arr

**Réalisé par : SIRGIANE Ouiçal Examiné par : PR.R. HANNANE**

**BOUDAINE Salma**

**Atelier 1**

**Rapport sur :** Compilation et programmation modulaires

Objectifs :

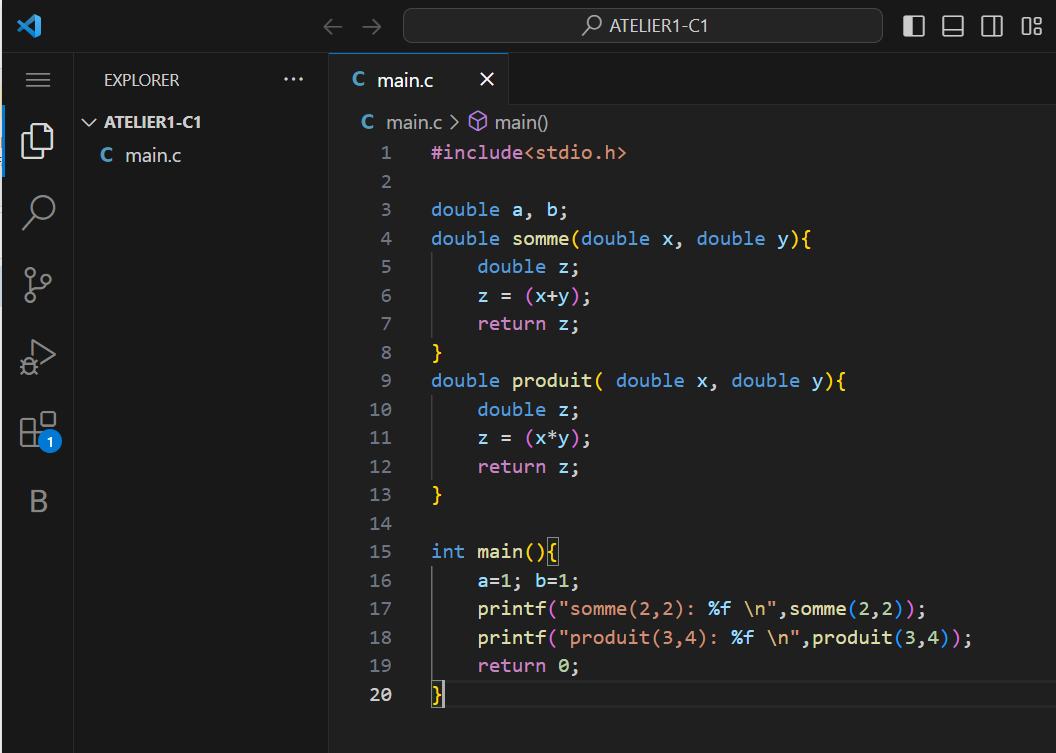
1. Compilation et Edition des liens.
2. Programmation modulaire et compilation séparée.
3. Fichiers d’entête (.h) et Bibliothèques (statiques .a ou dynamiques .so).

**Réalisé par : SIRGIANE Ouiçal Apo : 2020941 Examiné par : El Hassan Abdelwahed**

*2023/2024*

1. **Compilation et édition des liens :**

Soit le programme suivant saisi dans VS Code :



En utilisant le programme gcc pour le compiler et ensuite pour l’édition des liens afin générer le programme exécutable.

