Corrigé du TP6

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.

Manipuler les pointeurs

- Une variable est enregistrée dans un espace mémoire spécifique. L'adresse de la variable est le numéro de cette case mémoire.
- Pour obtenir l'adresse d'une variable var, il faut écrire *&var*.
- On peut également stocker l'adresse de la variable dans une nouvelle variable. Cette nouvelle variable est appelée un pointeur.

  Pour déclarer un pointeur Ptr, il faut écrire $*$ Ptr.

  Généralement, pour définir le pointeur d'une variable var de type type, on écrit

  type$*$ Ptr = &var ;

  Attention, lorsque vous déclarez un pointeur, le type du pointeur doit être le type de la variable vers laquelle le pointeur renvoie.
- Pour afficher la variable vers laquelle le pointeur renvoie, il faut écrire $*$ Ptr.

ATTENTION. Il n'est pas obligatoire, mais il est fortement conseillé de toujours initialiser un pointeur.

Lorsque vous ne le faites pas, C++ attribue une valeur arbitraire au pointeur et il pourrait pointer vers une case mémoire déjà utilisé par votre ordinateur. Il vaut mieux éviter de risquer de la modifier car cela risquerait d'endommager votre ordinateur.

Pensez donc à bien initialiser les pointeurs soit à une valeur donnée, soit à 0.

Mémoire dynamique

- La mémoire dynamique permet de demander à C++ d'allouer une case mémoire et de renvoyer le pointeur de la case correspondante. Vous ne manipulez alors la variable que par l'intermédiaire du pointeur.
- L'intérêt de cette façon de faire est que vous contrôlez la mémoire allouée.

  Si vous voulez utiliser de gros fichiers de données, vous n'avez pas forcément envie que C++ recrée le tableau en entier avant chaque utilisation, ce qui serait coûteux en temps et en espace mémoire, la mémoire dynamique vous permet d'éviter cela.

  Un autre intérêt est que la case mémoire survit à un bloc de code (entre deux accolades). Si vous n'utilisez pas de variables dynamiques, les variables créent dans un bloc disparaissent à la fin du bloc.
- Pour créer, par la gestion dynamique, une variable de type *type*, il faudra utiliser la commande *new type* et pour un tableau de variables de type *type*, ce sera *new type* [ ].
- Pour détruire une variable de pointeur *nomPointeur*, il faudra utiliser la commande *delete nomPointeur*

  Il est important de détruire la variable lorsque vous ne l'utilisez plus, sinon vous risquez d'inutilement encombrer la mémoire de votre ordinateur.

ATTENTION. Lorsque vous détruisez une variable, il est important de réinitialiser le pointeur à 0. Sinon, vous risquez de modifier une case mémoire qu'entre temps votre ordinateur pourra avoir utilisé pour autre chose.

Quelques conseils pour retravailler le TP

* La notion de pointeurs est complexe et il n'est pas étonnant qu'elle vous semble obscur de premier abord, c'est tout à fait normal.
* Le gros intérêt des pointeurs est la gestion de la mémoire et comme vous pouvez le voir dans l'exercice 3, de vous permettre de définir des fonctions hors de la fonction main sans que des choses trop étranges se produisent. Il n'est donc pas nécessaire d'utiliser systématiquement des pointeurs dans toutes vos lignes de code.
* Il est important cependant que vous ayez une certaine compréhension de la mécanique des pointeurs. Notamment, la différence entre utiliser le pointeur de la variable avec & ou au contraire manipuler la variable par son pointeur avec $*$, ce qui est modifié lorsque vous changer le pointeur ou la valeur associée, doit être clair dans votre esprit.
* Par la suite, vous verrez que la bibliothèque <vector> a sa propre gestion de la mémoire en utilisant sans l'expliciter totalement les pointeurs. Une bonne compréhension du fonction des pointeurs vous aidera à utiliser cette bibliothèque de façon plus naturelle.