

Chapitre N

DÉBOGAGE DE PROGRAMME

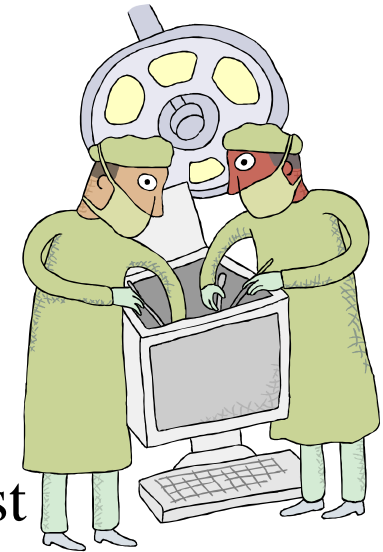
POLYTECHNIQUE
MONTRÉAL

LE GÉNIE
EN PREMIÈRE CLASSE



Débogage

- L'activité de débogage consiste à identifier les erreurs dans un programme et à les corriger.
- Pour détecter les erreurs, autres que syntaxiques, il faut **tester** le programme, manuellement ou automatiquement, et vérifier que la sortie correspond bien à vos attentes.
- Le test peut montrer que la sortie de votre programme est incorrecte, mais il ne vous donne en revanche généralement aucun indice sur la partie de votre code à l'origine du problème. C'est à ce niveau qu'intervient le débogage.
- Les analyseurs de programmes permettent aussi parfois de trouver des erreurs.



Débogage de programmes

1- Les erreurs potentielles

- Erreurs de syntaxe et d'exécution
- Identification des erreurs à la compilation et à l'exécution

2- Améliorer vos techniques de débogage

- Identifier les erreurs de logique et de sémantiques courantes
- Rechercher les lignes de code problématiques
- Corriger les erreurs

3- Services fournis par les outils de débogage

- Contrôle et observation de l'exécution d'un programme
- Exécution d'un programme pas à pas
- Points d'arrêts
- Visualisation des variables



1- Les erreurs potentielles



- Erreurs de syntaxe et erreurs d'exécution
 - Les erreurs de syntaxe sont celles qui sont détectées lors de la compilation.
 - Les erreurs d'exécution sont celles qui seront découvertes lors de l'activité de test du programme. On peut parler dans ce cas d'erreurs de logique ou d'erreurs de sémantique.
- Identification des erreurs à la compilation
 - Le compilateur énumère les erreurs de syntaxe qu'il a identifiées.
 - Une seule erreur peut entraîner une cascade d'erreurs. Pour cette raison il est conseillé de débiter la correction d'erreurs à partir de la première de la liste.
 - Le message d'avertissement ne doit pas être pris à la légère. Il identifie une ambiguïté qui pourrait résulter en une erreur d'exécution.
 - L'éditeur de VS est intelligent, car il est possible d'identifier certaines erreurs qui pourront être évitées lors de la compilation.

1- Les erreurs potentielles (suite)



- Identification des erreurs à l'exécution
 - Une erreur d'exécution est identifiée lorsqu'un comportement inattendu survient. Par exemple, une fin abrupte.
 - L'affichage d'un résultat erroné est également un indicateur d'une erreur d'exécution.
 - L'utilisation en entrée de données pour lesquelles le résultat est connu permet de vérifier efficacement un programme.
 - ATTENTION: l'utilisation de données erronées à l'entrée peut entraîner un comportement « aléatoire » du programme et par le fait même difficile à déboguer.

2- Améliorer vos techniques de débogage

❏ Identifier les erreurs de logique et de sémantiques

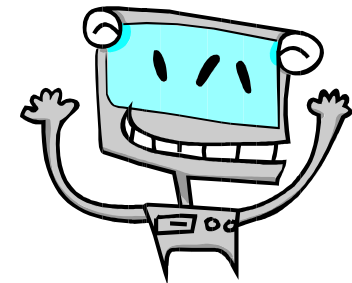
❏ Déclaration

- ❏ Utilisation inappropriée d'un type – un entier plutôt qu'un réel
- ❏ Initialisation manquante
- ❏ Dimension insuffisante d'un tableau

❏ Expression booléenne

- ❏ Logique inversée – `Fic.eof()` plutôt que `!Fic.eof()`
- ❏ Opérateur = plutôt que == : `if (Nbre = 0)` plutôt que `if (Nbre == 0)`
- ❏ Opérateur binaire & ou | plutôt que && ou ||
- ❏ Opérateur booléen && plutôt que || ou vice versa
- ❏ Expression toujours VRAIE ou toujours FAUSSE
 - ❏ `(i == 4 || i != 4)`
 - ❏ `(x > 0 && x <= 0)`

❏ Ordre des instructions non logique



2- Améliorer vos techniques de débogage (suite)

Autres expressions

Division entière

Moyenne = $1/4 * (W+X+Y+Z)$; donne 0

Angle en radian

Valeur = $\sin(45)$; correspond au sinus 45 radian

Structures de contrôle

Omission des accolades d'une instruction composée

Placement d'une instruction vide

for (i = 0; i < 10; i++);

if (Age > 10);

Tableau

Débordement

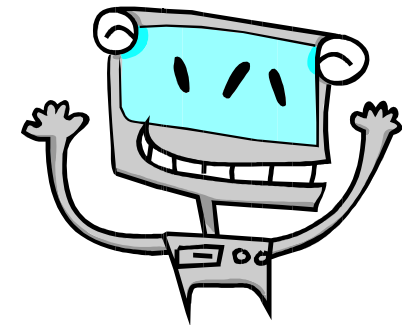
Utilisation d'un indice négatif ou dépassant le nombre d'éléments du tableau

for (i=1; i<=N; i++)

Autres

Intervertir les indices $\text{Tab}[i][j]$ ou $\text{Tab}[j][i]$

Omettre l'initialisation de tous les éléments du tableau



2- Améliorer vos techniques de débogage (suite)

- Rechercher les lignes de code problématiques
 - Le compilateur nous reporte à la ligne où l'erreur est détectée.
 - Il est fortement recommandé de tester chaque fonction individuellement.
 - Si une erreur survient, il faut initialement, se référer à l'emplacement dans le code correspondant à l'opération qui a pu provoquer cette erreur. Par contre, il est possible que l'erreur soit commise dans les opérations précédentes. Il faut alors penser à rebours l'exécution du programme.