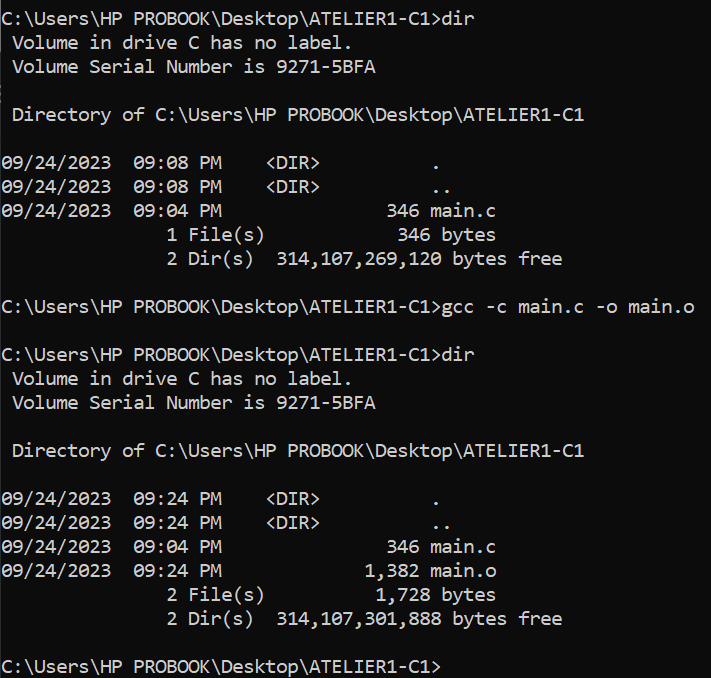
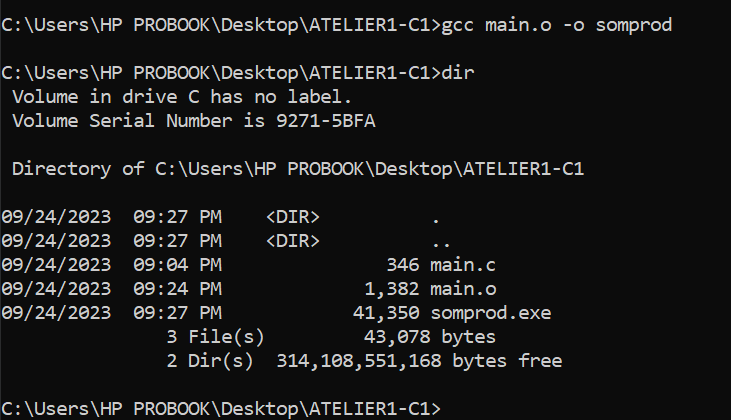


Figure 2 : Edition des liens et generation du programme executable.

Figure 1: Compilation du programme source.

Sur la ligne des commandes (Linux ou Windows), il suffit d’exécuter :

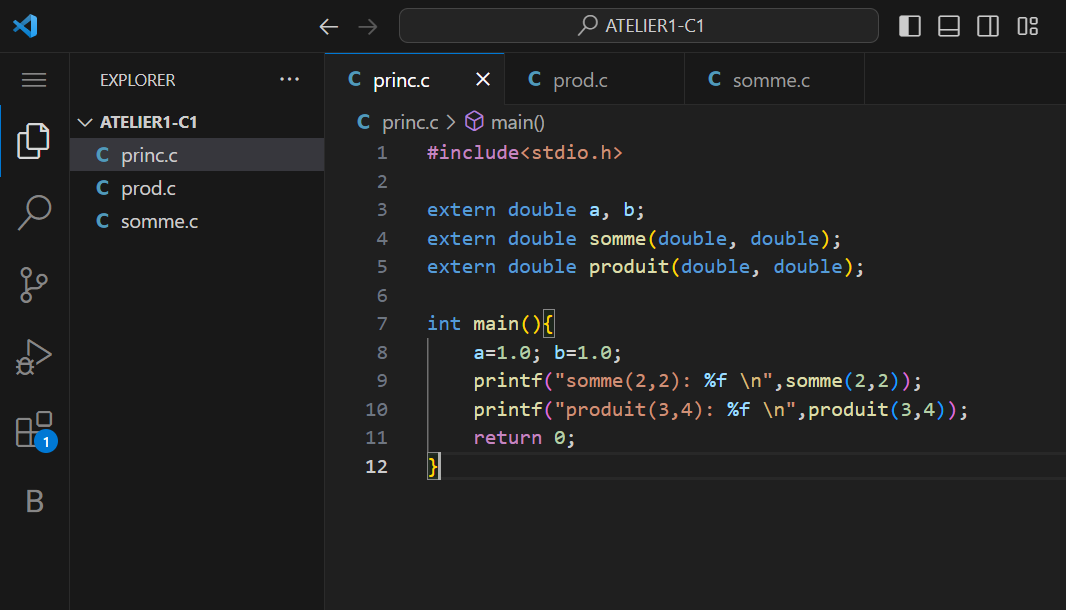
* gcc –c main.c –o main.o
* Pour compiler le programme main.c et générer le fichier (main.o) contenant le programme compilé :
* gcc main.o –o somprod
* Pour faire l’édition des liens (lier le programme main.o avec les autres bibliothèques, …) et le programme exécutable (somprod.exe)

