Soit le programme multi.c **écrit en langage C suivant :**

```
#include <stdio.h>
#include <stdlib.h>
#define scanf_s scanf

int main(void)
{
  int num, multip, count=0, product=0;
  printf("This program will give you the product of two numbers. Give me the
numbers:\n");
  scanf_s("%i", &num);
  scanf_s("%i", &multip);
  while (count < multip)
  {
            product += num;
            count++;
  }
  printf("The product of %i * %i is: %i\n", num, multip, product);
  return EXIT_SUCCESS;
}
```

1. Donner la ligne de commande de gcc permettant de créer le binaire appelé multi

Remarque : le programme prenant l'argument void entraîne l'obligation d'utiliser l'option -ansi, car void est défini dans la norme ansi du C. Les options -Wall (afficher tous les avertissements de compilation) et -std=c99 ne sont pas obligatoires.

gcc -Wall -ansi -std=c99 -o multi sources/multi.c

En jaune : -o multi qui donne le NOM du binaire final.

Attention avec cette option -o (qui signifie output), car le nom qui suit SERA le nom du binaire après la compilation (sauf si elle échoue). L'erreur souvent constatée, est d'écrire : -o multi.c (le fichier .c est écrasé et transformé en binaire : on a tout perdu !!). C'est la raison pour laquelle on place toujours le binaire dans un autre répertoire que les sources.

2. Que fait la fonction scanf_s() ?

La fonction scanf_s va LIRE un paramètre. C'est donc une fonction d'entrées/sorties (i/o en anglais).

3. combien d'arguments ou paramètres prend la fonction main ?

void (norme ansi), signifie "rien". Donc la fonction main prend 0 arguments.

4**. Que fait ce programme ?**

Ce programme réalise successivement les opérations suivantes :

- il demande à l'utilisateur de saisir 2 nombres entiers
- il lit les valeurs entières données par l'utilisateur
- il calcule le produit des deux nombres entiers saisis
- il écrit le résultat à l'écran avant de se terminer.

5. écrire l'algorithme associé

```
Déclaration des nombres entiers num, multip, count et product

DÉBUT programme  principal

        FAIRE
                num      <--  0  // "affecter la valeur entière 0 à num" conviendrait aussi
                multip   <-- 0
                count    <-- 0
                product <-- 0

        ÉCRIRE
                "This program .... "

        LIRE num
        LIRE multip

        TANT QUE  count est inférieur à multip
                FAIRE
                        product <--  product + num
                        count    <--   count + 1
        FIN TANT QUE

        ÉCRIRE "le produit de " num  " multiplié par " multip " vaut " product

FIN programme principal
```