Chapitre N

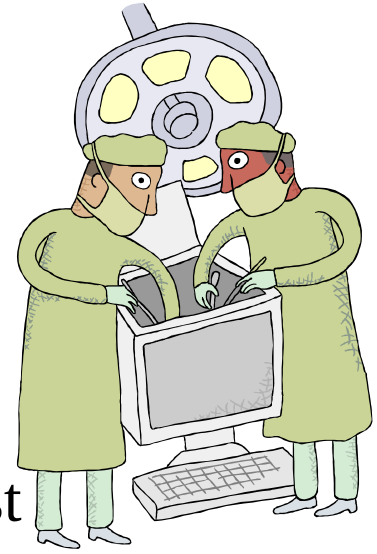
DÉBOGAGE DE PROGRAMME

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.

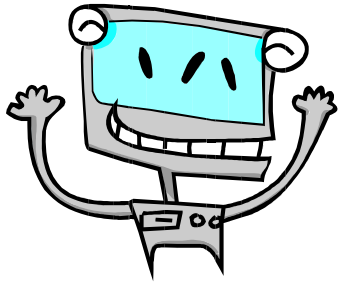
- Identification des erreurs à la compilation
  - Le compilateur énumère les erreurs de syntaxe qu'il a identifiées.
  - Une seule erreur peut entraîner une cascade d'erreurs. Pour cette raison il est conseillé de débuter la correction d'erreurs à partir de la première de la liste.
  - Le message d'avertissement ne doit pas être pris à la légère. Il identifie une ambiguïté qui pourrait résulter en une erreur d'exécution.

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.

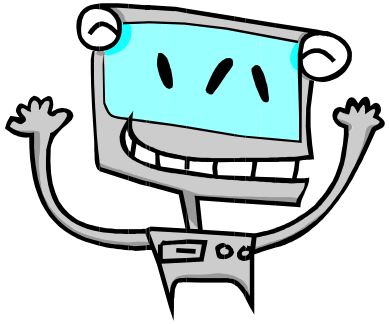
2- Améliorer vos techniques de débogage

- 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)



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 Tab[i][j] ou Tab[j][i]
    - Omettre l'initialisation de tous les éléments du tableau



2- Améliorer vos techniques de débogage (suite)

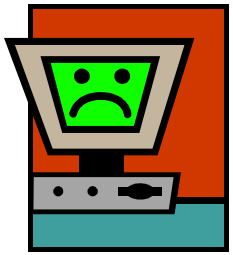
- Rechercher les lignes de code problématiques
  - Le compilateur nous reporte à la ligne où l'erreur est détectée.
  - Il est fortement recommandé de tester chaque fonction individuellement.
  - Si une erreur survient, il faut initialement, se référer à l'emplacement dans le code correspondant à l'opération 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.

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.
  - Ajouter quelques affichages (traces) vous permettant d'identifier le déroulement du programme et le contenu de certaines variables.



2- Améliorer vos techniques de débogage (suite)

- La première expérience de débogage de nombreux programmeurs consiste, lorsqu'ils essaient d'isoler un problème, à ajouter des affichages, à l'aide d'un énoncé **cout**, dans leur code.
- Cette méthode de débogage est parfaitement légitime. Mais une fois que vous aurez trouvé et résolu le problème, vous devrez reprendre tout votre code et supprimer tous ces énoncés d'affichage. A moins que vous ayez recours à une méthode systématique qui utilise des énoncés s'adressant au préprocesseur (#define, #ifdef et #endif)

2- Améliorer vos techniques de débogage (suite)

```
#define TRACE    // Placer au début du programme pour afficher les traces
                 //  Mettre en commentaire pour éviter d'afficher les traces

if (NbLu = = 0) {
  #ifdef TRACE
    cout << "[TRACE] Le nombre lu est 0." << endl;
  #endif
  NbZero++;
}
else {
  #ifdef TRACE
    cout << "[TRACE] Le nombre lu est différent de 0." << endl;
  #endif
  NbDifZero++;
}
```

2- Améliorer vos techniques de débogage (suite)

- Dans certaines situations occasionnelles, l'ajout de nouveau code, même un simple appel **cout**, peut modifier le comportement du code que vous essayez de déboguer !

3- Services fournis par les outils de débogage

Le débogueur de VC++ 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

3- Services fournis par les outils de débogage

- Démonstration de l'outil de débogage directement à l'ordinateur