

Utilização do DEV-C++ 5.11

Para instalação da ferramenta DEV-C++ clique no link a seguir:

<https://sourceforge.net/projects/orwelldevcpp/>

O DevC++ é um ambiente de desenvolvimento de programas em C/C++ que utiliza o compilador o GNU g++ ou o GNU gcc.

Como iniciar a utilização ?

Para criar seu programa em C/C++ e compilá-lo com o DEV-C++, siga os passos apresentados abaixo.

No texto, assume-se que você esteja utilizando a versão 5.11 do DevC++, em português. Caso você esteja usando outra versão, talvez alguns dos comandos e/ou menus não sejam os mesmos descritos aqui.

Primeiramente inicie o programa clicando em seu ícone.



Figura 1 - Ícone do DEV-C++

Como criar um projeto ?

Depois que o DevC++ tiver sido carregado, abra o menu Arquivo e selecione a opção Novo/Projeto. Na janela que surge (figura abaixo), clique no ícone Console Application, defina um nome para o projeto e selecione a linguagem a ser usada (C ou C++), no nosso caso C. Depois realizar estes procedimentos, clique no botão OK.

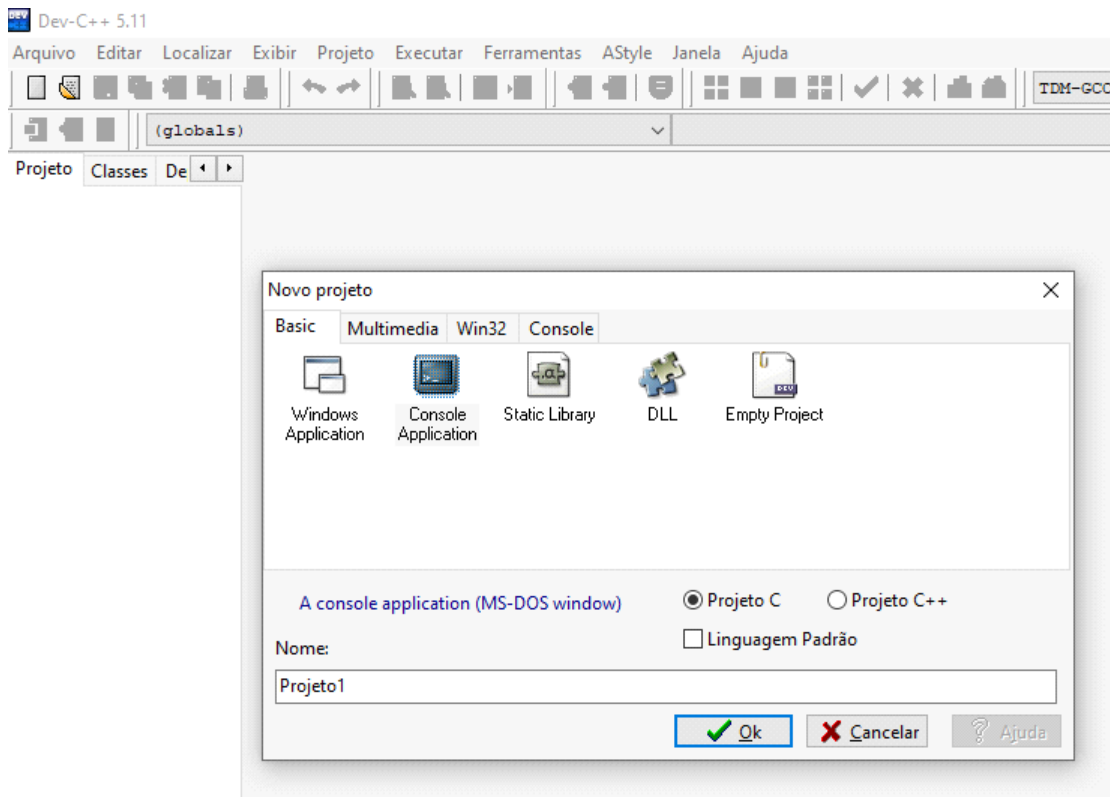


Figura 2 - Janela Project New

A seguir, o DevC++ solicita o nome do arquivo que irá guardar as informações do projeto. Defina um nome e salve o arquivo no diretório onde você desejar.

