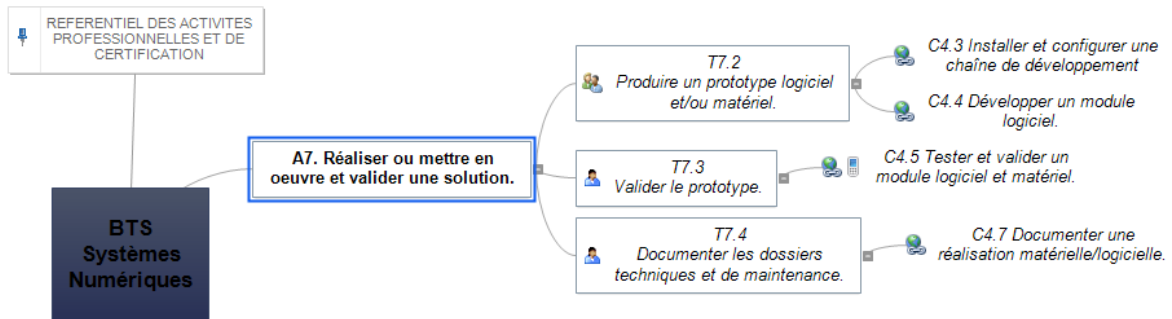


Débogage sous CodeBlocks



Créer dans votre répertoire personnel TPC++ un répertoire TP3

Etape 1 : Création d'un projet

- Créer un projet TP3 avec un fichier source minimal : main.cpp

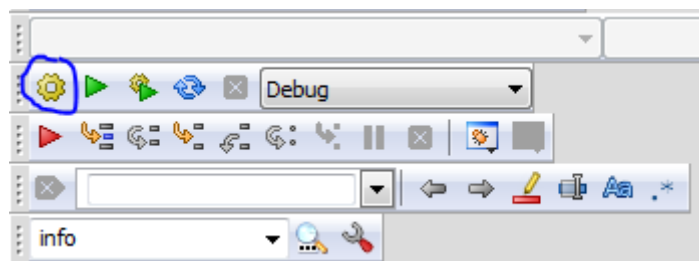
Etape 2 : Edition du code

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int i = 0;
8
9      cout << "veuillez saisir un entier : " ;
10     cin >> i;
11     cout << "Vous avez saisi:" << i << endl;
12     return 0;
13 }
  
```

Etape 3 : Compilation et linkage

- Cliquer sur l'icône en forme de roue dentée



