IFT1166 – TRAVAIL PRATIQUE #2 – 26 mai 2008

Les pointeurs de A à C++!

Mohamed Lokbani

Équipes: le travail est à faire en monôme (**une seule personne**).
Remise : une seule remise est à effectuer par voie électronique le **mercredi 18 juin 2008, 22h30 au plus tard, sans possibilités de prorogation**.
Conseils: n'attendez pas le dernier jour avant la remise pour engager votre travail. Vous n'aurez pas le temps nécessaire pour le réaliser.

But : ce TP a pour but de vous faire pratiquer la programmation non orientée objet du C++. Vous allez manipuler les pointeurs sous toutes les formes ainsi que la notion de « structure ».

Énoncé : ce travail est divisé en 2 exercices. Pour le premier, vous n'allez remettre que le rapport contenant la réponse aux questions posées, alors que pour le second, vous devez écrire et remettre un programme accompagné d'un rapport expliquant la démarche suivie.

Exercice -1- Le but de cet exercice est d'écrire une série d'instructions reliées à la notion de pointeurs. Pour chaque instruction vous devez nous expliquer ce qui se produit dans l'arrière scène. Si l'instruction à écrire provoque une erreur lors de la compilation ou l'exécution du programme, expliquez l'erreur et dites comment faudra-t-il la corriger. Il ne vous est pas demandé de remettre un programme pour cet exercice. Vos réponses et vos explications doivent figurer dans un rapport. Ceci écrit, vous pouvez quand même vous aider en écrivant un programme. Les questions sont reliées et se suivent. Les séries sont indépendantes mais nous vous conseillons de les faire dans l'ordre.

Série -1-
Q1- Déclarez les deux pointeurs « ptrx » et « ptry » qui pointent des valeurs du type « int ».
Q2- Déclarez la variable « varx » du type « int » sans l'initialiser et faites en sorte que « ptrx » pointe cette variable.
Q3- Déréférencez le pointeur « ptry » pour qu'il contient la valeur « 53 ».
Q4- Assignez « ptrx » à « ptry ».
Q5- Déréférencez le pointeur « ptry » pour qu'il contient la valeur « 99 ».
Q6- Quel est la valeur pointée par « ptrx »?
Q7- À l'aide de votre programme de test, indiquez la valeur des adresses mémoires associées aux variables « ptrx », « ptry » et « varx ».
Q8- Affichez la valeur contenue dans les variables « ptrx » et « ptry ». (Attention, il ne s'agit pas de déréférencer les pointeurs).

Série -2-
Q9- Déclarez deux pointeurs « ptra » et « ptrb » qui pointent des valeurs du type « int ».
Q10- Faites en sorte que pour chaque pointeur « ptra » et « ptrb » il lui soit alloué une zone mémoire permettant de contenir un seul élément du type « int ».
Q11- Déréférencez les pointeurs « ptra » et « ptrb » pour permettre de stocker respectivement les valeurs « 5 » et « 6 ».

Série -3-
Q12- Soit le fragment de code suivant :

```
int untab[] = {5, 19, 14, 8, -10, 99};
int *unptr;
```

Q12- Initialisez de deux manières différentes le pointeur « unptr » avec le premier élément du tableau.
Q13- Expliquez pourquoi cette instruction est incorrecte?

```
untab = unptr;
```

Q14- Expliquez le fragment de code suivant et que va-t-il s'afficher en sortie?

```
for (int i=0; i<6; i++){
   cout << "tab[" << i << "]" << tab[i] << endl; // ligne -A-
   cout << "unptr+" << i << "= " << *(unptr+i) << endl; // ligne -B-
}
```

Q15- Si vous permutez la ligne « -B » par ce qui suit, le fragment de code sera-t-il correct? Si oui, que va-t-il s'afficher en sortie?

```
cout << "unptr+" << i << "= " << *unptr++ << endl; // ligne -B-
```

Q16- Si vous permutez la ligne « -B » par ce qui suit, le fragment de code sera-t-il correct? Si oui, que va-t-il s'afficher en sortie?

```
cout << "unptr+" << i << "= " << *(++unptr) << endl; // ligne -B-
```

Série -4-

Le langage « C » et donc par extension le langage « C++ » dispose de la fonction « strcpy » permettant de copier une chaîne dans une autre. Soit le fragment de code suivant qui représente une version de cette fonction :

```
char *mon_strcpy(char destination[], char source[]) {
   int index = 0;
   while (source[index] != '\0') {
          destination[index] = source[index];
          index++;
   }
   destination[index] = '\0';
   return destination;
}
```

