Compte rendu du TP 6 en programmation C

Thomas BARUZIER - 01/19/23

I. Présentation et explication du code

```
// Insertion d'un nom
int insererpers(Ttabpers Rep, int *der, Tchaine nom) {
    if (*der >= MAXPERS) return 0; // the list is full

    int i = *der + 1;
    while (i > 1 && strcmp(Rep[i-1], nom) > 0) {
        strcpy(Rep[i], Rep[i-1]);
        i--;
    }

    strcpy(Rep[i], nom);
    (*der)++;
    return i+1;
}
```

Cette fonction insère un nom dans l’annuaire, en maintenant l'ordre alphabétique. La fonction prend en entrée un pointeur vers un entier "der" qui représente la dernière position utilisée dans le tableau, ainsi qu'une chaîne de caractères "nom" qui est le nom à insérer. Si le tableau est plein, la fonction retourne 0. Sinon, elle utilise une boucle while pour trouver la bonne position pour insérer le nom, en comparant les noms dans le tableau avec celui à insérer. Une fois la position trouvée, elle utilise strcpy pour copier le nom à insérer dans cette position et incrémente la valeur de "der" pour indiquer qu'une nouvelle position est utilisée. Enfin, la fonction retourne l'indice de la position où le nom a été inséré.

```
// Affichage de l'annuaire
void afficherrep(Ttabpers Rep, int der) {
    for (int i = 1; i <= der; i++) {
        printf("%d. %s\n", i, Rep[i]);
    }
}

```

Cette fonction affiche le contenu de l’annuaire en utilisant un boucle for. La fonction prend en entrée le tableau "Rep" et un entier "der" qui représente la dernière position utilisée dans le tableau. La boucle for parcourt chaque position dans le tableau de 1 à "der", en utilisant l'indice "i" comme compteur. Pour chaque itération, la fonction utilise la fonction printf pour afficher l'indice (i) suivi du nom (Rep[i]) correspondant séparé par un point.

```
// Recherche d'un nom
int chercherpers(Ttabpers Rep, int der, Tchaine nom) {
    int debut = 1; //premier element de la liste
    int fin = der; // dernier element de la liste
    while (debut <= fin) {
        int milieu = (debut + fin) / 2;
        int comp = strcmp(Rep[milieu], nom);
        if (comp == 0) return milieu;
        else if (comp < 0) debut = milieu + 1;
```

```
        else fin = milieu - 1;
    }
    return 0;
}
```

Cette fonction recherche un nom dans l'annuaire en utilisant la méthode de recherche dichotomique. La fonction prend en entrée le tableau "Rep", un entier "der" qui représente la dernière position utilisée dans le tableau, et une chaîne de caractères "nom" qui est le nom à rechercher. Enfin, elle renvoie la position de l'élément si il a été trouvé, sinon 0.

```
// Supprimer un nom
int supprimerpers(Ttabpers Rep, int *der, int position) {
    if (position > *der + 1 || position < 1) return 0; // position non valide
    for (int i = position; i < *der + 1; i++) strcpy(Rep[i], Rep[i+1]);
    (*der)--;
    return position;
}
```

Cette fonction supprime un nom de l'annuaire à une position donnée. La fonction prend en entrée le tableau "Rep", un pointeur vers un entier "der", et un entier "position" qui représente la position du nom à supprimer. La fonction vérifie si la position donnée est valide en comparant avec la dernière position utilisée et retourne 0 si elle n'est pas valide. Sinon, elle utilise une boucle for pour parcourir les positions du tableau à partir de la position donnée jusqu'à la dernière position utilisée, et utilise strcpy pour copier le nom de la position suivante dans la position courante. Enfin, la fonction décrémente la valeur de "der" pour indiquer qu'une position a été supprimée et retourne la position donnée.

II. Jeux de tests utilisés

Chacune des fonctions a été testée avec ses limites et un élément aléatoire, ainsi que dans les cas où il y a 0 ou 1 élément.

- L'ajout de 1 nom ou plus a été vérifié en utilisant la fonction afficherrep, en veillant à ce que l'ordre alphabétique soit respecté.
- La fonction afficherrep a été testée tout au long des autres tests, ainsi qu'aux limites, au milieu avec 0, 1 élément(s) et plus.
- La fonction chercherpers a été testée pour filtrer les doublons lors de l'ajout de personnes, ainsi qu'aux limites, au milieu avec 0, 1 element(s) et plus.
- La fonction supprimerpers a été testée en utilisant l'affichage du répertoire, ainsi qu'aux limites, au milieu, avec 0, 1 élement(s) et plus.
    -

