|  |
| --- |
| **SR01 - TD 2**  **Types de base, déclarations** |

**Remarque**    
Ce TD est à réaliser dans le répertoire **td2**

**Exercice1: fichier ex1**

Ecrire un programme ***ex1.c***qui affiche, en utilisant la fonction *sizeof()*, la taille de types de base en c:  
 void, char, short, int , long, float, double, long double,

**Exercice2: fichiers ex2.c et ex2.h**

Créer un fichier***ex2.h*** avec la définition des constantes symboliques suivantes:

*#define MAX 7  
#define VAL 20*

Ecrire un programme***ex2.c*** qui fait appel au fichier ***ex2.h*** (avec la directive *#include*),  
dans***ex2.c,*** déclarer un tableau *tab*de taille *MAX*, et initialiser ce tableau de la façon suivante:

*tab[i] = i + VAL;*

Affiche ensuite l'ensemble des valeurs de tab

**Exercice3: Les nombres complexes: Fichier ex3a.c, ex3b.c**

* Définir un nouveau type *Complexe*à partir de la structure suivante en utilisant *typedef*:

*struct complexe {  
                            float  x;  
                            float  y;  
                          }*  
Cette structure permet de représenter les nombres complexes *x + yi*

* Ecrire un programme ***ex3a.c***qui contient une fonction *Afficher*qui récupère la partie entière et  
  la partie imaginaire d'un complexe, et les affiche dans le format correspondant:

*x - yi     si y<0  
x           si y=0  
x + yi    si y>0*

voici le prototype (l'entête) de la fonction *Afficher*:

*Void  Afficher(Complexe);*

tester dans le *main*la fonction *Afficher*

* Ajouter dans ***ex3b.c*** , la fonction *Somme*qui permet de faire la somme de deux complexes.  
  Voici le prototype de la fonction *Somme*:

*Complexe  Somme( Complexe , Complexe);*Rappel:  (x + yi) + (u + vi) = (x + u) + (y + v)i

tester dans le *main* la fonction *Somme*

* Ajouter dans ce même programme ***ex3b.c*** , la fonction *Produit*qui permet de faire le produit de deux complexes.  
  Voici le prototype de la fonction Produit:

*Complexe  Produit( Complexe , Complexe);*Rappel:  (x + yi) \* (u + vi) = (x\*u - y\*v) + (y\*u + x\*v)i

tester dans le *main*la fonction *Produit*

**Exercice 4: énumération, unions et structures.  Fichiers: *ex4.c***  
  
Récupérer les déclarations des données suivantes (voir exemple\_typedef.c):

*struct complexe*: pour les nombres complexes  
*union valeur*: qui contient les champs suivants: int entier , float reel  et struct Complexe complexe,  
*enum genre* : qui regroupe les déclarations des constantes suivantes: entier, reel et complexe  
*nombre* : nouveau type (définit avec typedef) qui contient les champs suivant: enum genre type, et union valeur valeur.

Ecrire la fonction *Saisir\_Nombre()* qui permet à l'utilisateur de saisir un nombre entier , réel ou complexe.  
il faut demander d'abord le type du nombre, ensuite suivant ce type, demander le reste des champs.  
Voici le prototype de la fonction:   *nombre  Saisir\_Nombre( );*

Ecrire la fonction *Afficher\_nombre()* qui permet à l'utilisateur d'afficher un nombre entier , réel ou complexe.  
il faut vérifier d'abord le type du nombre, ensuite suivant ce type, afficher le nombre dans le format approprié.  
Voici le prototype de la fonction:  *void  Afficher\_Nombre( nombre );*  
  
Dans la fonction *main*tester les fonctions ci-dessus.

***-- SR01--A2022***