Sujet 3 : découverte des pointeurs	
Buts :	Tableaux dynamiques, pointeurs, partage de la mémoire, allocation dynamique, désallocation dynamique, passage par adresse.

Exercice 1 : passage par adresse

Dupliquez dans le répertoire qui contiendra les solutions que vous apporterez aux problèmes soulevés dans le Sujet 3, la solution que vous avez apporté à l'exercice 1 du Sujet 1 (Calculatrice - types natifs).

Modifiez la solution initiale en remplaçant tous les passages de paramètres des fonctions effectués par référence par des passages par adresse. Validez et commentez la solution apportée.

Exercice 2 : pointeurs sur des données existantes

1. Dans une fonction nommée `decouvertePointeurs` déclarez 5 variables dont les types seront `int`, `unsigned int`, `string`, `double` et `long double`. Vous les initialiserez aux valeurs respectives suivantes : -1, 1, "$Hello$", 3.14159 et $2.75e-3$. Affichez distinctement les valeur, adresse et taille (opérateur `sizeof`) de l'ensemble ces variables.

2. Créez un pointeur pointant sur chacune des variables précédemment définies. Vous les nommerez respectivement `pi`, `pui`, `ps`, `pd` et `pld`. Affichez distinctement les valeur, adresse et taille de chacun de ces pointeurs. Commentez le résultat obtenu entre autres concernant la valeur contenue dans les pointeurs.

3. Modifiez la valeur du `string` par l'intermédiaire du pointeur lui correspondant. Vérifiez que la chaîne a été bien modifiée en l'affichant une fois par le biais du pointeur déréférencé, une fois directement.

4. Faut-il libérer la mémoire associée aux 5 pointeurs avec l'opérateur `delete` ? Si ce n'est pas le cas, de quelle manière la mémoire va-t-elle être libérée ?

Exercice 3 : pointeurs et allocation dynamique

1. Définissez une fonction `initialisePointeur` qui prend en paramètre un pointeur de `int` que l'on nommera `ptr`. Cette fonction aura pour mission d'allouer dynamiquement le pointeur. Elle affichera ensuite sa valeur, son adresse et la valeur de l'objet pointé.

2. Dans la fonction `decouvertePointeurs` (qui devra impérativement être placée après la fonction `initialisePointeur` dans le fichier source) appelez la fonction `initialisePointeur` avec le pointeur `pi`. L'affichage de `ptr` et de `pi` est-il le même ? Justifiez votre réponse. Si ce n'est pas le cas proposez une(des) solution(s) pertinente(s) pour résoudre ce problème.

3. Est-on obligé de libérer la mémoire associée au pointeur `pi` avec l'opérateur `delete` ? Justifiez votre réponse.

Exercice 4 : tableaux dynamiques 1D et passage par adresse

Dans cet exercice on privilégiera le formalisme "arithmétique des pointeurs" (manipulation des adresses et l'opérateur d'indirection `*`) à l'utilisation de l'opérateur `[ ]`.

1. Écrire une fonction qui permet d'allouer dynamiquement un tableau dont la taille est passée en paramètre et donc les valeurs seront initialisées de façon aléatoire avec des valeurs comprises entre 0 et 100 (à l'aide de la fonction `rand()`).

2. Écrire une fonction qui affiche le contenu d'un tableau sous la forme :

```
15 ,  12  ,   56 ,   69 ,     3 ,    99 ,    75
```

4. Écrire une fonction qui renvoie l'indice de la valeur minimale contenue dans un tableau dynamique d'entiers.

5. Écrire une fonction qui renvoie la moyenne des valeurs contenues dans un tableau dynamique d'entiers.

6. Écrire une fonction qui permet d'échanger le contenu de deux cases du tableau dynamique d'entiers valeurs de ces deux indices.

7. Écrire une fonction qui permet de dupliquer un tableau dynamique d'entiers.

Exercice Complémentaire

L'objet de cet exercice est de programmer un jeu. L'idée de départ est de partir d'une suite de nombre entiers, par exemple : 8, 3, 5, 5, 7, 9, 1.

Deux joueurs jouent à tour de rôle, un coup consistant à retrancher une valeur à l'un des nombres de la suite de départ, par exemple :

- Le premier joueur désire retrancher 4 au 3^e nombre de la suite
  la suite devient : 8, 3, 1, 5, 7, 9, 1.
- Le second joueur désire retrancher 7 au 5^e nombre de la suite
  la suite devient : 8, 3, 1, 5, 0, 9, 1.

Le gagnant est celui qui n'a plus rien à retrancher lorsque arrive son tour :
Le joueur qui est devant la suite 0, 0, 0, 0, 0, 0, 0 après le coup de l'autre a gagné.

On souhaite réaliser un programme C++ qui permette à deux joueurs de jouer.

1. Écrire la fonction `creeSuiteVide` permettant de définir la taille de la suite initiale du jeu, et de créer un tableau d'entiers qui la stockera. L'initialisation de la taille consiste en un tirage aléatoire d'une valeur comprise entre 3 et une valeur maximale connue prise en paramètres.

2. Écrire une fonction `initialise` initialisant aléatoirement la suite de départ. Celle-ci consiste à générer des valeurs comprises entre 1 et une valeur maximale connue prise en paramètres. La suite sera stockée dans un tableau d'entiers `t` de taille `n`.

3. Écrire une fonction `saisieJeu` permettant de saisir simultanément sur quel élément de la suite le joueur veut agir (`quelelement`) ainsi que la valeur qu'il voudra lui retrancher (`valeur`). Cette fonction devra par ailleurs contrôler la validité des données saisies par le jouer. Ces saisies doivent respecter les règles suivantes :

- `quelElement` doit exister dans le tableau (entre autres, il ne peut excéder la taille de la suite) ;
- `quelElement` ne peut pas être à un emplacement de la suite dont le contenu est déjà nul ;
- `valeur` doit être strictement positive et ne peut excéder la valeur de l'élément présent à l'emplacement `quelElement` dans la suite.

4. Écrire une fonction `affiche` permettant d'afficher la suite.

5. Écrire un fonction `finJeu` permettant de savoir si le jeu est terminé, *i.e.* si la suite ne contient plus que des 0. Elle retournera vrai si c'est le cas, faux sinon.

6. Écrire une fonction `simulePartie` simulant le déroulement d'une partie entre 2 joueurs, connaissant la suite initiale. Cette fonction retournera le numéro du joueur gagnant.

7. Écrire la fonction `main` permettant de simuler une partie.