III. Logs des jeux de test

```
./annuaire.exe

 1-  Ajouter un nom de client
 2-  Rechercher la position d'un client
 3-  Supprimer un client de l'annuaire
 4-  Afficher l'annuaire
 5-  Quitter
  Votre choix :1

 Saisir un nom :test1

 Nom saisi : 'test1'
 1-  Ajouter un nom de client
 2-  Rechercher la position d'un client
 3-  Supprimer un client de l'annuaire
 4-  Afficher l'annuaire
 5-  Quitter
  Votre choix :4
1. test1

 1-  Ajouter un nom de client
 2-  Rechercher la position d'un client
 3-  Supprimer un client de l'annuaire
 4-  Afficher l'annuaire
 5-  Quitter
  Votre choix :1

 Saisir un nom :test2

 Nom saisi : 'test2'
 1-  Ajouter un nom de client
 2-  Rechercher la position d'un client
 3-  Supprimer un client de l'annuaire
 4-  Afficher l'annuaire
```

5- Quitter
Votre choix :1

Saisir un nom :test3

Nom saisi : 'test3'
1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :4
1. test1
2. test2
3. test3

1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :1

Saisir un nom :test11

Nom saisi : 'test11'
1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :1

Saisir un nom :test 0

Nom saisi : 'test'
1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :4
1. test
2. test1
3. test11
4. test2
5. test3

1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :1

Saisir un nom :test0

Nom saisi : 'test0'
1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire

4- Afficher l'annuaire
5- Quitter
Votre choix :4
1. test
2. test0
3. test1
4. test11
5. test2
6. test3

1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :1

Saisir un nom :abc

Nom saisi : 'abc'
1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :4
1. abc
2. test
3. test0
4. test1
5. test11
6. test2
7. test3

1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :2

Saisir un nom :abc

La position est 1 !!
1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :2

Saisir un nom :test

La position est 2 !!
1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :2

Saisir un nom :test11

La position est 5 !!
1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :2

Saisir un nom :test3

La position est 7 !!
1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :4
1. abc
2. test
3. test0
4. test1
5. test11
6. test2
7. test3

1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :3

Saisir un nom :abc

abc a ete supprimee de l'annuaire !!!

1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :4
1. test
2. test0
3. test1
4. test11
5. test2
6. test3

1- Ajouter un nom de client
2- Rechercher la position d'un client
3- Supprimer un client de l'annuaire
4- Afficher l'annuaire
5- Quitter
Votre choix :3

Saisir un nom :test3

test3 a ete supprimee de l'annuaire !!!

```
 1-  Ajouter un nom de client
 2-  Rechercher la position d'un client
 3-  Supprimer un client de l'annuaire
 4-  Afficher l'annuaire
 5-  Quitter
  Votre choix :4
1. test
2. test0
3. test1
4. test11
5. test2

 1-  Ajouter un nom de client
 2-  Rechercher la position d'un client
 3-  Supprimer un client de l'annuaire
 4-  Afficher l'annuaire
 5-  Quitter
  Votre choix :3

 Saisir un nom :test0

 test0 a ete supprimee de l'annuaire !!!

 1-  Ajouter un nom de client
 2-  Rechercher la position d'un client
 3-  Supprimer un client de l'annuaire
 4-  Afficher l'annuaire
 5-  Quitter
  Votre choix :4
1. test
2. test1
3. test11
4. test2

 1-  Ajouter un nom de client
 2-  Rechercher la position d'un client
 3-  Supprimer un client de l'annuaire
 4-  Afficher l'annuaire
 5-  Quitter
  Votre choix :5

 FIN !!!
```

IV. Test des performances

Reprise de la fonction de chronométrage réalisée en bonus au TP 2 pour cette partie.

Résultats

Sur RPI :
Ajout de 10000 éléments

Fait en 1180022 µs (1,18s)
Recherche de 10000 éléments
Fait en 5022 µs (5ms)
Suppression de 10000 éléments
Fait en 718190 µs (0,71s)

Sur Android :
Ajout de 10000 éléments
Fait en 214398 µs (0,21s)
Recherche de 10000 éléments
Fait en 2854 µs (3ms)
Suppression de 10000 éléments
Fait en 128899 µs (0,13s)

V. Conclusion

Au cours de ce TP en C, nous avons révisé une grande partie de nos connaissances en programmation C. Nous avons utilisé des fonctions, la recherche dichotomique, l'insertion, la manipulation de tableaux à une ou plusieurs dimensions, les makefile, la compilation séparée et le débogage pour créer un prototype fonctionnel d'un programme de gestion d'annuaire capable d'ajouter et supprimer des milliers de personnes en moins d'une seconde, même avec un processeur peu performant. Cependant, pour améliorer les performances, il serait possible d'optimiser le code en utilisant des structures de données plus avancées comme les arbres.