La fonction suppose que la destination a assez d'espace pour contenir la chaîne source. Récrivez cette fonction en n'utilisant que les pointeurs. Il ne doit y avoir aucune référence aux tableaux. La signature d'une telle fonction est comme suit :

```
char *mon_strcpy(char destination*, char source*);
```

Exercice -2- Le but de cet exercice est d'assimiler la notion de structure. Pour cet exercice, vous devez écrire un programme « C++ » et un rapport expliquant la démarche suivie.
Soit la structure :

```
struct Test {
   char* data;
   int index;
   Test* suivant;
};
```

Q1- Soit la variable « nbre » du type « int » initialisée avec une valeur aléatoire comprise entre « 1 » et « 20 ».
Q2- Soit « tabptr » un tableau de pointeurs du type « Test » pouvant contenir des pointeurs sur des structures du type « Test ». C'est un tableau de pointeurs de pointeurs du type « Test » ayant la taille « nbre ».
Q3- Écrivez la fonction « MemAlloc » qui sert à allouer de manière itérative le tableau « tabptr » et tous les éléments. Pour chaque structure allouée, la taille de « data » ainsi que son contenu sont tirés au hasard. Il en sera de même pour la valeur de la variable « index ». (voir annexes)
Q4- Écrivez la fonction « LibMem » qui sert à libérer tout l'espace mémoire alloué pour le tableau « tabptr » et ses composantes.
Q5- Soit la variable « debstr » un pointeur vers la première structure du tableau « tabptr », en utilisant « debstr » et le pointeur « suivant » de la structure « Test », écrivez la fonction « CompteStr » qui permet de compter le nombre de

structures définies. Attention pour cet exemple, le nombre obtenu doit être égal à la valeur de « nbre », sinon vous avez un bug dans le programme.
Q6- Écrivez la fonction « SommeIndex » qui permet de calculer la somme des « index ». Cette fonction retourne un « int ».
Q7- Écrivez la fonction « ChercheNOcc » qui permet de rechercher la nième occurrence d'une chaîne donnée dans la chaîne « data ». Cette fonction retourne un pointeur sur l'élément de « data » où commence l'occurrence recherchée. Pour cette question, vous ne pouvez pas utiliser les fonctions « strstr » et « find ». Par ailleurs, il ne vous est pas permis de convertir la chaîne du type « char* » vers le type « string ».

À noter que les questions ont été énoncées de manière à pouvoir suivre à la trace ce qui se passe en arrière plan. Nous aurions pu énoncer les questions autrement ou suivre carrément une autre approche, si l'optimisation était notre premier critère.

Pour résoudre cet exercice, je vous suggère d'abord de ne déclarer que 4 structures « à la main ». Par la suite suivre l'approche indiquée ci-dessous :
- Commencez par allouer les éléments de la structure en utilisant des caractéristiques fixes (valeurs connues).
- Libérez les éléments alloués.
- Écrivez les différentes fonctions demandées. Pour le comptage des structures, vous connaissez déjà le nombre total, donc c'est une façon de vérifier que la procédure codée est correcte. Pour la somme des « index », là aussi, la valeur de chaque « index » étant déterminée, vous pouvez calculer préalablement la somme totale et comparer ce résultat avec la valeur retournée par votre fonction. Pour la recherche d'une chaîne dans une autre, initialisez « data » avec l'une des chaînes proposées en annexe 2 et recherchez dans cette chaîne une expression qui s'y trouve déjà. Dupliquez la chaîne recherchée dans la chaîne data, et recherchez la deuxième ou la troisième occurrence.
- Ayant finalisé toutes ces étapes, initialisez vos paramètres avec des nombres aléatoires et vérifiez que tout est ok.
- Finalement prenez en considération que vous avez affaire à un tableau de pointeurs de pointeurs qui doit contenir l'ensemble des éléments.

Cette démarche nécessite plusieurs tests à chaque étape et c'est une bonne approche pour vous pousser à la comprendre dans globalité. Mais elle a un « inconvénient », le temps! Elle va prendre un certain temps. Un seul conseil, n'attendez pas la dernière minute pour commencer votre travail. **La panique de la feuille blanche conjuguée avec les pointeurs c'est un cocktail explosif!**

Pour les experts en programmation, vous pouvez oublier ma précédente suggestion et aller droit au but en codant le programme en un seul bloc.