2- Améliorer vos techniques de débogage (suite)

- Corriger les erreurs

- **Erreurs de compilation**

- Certaines erreurs de compilation sont évidentes, d'autres moins.
 - Il faut s'assurer de bien comprendre la cause de l'erreur pour apporter le bon correctif.
 - Il faut surtout éviter d'être soumis au compilateur et effectuer aveuglement la correction qu'il propose.
 - Éditeur de VS nous aide à résoudre certaines erreurs de compilation.

2- Améliorer vos techniques de débogage (suite)

- Corriger les erreurs

- **Erreurs d'exécution**

- Il faut s'assurer de bien comprendre la cause de l'erreur pour apporter le bon correctif.
 - Une fois les lignes de code provoquant l'erreur bien identifiées, les corrections peuvent être effectuées.
 - Il n'est pas rare que la correction d'une erreur puisse nous permettre de découvrir une ou d'autres erreurs (parfois plus grave).
 - Si les fonctions ont été testées individuellement, il faut alors se concentrer sur l'information qui transige d'une fonction à l'autre.

- Outil de débogage permet d'exécuter le programme séquentiellement (instruction par instruction) pour vérifier les résultats partiels de chaque instruction

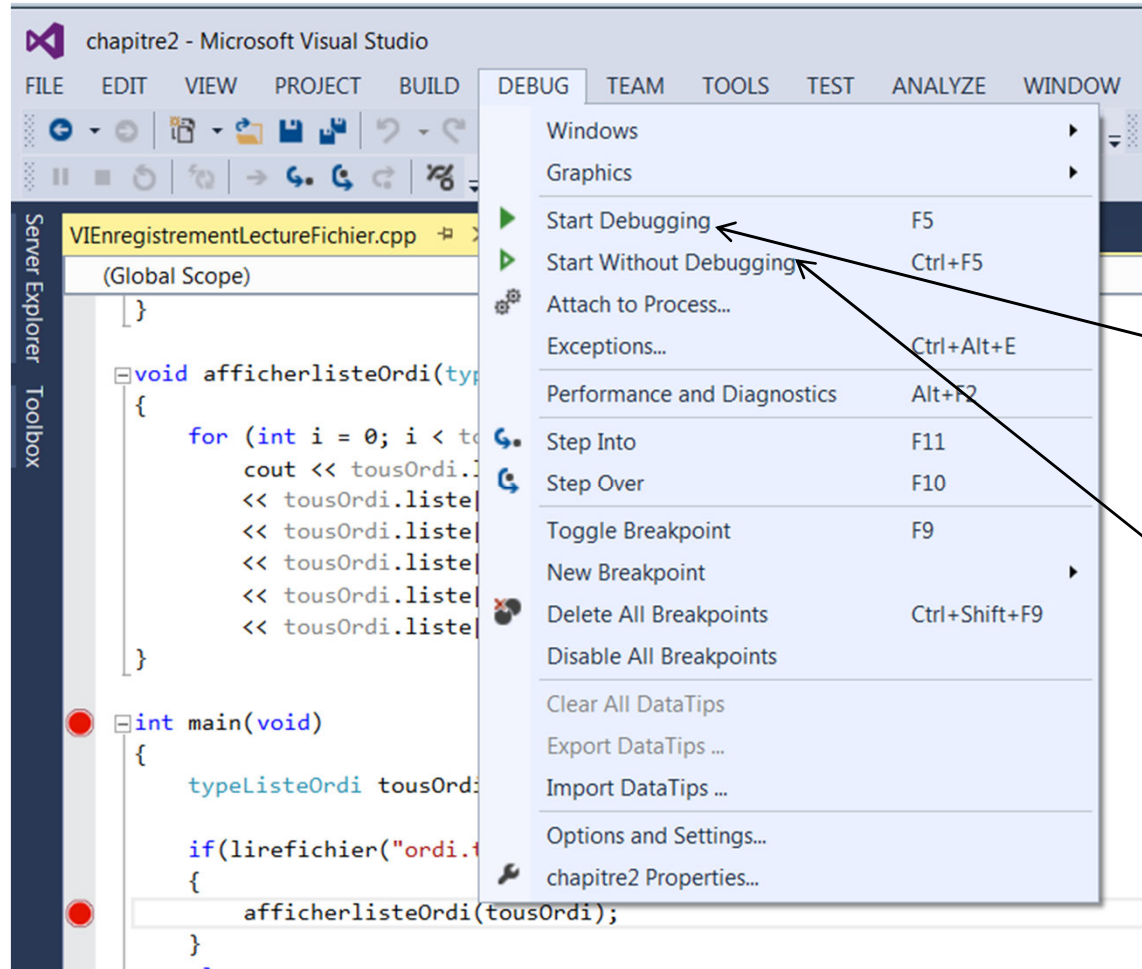
3- Services fournis par les outils de débogage

Le débogueur de VS permet le contrôle et l'observation de l'exécution d'un programme