On peut aussi simplement exécuter (compilation suivie de l’édition des liens) :

* gcc main.c –o somprod

1. **Programmation modulaire et compilation séparée :**

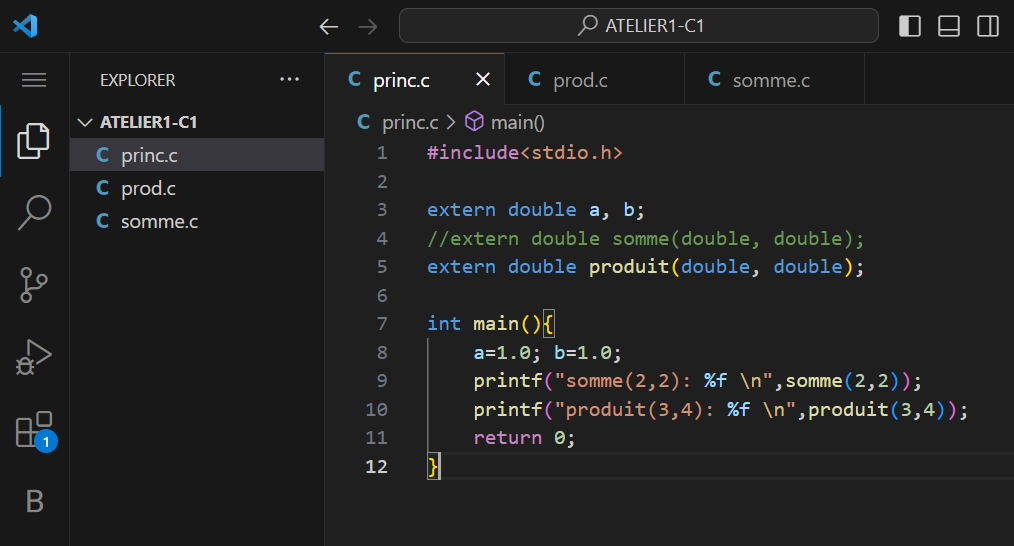
On va répartir le programme précédent (main.c) sur trois fichiers (somm.c, prod.c et princ.c) contenant chacun une fonction (module) donnée : la fonction de calcul de la somme, la fonction de calcul du produit et le programme principal.



Qu’est-ce qu’une variable externe ?

Ce qui signifie qu'elle est définie ailleurs dans le programme. Cela permet d'accéder à ces variables ou fonctions à partir de différents fichiers source, facilitant ainsi la modularité et la séparation du code.

* Mettez la ligne 4 (et/ou la ligne 5) du programme (princ.c) en commentaire. Exécutez de nouveau le programme. Les résultats après exécution du programme sont-ils corrects ? Expliquez.



Sur la ligne des commandes (Linux ou Windows), il suffit d’exécuter :

* Compilation des trois programmes :

