Language de programmation L2

Le langage C

TP2

Alain Plantec

Carnet de rendez-vous : seconde et troisième versions

Pour cette seconde version nous utilisons l'allocation dynamique.

Rappels de cours

Les fonctions *malloc* et *free* permettent respectivement d'allouer et de libérer dynamiquement de la mémoire. La fonction *malloc* réserve le nombre d'octets qui lui est passé en argument alors que free libère la mémoire dont l'adresse est passée en argument.

```
void * malloc (int nb_octets) ;
void free (void * ptr) ;
```

En C, les chaines de caractères sont manipulées via des tableaux de caractères. Le dernier caractère est suivi en mémoire par le caractère '\0'. Pour créer une chaine de caractère vide avec *malloc* on a donc besoin d'allouer l'espace mémoire pour un caractère.
De façon plus générale, pour *nb* caractères, le nombre d'octets à allouer est calculé par l'expression suivante :

```
sizeof (char) * (n + 1)
```

Exercice 1

On utilise les structures C suivantes :

```
typedef struct {
  char * nom ;
  char * num_tel ;
  char * lieu ;
  char * date ;
} t_rendez_vous;

typedef struct {
  int nb;
  t_rendez_vous * rdvs; // Tableau de t_rendez_vous
} t_carnet;
```

Les chaînes de caractères et les rendez-vous sont gérés par allocation dynamique.

Il est demandé de mettre en oeuvre les fonctions d'initialisation et de destruction suivantes :

1. *`rendez_vous_creer (t_rendez_vous * self)`*,
2. *`carnet_creer(t_carnet * self)`*,
3. *`rendez_vous_detruire(t_rendez_vous * self)`*
4. *`carnet_detruire (t_carnet * self)`*

Il est ensuite demandé de programmer les fonctions suivantes pour la gestion du carnet :

1. *`rendez_vous_afficher(t_rendez_vous * self)`* pour l'affichage du rendez_vous qui lui est passée en paramètre
2. *`carnet_afficher(t_carnet * self)`* pour l'affichage de tous les rendez-vous d'un carnet.

3. *rendez_vous_saisir (t_rendez_vous * self)* qui permet la saisie au clavier des données d'un rendez_vous;
4. *int carnet_ajouter_rendez_vous(t_carnet * self, t_rendez_vous * rdv)* qui ajoute un rendez_vous dans un carnet; retourne 0 si l'ajout c'est bien passé, 1 sinon.
5. *carnet_rechercher_rendez_vous(t_carnet * self, char * date, t_carnet * resultat)* qui recherche tous les rendez_vous pour une date donnée; le résultat est inséré dans un carnet vierge passé en argument;
6. *carnet_retirer_rendez_vous(t_carnet * self, char * nom, char * date)* qui retire un rendez_vous du tableau.

Exercice 2

Pour cette dernière version, on utilise la structure `t_carnet` suivantes :

```
typedef struct {
  int nb;
  t_rendez_vous ** rdvs; // Tableau d'adresses de t_rendez_vous
} t_carnet;
```

Le type *t_rendez_vous* reste inchangé.

On remarque que les rendez-vous sont stockés dans un tableau d'adresses et non plus dans un tableau de *t_rendez_vous*. Autrement dit, un élément du tableau *rdvs* n'est plus un *t_rendez_vous* mais un t*_rendez_vous* * et contient donc l'adresse d'un *t_rendez_vous*.

Il est demandé de mettre en oeuvre les même fonctions que pour la version précédente.

Programme principal

Quelque-soit la version , le programme principal du donné dans le TP1 doit fonctionner sans modification.
En rappel, voici le programme principal à mettre en œuvre :

```
int main()
{
  char c = 0;
  t_carnet carnet;
  carnet_creer(&carnet);

  while (c != 'q') {
    printf("(a)jouter (r)echercher (s)upprimer (l)ister (q)uitter\n");
    scanf("%c", &c);
    switch (c) {
      case 'a' : {
           t_rendez_vous nouv;
           rendez_vous_creer(&nouv);
           printf("nom, num.tel. lieu et date : ");
           rendez_vous_saisir(&nouv);
           if (!carnet_ajouter_rendez_vous(&carnet, &nouv)) {
                   fprintf(stderr, "Carnet plein\n");
           }
           rendez_vous_detruire(&nouv);
           break;
      }
      case 's' : {
            char bufdate[1024];
            char bufnom[1024];
            printf("nom et date : ");
            scanf("%s %s", bufnom, bufdate);
            carnet_retirer_rendez_vous(&carnet, bufnom, bufdate);
            break;
      }
      case 'l' : {
            carnet_afficher(&carnet);
```

```
            break;
        }
        case 'r' : {
            char bufdate[1024];
            t_carnet resultat;
            carnet_creer(&resultat);
            printf("date : ");
            scanf("%s", bufdate);
            carnet_rechercher_rendez_vous(&carnet, bufdate, &resultat);
            carnet_afficher(&resultat);
            carnet_detruire(&resultat);
            break;
        }
      }
    }
    carnet_detruire(&carnet);
    return EXIT_SUCCESS;
}
```