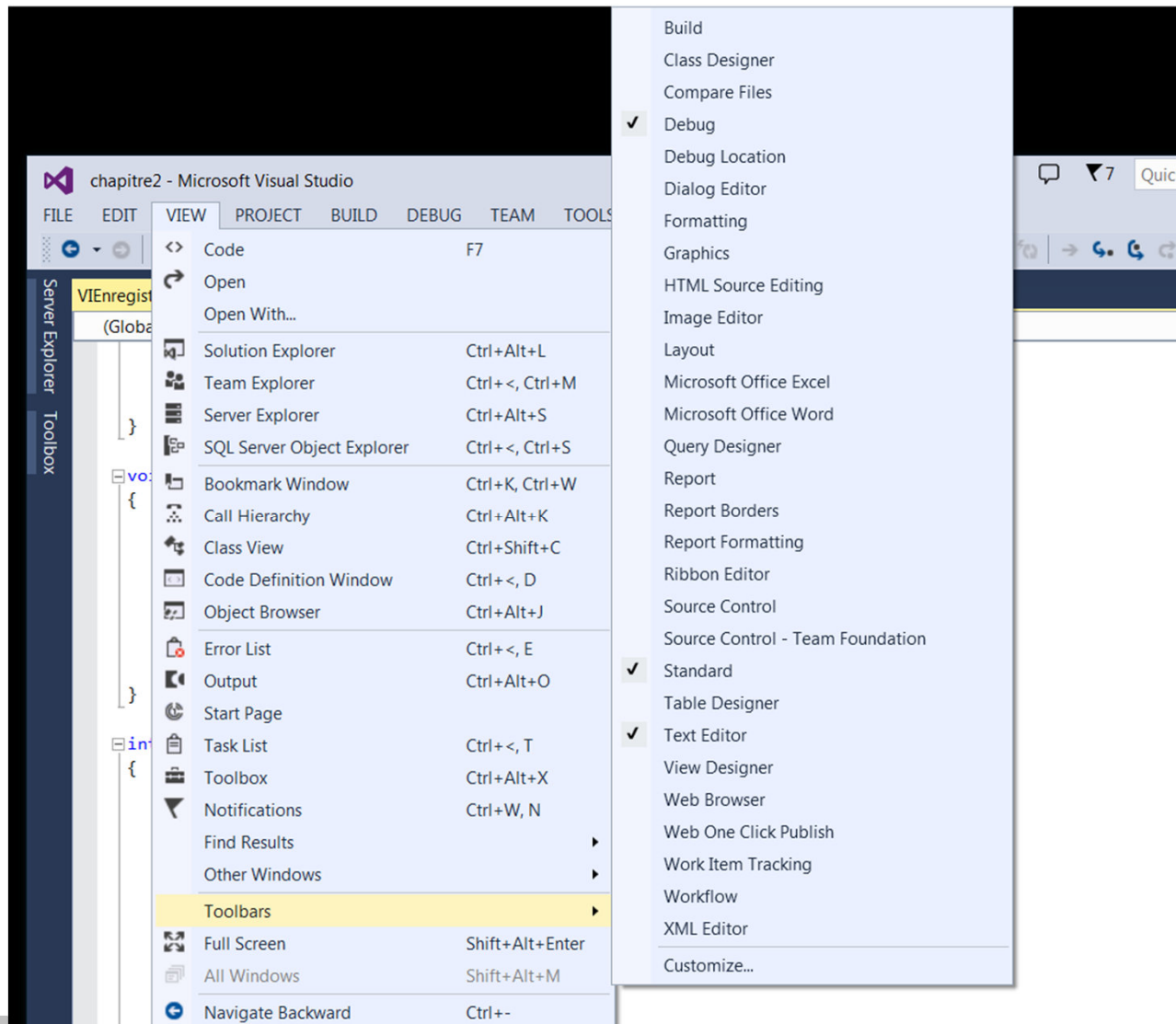
- Démarrage ou poursuite de l'exécution
- Interruption de l'exécution
- Exécution d'un programme pas à pas
- Points d'arrêts permettant l'arrêt de l'exécution
- Exécution jusqu'à un emplacement spécifié
- Visualisation des variables



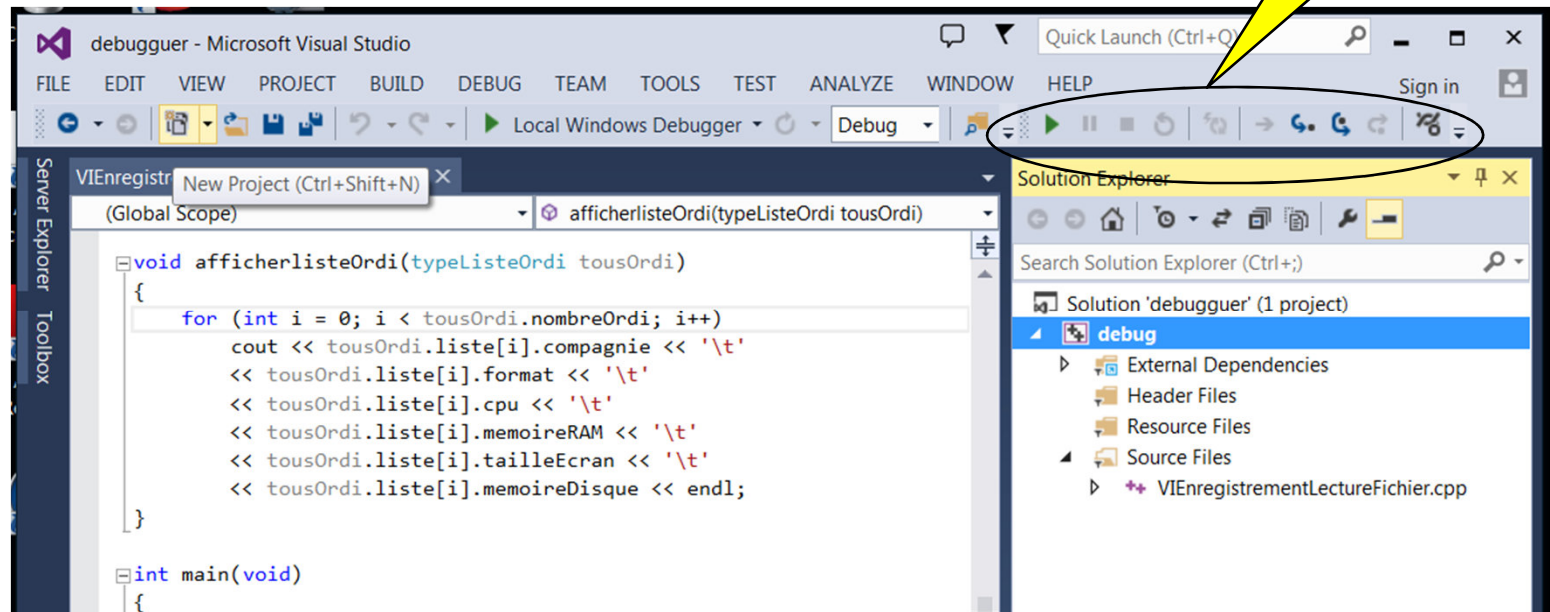
Exécution du programme



Afficher le menu raccourci Deboguer




ShortCut
Debugger




3- Services fournis par les outils de débogage

F9: Point d'arrêt (BreakPoint) pour suspendre l'exécution d'un programme à une instruction spécifique.

