IUT de BAYONNE PAYS BASQUE - DÉPARTEMENT INFORMATIQUE
Semestre 1- 2015-2016

M1103 : Structures de Données & Algorithmes fondamentaux

Feuille TP n° 1 – distribuée en semaine DUT n°10

Algorithmes classiques sur des tableaux

OBJECTIFS PEDAGOGIQUES :

1.- Codage d'algorithmes sous forme modulaire : utilisation de sous-programmes, séparation de la spécification et de l'implémentation d'un sous-programme
2.- S'exercer à l'écriture progressive de programmes.

ÉVALUATION :

- Les exercices demandés pourront faire l'objet d'une évaluation durant le 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 (Ressources complémentaires) :
  Une archive (.zip) contenant l'algorithme de recherche de première occurrence dichotomique appliquée aux tableaux (transp. 21 du cours-chap. 1). A adapter pour compléter la documentation du programme de la question 2.

DIRECTIVES GENERALES – ORGANISATION DES REPERTOIRES DE TP

1. Dans votre espace de travail, créer un répertoire M1103 pour accueillir tous les TPs qui seront réalisés dans le cadre de ce module.

Les directives qui suivent sont identiques à celles données pour les TPs du module M1102.

2. Dans cet espace de travail, vous créerez un nouveau répertoire pour chaque feuille de TP qui vous sera distribuée. Créer donc dès aujourd'hui le répertoire TP1. Il contiendra tous les exercices qui vous sont demandés dans ce TP.
3. Workspace : Un workspace vous sera demandé par répertoire de TP.

EXERCICES A CODER DURANT CETTE SEANCE

Conformément à la Feuille de TD n°1.

DIRECTIVES PARTICULIERES

Dans chaque répertoire de projet, le code sera découpé selon le principe suivant :

- Un fichier `main.cpp`, contenant le programme de test appelant les sous-programmes cités dans la feuille de td1
- Pour chaque sous-programme que vous créez et qui sera appelé par le `main`, 2 fichiers : 1 fichier de déclaration (.h) et un fichier de définition (.cpp)

Pour comprendre un des intérêts de la séparation entre spécification de sous-programme (fichier .h) et implémentation de sous-programme (fichier .cpp), vous pouvez compléter/développer votre code (en le compilant au fur et à mesure) dans l'ordre suivant :

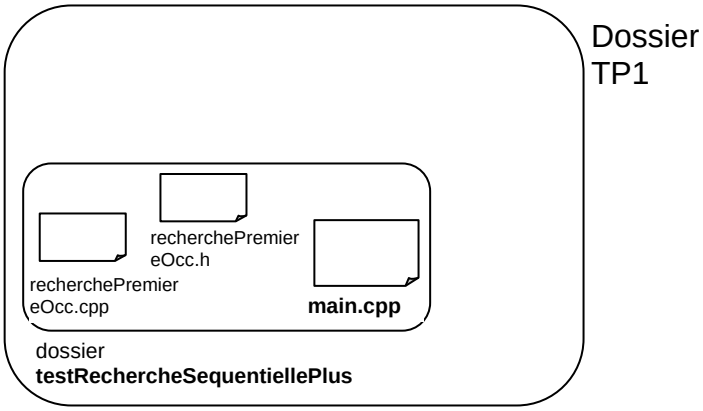
- Fichier sous-programme.h
- Fichier main.cpp
- Fichier sous-programme.cpp

1.- Recherche séquentielle (améliorée) d'une valeur dans un tableau strictement/totalement ordonné

Créer un projet **`testRechercheSequentiellePlus`** contenant un programme (**`main`**) de test du sous-programme **`recherchePremiereOcc`** (modèle recherche 1èer occ. séquentielle) vu en TD n°1.
Le programme (**`main`**) :

- Initialisera un tableau 'en dur' cad en fournissant les valeurs du tableau dans la déclaration
  Demandera à l'utilisateur de saisir une valeur à chercher dans le tableau
- Effectuera la recherche (= appel du
- sous-programme **`recherchePremiereOcc`**)
- Affichera le résultat de la recherche

A l'issue de cette opération, le répertoire TP1 aura une organisation semblable à l'illustration ci-contre :



Simplification par rapport au TD :

1) La déclaration

```
const unsigned int TAILLE = 10;
```

sera placée dans le fichier main.cpp

2) On n'utilisera pas le type présenté dans la feuille de TD :

```
typedef int UnTableauEntiers[TAILLE];     // UnTableauEntiers devient un type 'tableaux
                                          // d'entiers de 10 cases'
```

Les paramètres tableau seront donc passés selon la forme habituellement utilisée en M1102.
Exemple :

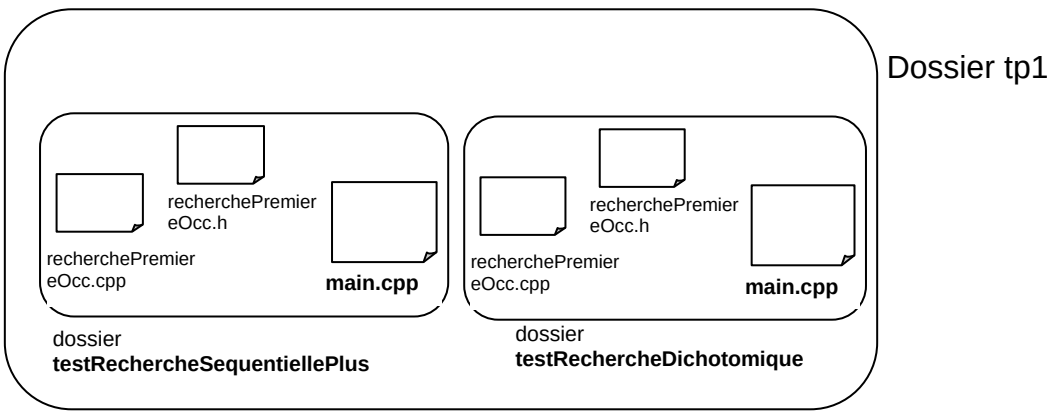
```
void afficherTableau (const int tab[], unsigned int nbTab) ;
// But : affiche à l'écran sur une même ligne tous les éléments du tableau d'entiers tab
// ayant nbTab cases.
```

2.- Recherche dichotomique d'une valeur dans un tableau d'entiers strictement/totalement ordonné

Créer un projet **`testRechercheDichotomique`** contenant un programme (**`main`**) de test du sous-programme **`recherchePremiereOcc`** (modèle recherche dichotomique) vu en TD n°1.
Le programme (**`main`**) :

- Initialisera un tableau 'en dur'
- Demandera à l'utilisateur de saisir une valeur à chercher dans le tableau
- Effectuera la recherche (= appel du sous-programme **`recherchePremiereOcc`**)
- Affichera le résultat de la recherche

A l'issue de l'opération, le répertoire TP1 aura une organisation semblable à l'illustration ci-dessous :



Simplification par rapport au TD : les mêmes que précédemment.

RAPPELS 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 le document Sous-Programmes : Bonnes pratiques, disponibles sur le webcampus dans la section TP du module M1102.

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.