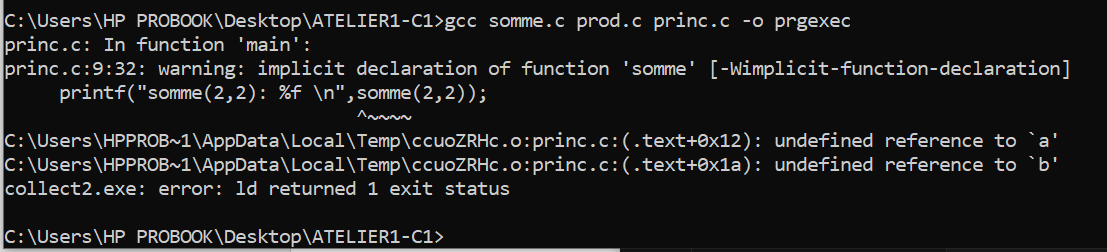
gcc –c somm.c –o somm.o

gcc –c prod.c –o prod.o

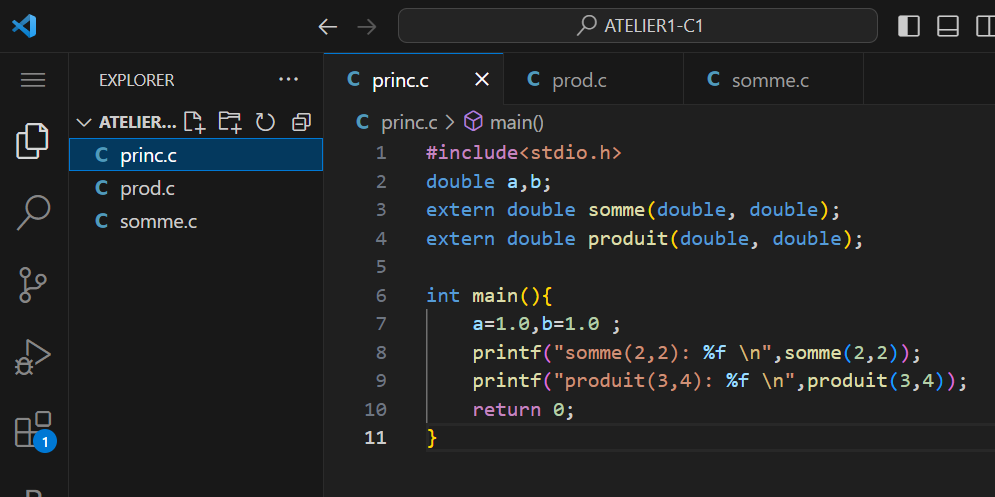
gcc –c prin.c –o princ.o

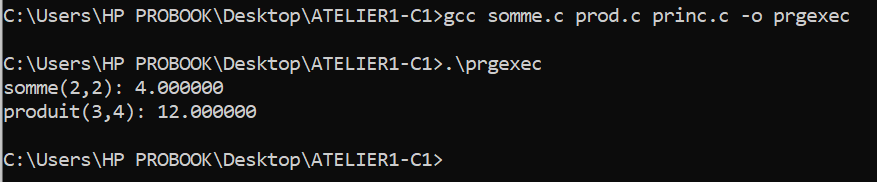
* Edition des liens et génération du programme exécutable prgexec :

gcc somm.o prod.o princ.o – prgexec

On peut aussi simplement exécuter la commande ci-dessous pour compiler les trois fichiers suivi de l’édition des liens pour générer le programme exécutable : - gcc somm.c prod.c princ.c –o prgexec

* On observe qu’on a deux problèmes : le premier c’est que lorsque nous avons rendre la ligne 4 comme commentaire on a le problème dans le main qui na pas connait la fonction somme. Le deuxième problème c’est que on a déclarer les deux variables a et b dans un fichier et on les utilise dans un autre.
* J’ai enlevée seulement le mot extern et le commentaire le programme marche bien.





1. **Fichiers d’entête (.h) et Bibliothèques (statiques .a ou dynamiques .so) :**

* Il ne faut pas confondre les fichiers d’entête (.h) introduits par l’instruction #include comme par exemple #include et les bibliothèques ayant des extensions .a pour les librairies statiques et .so pour les librairies dynamiques.
* La différence entre les deux résidents dans le fait que les bibliothèques statiques sont incorporées à l’exécutable final, alors que les bibliothèques dynamiques sont chargées lors de son exécution.
* Un fichier d’entête est un fichier texte qui contient principalement des déclarations des variables, des définitions de structures de données et des déclarations de fonctions (signatures de fonctions). Une bibliothèque (librairie) est en quelque sorte un regroupement d’une collection de programmes (fonctions) compilés.