F5: Démarrer le débogage (Start Debugging). Continuer l'exécution au prochain point d'arrêt. 

F11: Pas à pas détaillé (Step into). Si l'instruction est un appel à une fonction, F11 permet d'entrer dans la fonction et d'arrêter à la première instruction de celle-ci. 

F10: Pas à Pas principal (Step Over). Si l'instruction est un appel à une fonction, F10 permet d'exécuter la fonction et de continuer à l'instruction suivante à la fonction. 

Maj+F11: Pas à pas sortant (Step Out). Cette option permet de terminer l'appel d'une fonction et de revenir au programme appelant de la fonction. 

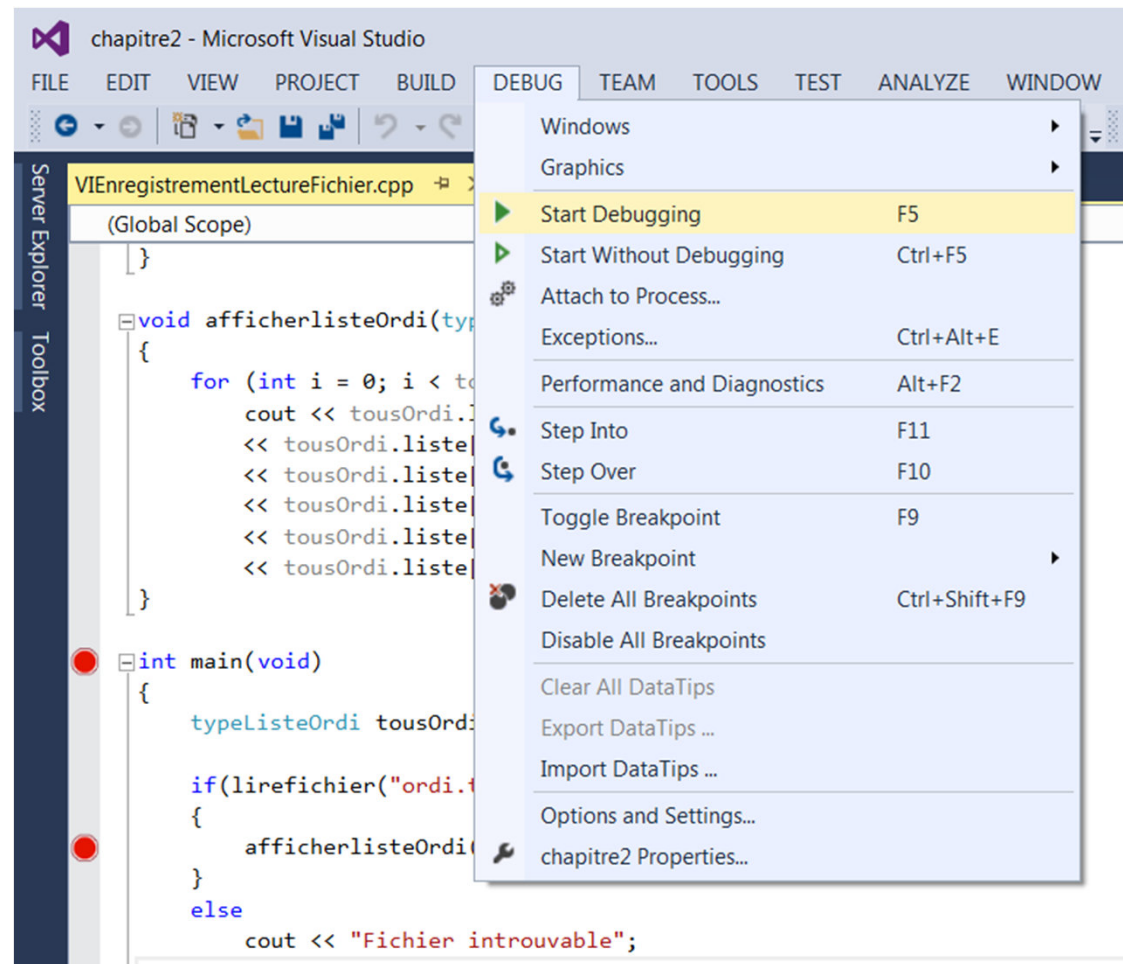
Shift+F5: arrêt du débogage (Stop Debugging) 

Point d'arrêt (BreakPoint)

