

TP N°7

Débogueur sous Code::Blocks

Notions de base

Objectifs :

S'initier aux opérations de mise au point d'un programme en utilisant les techniques de base d'un débogueur intégré.

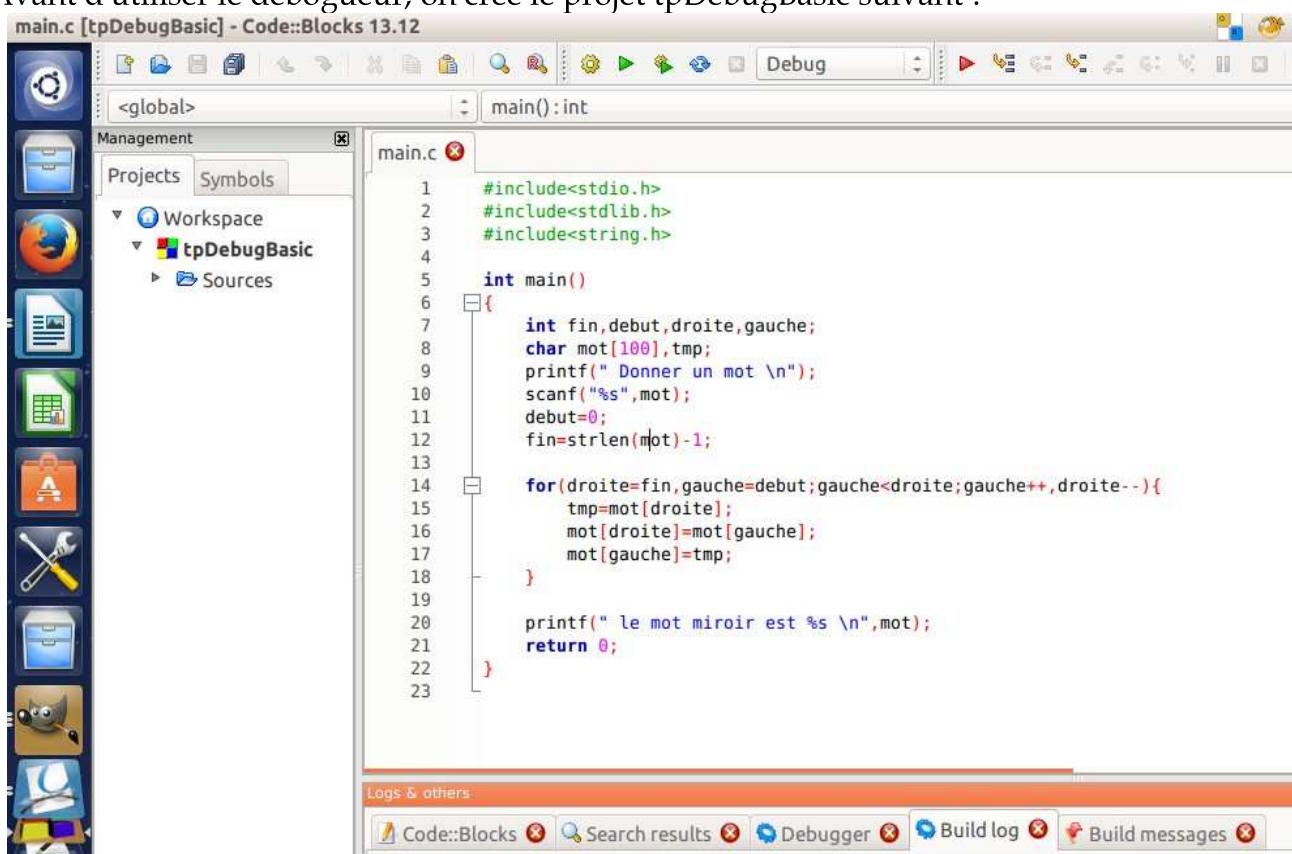
Énoncé :

Reprendre les manipulations suivantes en répondant aux questions :

Étape 1 : Préparation de l'agencement de fenêtres

Q1 : Que signifie le mot débogueur, et quel est son origine ?