On va reprendre l’exemple du programme précédent en créant deux fichiers d’entête (ve.h et vg.h) et une bibliothèque (libsmi.a) :

* ve.h : contient la déclaration des variables externes
* vg.h : contient la déclaration des variables globale

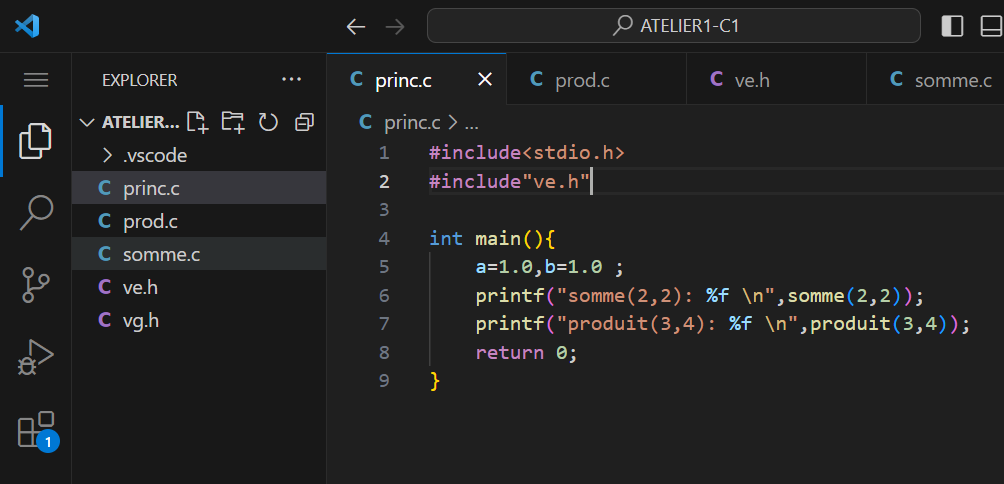


Figure 3 : le programme principal princ.c

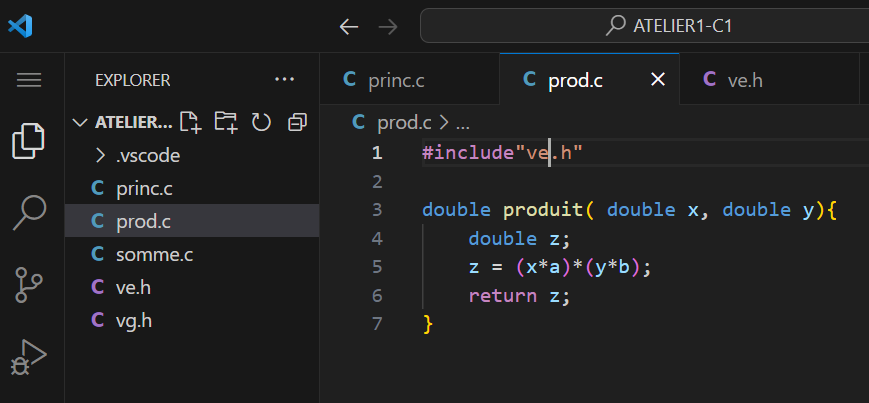
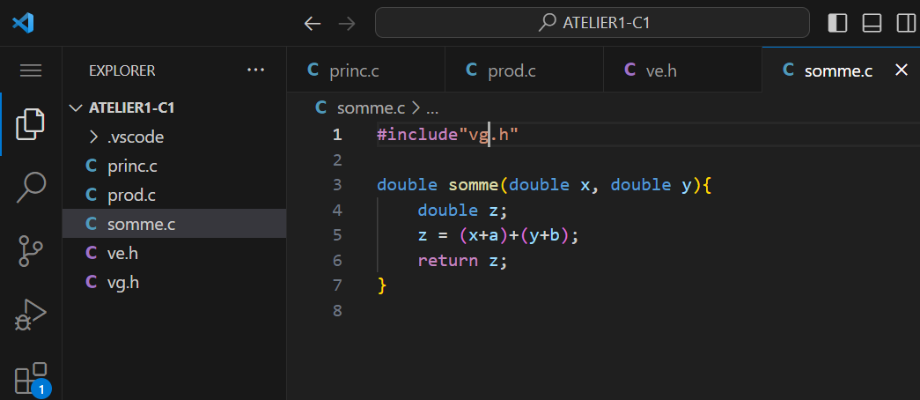


