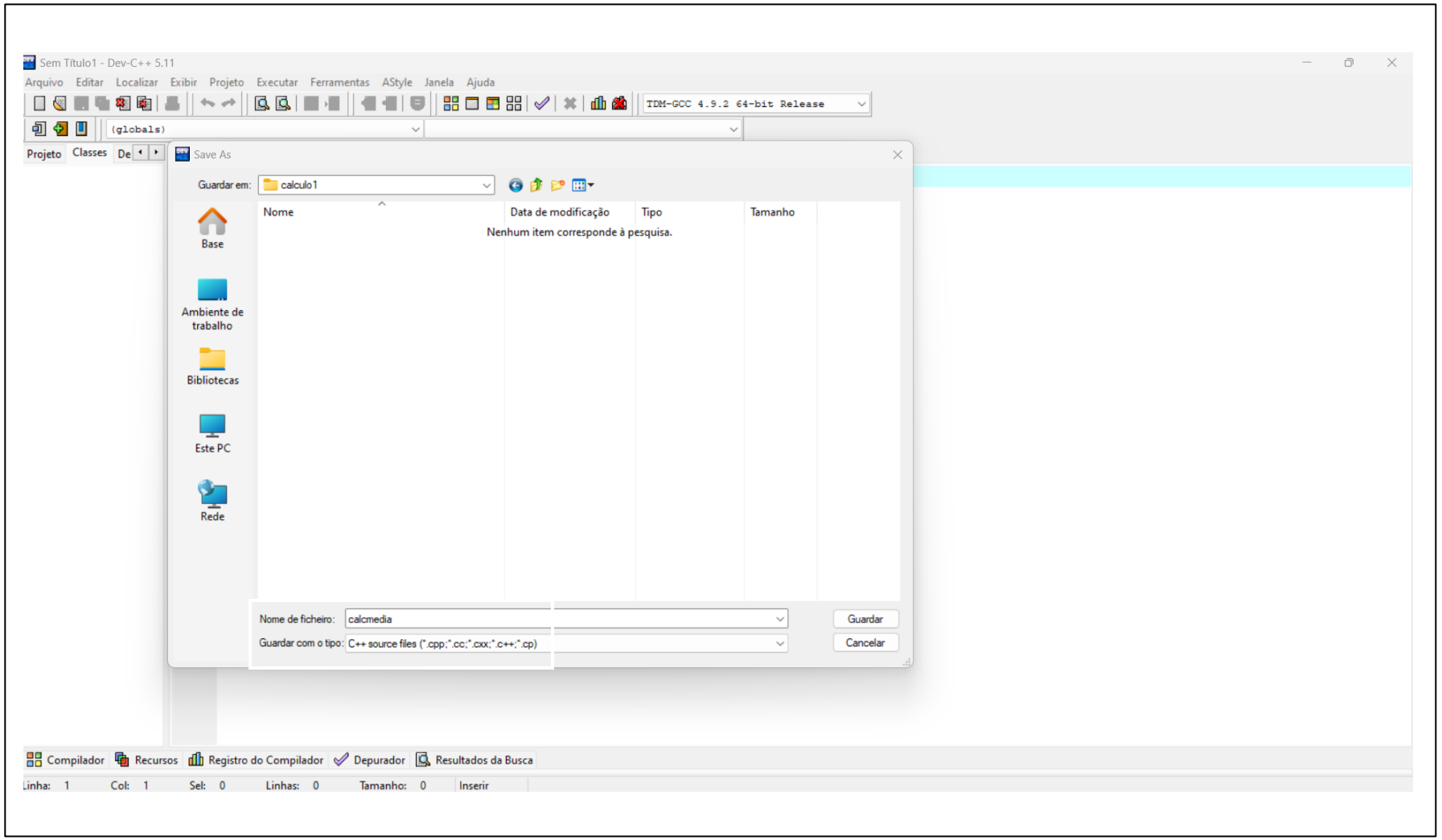
Passo 3 – Vamos agora salvar o ficheiro e vamos a salvar como…



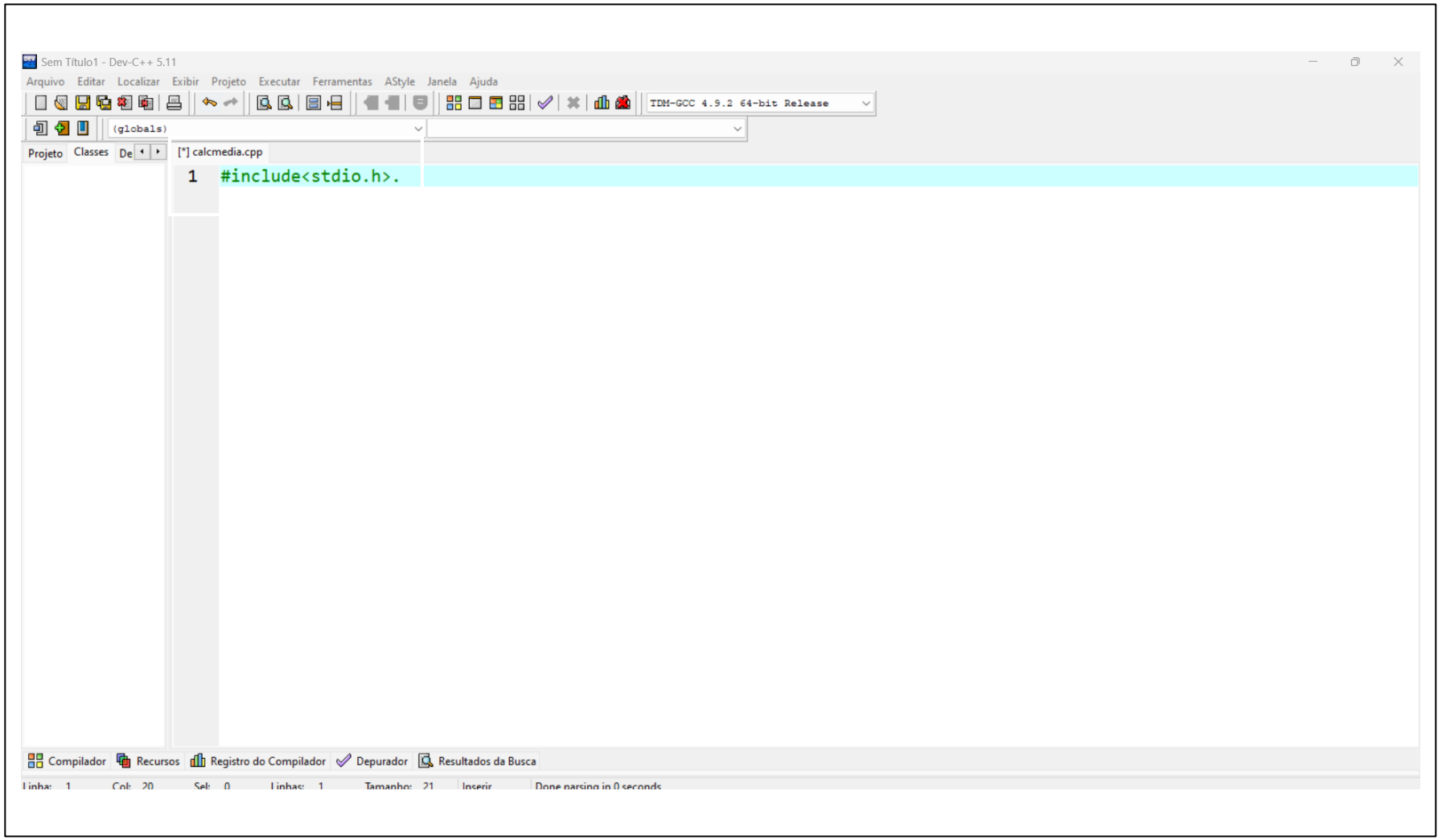
