Université Hassan 1er

Faculté des Sciences et Techniques de Settat

PROGRAMMATION EN «C» ET STRUCTURES DE DONNÉES TRAVAUX PRATIQUES

Professeur Laachfoubi Nabil

Département des Mathématiques et Informatique

Sommaire

TP 1 – Les Structures : Tableau de Structures 3
TP 2 – Les Structures : Liste Simplement Chaînée de Structures 4
TP 3 – Les Structures : Liste Doublement Chaînée de Structures 5

TP 1 – Les Structures : Tableau de Structures

Soient les déclarations suivantes :

```
struct date
{
   unsigned int j ;
   unsigned int m ;
   unsigned int a ;
};
typedef struct date Date ;

struct client
{
   unsigned int Code ;
   char * Nom ;
   char * Prenom ;
   Date DateNaissance ;
};
typedef struct client Client ;

static unsigned int CC = 0 ;  // Variable globale : Compteur de clients devant être auto-incrémenté
static Client * TC = NULL ;   // Variable globale : Pointeur pour créer un tableau dynamique de clients
```

Développer un projet modulaire permettant des répondre aux fonctionnalités suivantes :

a. Afficher les informations d'un client en se basant sur la structure client en paramètre

   Prototype : **void** AfficherClient_TSC(**Client**)

b. Afficher les informations de l'ensemble des clients

   Prototype : **void** AfficherClients_TSC()

c. Saisir les informations liées à un client à partir de son adresse mémoire passée en paramètre

   Prototype : **void** SaisirClient_TSC(**Client** ***)

d. Ajouter un client en début du tableau de clients

   Prototype : **void** AjouterClientDT_TSC()

e. Ajouter un client en fin du tableau de clients

   Prototype : **void** AjouterClientFT_TSC()

f. Supprimer un client de la liste en se basant sur son code en paramètre

   Prototype : **void** SupprimerClient_TSC(**unsigned int**)

g. Ordonner la liste des clients selon l'ordre croissant du code

   Prototype : **void** OrdonnerClientsViaCode_TSC()

h. Créer un menu permettant de faire fonctionner les différentes options

   Prototype : **void** Menu_TSC()

TP 2 – Les Structures : Liste Simplement Chaînée de Structures

Soient les déclarations suivantes :

```
struct date
{
   unsigned int j ;
   unsigned int m ;
   unsigned int a ;
};
typedef struct date Date ;

struct client
{
   unsigned int Code ;
   char * Nom ;
   char * Prenom ;
   Date DateNaissance ;
   struct client * svt ;
};
typedef struct client Client ;

static unsigned int CC = 0 ;  // Variable globale : Compteur de clients devant être auto-incrémenté
static Client * DL = NULL ;   // Variable globale : Pointeur sur le début de la liste
static Client * FL = NULL ;   // Variable globale : Pointeur sur la fin de la liste
```

Développer un projet modulaire permettant des répondre aux fonctionnalités suivantes :

a. Afficher les informations d'un client en se basant sur la structure client en paramètre

   Prototype : **void** AfficherClient_LSC(**Client**)

b. Afficher les informations de l'ensemble des clients

   Prototype : **void** AfficherClients_LSC()

c. Saisir les informations liées à un client à partir de son adresse mémoire passée en paramètre

   Prototype : **void** SaisirClient_LSC(**Client ***)

d. Ajouter un client en début de la liste des clients

   Prototype : **void** AjouterClientDL_LSC()

e. Ajouter un client en fin de la liste des clients

   Prototype : **void** AjouterClientFL_LSC()

f. Supprimer un client de la liste en se basant sur son code en paramètre

   Prototype : **void** SupprimerClient_LSC(**unsigned int**)

g. Ordonner la liste des clients selon l'ordre croissant du code

   Prototype : **void** OrdonnerClientsViaCode_LSC()

h. Créer un menu permettant de faire fonctionner les différentes options

   Prototype : **void** Menu_LSC()

TP 3 – Les Structures : Liste Doublement Chaînée de Structures

Soient les déclarations suivantes :

```
struct date
{
   unsigned int j ;
   unsigned int m ;
   unsigned int a ;
};
typedef struct date Date ;

struct client
{
   unsigned int Code ;
   char * Nom ;
   char * Prenom ;
   Date DateNaissance ;
   struct client * pre ;
   struct client * svt ;
};
typedef struct client Client ;

static unsigned int CC = 0 ;  // Variable globale : Compteur de clients devant être auto-incrémenté
static Client * DL = NULL ;   // Variable globale : Pointeur sur le début de la liste
static Client * FL = NULL ;   // Variable globale : Pointeur sur la fin de la liste
```

Développer un projet modulaire permettant des répondre aux fonctionnalités suivantes :

a. Afficher les informations d'un client en se basant sur la structure client en paramètre

   Prototype : **void** AfficherClient_LDC(**Client**)

b. Afficher les informations de l'ensemble des clients

   Prototype : **void** AfficherClients_LDC()

c. Saisir les informations liées à un client à partir de son adresse mémoire passée en paramètre

   Prototype : **void** SaisirClient_LDC(**Client ***)

d. Ajouter un client en début de la liste des clients

   Prototype : **void** AjouterClientDL_LDC()

e. Ajouter un client en fin de la liste des clients

   Prototype : **void** AjouterClientFL_LDC()

f. Supprimer un client de la liste en se basant sur son code en paramètre

   Prototype : **void** SupprimerClient_LDC(**unsigned int**)

g. Ordonner la liste des clients selon l'ordre croissant du code

   Prototype : **void** OrdonnerClientsViaCode_LDC()

h. Créer un menu permettant de faire fonctionner les différentes options

   Prototype : **void** Menu_LDC()