Figure 4: les deux fichiers prod.c et somme.c

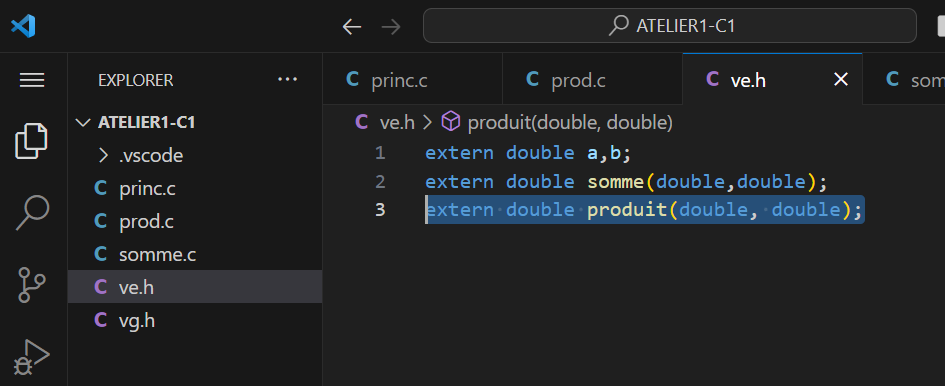


Figure 5 : le fichier header ve.h

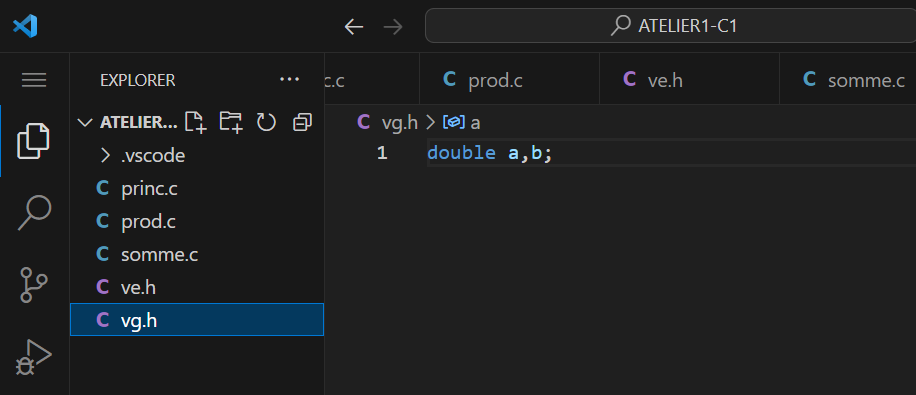
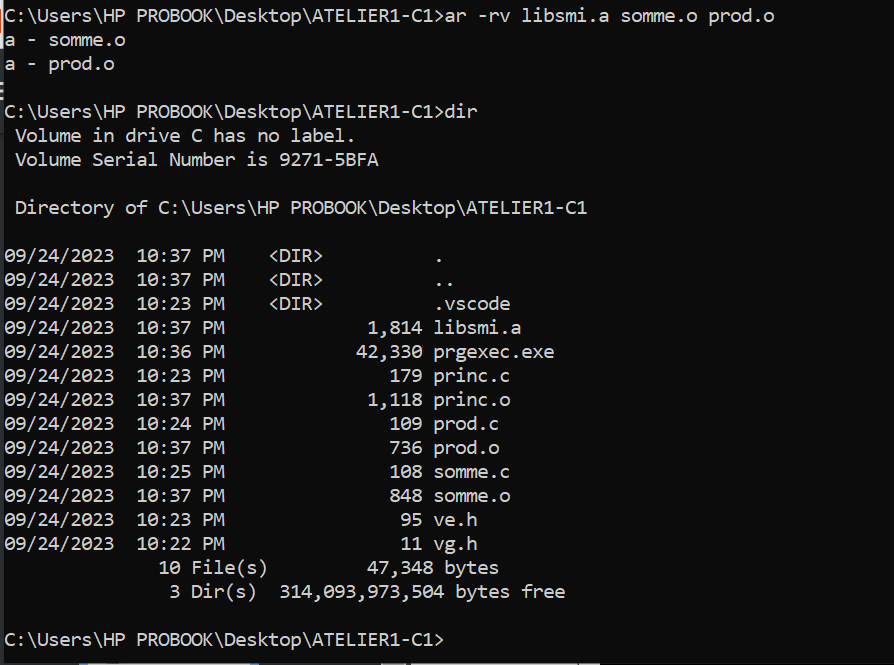