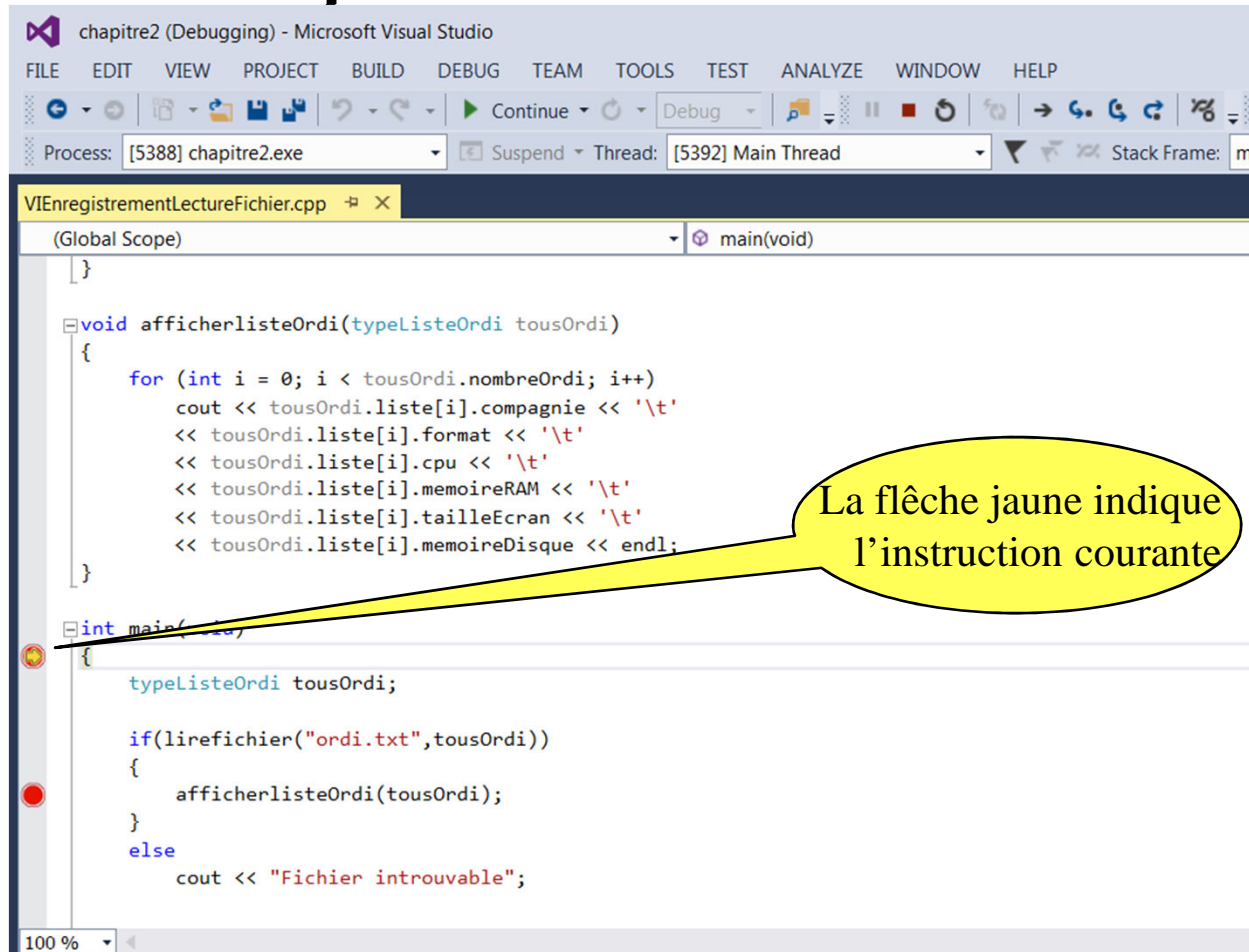


- Point d'arrêt (Breakpoint) est un élément de débogage qui permet de suspendre l'exécution de votre programme à une instruction que le développeur a défini.
- Dans la barre verticale grise de l'éditeur, cliquer 2 fois pour indiquer les instructions à suspendre (c'est à dire mettre des points d'arrêts)