Passo 4 – Vamos agora salvar o ficheiro com o nome “calcmedia” numa pasta no ambiente de trabalho dentro da pasta “calculo1”



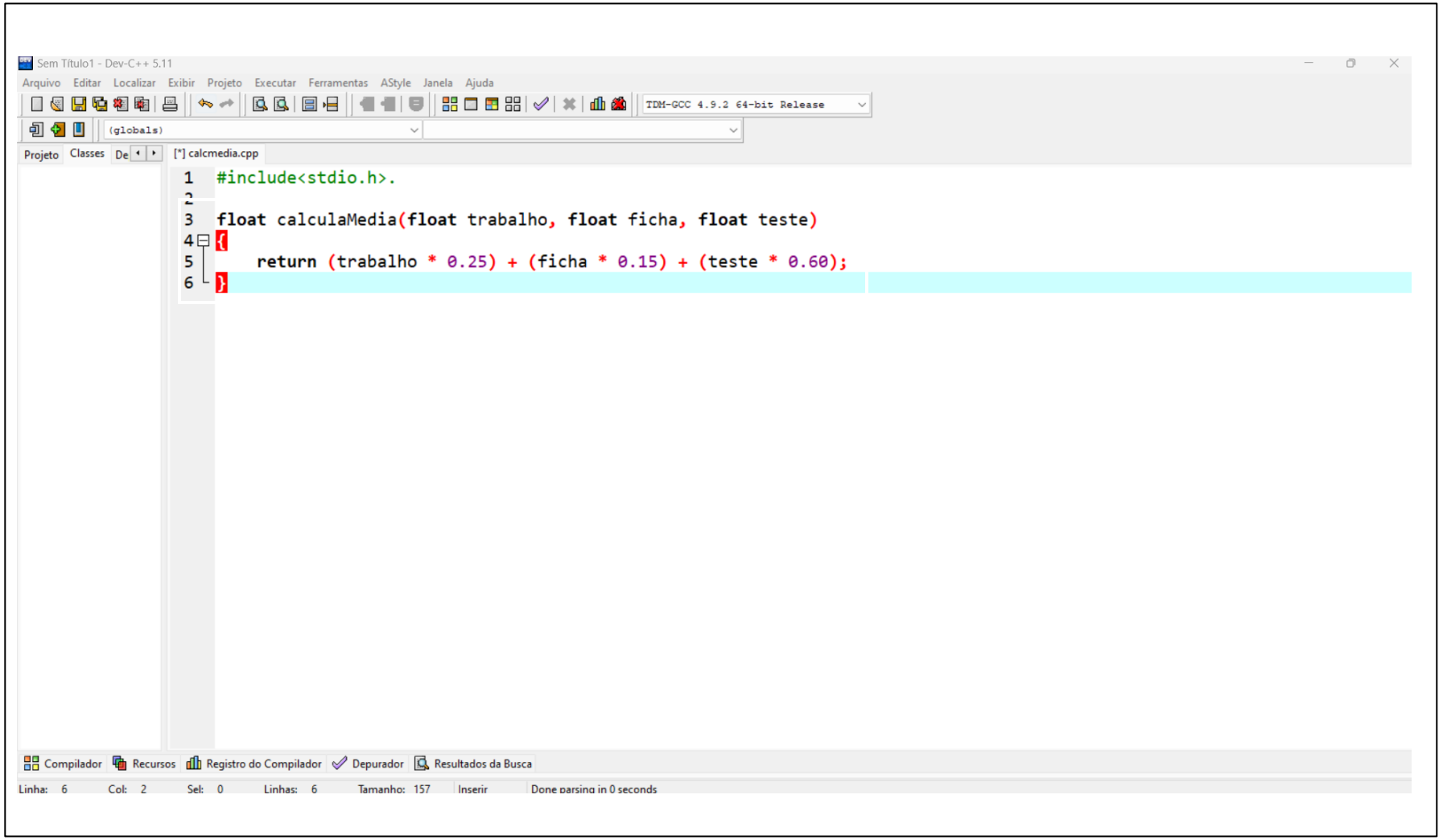