Nous vous proposons dans le fichier « tp2exo2.cpp », qu'il faudra compléter, la signature des différentes fonctions..

Hypothèses et contraintes :

- pour commencer, vous devez déjà respecter les différentes contraintes de programmation précédemment décrites.
- IFT1166 est un cours en C++, donc votre programme doit être écrit en C++ et non pas en C ni en Java! Nous n'autoriserons aucune référence au langage C. Par exemple, l'instruction [#include <stdio.h>] fait référence au langage C donc elle n'est pas autorisée. Utilisez plutôt [#include <iostream>].
- ce travail n'est pas un exercice d'algorithmique donc pas besoin de compliquer le travail pour rien!
- vous devez vous assurer de ne pas changer les noms des fichiers et de respecter par la même occasion le format de l'affichage en sortie.

Rapport : le fichier « rapport.pdf » va décrire comment vous avez procédé pour réaliser ce travail (pour le contenu d'un rapport voir aussi la « FAQ » sur la page web du cours). Le rapport doit être au format « pdf » (voir là aussi la FAQ sur la page web du cours sur la ma façon de générer un fichier au format « pdf »).

Remise : il est important de noter que votre TP sera compilé avec gcc3.4.2. Si par choix, vous décidez d'utiliser un autre compilateur, vérifiez que le code que vous avez produit (devant normalement fonctionner correctement chez vous) fonctionne aussi sur les ordinateurs de la DESI. Pour avoir la version du compilateur, utilisez la commande "**gcc –v**", qui devra donner le numéro de version "**3.4.2**".

Par ailleurs, assurez-vous de la présence de l'option « pedantic » sur la ligne de compilation. (Cette option n'est pas activée par défaut dans l'utilitaire « devcpp ». Il faudra donc penser à l'activer).
Si vous avez regroupé l'ensemble des fichiers dans un seul fichier « tp2.zip », vous devez faire la remise comme suit :

1. commencez d'abord par vous connecter sur la machine « remise » comme il a été pratiqué dans la démo #01.
2. envoyez l'ensemble de vos fichiers par la procédure de remise électronique habituelle (Pour obtenir de l'aide sur cette commande, tapez dans un Xterm : man remise). Respectez les noms des fichiers.

   remise ift1166 tp2 tp2.zip

3. Vérifiez que la remise s'est effectuée correctement.

   remise –v ift1166 tp2

Barème : ce TP2 est noté sur **15 points**.

Exercice -1-		**6 points**
Exercice -2-		**9 points**
	Compilation et respect des spécifications	1
	Codage, commentaires etc.	3
	Rapport	2
	Tests	3

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ….

- La non remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- Les programmes ne contenant pas d'en-tête, -1 point.
- Un programme qui ne compile pas : 0.
- Un programme qui compile mais ne réalise pas les choses prévues dans la spécification : 0.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.25 et plus.
- Le non respect du nom du fichier va générer une erreur de compilation donc un des points de la spécification n'a pas été respecté : 0.
- Aberration dans le codage : même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long!

Des questions à propos de ce TP?

Une seule adresse : dift1166@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre email, au moins la chaîne: [IFT1166] et une référence au tp02.

Mise à jour

26-05-2008 diffusion

Annexe -1-

Génération des nombres aléatoires

```
#include <iostream>

using namespace std;

int main() {

    // Pour générer des nombres différents à chaque fois
    // sinon mettre une valeur égale à 1 à la place de NULL

    srand(static_cast<unsigned>(time(NULL)));

    // Retourne une valeur aléatoire du type int

    int x = rand();

    cout << "x: " << x << endl;

    // Retourne une valeur aléatoire entre 0 et 99

    int y = rand()%100;

    cout << "y: " << y << endl;

    // Retourne une valeur aléatoire entre 1 et 100

    int b = (1 + rand()%100);

    cout << "b: " << b << endl;

    char tab[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

    // Retourne une valeur aléatoire entre 0 et 51 qui va servir à récupérer
    // la lettre du tableau « tab »


    char z = tab[rand()%52];

    cout << "z: " << z << endl;

    return 0;

}
```

Annexe -2-

Initialisation de « data »

Cette dernière doit être initialisée aléatoirement avec l'une des 5 chaînes suivantes :

"un tableau et une armoire et une chaise"
"ceci cela mais pas ceci"
"est un jour est une nuit est une journee"
"do re mi fa sol do la si do mi"
"les bateaux, les plages, les mers, les poissons"