Démarrer le débogage F5 ou



Éxecuter jusqu'au premier point d'arrêt F5 ou

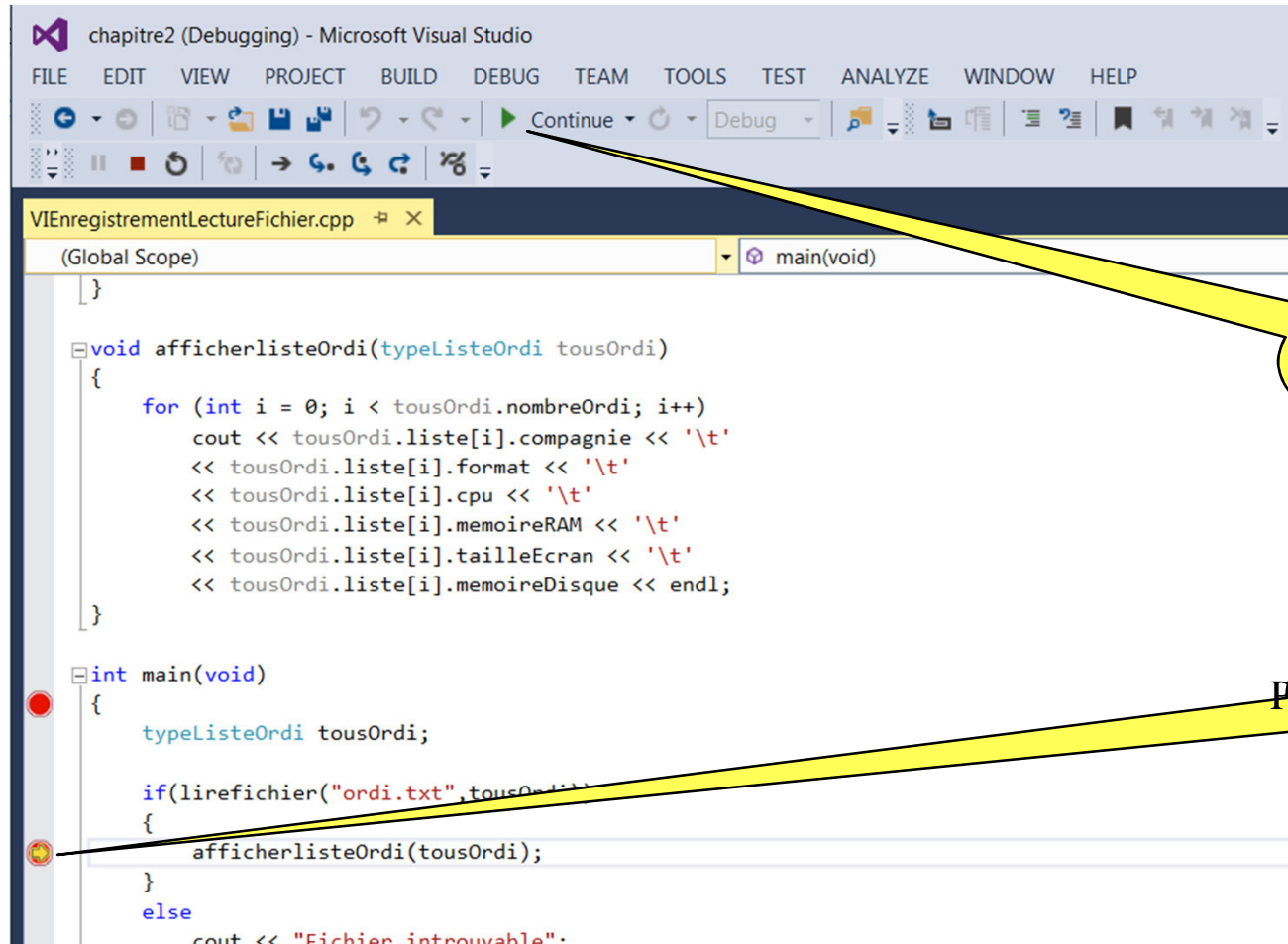


```
chapitre2 (Debugging) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Process: [5388] chapitre2.exe Suspend Thread: [5392] Main Thread Stack Frame: m
VEnregistrementLectureFichier.cpp
(Global Scope) main(void)
}
void afficherlisteOrdi(typeListeOrdi tousOrdi)
{
    for (int i = 0; i < tousOrdi.nombreOrdi; i++)
        cout << tousOrdi.liste[i].compagnie << '\t'
        << tousOrdi.liste[i].format << '\t'
        << tousOrdi.liste[i].cpu << '\t'
        << tousOrdi.liste[i].memoireRAM << '\t'
        << tousOrdi.liste[i].tailleEcran << '\t'
        << tousOrdi.liste[i].memoireDisque << endl;
}
int main()
{
    typeListeOrdi tousOrdi;

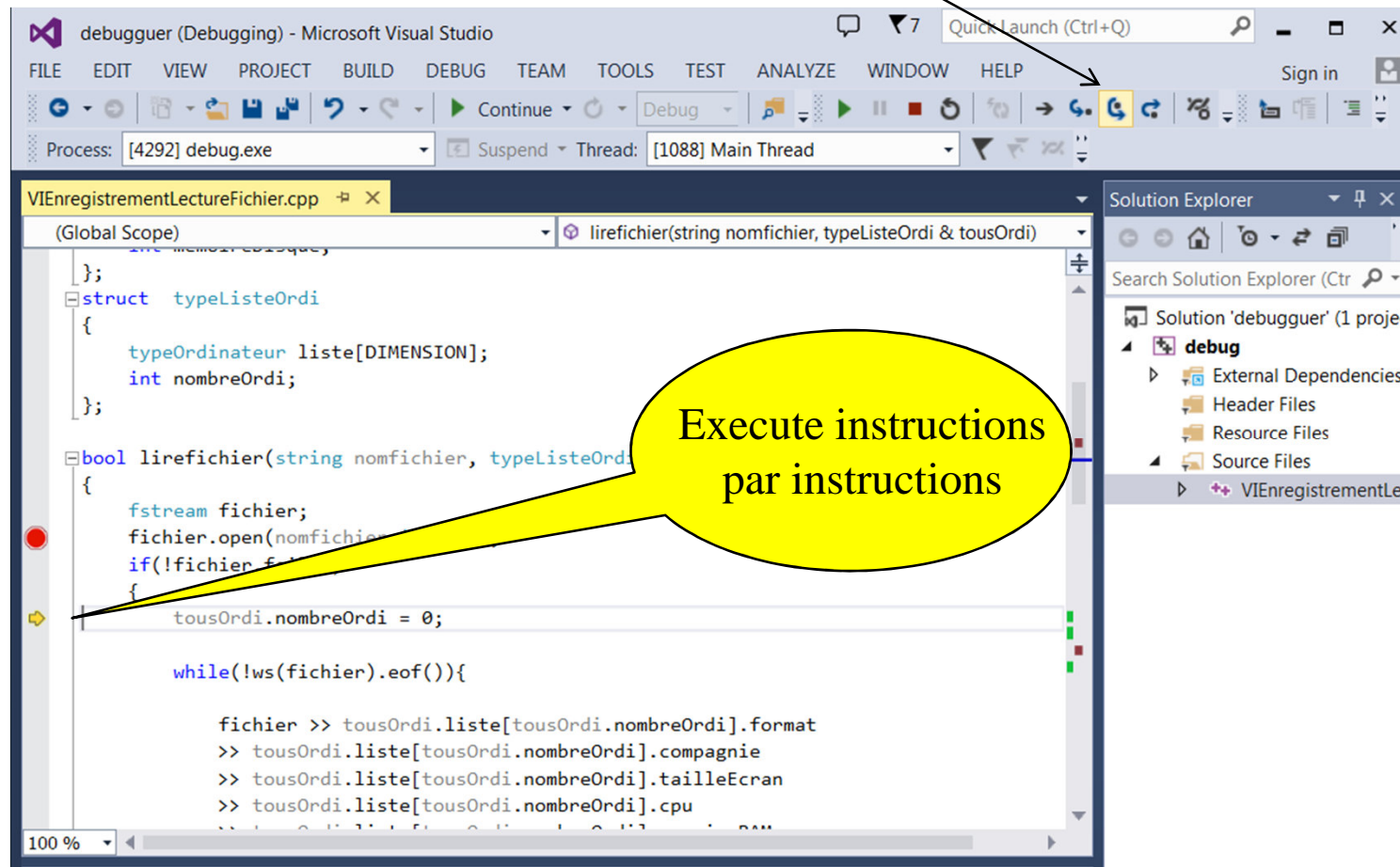
    if(lirefichier("ordi.txt",tousOrdi))
    {
        afficherlisteOrdi(tousOrdi);
    }
    else
        cout << "Fichier introuvable";
}
```