Passo 5 – Vamos agora salvar o ficheiro com o nome “calcmedia” numa pasta no ambiente de trabalho dentro da pasta “calculo1”



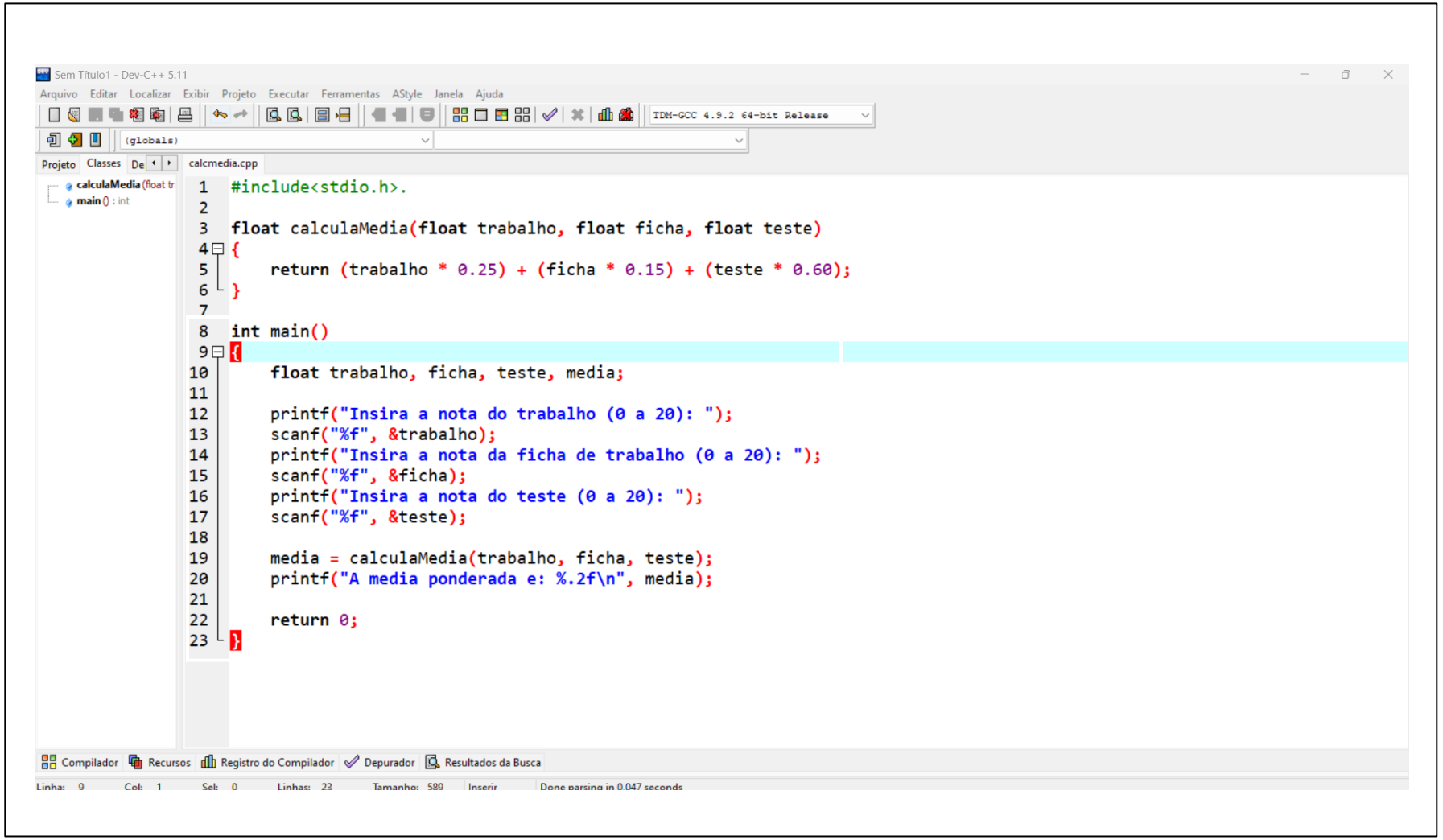