Avant d'utiliser le débogueur, on crée le projet tpDebugBasic suivant :



The screenshot shows the Code::Blocks 13.12 IDE interface. The main window displays a C program named 'main.c' with the following code:

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

int main()
{
    int fin,debut,droite,gauche;
    char mot[100],tmp;
    printf(" Donner un mot \n");
    scanf("%s",mot);
    debut=0;
    fin=strlen(mot)-1;

    for(droite=fin,gauche=debut;gauche<droite;gauche++,droite--){
        tmp=mot[droite];
        mot[droite]=mot[gauche];
        mot[gauche]=tmp;
    }

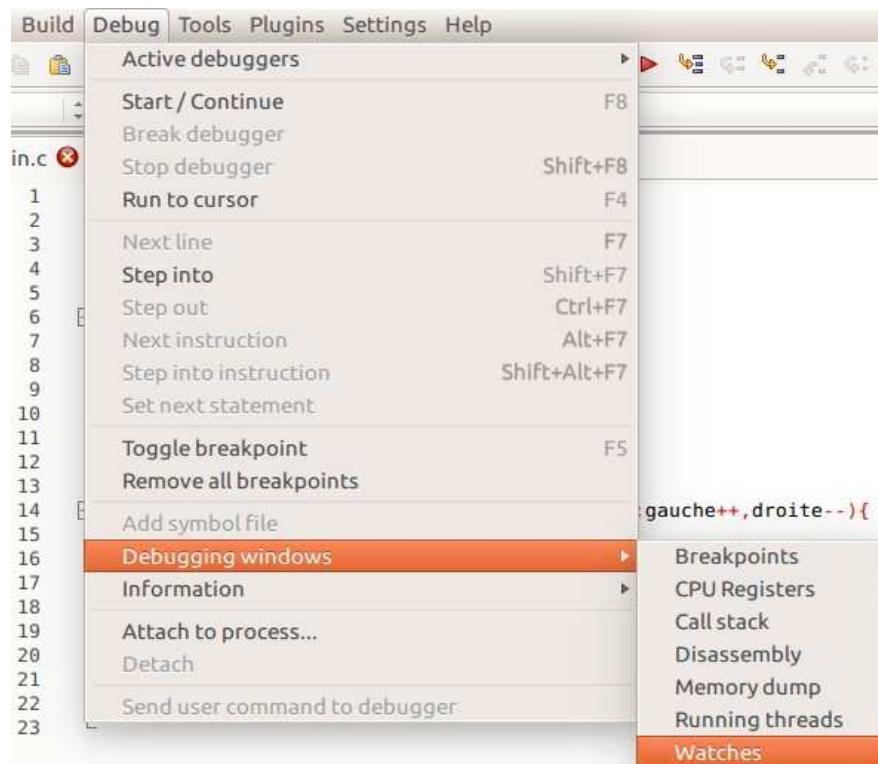
    printf(" le mot miroir est %s \n");
    return 0;
}
```

The IDE features a toolbar at the top with various icons for file operations, search, and build. Below the toolbar is a menu bar. The left sidebar contains icons for file management, browser, terminal, and other tools. The central workspace shows the project structure under 'Management' > 'Projects' > 'tpDebugBasic' and the source code in the main editor window. At the bottom, there is a docked window titled 'Logs & others' containing tabs for 'Code::Blocks', 'Search results', 'Debugger', 'Build log', and 'Build messages'.

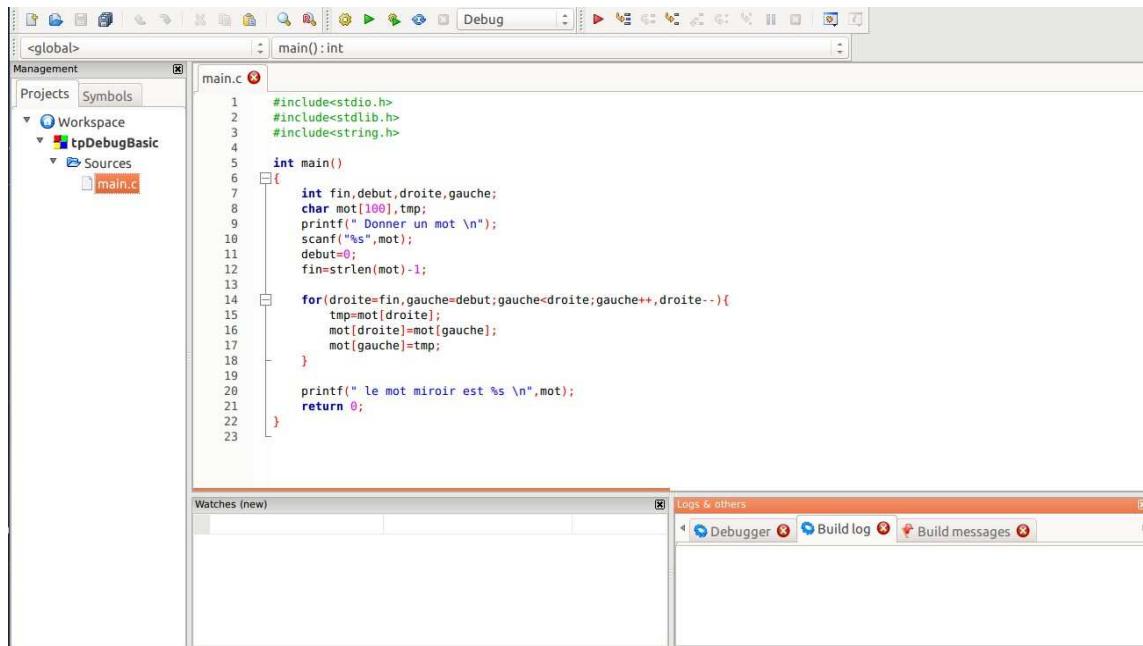
Il ne faut pas oublier de cocher la case debug de la boîte de dialogue suivante :