```

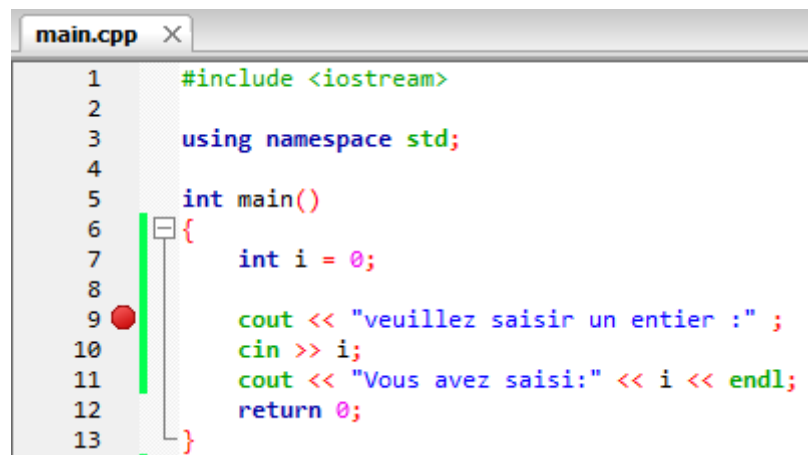
----- Build: Debug in TP2 (compiler: GNU GCC Compiler)-----
mingw32-g++.exe -Wall -fexceptions -g -c "I:\Utilisateurs\Documents\Cours2014-2015\SNIR1\ressources TP\streamVideo\wxwidgetGUI\TP2\main.cpp" -o obj\Debug\main.o
mingw32-g++.exe -LI:\Utilisateurs\Documents\Cours2013-2014\Iris2\plaisance\cbsql -o bin\Debug\TP2.exe obj\Debug\main.o
Output file is bin\Debug\TP2.exe with size 942.90 KB
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

```

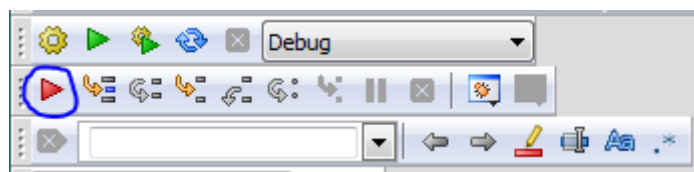
- Si pas d'erreur alors exécution du programme, sinon il peut être nécessaire d'effectuer le débogage du code source lorsque l'erreur ne nous saute pas aux yeux.

Etape 4 : Débogage de l'exécutable généré.

Pour effectuer un débogage, il est nécessaire de placer au moins un point d'arrêt sur une des lignes de code, soit en cliquant sur la colonne de gauche, soit en invoquant le menu contextuel (click bouton de droite) et en cliquant sur **Toggle breakpoint** au niveau de la ligne de code ; un point rouge devrait apparaître :

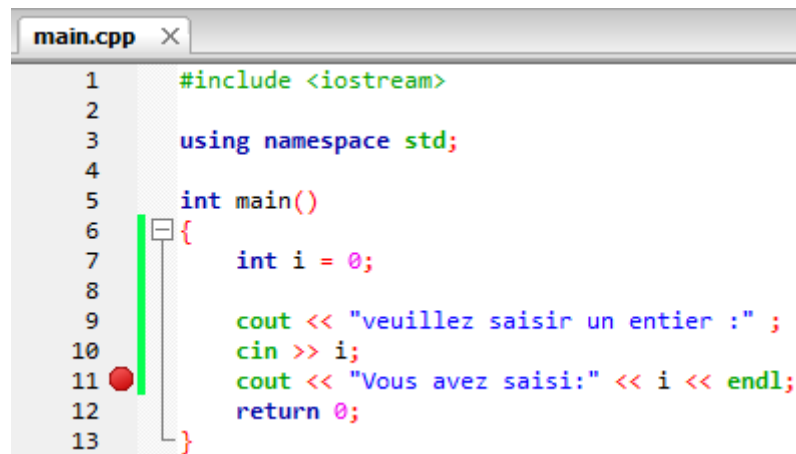


Ensuite on lance l'exécution du programme à l'aide de la commande **Start** du menu **Debug** (ou on clique sur le triangle rouge)



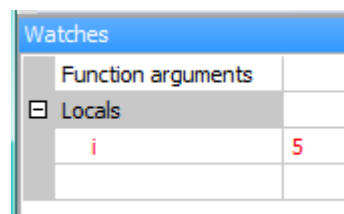
Le programme s'arrête sur la première ligne qui possède un point d'arrêt (un curseur apparaît) :

A ce stade, on peut consulter le contenu des variables actives dans la fenêtre **Watches** que l'on fait apparaître en cochant la case **Watches** dans le menu **Debugging windows**



```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int i = 0;
8
9      cout << "veuillez saisir un entier : " ;
10     cin >> i;
11     cout << "Vous avez saisi:" << i << endl;
12     return 0;
13 }
```

La fenêtre **Watches** fournit par défaut le contenu des variables actives et on peut en rajouter d'autres à l'aide du menu contextuel de l'éditeur.



On peut dérouler l'exécution suivant différents modes : commandes **Step over (F7)**, **Step into (Shift-F7)** ou **Step out (Ctrl-Shift-F7)** et **Abort** du menu **Debug** ou via les boutons d'accès rapide :



A vous de faire :

Vous produirez un compte rendu numérique au format pdf agrémenté :

- du code source
- de copie d'écran prouvant le bon fonctionnement de l'application
- des questions et réponses (si présente).

FAITES VALIDER CHAQUE EXERCICE PAR VOTRE PROFESSEUR

Exercice 1

Ecrire un programme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne. En cas de réponse supérieure à 20, on fera apparaître un message : " Plus petit ! ", et inversement, " Plus grand ! " si le nombre est inférieur à 10.

Exercice 2

Ecrire un programme permettant la saisie de la réponse à la question : " aimez-vous l'informatique (O/N) ". Vous devez afficher un message en cas de mauvaise réponse, et renouveler la question jusqu'à ce que la réponse soit correcte.

Exercice 3

La racine carrée d'un nombre peut être obtenue par soustractions successives. L'exemple ci-dessous illustre ce principe :

Recherche de la racine de 49 :

$$49 - 1 = 48$$

$$48 - 3 = 45$$

$$45 - 5 = 40$$

$$40 - 7 = 33$$

$$33 - 9 = 24$$

$$24 - 11 = 13$$

$$13 - 13 = 0$$

Le nombre de soustractions effectuées donne la racine. Dans ce cas, nous avons effectué 7 soustractions.

Ecrire un programme correspondant à cette méthode. On suppose que le nombre saisi est un entier et est carré d'un entier (exemple : 9, 16, 25 ...).