

Exercice 1 – Manipulation basique de pointeurs en C++

5 septembre 2017

Préparé par
Benjamin Lemelin

1 Travail à effectuer

Ouvrez le projet Visual Studio fourni avec cet énoncé. Ce projet contient plusieurs fichiers « `.h` » et « `.cpp` » à compléter. L'exercice est en deux parties : quelques expériences guidées avec les pointeurs suivies d'exercices avec les tableaux dynamiques.

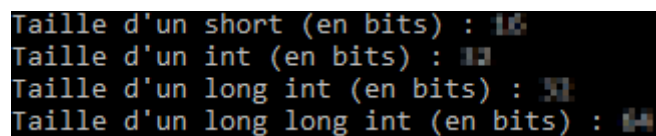
Notez que les défis de cet exercice ne sont pas à prendre à la légère. Ne vous lancez pas dedans si vous ne savez pas comment vous y prendre ou si vous n'avez pas le temps.

1.1 Expériences sur la taille des variables en C++

Créez 4 variables : un « `short` », un « `int` », un « `long int` » et un « `long long int` ». Ensuite, utilisez le mot clé « `sizeof` » pour obtenir la taille (en octets) de chaque variable. Par exemple :

```
int a = 1;
int sizeOfInts = sizeof(a);
```

Imprimez la taille en bits (pas en octets) de chaque variable dans la console.



```
Taille d'un short (en bits) : 16
Taille d'un int (en bits) : 32
Taille d'un long int (en bits) : 64
Taille d'un long long int (en bits) : 128
```

Questions méritant réflexion :

- Quelle est la taille de leurs versions non signées (« `unsigned long long int` » par exemple) ?
- Quelle est la taille d'un « `bool` » étant donné qu'il n'a que 2 valeurs possibles ?

1.2 Expériences sur la taille des pointeurs en C++

Créez 4 pointeurs : un pointeur vers un « `short` », un pointeur vers un « `int` », un pointeur vers un « `long int` » et pointeur vers un « `long long int` ». Ensuite, utilisez le mot clé « `sizeof` » pour obtenir la taille (en octets) de chaque pointeur. Par exemple :

```
int* a = new int(1);
int sizeOfIntPtr = sizeof(a);
```

Imprimez la taille en bits (pas en octets) de chaque pointeur dans la console.

Questions méritant réflexion :

- Quelle est la taille d'un pointeur de « `sf::window` » ?
- Quelle serait la taille d'un tableau de cinq (5) pointeurs de « `unsigned long long int` » ?

1.3 Expériences sur les adresses

Déclarez une variable de type « `float` » initialisé à « `3.1416f` » nommée « `pi` ». Déclarez ensuite ceci :

```
float* piPointer = &pi;
```

L'opérateur « `&` » permet d'obtenir l'adresse d'un élément. Nous avons donc obtenu l'adresse de la variable « `pi` » déclarée sur la pile et l'avons placé dans un pointeur.

Imprimez leurs valeurs dans la console.

```
Valeur de PI : 3.1416
Valeur de PIPointer : 3.1416
```

Déclarez maintenant ceci, de sorte à obtenir l'adresse de « `piPointer` ».

```
float** piPointerPointer = &piPointer;
```

Imprimez « `piPointer` » et « `piPointerPointer` » dans la console, mais ne les déréférenciez pas de manière à voir les adresses.

```
Adresse de PI : 00401000
Adresse de PIPointer : 00401004
```

Questions méritant réflexion :

- Qu'observez-vous en imprimant les adresses de « `piPointer` » et « `piPointerPointer` » ?
- Que se passe-t-il si vous faites un « `delete` » sur « `piPointer` » ? Pourquoi ?
- Comment ferez-vous pour obtenir la valeur de « `piPointerPointer` » ?