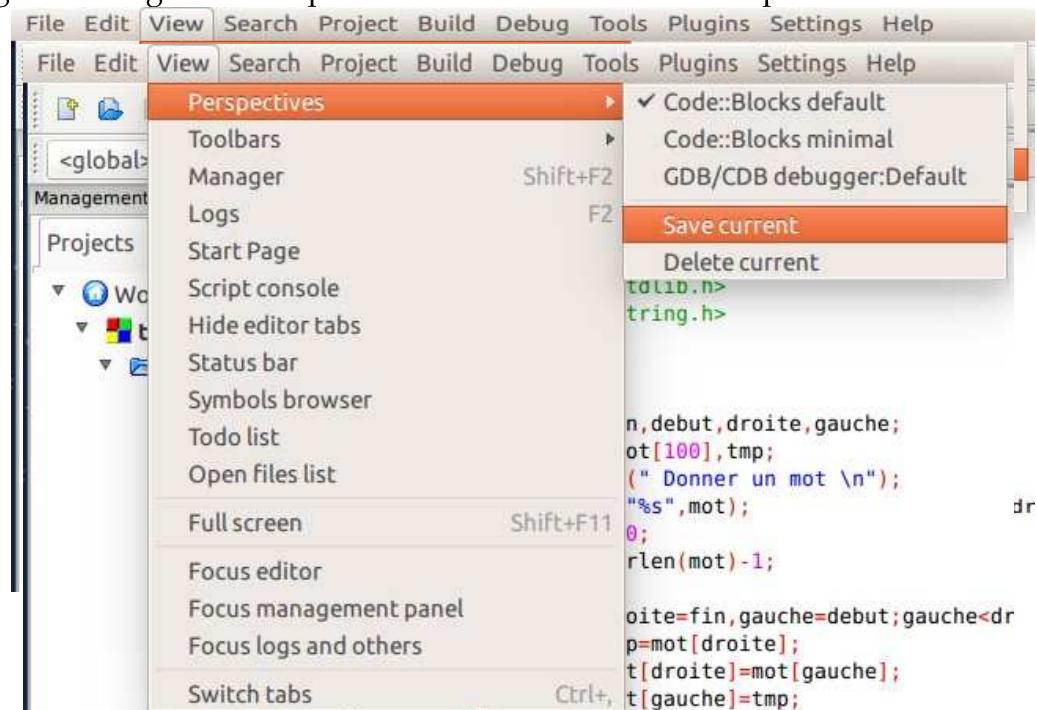
Pour utiliser le débogueur, on commence par visualiser la fenêtre watches (Debug → Debugging windows → Watches) comme suit :



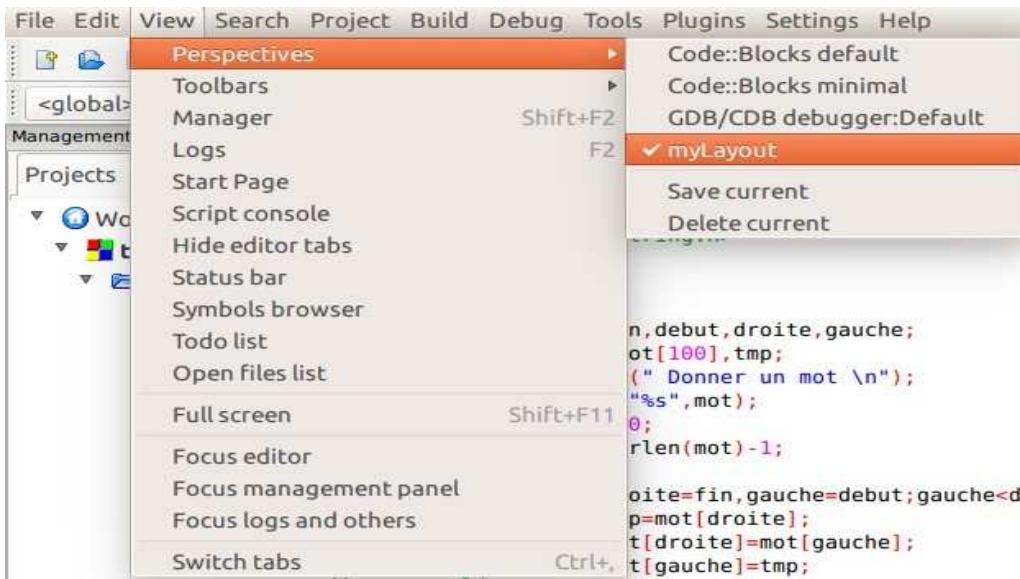
Puis on place cette fenêtre en bas par un cliquer-glisser. Nous aurons donc l'agencement de fenêtres suivant :



On sauvegarde cet agencement par la commande : View → Perspective → Save current

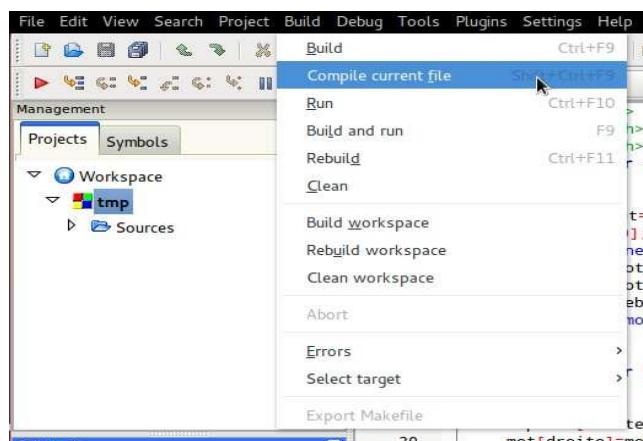


Ce qui ouvre une boîte de dialogue demandant un nom de perspective. On entre par exemple `myLayout` et on tape entrer. La prochaine fois, si on ouvre le sous-menu perspective, on trouve le nom de notre agencement sauvé. Cliquer dessus va ramener les fenêtres, quelles que soient leurs positions, à l'agencement sauvé.



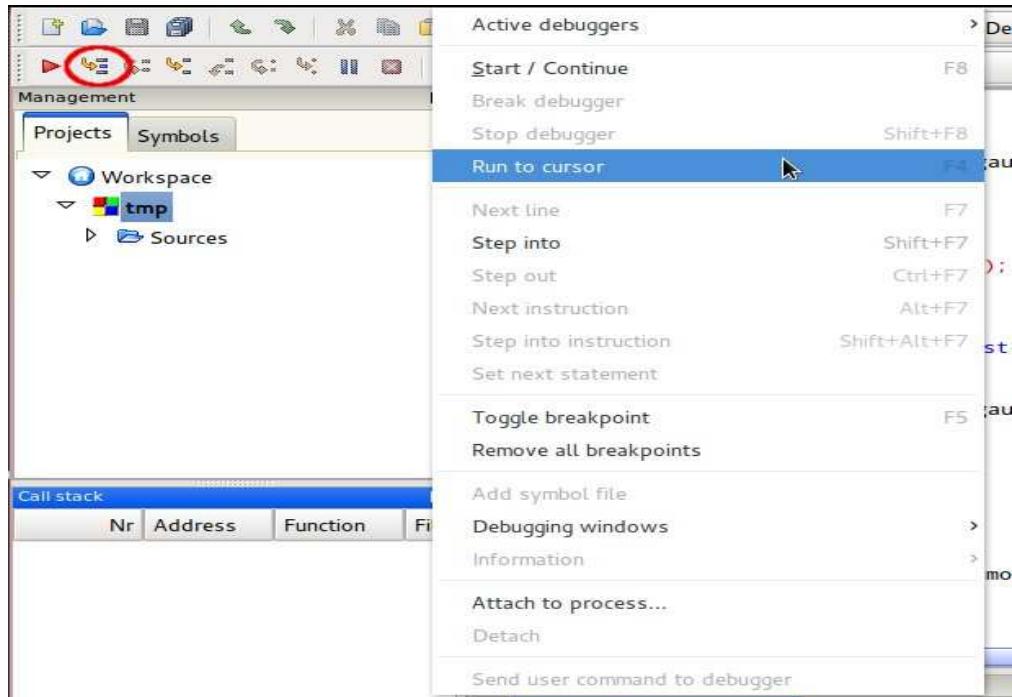
Étape 2 : Exécution jusqu'à un point donné

Avant de commencer le déroulement pas par pas de notre programme, il faut le compiler (en mode Debug). Pour cela on utilise la commande :



Si la création du code objet échoue, corriger les erreurs syntaxiques en utilisant le log de compilation.

Une fois la compilation réalisée avec succès, on met le curseur sur la ligne 12 (par exemple). Ensuite, on exécute la commande **Run to cursor** (On peut aussi cliquer sur l'icône associée (voir l'icône entourée en rouge) ou bien utiliser le raccourci clavier F4) comme suit:



