

**UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

Semana 02 - Ambiente de Programação Linux

Professor: Éder Alves de Moura
Aluno: Wander Victor Verçosa Mares - 11811EAU010

Uberlândia
18/12/2021

Questão 1

Pré-processador - Nesta etapa são removidos comentários, inclusão de arquivos (include, define, etc) de cabeçalho no código fonte e substituição do nome da macro pelo código. A saída desta etapa é um código “puro” sem diretivas de pré-processador.

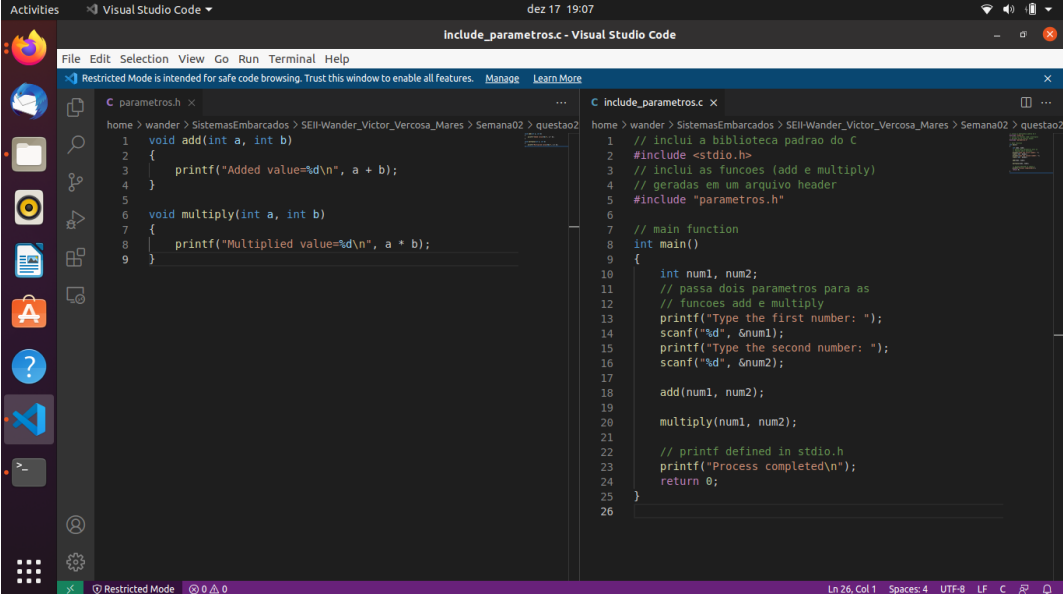
Compilação - O compilador pega a saída da etapa anterior e produz um arquivo objeto a partir dela. Dessa forma, por exemplo, um arquivo “HelloWolrd.c” é traduzido em um “a.out” por exemplo.

Montador - Nesta etapa, o arquivo anterior é traduzido em linguagem de máquina e se torna um arquivo objeto.

Ligante - Nesta etapa são vinculados os arquivos de objeto para produção de um arquivo executável final, pode ser criado também uma biblioteca ou arquivo header, que será melhor exemplificado na Questão 2.

Questão 2

A aplicação consiste em definir um código header que formule os cálculos a serem efetuados e em seguida um código principal que irá incluir os parâmetros a serem calculados via inserção do usuário.

