Université Paris Diderot
M2 Modélisation Aléatoire

Programmation C++
Année 2013-2014

TP n°2

Page du cours : `http://www.liafa.univ-paris-diderot.fr/~carton/Enseignement/C++/`

Exercice 1 Écrire un programme qui déclare une variable entière i puis qui affiche son adresse. Le programme est court.

Exercice 2 (Pointeur void*)
Écrire une fonction `void adresse(void* a)` qui affiche l'adresse donnée en paramètre.

Exercice 3 (Échanger les valeurs de deux paramètres)
Écrivez une fonction `void echanger(int *pa, int *pb)` qui permet d'échanger les valeurs de deux entiers. On l'utilisera dans la fonction `main` de la façon suivante :

```
int a = 1, b = 2;

echanger(&a, &b);
cout << "a = " << a << ", b = " << b << endl;
```

Exercice 4 (Tableau Aléatoire)
Écrire un programme qui remplit un tableau de 20 entiers avec des nombres aléatoires allant de 1 à 100. Puis afficher les éléments de ce tableau.

Exercice 5 (Chaîne de caractère)
Une chaîne de caractère est un tableau de `char` dont le dernier caractère est null (ie vaut 0). Écrire une fonction`void supprimerVoyelle(char** chaine)` qui retire toutes les voyelles d'une chaîne.
Aide : chaque caractère est codé par un nombre entre 0 et 255. On appelle ce nombre le code ASCII d'un caractère. En C++ pour obtenir le code d'un caractère on le place entre apostrophe. Exemple : `'a'` représente le code du caractère `a` et `'0'` représente le code du caractère `0`. Attention `0!='0'`.

Exercice 6 (Passage d'un tableau comme paramètre)
Écrivez une fonction `void afficherTableau(int len, char **tableau)` qui affiche le contenu d'un tableau de chaînes de caractères. (un tableau de chaine de caractère est un tableau de tableau de `char`) Vérifiez votre fonction comme ceci :

```
char **tab = {"Bonjour", "Monsieur", "Madame" };
afficherTableau(3, tab);
```

Affichez également le contenu du tableau `char **argv` que la fonction `main` a reçu comme paramètre lors de l'exécution de la commande suivante

```
./exo2 Bonjour tous .
```

Exercice 7 *(Nombre premier)
Écrivez un programme qui affiche les nombres premiers compris entre 1 et n en utilisant le crible d'Ératosthène. Vous utiliserez un tableau d'entiers pour stocker les nombres de 1 à n. n sera un argument du programme. On rappelle que la première case d'un tableau est la case 0. Le paramètre n sera passé dans la ligne de commande.
Exemple : `$ ./premier 20`
`2 3 5 7 11 13 17 19` Aide : le paramètre `int argc` de la fonction `main` prend la valeur du nombre d'arguments de la ligne de commande. Le paramètre `char** argv` est un tableau qui contient les chaines de caractères.

Exercice 8 (Tri Tableau)
On veut créer une fonction qui trie un tableau d'entier par insertion.

- Écrire une fonction `inserer(int tab[], int p, int q)` qui insère la valeur de la case q du tableau `tab` à la position p et qui décale les autres valeurs du tableau vers la droite. On suppose que $p <= q$. Par exemple :
  ```
  // tab: 1 7 8 6 22
  inserer(tab, 1, 3);
  // tab 1 6 7 8 22
  ```
  Remarque : l'indice le plus grand est par convention l'indice le plus "à droite" et l'indice 0 (le plus petit) est l'indice le plus "à gauche".
- Écrivez une fonction récursive `int chercher(int* T,int taille,int val)` qui retourne l'indice de la case où l'entier `val` devrait être inséré par insertion. On suppose que T est trié donc procédez par dichotomie.
  (Les valeurs contenues dans la case retournée et dans les cases à sa droite devront être décalées vers la droite avant l'insertion).
- Écrivez une fonction `void trier(int T[],int taille)` qui trie le tableau T par insertion en utilisant les fonctions `inserer` et `chercher`.
- Essayez la fonction sur un tableau rempli par des nombres aléatoires.

Exercice 9 (Intégrer de façon approchée)
Dans cet exercice on intégrera des fonctions polynomiales de façon approchée.

1. Commencez par définir une classe `Polynome` qui répresente une fonction polynomiale
   $$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots a_1 x + a_0.$$
   Le constructeur prend un tableau des coefficients $a_0, \dots, a_n$ ainsi que le degré n.
   Écrivez la fonction `double Polynome::IntegrationExacte()` qui renvoie la valeur
   $$\int_0^1 f(x)dx.$$
2. Pour la méthode d'intégration approchée on aura besoin d'une classe `Point` qui répresente un point dans $[0,1] \times [0,a]$. Définissez la classe `Point`. Son constructeur prend un paramètre a et initialise les coordonnées x, y du point avec des valeurs aléatoires dans $[0,1] \times [0,a]$.
3. Écrivez maintenant une fonction `int Polynome::enDessous(Point p)` qui renvoie 1 si le point p est situé en dessous de la courbe du polynome.
4. Maintenant on peut définir la fonction `double Polynome::IntegrationApprochee(int N)`. Cette fonction tire aléatoirement N fois un point dans $[0,1] \times [0, \sum_{i=0}^{n} a_i]$ et compte le nombre de fois où ce point est en dessous de la courbe. En fonction de cette valeur calculez une valeur approchée de l'intégrale.

Testez vos classes avec le code suivant :

```
int N = 1;
int a[] = {1, 2, 3, 4, 5};
Polynome p(a, 5);

cout << "Integration exacte: " << p.IntegrationExacte() << endl;

for(int i=0; i < 6; i ++)
{
  cout << "Integration approchee(" << N << "): " << p.IntegrationApprochee(N) << endl;
  N = N * 10;
}
```

Annexes :

La fonction `int rand()` renvoie des nombres aléatoires compris 0 et la constante `RAND_MAX`, pour utiliser `rand` il faut inclure `<cstdlib>`. De plus cette fonction est définie dans l'espace de nom `std`.