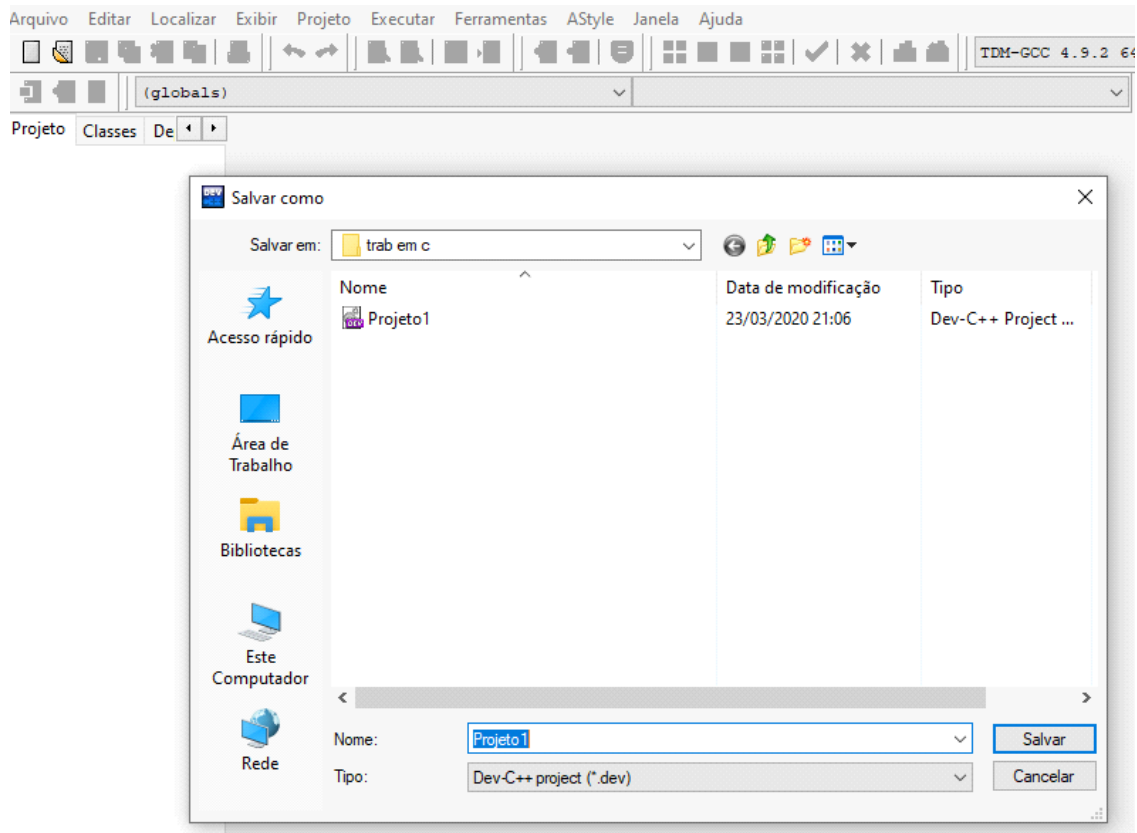


Figura 3 - Projeto Criado

Com isto o ambiente do DevC++ deverá parecer-se com a figura abaixo, na qual você deve clicar no sinal de “+” assinalado na figura. Isto abrirá uma árvore de pastas na qual você poderá colocar seus programas-fonte.

