Introduction à la programmation-S6-TP2

Jasmine Chaid Akacem-Nicolas Morvan

Première étape

Nous avons défini notre structure contact

```c
struct contact{
    char nom[100];
    char prenom[100];
    char tel[100];
    };
```

Nous avons laissé 100 places disponibles pour chaque champ, ce qui est un peu excessif dans cet exercice, surtout pour le numéro de téléphone qui, sauf cas particulier, sera toujours de la même longueur à chaque fois.

Seconde étape

Fonction lecture_contacts

```c
int lecture_contacts(char* nom_fichier,struct contact tabc[1000]){
   FILE* fichier;
   char *champ, *l, ligne[100];
   int i=0,a=0;
   fichier=fopen(nom_fichier,"r");
   while( fgets(ligne, sizeof ligne, fichier)!=NULL){
      i=i+1;
      l=strdup(ligne);
      l[strlen(l)-1]=0;
      while((champ=strsep(&l,":"))!=NULL){
            a=a+1;
            if(a==1){
               strcpy(tabc[i].nom,champ);}
            else if(a==2){
               strcpy(tabc[i].prenom,champ);}
            else{
               strcpy(tabc[i].tel,champ);}
      }
      a=0;
      }
   return i;
   }
```

La fonction lecture_contacts prends en argument le nom du fichier, et un tableau de 1000 lignes de type contact.
Nous ouvrons le fichier en mode lecture, afin de pouvoir lire les lignes du fichier.

Avec une boucle while, nous prenons chaque ligne une par une dans le fichier et la stockons dans une variable nommée ligne, la condition étant « Tant que la ligne existe ». La variable i est incrémentée de 1 à chaque fois que la boucle tourne, ce qui permet de compter le nombre de ligne. Nous copions le contenu de la variable ligne dans une variable l.
Avec cette variable l et une autre variable, nommée champ, nous créons une nouvelle boucle while avec une condition assez complexe.
Strsep est une fonction qui permet de prendre l'adresse d'une variable et un caractère en argument, et elle retournera une chaîne de caractère qui partira du début de la chaîne initiale jusqu'au caractère rentré en argument, nommé séparateur. Puis cette fonction recommencera le travail jusqu'à ce que la chaîne initiale soit parcourue entièrement.
Le résultat de strsep est envoyé dans la variable champ. Ce que nous voulons, c'est que les différents champs de la ligne soient placés dans les bons champs de notre tableau structuré. Nous savons que la ligne est structurée de façon suivante : nom:prenom:tel. Avec une variable compteur a, incrémentée de 1 à chaque boucle, nous faisons des tests pour savoir si le champ est le premier de la ligne, le deuxième ou le troisième. Selon la valeur de a, nous copions la chaîne de champ à l'emplacement du tableau correspondant. Après cette seconde boucle while, a est remis à zéro.

Lorsque le fichier a été totalement parcouru, la première boucle while s'arrête et la fonction retourne la variable i, qui a compté toutes les lignes.

Troisième étape

Fonction affiche_contact

```
void affiche_contact(struct contact c){
   printf("%s, %s, %s\n",c.nom, c.prenom, c.tel);}
```

La fonction affiche_contact retourne un affichage, elle est donc de type void, et prends en argument une variable c de type contact.
Elle se contente d’afficher dans un ordre précis les chaînes contenus dans chaque champ.

Fonction affiche_liste_contact

```
void affiche_liste_contacts(struct contact l_c[1000], int n){
   int i;
   for(i=0;i<=n;i++){
      affiche_contact(l_c[i]);}
   }
```

La fonction affiche_liste_contacts prends en argument un tableau de type contact préalablement rempli, et sa taille n.
Avec une boucle For allant de 0 jusqu'à n, elle exécute la fonction affiche contact pour chaque ligne du tableau.

Quatrième étape

Fonction saisie_nouveau_contact

```
void saisie_nouveau_contact(struct contact l_c[1000],int k){
    char nom[100];
    char prenom[100];
    char tel[100];
    printf("Saisir nom,prénom et n°de téléphone:");
    scanf("%s",&nom);
    scanf("%s",&prenom);
    scanf("%s",&tel);
    strcpy(l_c[k].nom,nom);
    strcpy(l_c[k].prenom,prenom);
    strcpy(l_c[k].tel,tel);

}
```

La fonction saisie_nouveau_contact prends en argument un tableau de type contact préalablement rempli et un entier k.
Trois tableaux de caractères avec une taille fixe, nom, prenom et tel, sont initialisés, afin de recevoir les données entrées par l'utilisateur. Ensuite, la fonction copie une à une les chaînes dans les tableaux de caractères, à la « case » du tableau correspondant (nom, prenom ou tel) à la ligne correspondante, indiquée par l'entier k pris en argument.

Fonction ajout_contact_fichier

```
void ajout_contact_fichier(char* nom_fichier,struct contact c){
   FILE* fichier;
   char nom[100],prenom[100],tel[100];
   fichier=fopen(nom_fichier,"aw");
   fseek(fichier,0,SEEK_END);
   strcpy(nom,c.nom);
   strcpy(prenom,c.prenom);
   strcpy(tel,c.tel);
   fprintf(fichier,"\n%s:%s:%s",nom, prenom, tel);
   fclose(fichier);

}
```

La fonction ajout_contact_fichier prends en argument le nom d'un fichier, en chaîne de caractères, et une variable c de type contact.
Le fichier est ouvert en mode « aw », ce qui signifie « ouverture avec écriture à la fin du fichier ».
La fonction se positionne à la fin du fichier, pour se préparer à l'écriture des données.
Trois tableaux de caractères, nom, prenom et tel sont utilisés afin de copier le contenu des différents champs de la variable c dans ces chaînes. Ensuite, la fonction écris ces données dans le fichier, en structurant la ligne de façon suivant : nom:prenom:tel.
Enfin, le fichier est fermé.

Différences principales entre les objets Python et les structures en C

Lorsque l'on définit une structure en C, il est obligatoire de préciser tout les champs qui seront contenus dans cette structure, avant de les utiliser. En Python, il est possible de définir et créer des champs grâce aux objets créés avec cette classe, le tout au fur et à mesure des fonctions ou du programme principal.

De plus, en Python nous pouvons directement utiliser les champs, afin d'écrire dans un fichier par exemple, alors qu'en C, nous devons copier leur contenu dans une variable de type chaîne de caractères (un tableau contenant des caractères).

En Python nous pouvons également se servir directement de la variable structurée pour l'ajouter dans une liste, elle gardera sa structure, alors qu'en C, il faut copier un à un les champs dans les cases d'un tableau du type de la structure.