1.4 Expériences sur les tableaux de caractères

La classe « `string` » est essentiellement un tableau dynamique de « `char` », comme ceci :

```
char* characters = new char[100];
```

Il est donc courant de voir un « `char*` » au lieu d'un « `string` », car c'est un peu plus léger. De plus, le langage possède des particularités facilitant l'usage de « `char*` ».

Déclarez ceci en guise de démonstration :

```
char* helloText = "Bonjour a vous!";
```

Notez que cette chaîne de caractères est déclarée statiquement, même si le type est « `char*` ».

Placez un point d'arrêt (« Breakpoint ») à la dernière ligne de votre main. Exécutez votre programme en mode « Debug ». Dans la fenêtre « Espion », ajoutez un espion pour « **helloText** ».

Espion 1	
Nom	Valeur
helloText	0x001be308 "Bonjour a vous!"
	66 'B'

Remarquez que le débogueur ne vous montre qu'une seule case mémoire. Ajoutez un autre espion, cette fois nommé « **helloText,16** ». Cela devrait vous permettre de voir les 16 cases du tableau.

Questions méritant réflexion :

- Pourquoi le débogueur vous montre-t-il, par défaut, qu'une seule case ?
- Envoyez « **helloText** » directement à un « **std::cout** ». Qu'obtenez-vous ? Est-ce logique, sachant que vous ne lui avez pas envoyé la taille de la chaîne de caractères ?
- La chaîne de caractères « **helloText** » utilise 16 cases mémoire, et non pas 15 comme on pourrait s'y attendre. Observez donc le 16^e caractère et tentez de déterminer à quoi il sert en vous basant sur la question précédente.
- Que se passe-t-il si vous faites un « **delete** » sur « **helloText** » ? Pourquoi ?

1.5 Lire un tableau d'entiers à partir de la console

Dans votre fonction « **main** », utilisez la fonction « **readInt** » définie dans « **console.h** » pour lire un tableau d'entiers (soit un « **int*** »).

La taille du tableau doit être choisie par l'utilisateur. Demandez donc la taille du tableau en premier.

```
Combien de nombres?
2
Entrez un nombre :
5
Entrez un nombre :
8
Appuyez sur une touche pour terminer le programme...
```

Afin que votre code soit bien découpé, complétez et utilisez la fonction « **readIntArray** » définie dans « **console.h** ». Consultez la documentation de cette méthode pour les détails.

Questions méritant réflexion :

- Qui est responsable de supprimer le tableau retourné par la fonction « **readIntArray** » ?
- Pourquoi « **readIntArray** » nécessite-t-il de connaître la taille du tableau à l'avance ?

1.6 Afficher le tableau dans la console

Dans « `console.h` », ajoutez une fonction nommée « `showIntArray` » dont le but est d'afficher un tableau dans la console. Affichez donc le tableau saisi par l'utilisateur dans la console en utilisant cette fonction.

```
Voici votre tableau :  
{2 ,9 ,7 ,1 ,4}
```

Faites aussi la documentation pour votre fonction.

1.7 Trier le tableau

Faites une fonction nommée « `sortIntArray` », dans un nouveau fichier « `.h` », dont le but est de trier un tableau en ordre croissant. Cette fonction doit modifier le tableau original, et non pas créer un nouveau tableau.

Pour ce faire, utilisez le « Tri à Bulle », dont voici le pseudocode :

```
POUR i DE taille - 1 À 1  
    POUR j DE 0 À i - 1  
        SI tableau[j+1] < tableau [j] ALORS  
            Échanger tableau[j+1] et tableau[j]  
        FIN SI  
    FIN POUR  
FIN POUR
```

1.8 Afficher le tableau trié dans la console

Affichez le tableau trié dans la console.

```
Voici votre tableau trie :  
{1 ,2 ,4 ,7 ,9}
```

1.9 Éliminer les fuites de mémoire

Vérifiez que votre programme n'a pas de fuite de mémoire.

4 Modalités de remise

Remettez le projet Visual Studio sur LÉA à l'heure et à la date indiquée par votre professeur. Faites attention à ne pas remettre de fichiers inutiles.