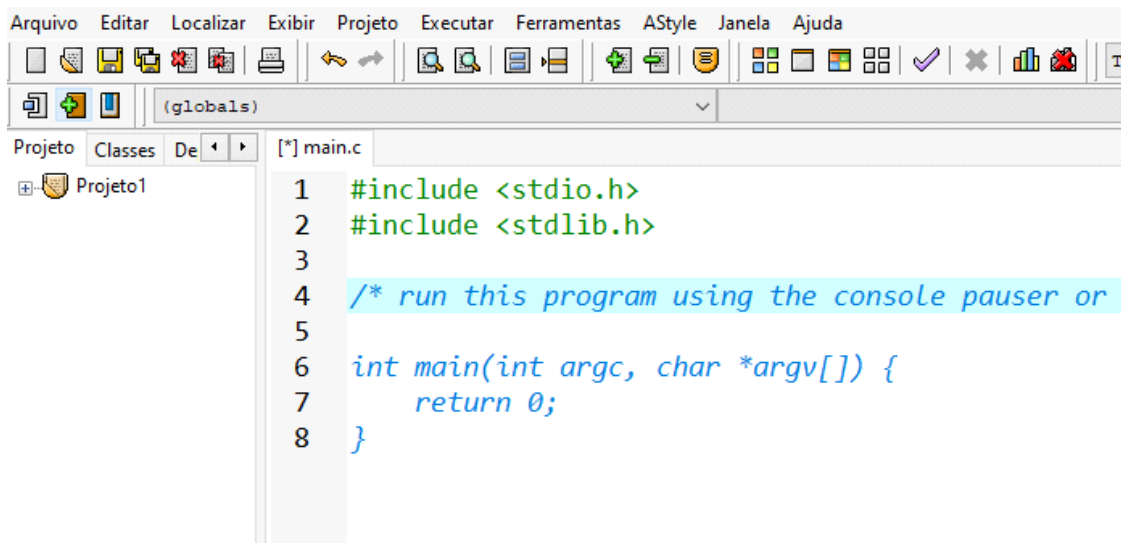


Figura 4 - Pasta projeto1

Ao clicar sobre o sinal de “+” surge o nome do arquivo “main.cpp” que está na área de edição do DevC++. Remova este arquivo do projeto clicando com o botão da direita do mouse sobre o arquivo (veja figura abaixo).