Passo 6 – No topo do programa do Dev C++ linha 1 devemos incluir a biblioteca - #include<stdio.h>



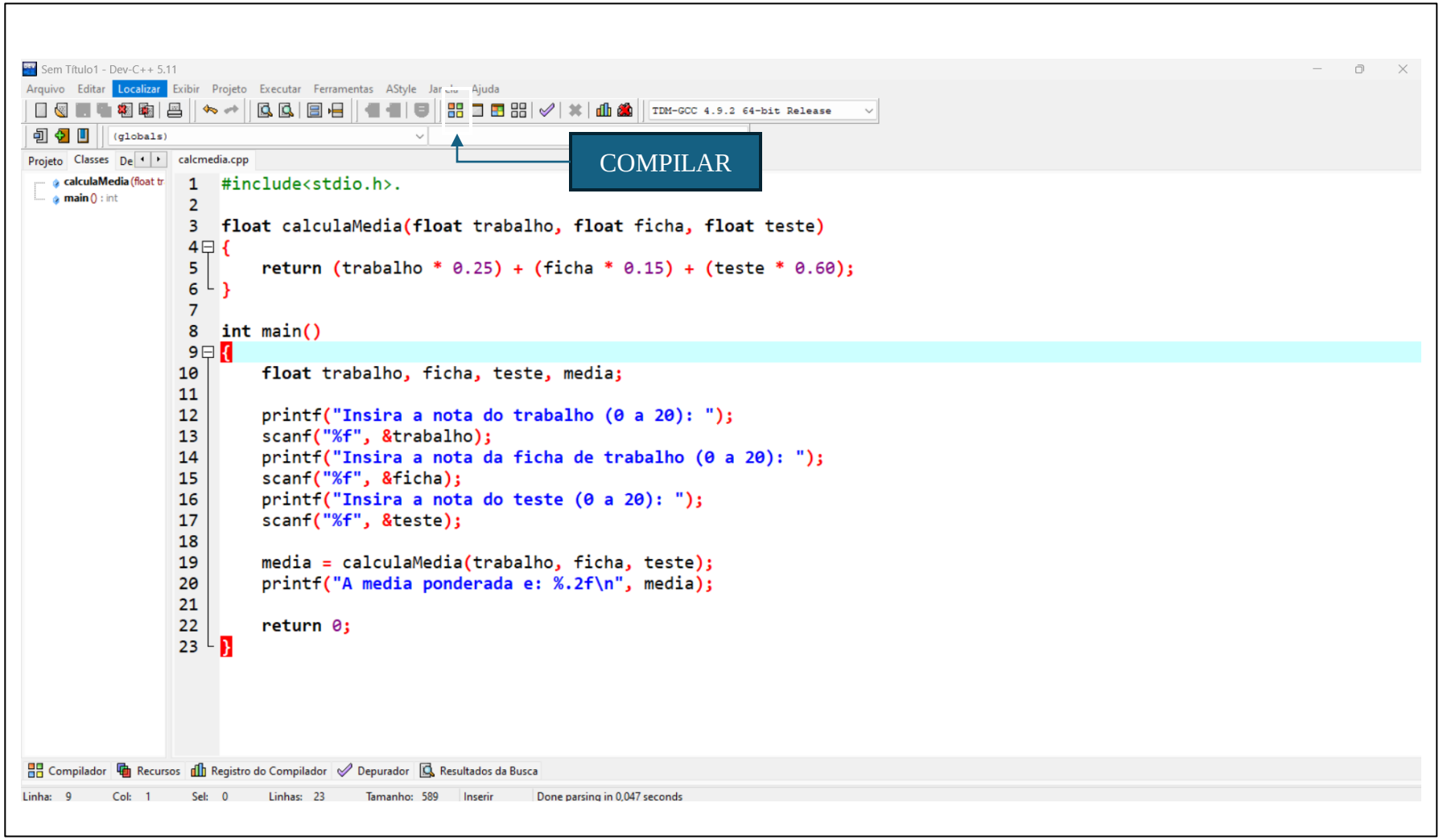
Passo 7 – Vamos declarar a função float calculaMedia(float