Corrigé du TP1

Disclaimer : Ce corrigé n'a pas pour but de vous fournir un détail des algorithmes répondant aux questions de l'énoncé, mais de vous résumer les notions clés dont vous aurez besoin pour résoudre ces exercices, les avertissements vis à vis de certaines erreurs classiques et les détails des passages les plus techniques, voire des précisions si vous souhaitez aller plus loin sur ces questions et plus généralement de synthétiser dans un même document la plupart des commentaires fait en TP afin qu'ils puissent profiter à tous. Pour obtenir un corrigé détaillé, il vous faudra donc vous référer à ce que vous avez pu faire en TP ou reprendre vous même les exercices à partir des notions présentées dans le présent corrigé pour écrire les algorithmes qui vous manquent.

En cas de difficultés, **n'hésitez pas à me contacter**, nous pourrons en discuter soit par mail soit en début de séance suivante. Ce corrigé n'a pour but que d'être une base de travail pour vous et de servir de complément par rapport à ce que vous avez fait en séance, il n'est en aucun cas fait pour se suffire à lui même.

Premiers pas avec Code : :Blocks

* Si vous souhaitez installer Code : :Blocks sur votre ordinateur, il est important d'utiliser un fichier d'installation comprenant **mingw** dans le nom.

  Un fichier ne comprenant pas mingw ne contiendra pas de compilateur. Dans certains cas, il est possible de s'en passer, notamment si votre ordinateur possède déjà un compilateur, mais prendre le fichier en mingw fonctionnera à tous les coups et l'inverse n'est pas vrai.

* Pour créer un nouveau projet, il faut faire **File** → **New** → **Project**

  Si ce n'est pas fait par défaut, cochez le Debug, c'est une bonne habitude à prendre.

* Pour l'instant, tout votre code s'écrira à l'intérieur de la fonction main.

  Ne supprimez rien dans le fichier main.cpp, la plupart de ces lignes sont indispensables, à part le cout « "Hello world !" « endl ;, que vous pouvez supprimer.

Écrire un code compréhensible

* Il est important de faire des efforts sur la syntaxe, lorsque votre code fait un millier de lignes, ça peut être d'une aide précieuse pour corriger les éventuelles coquilles dans le code.
* Il faut mettre un ; après chaque commande.
* Ne pas mettre de ; après les if, else if, else, for, while.

  Il faut cependant mettre un ; après un while d'une boucle do while.
* Préférez utiliser des noms de variables compréhensibles, pas seulement k, z, etc ... à moins bien sûr que ces variables vous soient données dans l'énoncé.
* Commentez vos codes, ainsi même en reprenant votre code pour réviser vos examens, vous ne perdrez pas 30 minutes à comprendre ce que vous vouliez faire.
* Les commandes dans une boucle ou une structure conditionnelle sont entre accolades. On dit qu'elles sont encapsulées.
* Utilisez les indentations, une commande égale une ligne, toutes les commandes d'une boucle (sauf en cas de boucle imbriquée) est sur la même indentation, ...
* Lorsque vous utilisez des doubles, il vaut mieux mettre un . après le chiffre, écrivez 2. ou 2.0 au lieu de 2

  Cela évitera que c++ convertisse arbitrairement des double en int.

Lecture et écriture de données

Ne vous trompez pas de sens dans les flèches > et <. En cas de doute, un moyen mémotechnique est de voir cin et cout comme votre terminal.

Dans un cas vous envoyez les données vers le terminal cout << et dans l'autre vous les récupérez cin >>

Utilisation de l'instruction if, else if, else

* Ces instructions permettent de n'exécuter les commandes uniquement si la condition est vérifiée
* Une structure qui fonctionne pour une boucle if est

```
if (condition 1)
{
    commandes 1 ;
} else if (condition 2)
{
    commandes 2 ;
} else
{
    commandes 3 ;
}
```

  Même s'il existe des notations plus compactes.
* N'oubliez pas de mettre la condition entre parenthèse et toutes les commandes entre les accolades.

  Sauf pour else qui n'est pas suivi de parenthèses.
* Ne mettez pas de ; après la condition, mais il faut en mettre après chacune des commandes dans les accolades.
* Vous n'êtes pas obligés d'utiliser les commandes **else if** ou **else** si ce n'est pas nécessaire.

Écriture d'une boucle

* Les boucles while et do..while servent à répéter une commande ou une suite de commande tant que la condition est vérifiée.

  Faites attention, C++ ne comprend pas quand vous écrivez une condition seule, c'est une erreur qui revient très souvent. Le programme ne va pas bugger et ne vous renverra pas d'erreur, mais C++ passera dessus comme si vous n'aviez rien écrit.
* Une utilisation classique de la boucle do..while est de forcer l'utilisateur à entrer une valeur précise par un cin.