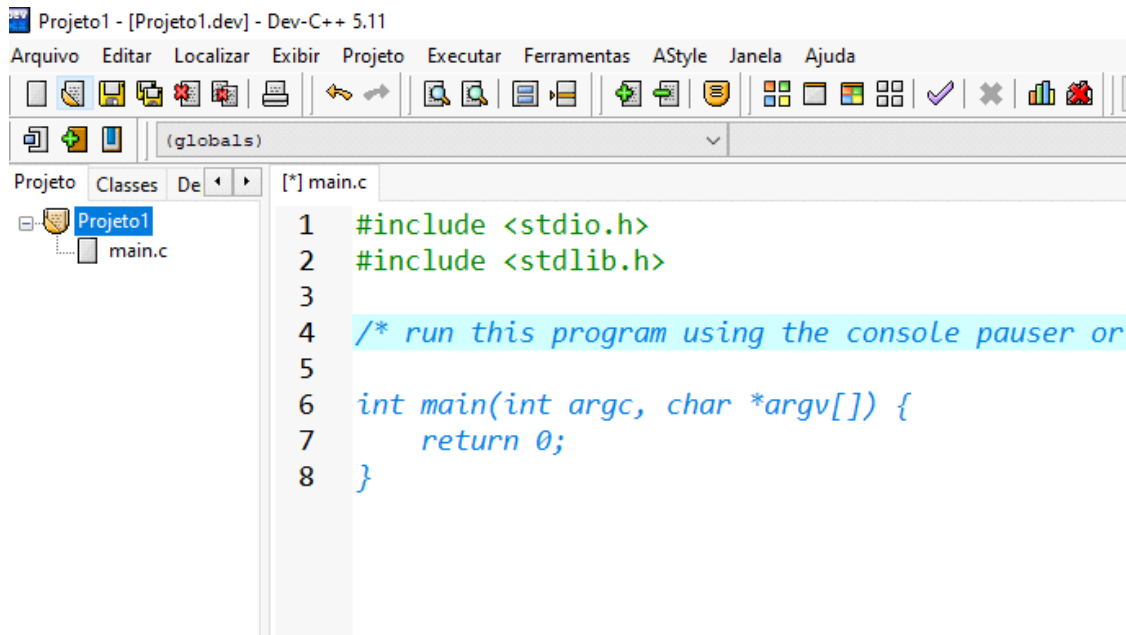


Figura 5 - Remoção do Arquivo main.c

Como Criar um Programa-Fonte ?

Para criar um programa-fonte, clique com o botão da direita sobre o nome do projeto e selecione o “Novo arquivo” ou “new file”. Isto cria um arquivo vazio e insere o mesmo no projeto. Tecele CTRL-S e salve o arquivo recém criado, com o nome programa1. Lembre-se de verificar se está a extensão.C no final do nome do arquivo.

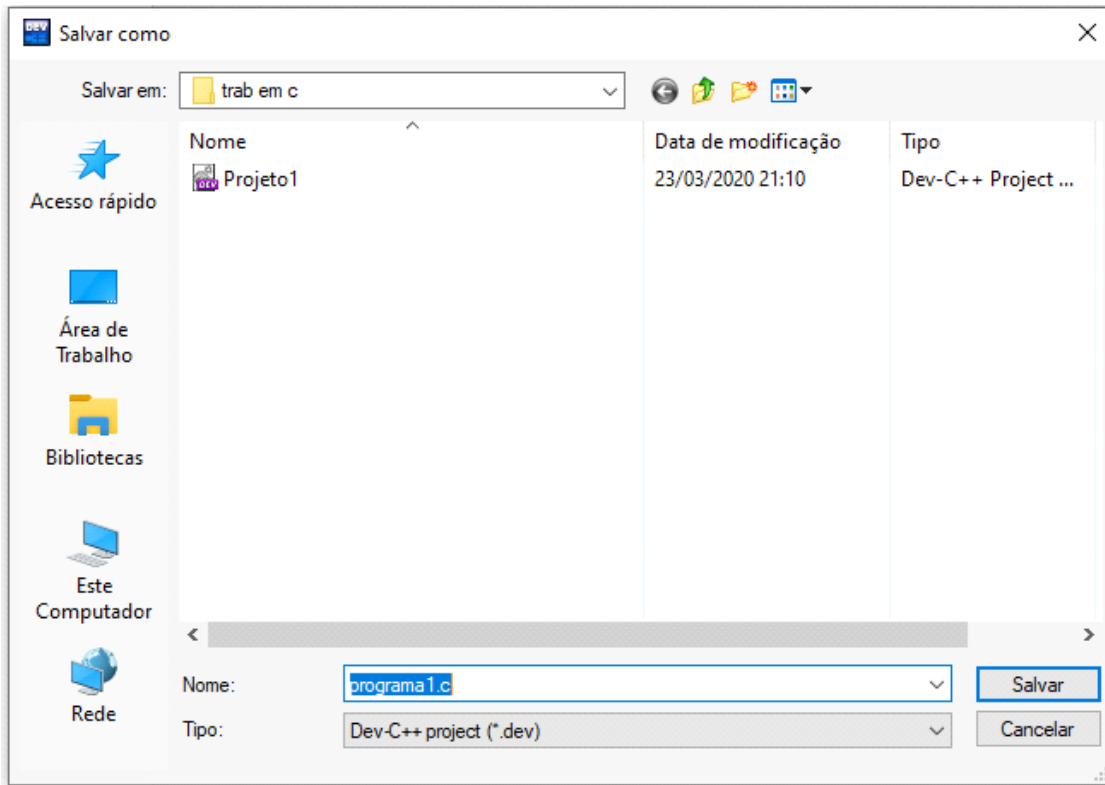


Figura 6- Criação de Arquivo

A partir deste ponto você pode digitar seu programa e testá-lo. A título de exemplo, digite o programa abaixo e salve o arquivo criado..

```
// *****  
  
//  Programa de teste  
  
// *****
```

