Handout – Boucles et debugging

Préambule

Tout le code de ce handout doit être placé à l'intérieur de la fonction `main()` comme suit:

```
#include <iostream>
using namespace std;

int main() {
  // ton code
}
```

Boucle

Boucle while

Répète les instructions du corps de la boucle aussi longtemps que la condition reste vérifiée.

```
int i = 0;
while(i < 10) {
  // imprime le message 10 fois
  cout << "hello from loop number " << i << "\n";
  i++;
}
```

Parfois nous voulons quitter la boucle plus tôt, avant que la condition ne devienne fausse, et nous utilisons le mot-clé **break** dans ce cas. Cela effectuera un saut jusqu'à l'instruction suivant la fin de la boucle.

```
int i = 0;
while(i < 10) {
  // imprime le message seulement 3 fois
  cout << "hello from loop number " << i << "\n";
  if(i == 2)
    break;
  i++;
}
```

Si nous voulons sauter le reste du corps de la boucle sans la quitter totalement, nous utilisons le mot-clé **continue**.

```
int i = 0;
while(i < 10) {
  if(i % 2 == 1)
    continue;
  // n'imprime le message que quand i est pair
  cout << "hello from loop number " << i << "\n";
  i++;
}
```

Boucle for

Ce schéma d'initialisation d'une variable, utilisation de la variable dans une condition et mise à jour de la variable à la fin est très courant. Pour cette raison, le C++ nous permet de faire cela plus rapidement grâce à la boucle for.

```
for(int i = 0; i < 10; i++) {
  // imrpime le message 10 fois
  cout << "hello from loop number " << i << "\n";
}
```

Nous n'avons pas besoin d'utiliser tout le temps les 3 parties de la boucle for, nous pouvons les laisser vides. Cependant, si nous en laissons 2 ou plus vides, nous pourrions aussi bien utiliser une boucle while.

```
int i = 0;
for(;i < 10;) { // ne fais pas ça
  // imrpime le message 10 fois
  cout << "hello from loop number " << i << "\n";
  i++;
}
```

Exemple: test de primalité

Exercice: vérifie pour un certain nombre n s'il est un nombre premier.

```
bool prime = true;
for(int i = 2; i < n; i++) {
  if(n % i == 0) {
    prime = false;
    break;
  }
}
```

Debugging

Que faire si ton programme ne fonctionne pas?

- Compile ton programme et laisse le compilateur te dire s'il y a une erreur de syntaxe et si oui, à quelle ligne.
- Si le programme crashe, vérifie tes suppositions avec `assert(conditions)`.
- Utilise `cerr << var` pour imprimer de l'information qui pourrait t'aider à déterminer ce qui ne fonctionne pas.
- Utilise le debugger de VSCode.

Debugger VSCode

Comment utiliser le debugger de VSCode:

- Vérifie que *task.json* et *launch.json* sont bien configurés dans le dossier *.vscode*
- Construis le projet avec Terminal->Run Build Task (*Ctrl-Shift-B*)
- Ajoute au moins un "breakpoint" en cliquant à gauche du numéro de ligne.
- Commence à debugger avec Debug->Start Debugging (*F5*)