Si vous avez suivi ces étapes, le programme s'exécute de façon normale. Donc il va vous demander d'introduire un mot :



Remarques :

1. Si la fenêtre d'exécution n'apparaît pas allez la chercher dans la barre de tâches ;
2. Si vous avez un message 'warning: GDB: Failed to set controlling terminal: Operation not permitted', ignorez le.

Écrivez par exemple bonjour puis tapez entrer. Vous allez avoir un scintillement (ou un clignotement) de l'icône de codeBlocks dans la barre de tâches. Cliquez dessus pour visualiser la fenêtre de CodeBlocks. Vous aurez l'image suivante indiquant où l'exécution s'est arrêtée par un triangle jaune sur la marge.

Q2 : Que se passe-t-il dans la fenêtre watches ?

Q3 : Quelle est la valeur de la variable fin ? Idem pour debut ? Pourquoi cette différence ?

Q4 : Les valeurs trouvées par vos camarades sont-elles identiques à vos valeurs ? Pourquoi ?

Q5 : Après exécution de la ligne 12 quels changements allez vous constater dans ces valeurs ?

Q6 : Expliquer le contenu de la variable mot.

The screenshot shows the Code::Blocks IDE interface. The main window displays a C program named `main.c` with a breakpoint set at line 12. The code reverses a string input by the user.