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

```
int main()  
{  
    printf("Olá Mundo!\n");  
    system("PAUSE");
```

```
return 0;  
  
}
```

Ou seja, deve ser inserido conforme figura a seguir:

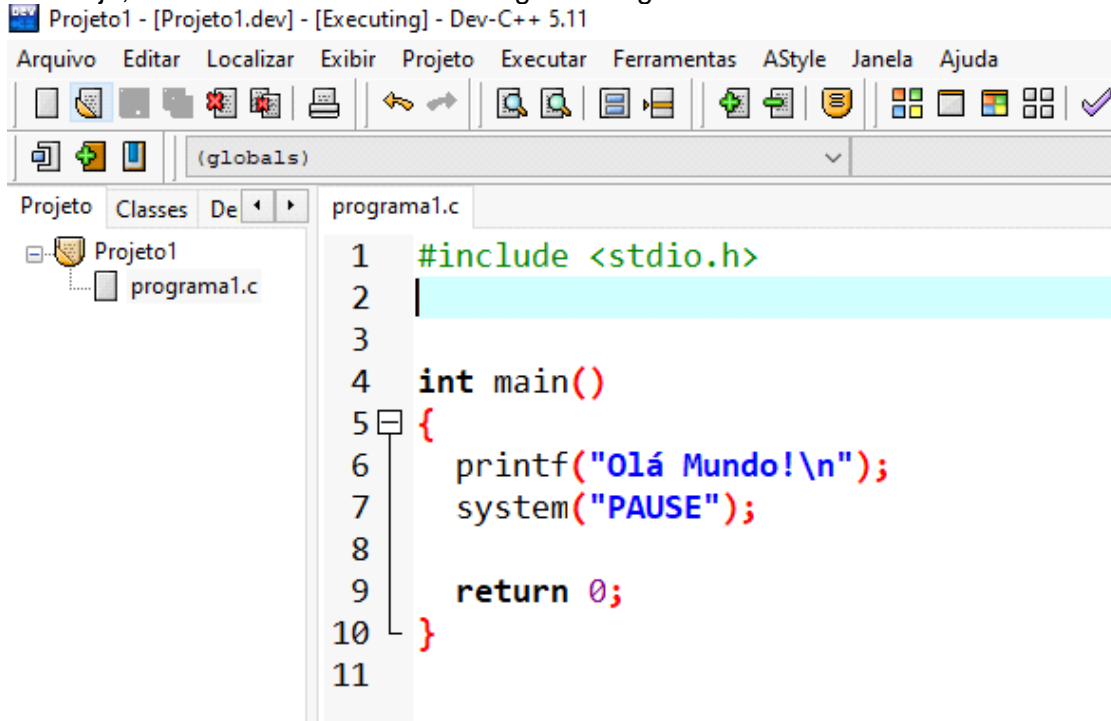


Figura 7 - Código fonte

Como Compilar e Executar um Programa ?

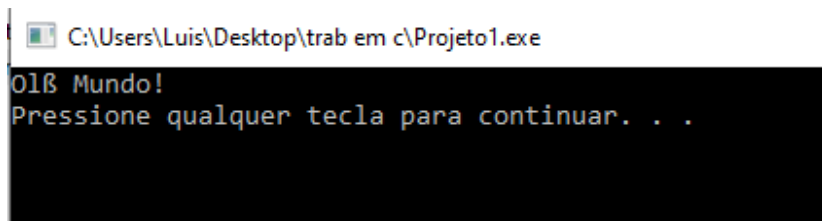
Para compilar e executar o programa, clique no botão de compilação (veja a figura abaixo).



Figura 8 - Compilar & Executar(F11)

Ou, na barra de menu **Executar** (compilar - F9).