Figure 6 : le fichier header vg.h

On va créer une librairie libsmi.a regroupant les deux programmes compilés (somm.o et prod.o) contenant des fonctions (fonctions somme( ) et produit( )) autres que programme principal (contenant la fonction main( )).

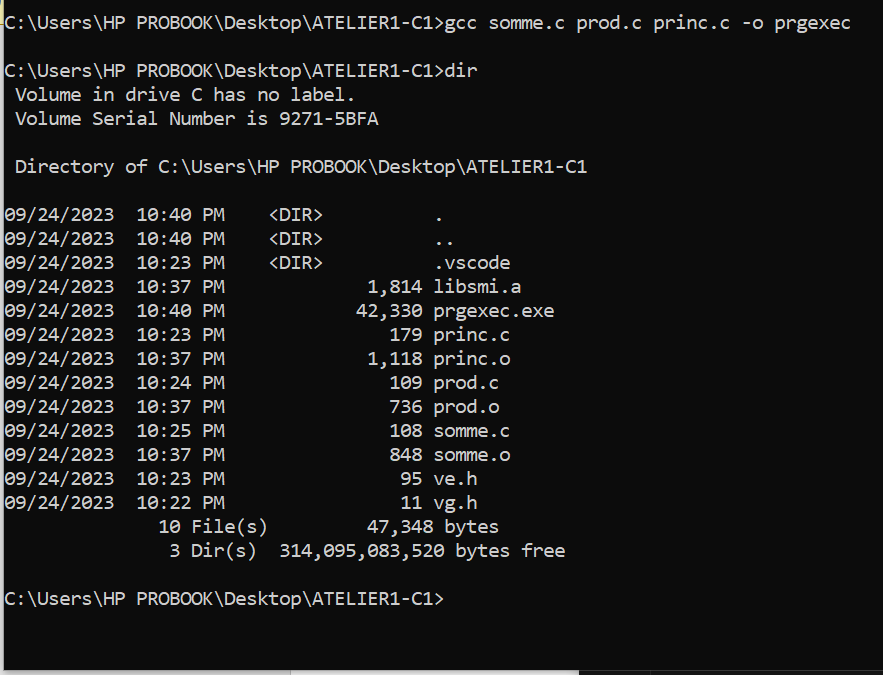
La commande **ar** pour créer une librairie et pour créer libsmi.a en y ajoutant les deux fichiers somm.o et prod.o où l’option **–r** permet d’insérer (archiver) des fichiers objets dans la librairie et **–v** permet d’afficher à l’écran les noms des fichiers insérés :

**ar –rv libsmi.a somm.o prod.o**



La commande précédente pour l’édition des liens et la génération du programme exécutable :

* gcc somm.c prod.c princ.c –o prgexec



On va créer dans le dossier de notre projet un dossier include dans le lequel nous allons déplacer nos propres fichiers d’entête (ve.h et vg.h). Dans nos programmes nous allons changer l’instruction #include «ve.h» (indiquant que fichier ve.h est dans le dossier courant) par #include indiquant au système qu’il doit chercher le fichier ve.h dans le dossier spécifier :