```

1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4
5 int main()
6 {
7     int fin,debut,droite,gauche;
8     char mot[100],tmp;
9     printf(" Donner un mot \n");
10    scanf("%s",mot);
11    debut=0;
12    fin=strlen(mot)-1;
13
14    for(droite=fin,gauche=debut;gauche<droite;gauche++,droite--){
15        tmp=mot[droite];
16        mot[droite]=mot[gauche];
17        mot[gauche]=tmp;
18    }
19
20    printf(" le mot miroir est %s \n",mot);
21    return 0;
22 }

```

The **Watches (new)** window shows local variable values:

Variable	Value
<code>fin</code>	-1073744808
<code>debut</code>	0
<code>droite</code>	46137407
<code>gauche</code>	0
<code>mot</code>	"bonjour\000\271\277\267\00

The **Logs & others** window shows build logs:

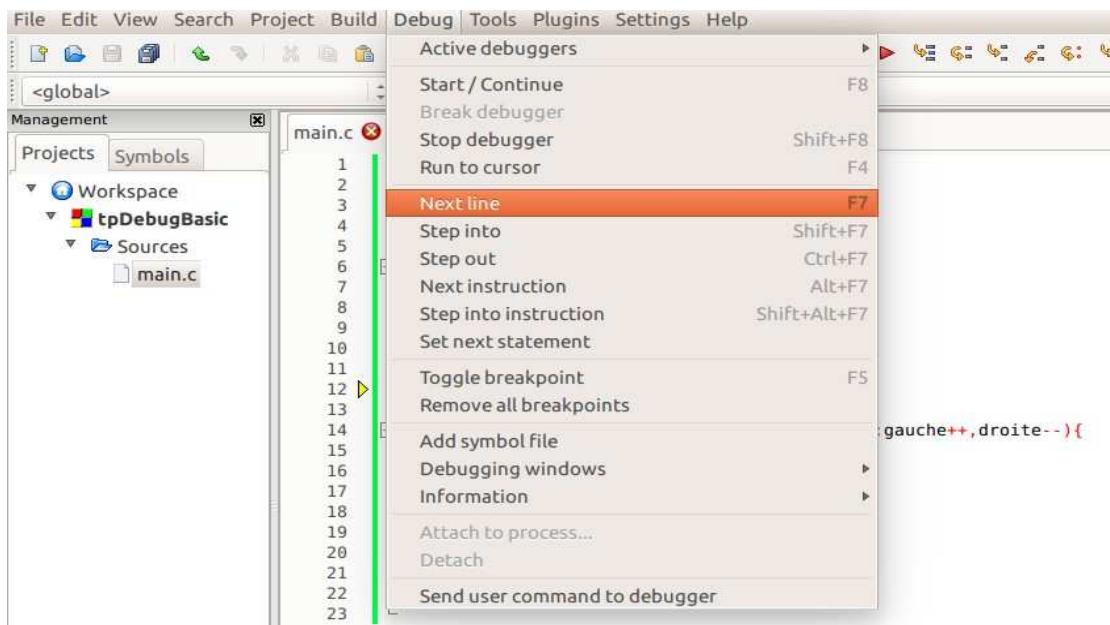
```