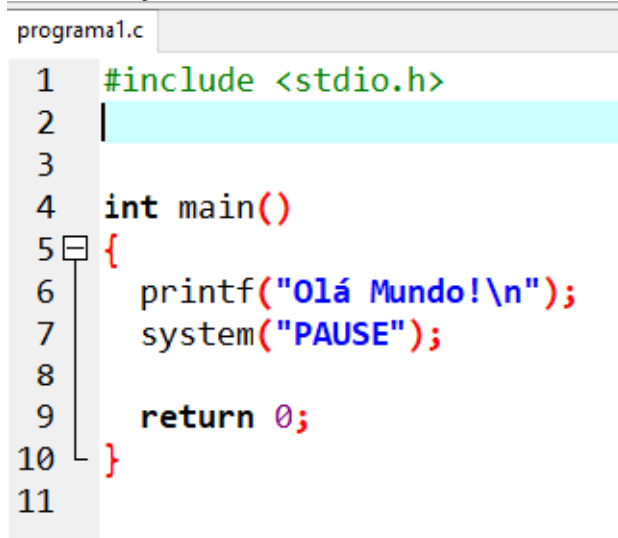
Vai surgir a tela de console de saída conforme figura a seguir:



```
C:\Users\Luis\Desktop\trab em c\Projeto1.exe
Olá Mundo!
Pressione qualquer tecla para continuar. . .
```

Figura 9 - Compilação do algoritmo

Observação:



```
programa1.c
1  #include <stdio.h>
2
3
4  int main()
5  {
6      printf("Olá Mundo!\n");
7      system("PAUSE");
8
9      return 0;
10 }
11
```

Figura 10 - Código fonte

Conforme figura 10, na linha 1 temos **#include <stdio.h>** o termo **#include** declaramos como parametrô para inserção da biblioteca do C que vem a seguir **<stdio.h>**. A biblioteca **stdio**, significa os comandos de **entradas** e **saídas** da linguagem C, que vem de **input/ output**.

Sempre que for necessário usar no código o comando de saída(**escreva()**), conforme linha 6 da figura 10 (**printf()**) devemos ter essa biblioteca declarada no cabeçalho antes da função principal main.

Na linha 4 a função **main()** é a função principal como em português estruturado o comando inicio/finalgoritmo. Entre chaves { } colocamos o algoritmo, que fizemos em portugues estruturado que agora estamos colocando em uma linguagem de programação que é linguagem C.

Na linha 7 temos a função **system(pause)** que é opcional, faz dar uma pausa na tela até o usuário do sistema teclar qualquer tecla para continuar.

```
C:\Users\Luis\Desktop\trab em c\programa1.exe
Olá Mundo!
Pressione qualquer tecla para continuar. . .
```

Figura 11 - Compilação

Na linha 9, ao final da função main() foi colocado um **return 0**. Isso serve para informar ao compilador que ocorreu tudo certo com a função main(). Se main retornasse um outro valor diferente de 0 haveria um problema em sua execução, que seria informada ao compilador.

```
programa1.c
1  #include <stdio.h>
2  #include <locale.h>
3
4  int main()
5  {
6      setlocale(LC_ALL, "portuguese");
7      printf("Olá Mundo!\n");
8      system("PAUSE");
9
10     return 0;
11 }
12
```

Figura 12 - Biblioteca locale.h

Conforme figura 12, na linha 2 foi inserido mais uma biblioteca <locale.h> Chamamos de “localizar” um programa quando fazemos a adaptação deste às características de um determinado idioma.