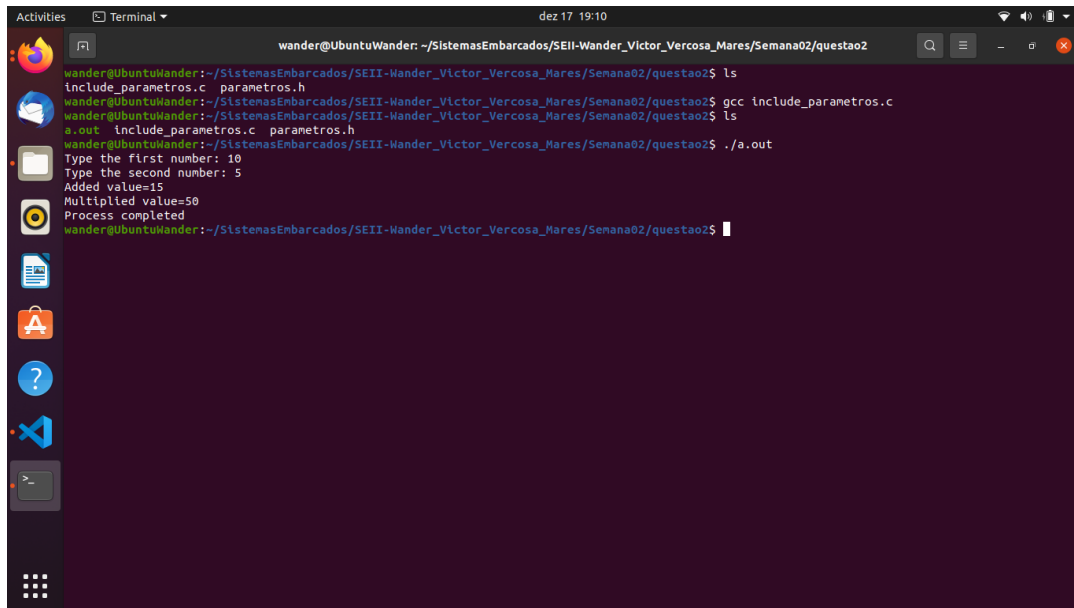


The screenshot shows the Visual Studio Code interface with two files open. The left file, 'parametros.h', contains two function declarations: 'void add(int a, int b)' and 'void multiply(int a, int b)'. The right file, 'include_parametros.c', contains the implementation of these functions. It includes the standard C library, the header file, and a main function that prompts the user for two numbers, calls the 'add' and 'multiply' functions, and prints the results.

```
home > wander > SistemasEmbarcados > SEII-Wander_Victor_Vercosa_Mares > Semana02 > questao2
1 void add(int a, int b)
2 {
3     printf("Added value=%d\n", a + b);
4 }
5
6 void multiply(int a, int b)
7 {
8     printf("Multiplied value=%d\n", a * b);
9 }

home > wander > SistemasEmbarcados > SEII-Wander_Victor_Vercosa_Mares > Semana02 > questao2
1 // inclui a biblioteca padrao do C
2 #include <stdio.h>
3 // inclui as funcoes (add e multiply)
4 // geradas em um arquivo header
5 #include "parametros.h"
6
7 // main function
8 int main()
9 {
10     int num1, num2;
11     // passa dois parametros para as
12     // funcoes add e multiply
13     printf("Type the first number: ");
14     scanf("%d", &num1);
15     printf("Type the second number: ");
16     scanf("%d", &num2);
17
18     add(num1, num2);
19
20     multiply(num1, num2);
21
22     // printf defined in stdio.h
23     printf("Process completed\n");
24     return 0;
25 }
26
```

Figura 1: função header “parametros.h” e função principal “include_parametros.c”.



```
wander@UbuntuWander: ~/SistemasEmbarcados/SEII-Wander_Victor_Vercosa_Mares/Semana02/questao2
wander@UbuntuWander:~/SistemasEmbarcados/SEII-Wander_Victor_Vercosa_Mares/Semana02/questao2$ ls
include_parametros.c  parametros.h
wander@UbuntuWander:~/SistemasEmbarcados/SEII-Wander_Victor_Vercosa_Mares/Semana02/questao2$ gcc include_parametros.c
wander@UbuntuWander:~/SistemasEmbarcados/SEII-Wander_Victor_Vercosa_Mares/Semana02/questao2$ ls
a.out  include_parametros.c  parametros.h
wander@UbuntuWander:~/SistemasEmbarcados/SEII-Wander_Victor_Vercosa_Mares/Semana02/questao2$ ./a.out
Type the first number: 10
Type the second number: 5
Added value=15
Multiplied value=50
Process completed
wander@UbuntuWander:~/SistemasEmbarcados/SEII-Wander_Victor_Vercosa_Mares/Semana02/questao2$
```

Figura 2: execução do código “include_parametros.c” via terminal.

Os arquivos citados nesta questão, encontram-se em anexo junto a pasta no Git.

Questão 3

a) **-static**: essa opção entra em jogo quando o compilador vincula arquivos-objetos em um arquivo de saída executável. Este parâmetro não tem sentido se o compilador não estiver realizando uma etapa de link (ligante).

b) **-g**: produz informações de depuração no formato nativo do sistema operacional (stabs, COFF, XCOFF ou DWARF).

c) **-pedantic**: emite todos os avisos exigidos pela ISO C; rejeita todos os programas que usam extensões proibidas e alguns outros que não seguem a ISO C. Os programas que fazem parte da ISO serão compilados tranquilamente sem esta extensão.

d) **-Wall**: isso ativa todos os avisos sobre construções que alguns usuários consideram questionáveis e fáceis de evitar (ou modificar para evitar o aviso), mesmo em conjunto com macros. Isso também permite algumas linguagens específicas, presentes no manual.

e) **-Os**: os parâmetros “o” otimizam a compilação reduzindo o tamanho do código e o tempo de execução. O parâmetro “-Os” permite todas as otimizações do “-O2” exceto aquelas que frequentemente aumentam o tamanho do código.

f) -O3: Otimiza todas as especificações descritas por “O2” e também ativa um grupo de sinalizadores descritos no manual.