----- Build: Debug in tpDebugBasic (compiler: GNU GCC Compiler) -----
Target is up to date.
Nothing to be done (all items are up-to-date).

```

Étape 3 : Exécution pas par pas

Pour exécuter notre programme pas à pas à partir de la ligne 12, on utilise la commande **Debug → Next line**.



Q7 : Qu'est ce que vous remarquerez ? (position du triangle jaune, valeurs des variables).

Q8 : Exécuter le programme pas par pas (on peut utiliser le raccourci F7) jusqu'à sa fin. Commenter.

Étape 4 : Détection d'erreurs logiques (Ne pas regarder le code précédent si vous voulez apprendre)

Q9 : Quelles est la différence entre erreurs logiques et syntaxiques ?

En vue de réaliser une deuxième version du programme précédent, on propose le code suivant :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int fin,debut,gauche;
char mot[100];
int main()
{
    char tmp;
    printf(" Donner un mot ");
    scanf("%s",mot);
    debut=0;
    fin=strlen(mot);

    for(gauche=debut;gauche<fin/2;gauche++){
        tmp=mot[fin-gauche];
        mot[fin-gauche]=mot[gauche];
        mot[gauche]=tmp;
    }

    printf(" le mot miroir est %s \n",mot);
    return 0;
}
```

Créer un tel projet puis l'exécuter avec le même mot en entrée.

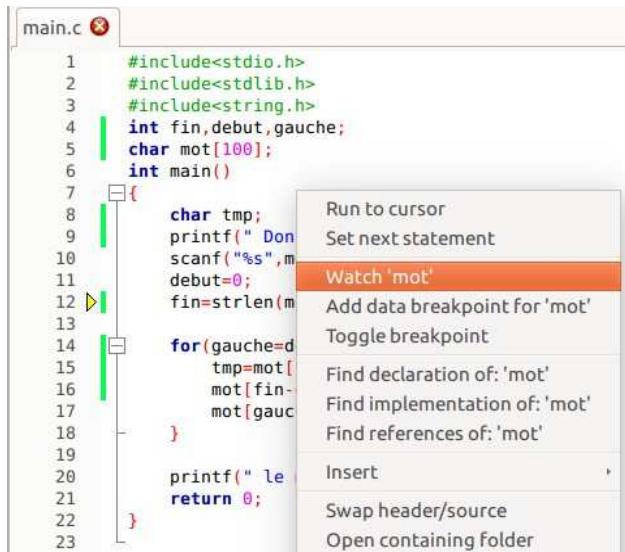
Q10 : Est ce que le résultat affiché est celui attendu ?

Q11 : Quelle est la position à partir de laquelle il est préférable de commencer l'exécution pas par pas ?

Commencer l'exécution pas par pas avec la commande Run to cursor. Introduisez le mot bonjinob et tapez entrer.

Q12 : Qu'est ce que vous remarquez dans la fenêtre Watches ?

Pour visualiser une variable, on peut ramener le pointeur de souris dessus puis on clique sur le bouton droit puis sur Watch 'mot'. Utiliser cette commande pour visualiser les variables mot, debut, fin, gauche.



On peut aussi visualiser une variable ou même une expression en double-cliquant sur une ligne de la fenêtre watches puis en saisissant la variable voulue.

Q13 : Utiliser cette méthode pour visualiser l'expression fin-gauche.



Q14 : Continuer votre exécution pas par pas jusqu'à identification de (ou des) l'erreur.