trabalho, float ficha, float teste) antes da função main()



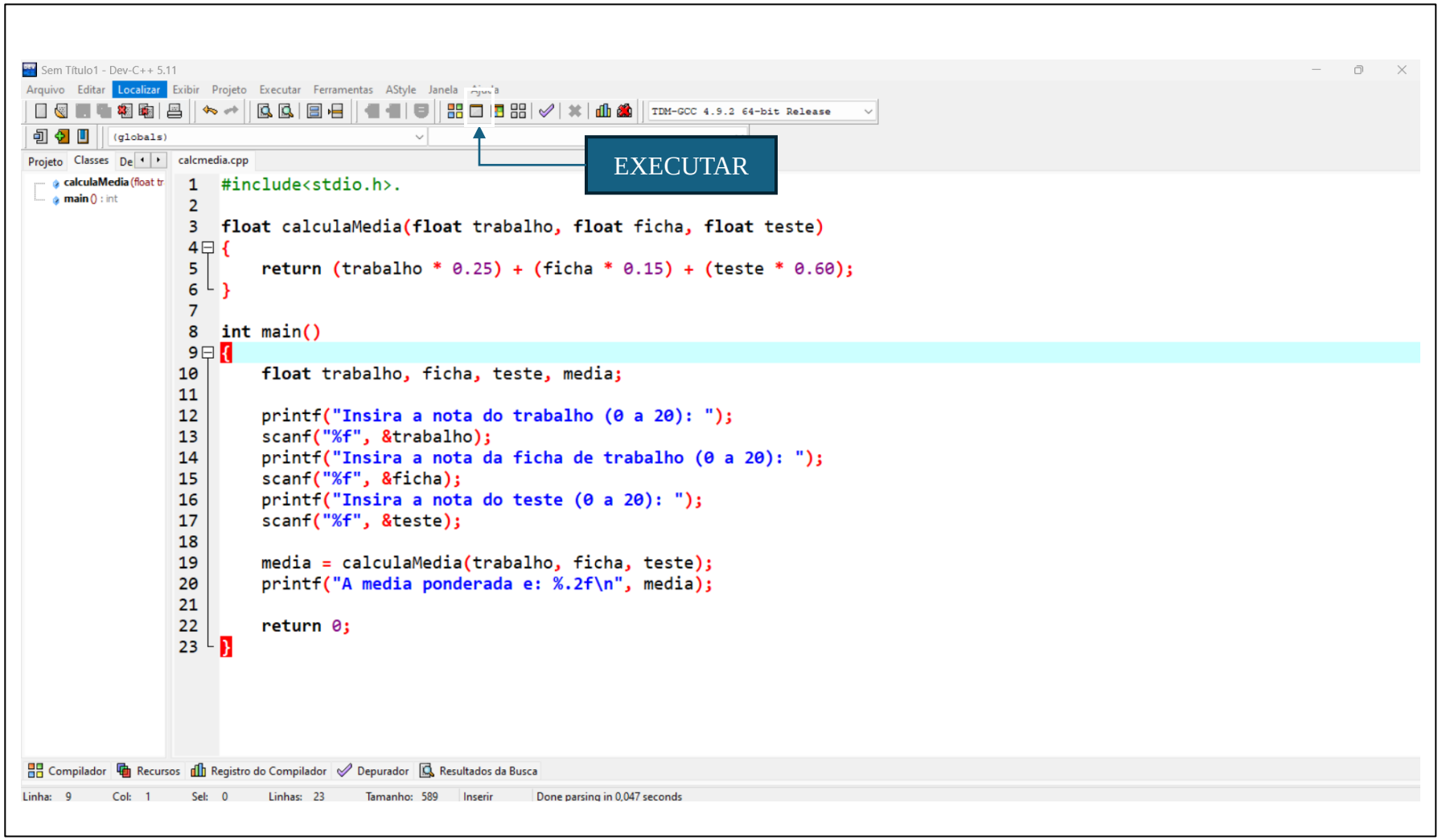
Passo 8 – Vamos agora escrever o resto do programa utilizando a função main()