Veremos dois exemplos para permitir suporte a língua portuguesa. A utilização do arquivo **locale.h** e da função **setlocale()** configurada adequadamente vai garantir que caracteres como “ç” e acentuação sejam exibidos normalmente em nosso programa.

```
C:\Users\Luis\Desktop\trab em c\programa1.exe
Olá Mundo!
Pressione qualquer tecla para continuar. . .
```

Figura 13 - Acentuação com setlocale()

A seguir, um programa que pede para o usuário informar um valor do tipo inteiro e após mostrar esse valor para o usuário.

```
programa1.c
1  #include <stdio.h>
2  #include <locale.h>
3
4  int main()
5  {
6      setlocale(LC_ALL, "portuguese");
7      int a;
8      printf("Informe um número inteiro: ");
9      scanf("%d", &a);
10
11     printf("Soma = %d \n", a);
12
13     system("PAUSE");
14
15     return 0;
16 }
```

Figura 14 - Função scanf()

Na figura 14, temos na linha 7, a declaração da variável **a** do tipo inteiro (primeiro colocamos o tipo e depois o identificador, ou seja, o nome da variável).

Na linha 9, temos um comando `scanf()`, ou seja, `scanf()`, que permite armazenar por tempo de execução o valor do tipo inteiro nessa variável **a**.

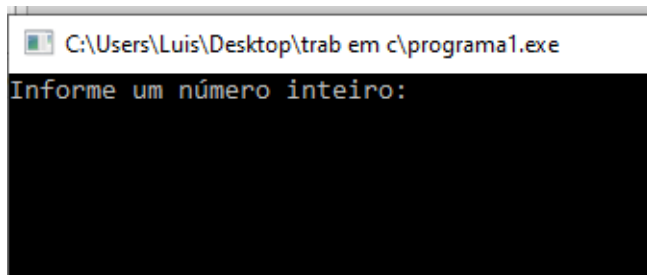
Sintaxe:

```
scanf("tipo_de_dado_armazenado", nome_variavel)
```

O programa acima usa a função `scanf` para obter um valor para a variável **a**, a partir dos caracteres do teclado.

Espera-se que sejam fornecidos como parâmetros o **endereço das variáveis** (`&a`) onde devem ser armazenados os valores obtidos no fluxo de entrada.

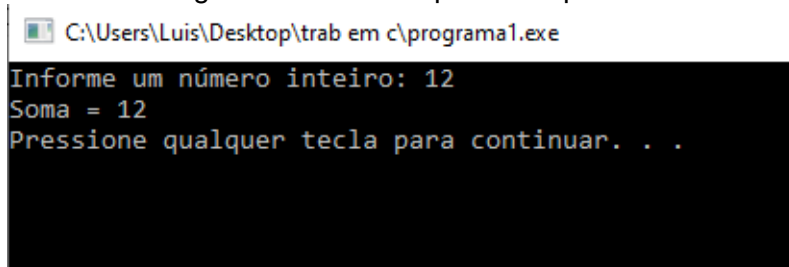
Um erro muito comum de programadores que usam a função `scanf` é esquecer de utilizar o caractere "e comercial" (`&`) antes do nome da variável que deve receber o valor convertido a partir dos caracteres do fluxo de entrada, significando mostrar onde fica o endereço dessa variável na memória do computador.



```
C:\Users\Luis\Desktop\trab em c\programa1.exe
Informe um número inteiro:
```

Figura 15 - Comando printf()

A figura 15, mostra o comando de saída na linha 8 conforme figura 15. Aguarda até o usuário do sistema digitar um valor do tipo inteiro para ser armazenado na variável.



```
C:\Users\Luis\Desktop\trab em c\programa1.exe
Informe um número inteiro: 12
Soma = 12
Pressione qualquer tecla para continuar. . .
```

Figura 16 - Compilação do programa

A linguagem C define duas funções que podem ser usadas para escrever e ler valores. Os valores são escritos em um chamado “fluxo de saída” que normalmente corresponde a um dispositivo referido como tela ou console. Os valores são lidos de um chamado “fluxo de entrada” que normalmente corresponde a um dispositivo referido como teclado.

Exercícios de fixação

- Elabore um programa que solicite para o usuário inserir dois valores inteiros e mostre na tela ao executar a soma desses dois valores.
- Elabore um algoritmo que leia o nome do usuário e a qualidade dessa pessoa, e exiba na mensagem "< nome > é uma pessoa que tem <qualidade>".