La flèche jaune indique l'instruction courante

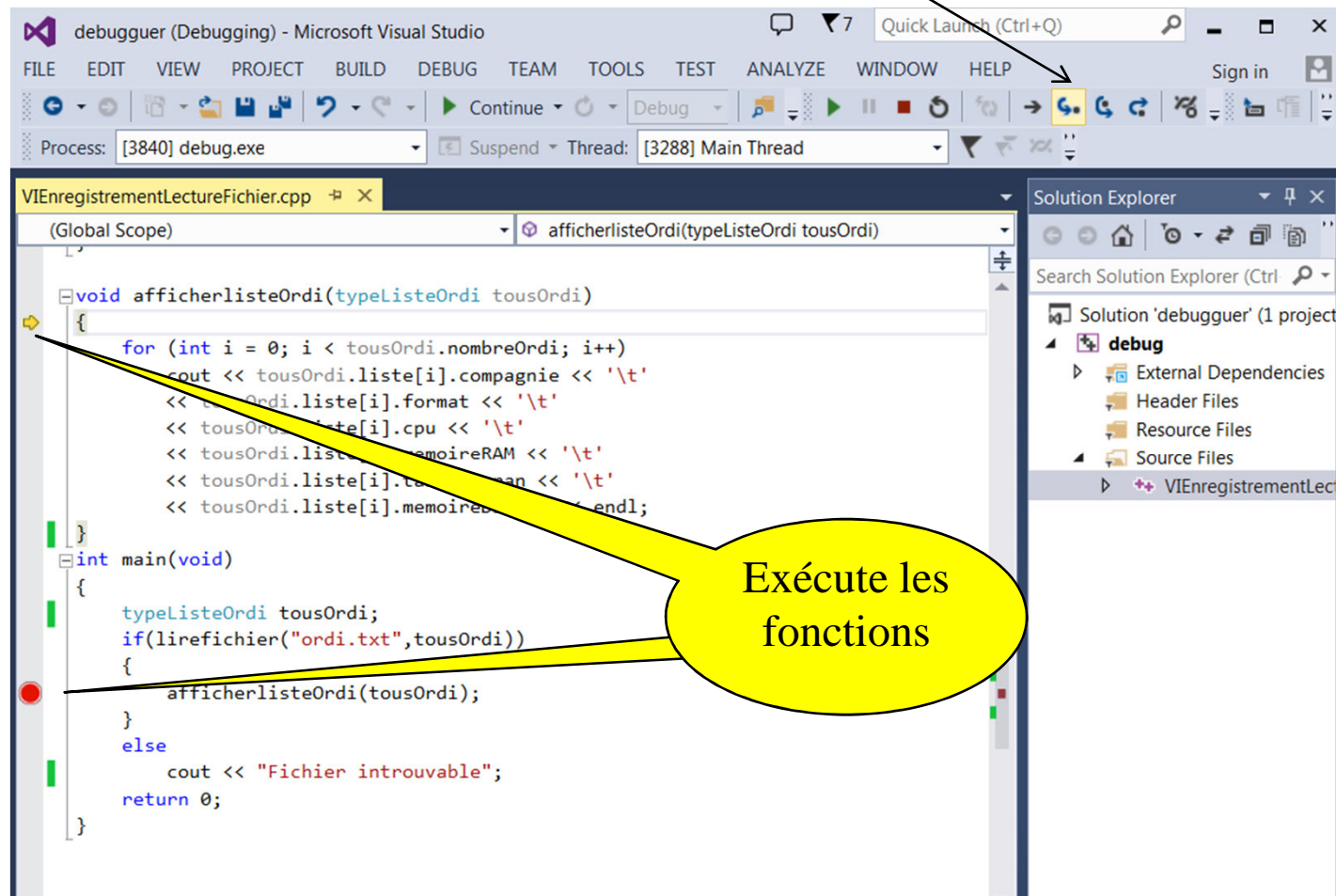
Éxecuter jusqu'au prochain Point d'arrêt F5 ou



F10: Pas à Pas principal (Step Over).



F11: Pas à pas détaillé (Step into).



Maj+F11: Pas à pas sortant (Step Out).



avant

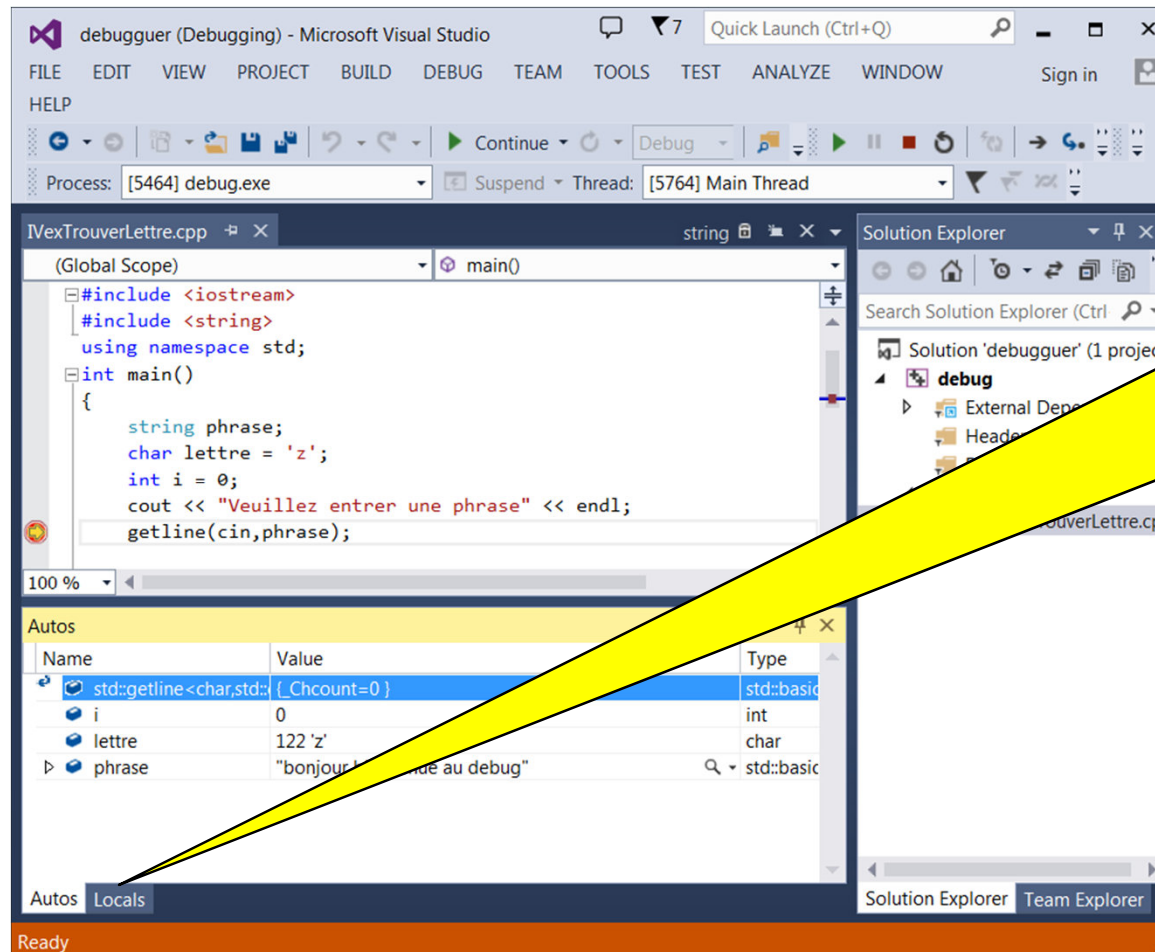
Après step out,
la fonction est
terminée et le
prg revient à
l'appellant

