M1103 : Structures de Données & Algorithmes fondamentaux

Feuille TP n° 2 – distribuée en semaine DUT n°11

Algorithmes classiques sur des tableaux

OBJECTIFS PEDAGOGIQUES :

1.- Codage d'algorithmes sous forme modulaire.
2.- S'exercer à l'écriture progressive de programmes.

ÉVALUATION :

- Les exercices demandés pourront faire l'objet d'une évaluation en fin de module
- Il en est de même pour les techniques de développement et de débogage apprises en M1102 qui vous sont demandées d'appliquer

DOCUMENTS A VOTRE DISPOSITION POUR REALISER CE TP :

- Sur le WebCampus, dans la zone associée à ce module APL-M1103 :
  - Tp2.pdf
  - Fichiers `modulesCommunsTri.h et modulesCommunsTri.cpp`

DIRECTIVES GENERALES – ORGANISATION DES REPERTOIRES DE TP

1. Dans votre répertoire consacré aux TPs du module M1103, créer un répertoire TP2.
2. Dans le répertoire TP2, vous créerez un unique projet : Projet **`testTrisTableaux`**

DIRECTIVES PARTICULIERES POUR CETTE SEANCE DE TP

Découpage du code. Le code sera découpé selon le principe suivant :

- Un fichier `main.cpp`, contenant le programme de test appelant les sous-programmes que vous avez décidé de coder. A titre de modèle, voici le code d'un programme testant **`triSelectionDePlace.`**

```
#include "modulesCommunsTri.h"
#include <iostream>

using namespace std;

int main()
{
   const unsigned int TAILLE = 10 ;
   int monTab [TAILLE] = {7, -6, 8, 10, 2, -3, 6, 9, 0, 2}; // non trié
//   int monTab [TAILLE] = {-6, -3, 0, 2, 2, 6, 7, 8, 9, 10}; déjà trié croissant
//   int monTab [TAILLE] = {10, 9, 8, 7, 6, 2, 2, 0, -3, -6}; trié à l'envers

   cout << "TRI par SELECTION de PLACE" << endl << endl;

   // tab, TAILLE >> afficher tableau >> (écran)
   cout << "AVANT, monTab = " ;
   afficherTableauEntiers(monTab, TAILLE);
   cout << endl;

   // monTab, TAILLE >> TRI Insertion de Place >> monTab
   triSelectionPlace (monTab, TAILLE);

   // tab, TAILLE >> afficher tableau >> (écran)
   cout << "APRES, monTab = " ;
   afficherTableauEntiers(monTab, TAILLE);
   return 0;
}
```

`modeleMain.cpp`

Vous le complèterez avec les tests des autres sous-programmes de tri.

- Les déclarations des sous-programmes de tri et des sous-programmes communs (par exemple, **`echanger`** ou encore **`afficherTableauEntiers`**) seront tous ajoutés aux fichiers `modulesCommunsTri.h` mis à votre disposition :

```
#ifndef MODULESCOMMUNSTRI_H
#define MODULESCOMMUNSTRI_H

void afficheTableauEntiers  (const int tab[],
                             unsigned int nbTab);
// But : affiche à l'écran le contenu d'un tableau tab de nbTab éléments

void echanger (int& a, int& b);
// échange le contenu des 2 paramètres entiers a et b

#endif  // MODULESCOMMUNSTRI_H
```

`modulesCommunsTri.h`

- Les corps des sous-programmes communs (par exemple, **`echanger`** ou encore **`afficherTableauEntiers`**) seront dans le fichier `modulesCommunsTri.cpp` mis à votre disposition :

```
#include<iostream>
#include "modulesCommunsTri.h"
using namespace std;

void afficheTableauEntiers  (const int tab[],
                             unsigned int nbTab)
{
    cout << "{";
    for (unsigned int i = 0; i<nbTab-1; i++)
    {
        cout << tab[i] << ", " ;

    };
    cout << tab[nbTab-1] << "} "<< endl;
}

void echanger (int& a, int& b);
// échange le contenu des 2 paramètres entiers a et b
{
  // à vous de faire
}
```

`modulesCommunsTri.cpp`

- Les corps des sous-programmes de tri seront chacun dans leur propre fichier `.cpp`, avec éventuellement des sous-programmes spécifiques.

EXERCICES A CODER DURANT CETTE SEANCE

Vous coderez les sous-programmes choisis conformément à la Feuille de TD n°2 et aux directives de découpage des sources vues au point 1.-.

Production de jeux d'essais

Dans le programme principal, vous devrez faire varier les contenus des tableaux pour produire les différents jeux d'essais qui permettront de tester la validité du programme.

Consignez les différents jeux d'essais utilisés afin de montrer à votre enseignant l'exhaustivité de vos tests. (cf. fichier modeleMain.cpp)

L'ensemble des jeux d'essais pourra être regroupé dans le propre main.cpp, ou bien dans un autre document (JeuxEssais.odt) qui sera également rangé dans votre répertoire de TP.

3.- Si vous avez terminé, finalisez le codage de la feuille de TP1.

RAPPEL DES PRINCIPALES BONNES PRATIQUES VUES JUSQU'A MAINTENANT

- Écriture progressive du code en validant chaque étape par une opération de compilation ;
- Respect des règles de nommage des variables et des constantes ;
- Chaque variable déclarée est accompagnée d'un commentaire indiquant son rôle. Ce commentaire est écrit au moment où on déclare la variable et non à la fin une fois que le programme est terminé . . .
- Chaque variable est définie par le type qui la représente au mieux : unsigned short int pour un pourcentage plutôt qu'un simple int par exemple ;
- Le code doit toujours être indenté ;
- Les structures de contrôles sont toujours écrites en deux étapes : écriture du squelette de la structure (soit manuellement, si possible en privilégiant les abréviations de Code::Blocks) puis remplissage de la structure ;
- Les paramètres des sous-programmes que vous écrivez devront répondre aux bonnes pratiques vues en cours et résumées dans les documents du module M1102 disponibles sur le webcampus.

Pensez à intégrer chacune de ces bonnes pratiques dès que vous codez et surtout, sollicitez votre enseignant pour qu'il vous donne un avis sur les codes que vous avez terminés.

Ne codez jamais un exercice sans avoir, au préalable, réalisé l'algorithme correspondant.

Même en TP, si vous devez écrire un code dont vous n'avez pas l'algorithme, lâchez le clavier, prenez une feuille et un crayon et concevez votre solution sur papier puis faîtes la valider par votre enseignant.

Vous demandez à vos enseignants ce que vous ne comprenez pas.

Aucune aide ne vous sera fournie en TP si vous n'êtes pas en mesure de montrer l'algorithme sur lequel vous vous appuyez pour élaborer votre code.