  Pour ce faire, il faut que tant que l'utilisateur ne rentre pas une valeur correcte, la commande cin se répète.

  Il est important que la condition soit "la valeur n'est pas correct" et non pas "la valeur est correcte".
* Une utilisation classique des boucles while est de répéter une instruction un nombre n de fois.

  Pour ce faire, il faut commencer par définir un entier valant 0 que l'on appelle compteur.

  Ensuite on utilise une boucle while et on répète les instructions jusqu'à ce que le compteur vaille n, en ajoutant la commande "compt++ ;" qui incrémente le compteur à chaque passage dans la boucle.
* Une structure qui fonctionne pour une boucle do .. while est

```
do
{
    commandes ;
} while (condition) ;
```

* Une structure qui fonctionne pour une boucle while est

```
while (condition)
{
    commandes ;
}
```

* Pour do .. while, n'oubliez pas de mettre le ; après la condition qui elle même doit être entre parenthèse. Pour while, il n'y a pas de ; après la condition.
* Les boucles while et do .. while fonctionne presque de la même façon, la différence est que la condition de la boucle while est testée avant la boucle et celle de la boucle do .. while est testée après la boucle.

Éléments logiques

* Il est possible de lier des propositions entre elles par des opérateurs logiques.
* Pour "et", on utilise la commande && Pour écrire &, il faut appuyer sur la touche 1 sans activer la touche majuscule, on appelle ce symbole l'esperluette.
* Pour "ou", on utilise la commande || Pour écrire |, il faut appuyer sur la touche 6 sans activer la touche majuscule.
* Pour "non", on utilise la commande !(...)

Quelques conseils pour retravailler le TP

* Les quelques remarques sur 'Écrire un code compréhensible' sont plus importantes qu'elles n'y paraissent, vous vous en rendrez compte si vous êtes amenés à travailler avec des programmes informatiques plus tard. Ce sont quelques automatismes qui augmenteront beaucoup votre confort de travail, plus tôt vous les prenez, plus simple ce sera pour vous. Il en reste que ce n'est pas indispensable pour valider votre semestre, essayez donc d'y faire attention en TP, mais ne vous prenez pas non plus trop la tête avec ça.
* La lecture et l'écriture de données revient très régulièrement en TP. Cette distinction entre utilisateur/rice du programme (la personne qui entre la valeur de la variable) et programmateur/rice (la personne qui définit comment on utilise cette variable) peut poser quelques difficultés. Si c'est votre cas, il est important que vous travaillez ce point pour qu'il soit clair dans votre tête pour ne pas perdre du temps inutilement en TP.
* L'utilisation de l'instruction if et des boucles while et do ... while, sont centrales en informatique. Si vous n'avez jamais vu ces notions, il est normal qu'elle puisse sembler difficiles à manipuler.

  N'hésitez pas à tester dans un premier temps à la main les différentes possibilités dans le cas d'un if ou les premières itérations des boucles. Vous pouvez d'ailleurs le faire sur les exemples vus en cours, TD et TP pour les retravailler après la séance. C'est important pour vous familiariser avec ces structures.

  En tous les cas, nous reverrons et retravaillerons ces questions tout au long du semestre. Si vous peinez à faire le calcul à la main dont je vous parle précédemment, n'hésitez pas à m'en parler en TP, nous en rediscuterons, il est important que vous arriviez à le faire correctement.
* Il vous sera utile de retenir le morceau de code permettant de demander à l'utilisateur une variable possédant certaines propriétés données (note entre 0 et 20). Nous le réutiliserons beaucoup par la suite. Et surtout ne vous contentez pas d'écrire une commande disant que la note doit être entre 0 et 20, C++ ne comprendra pas, il vous faut une boucle.

Pour aller plus loin

Nous n'avons pas eu l'occasion de le voir en TP, mais vous pouvez choisir le nombre de chiffres affichés.

Pour ce faire, déclarez le package **iomanip** en début de programme avec un include.

Vous pourrez alors utiliser la fonction **cout.precision** qui prend en argument un entier. Cet entier est le nombre de caractères utilisés pour écrire la réponse (en incluant le point).

Question bonus : Utilisez cette commande dans l'exercice 4 pour afficher le résultat obtenu avec 4 chiffres après la virgule.

Question bonus 2 : Quel devrait être le dernier chiffre (au sens de quel est le chiffre significatif) affiché par le programme en fonction de la précision voulue ? Utilisez la commande précédente pour afficher le résultat avec ce nombre de chiffres après la virgule.