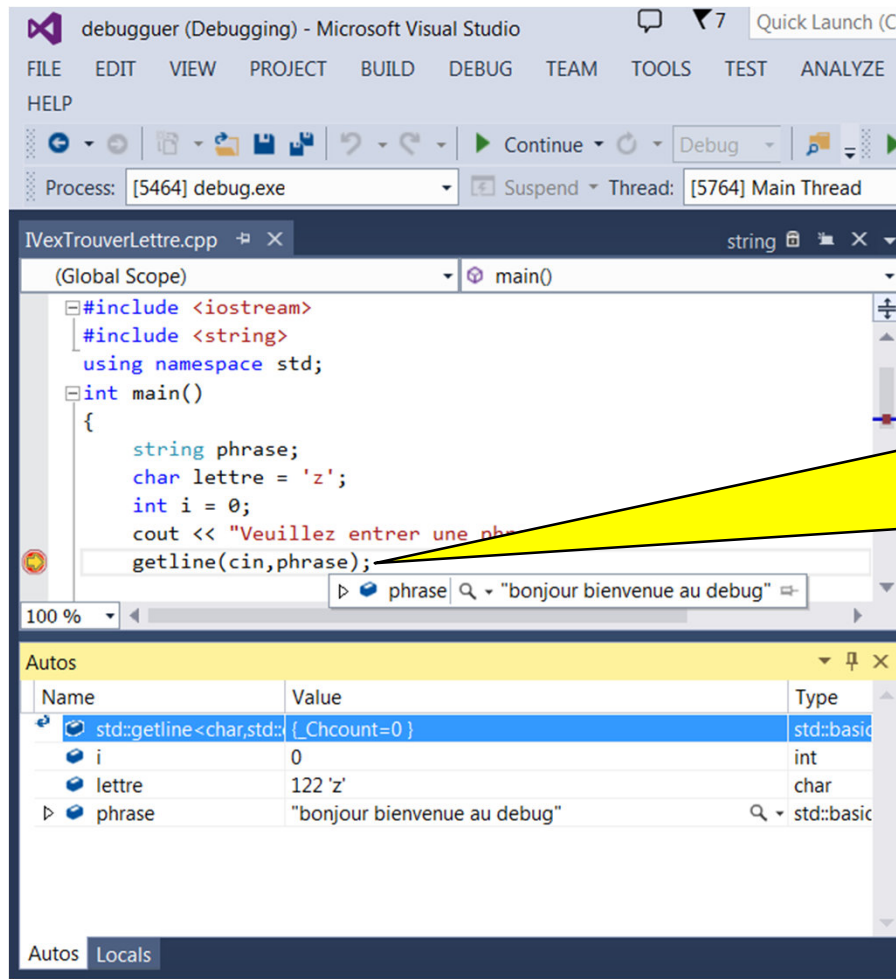
```

debugger (Debugging) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Process: [3840] debug.exe Suspend Thread: [3288] Main Thread
VIEnregistrementLectureFichier.cpp
(Global Scope)
afficheurlisteOrdi(typeListeOrdi tousOrdi)
{
    void afficheurlisteOrdi(typeListeOrdi tousOrdi)
    {
        for (int i = 0; i < tousOrdi.nombreOrdi; i++)
            cout << tousOrdi.liste[i].compagnie << '\t'
                << tousOrdi.liste[i].format << '\t'
                << tousOrdi.liste[i].cpu << '\t'
                << tousOrdi.liste[i].memoireRAM << '\t'
                << tousOrdi.liste[i].tailleEcran << '\t'
                << tousOrdi.liste[i].memoireDisque << endl;
    }
}
int main(void)
{
    typeListeOrdi tousOrdi;
    if(lirefichier("ordi.txt",tousOrdi))
    {
        afficheurlisteOrdi(tousOrdi);
    }
    else
        cout << "Fichier introuvable";
    return 0;
}
  
```

```

debugger (Debugging) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Process: [3840] debug.exe Suspend Thread: [3288] Main Thread
VIEnregistrementLectureFichier.cpp
(Global Scope)
afficheurlisteOrdi(typeListeOrdi tousOrdi)
{
    void afficheurlisteOrdi(typeListeOrdi tousOrdi)
    {
        for (int i = 0; i < tousOrdi.nombreOrdi; i++)
            cout << tousOrdi.liste[i].compagnie << '\t'
                << tousOrdi.liste[i].format << '\t'
                << tousOrdi.liste[i].cpu << '\t'
                << tousOrdi.liste[i].memoireRAM << '\t'
                << tousOrdi.liste[i].tailleEcran << '\t'
                << tousOrdi.liste[i].memoireDisque << endl;
    }
}
int main(void)
{
    typeListeOrdi tousOrdi;
    if(lirefichier("ordi.txt",tousOrdi))
    {
        afficheurlisteOrdi(tousOrdi);
    }
    else
        cout << "Fichier introuvable";
    return 0;
}
  
```





En mode
debogage, en
placant le
curseur, on
peut
visualiser le
contenu de
variables