Algorithmique et Structures de Données 1
Travaux Pratiques

Premier TP : compilation, enregistrements, complexité

Le travail en séance de Travaux Pratiques s'effectue dans l'environnement Linux avec le compilateur Gnu g++. Vous pouvez utiliser d'autres compilateur/système d'exploitation/ environnement de développement sur d'autres machines ou sur celles des salles de TP, par exemple Visual Studio Code. Vous travaillerez autant que possible en binômes.

Partie 1. Rappel

Votre espace de travail comporte une fenêtre de terminal pour lancer les commandes de compilation et d'exécution, une fenêtre d'éditeur de texte pour écrire le code, une fenêtre avec le sujet du TP disponible.

(i) ***Compilation***

Téléchargez l'archive `TP1-1.zip` sur Madoc, extrayez son contenu et placez-le dans un répertoire adéquat de la hiérarchie de vos documents (par exemple `~/X21I010_ASD1/TP1/`). Dans le terminal, utilisez les commandes de changement de répertoire courant (e.g. `cd X21I010_ASD1/TP1`) pour accéder à ce répertoire et lancez la compilation du fichier `progCorrect.cpp` (la commande est, par exemple,

```
gpp -std=c++11 progCorrect.cpp -o  progCorrect.exe
```

La compilation devrait bien se passer : pas de message d'erreur. Demandez alors l'exécution du code machine obtenu avec la commande

```
./progCorrect.exe
```

(ii) Correction

Compilez de même le fichier supposé contenir un programme de résolution d'une équation du second degré. En étudiant le premier message d'erreur obtenu à la compilation, corrigez un première fois le code (ouvrir le fichier dans l'éditeur de texte), puis relancez la compilation. Compilez une seconde fois puis corrigez à nouveau le code, et ainsi de suite jusqu'à ce que la compilation ne provoque plus de message d'erreur.

Trouvez les autres fautes qui ne provoquent pas d'erreurs de compilation. Lorsque vous estimez avoir obtenu un programme correct, montrez à l'enseignant son fonctionnement (en l'exécutant).

Partie 2. Complexité

Téléchargez l'archive `TP1-2.zip` sur Madoc, extrayez son contenu et placez-le dans le sous-repertoire de votre TP1 crée en partie 1. Vous trouverez 5 algorithmes dans ce repertoire, compilez-les et testez l'execution. Il s'agit de 5 algorithmes, chacun prenant en paramètre un entier qui influencera le nombre de tour de l'algorithmes. Par exemple, algorithme 2 prend en entrée la taille d'un tableau de N entiers consécutifs et ensuite effectue la recherche dichotomique d'un entier aléatoire:

```
>./algo2 10000
Found 9392 in the array of 10000 elements in 14 steps
```

Votre but est, pour chaque algorithme, de :

1) construire un tableau de relation entre le paramètre d'entrée N et le nombre des étapes rapportée par l'algorithme pour arriver à la solution;
2) tracer un graphique de nombre des étapes en fonction de N (une courbe par algorithme, le tout sur le même graphique). Que pouvez-vous dire sur le comportement de ces courbes ? Classez leur comportement dans les classes: exponentielle, polynomiale, logarithmique, constante, linéaire;

3) mesurer le temps d'execution de chaque algorithme pour des différents N. Vous pouvez le faire en rajoutant la commande `time` avant chaque commande d'execution (e.g. `time ./algo2 10`). Analysez le résultat.

Partie 3. Travaux dirigés

Implémentez les types et algorithmes définis ou à définir dans les exercices de travaux dirigés. Soignez la qualité du code (commentaires, indentation, noms des variables) et l'interface (saisies et affichages).