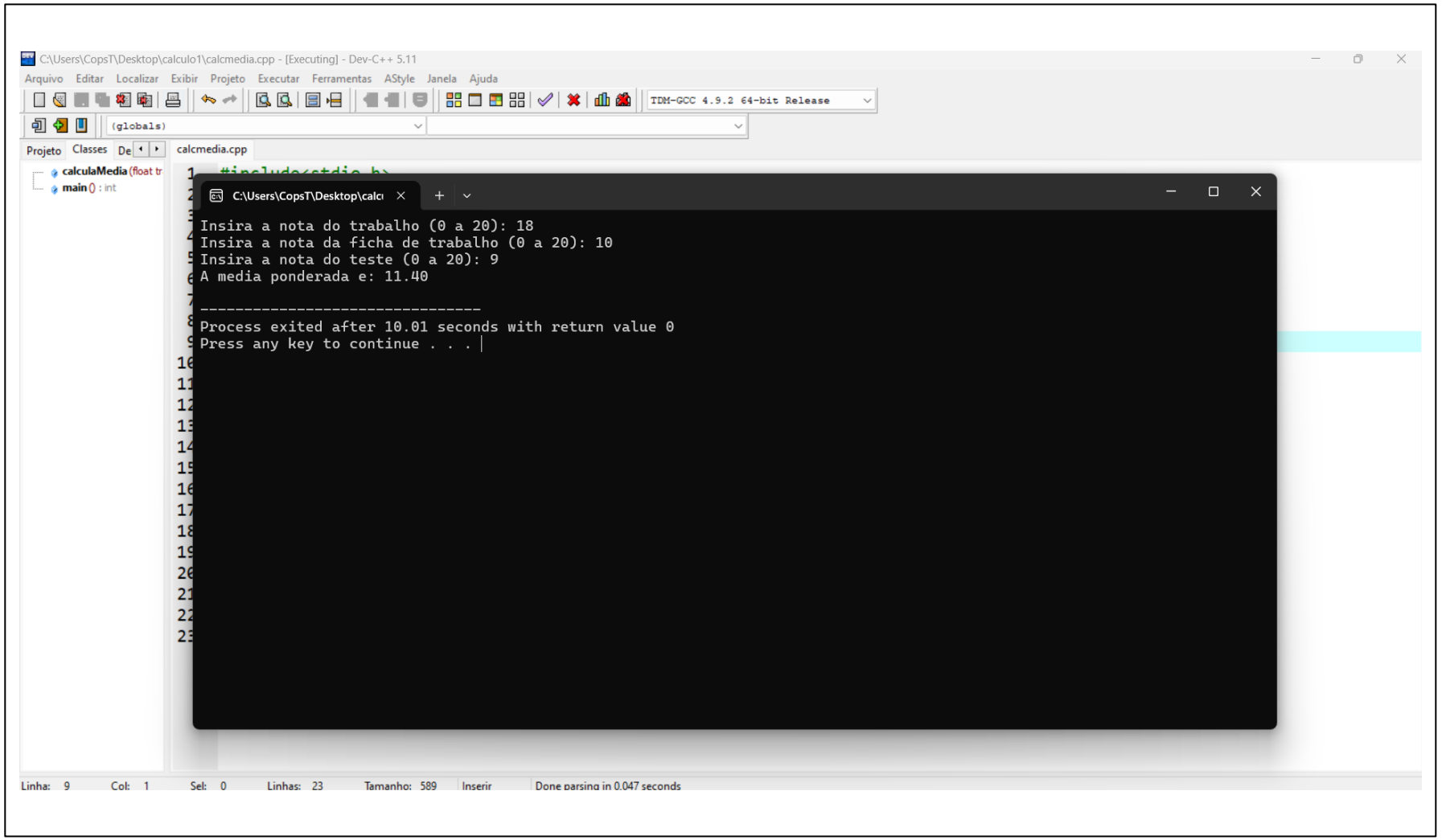
Passo 9 – Vamos agora compilar o programa para verificarmos se existem erros no código



Passo 10 – Vamos agora executar o programa e ver o resultado



Passo 11 – Vamos agora executar e ver a nossa consola



INICIO DO EXERCICIO 3 CONDUZIDO STEP BY STEP

FIM DA AULA 2 e 3