SYSTEME DE GESTION DE TRANSPORT SCOLAIRE

Langage C

Ce programme offre à l'utilisateur la possibilité de saisir un nouvel étudiant ainsi que la station où il descend. Il permet également d'afficher la liste de tous les étudiants ajoutés dans le système, de rechercher un étudiant pour vérifier s'il est présent dans la base de données, et d'agir en conséquence : l'ajouter s'il n'existe pas, ou le modifier ou le supprimer s'il est déjà enregistré. Toutes ces fonctionnalités ont été mises en œuvre grâce aux listes chaînées, qui ont rendu ce projet possible.

[Idrissi Kandri Badreddine]

15/12/2024

Système de gestion de transport scolaire

Les fonctions utilisées :

Pour réaliser toutes ces fonctionnalités j'avais eu besoin de 12 fonctions que j'ai détaillé dans ce rapport de projet ;

```
1-clear_screen ()
```

En C, cette fonction permet d'effacer le contenu affiché dans la console afin de donner une apparence "propre" à l'écran. Elle est généralement implémentée en utilisant la commande système spécifique à l'environnement d'exécution.

```
2-aloc_node ()
```

Cette fonction alloue de la mémoire pour un nouveau nœud et enregistre les informations obligatoires saisies par l'utilisateur (**Name** et **Bus**) à l'aide d'un tableau de type char de taille 20. La fonction strcpy () est utilisée pour copier la valeur du tableau dans les champs Name et bus du nœud. Enfin, elle initialise le pointeur **Next** à NULL et retourne le nouveau nœud.

```
3-addStudent ()
```

Cette fonction crée un pointeur **temp**, initialisé au nœud créé par la fonction alloc_node (), et effectue les calculs nécessaires pour ajouter un nouvel étudiant. Elle vérifie ensuite si l'ajout doit se faire au début ou à la fin de la liste, car les opérations diffèrent bien évidement selon le cas. Enfin, cette fonction retourne le pointeur temp, représentant le nouveau nœud prêt à être relié aux autres pour former une liste chaînée de données.

```
4-addstudent_intrv ()
```

Cette fonction est appelée lorsque l'utilisateur recherche un étudiant qui n'existe pas dans la base de données et décide de l'ajouter au système. Elle est similaire à la fonction précédente, à la différence qu'au lieu d'appeler la fonction alloc_node (), elle reçoit en entrée un pointeur contenant le nom ou le bus déjà saisis par l'utilisateur. Cela évite de lui redemander ces informations qu'il a déjà fournies.

```
5-displayallstudents ()
```

Cette fonction prend en entrée la tête de la liste et affiche tous les étudiants enregistrés dans le système. Elle affiche également, en bas, le nombre total d'étudiants.

```
6-modifier ()
```

Cette fonction est appelée lorsque l'utilisateur recherche un étudiant existant dans la base de données et décide de modifier ses informations. Elle prend en entrée le nœud à modifier. De manière similaire à alloc_node (), un tableau t de taille 20 est créé, et les mêmes calculs sont effectués pour mettre à jour les données du nœud.

7-supprimer ()

Cette fonction est appelée lorsque l'utilisateur souhaite supprimer définitivement un étudiant existant dans la base de données. Elle prend en entrée le pointeur **temp4** représentant le nœud à supprimer, le pointeur **preced** correspondant au nœud précédent, ainsi qu'une variable **cnt** utilisée auparavant pour vérifier si le nœud à supprimer se trouve en tête de liste ou ailleurs. La fonction effectue ensuite les calculs nécessaires pour supprimer le nœud temp4 tout en maintenant la liaison entre les autres nœuds de la liste. Enfin, elle libère l'espace mémoire alloué à temp4 et décrémente le nombre total d'étudiants de 1.

8-mdf_sup ()

Lorsque l'utilisateur recherche un étudiant existant dans la base de données, cette fonction affiche un menu permettant de choisir entre modifier ou supprimer l'étudiant. En fonction du choix de l'utilisateur, elle appelle soit la fonction modifier (), soit la fonction supprimer (). Pour localiser l'étudiant, la fonction compare la variable **to_find**, qui contient la donnée recherchée, avec les champs de tous les nœuds de la liste jusqu'à ce qu'une correspondance soit trouvée. Comme il n'est pas possible de comparer directement des types de données différents en C, la fonction strcmp () a été utilisée pour effectuer cette comparaison. La suppression d'un élément dans une liste simplement chaînée étant relativement complexe, une variable cnt a été introduite pour vérifier si le nœud à supprimer se trouve au début, au milieu ou à la fin de la liste. Cette variable permet également de conserver une référence au nœud précédent, essentielle pour exécuter correctement la fonction supprimer (). Enfin, grâce à la bibliothèque **<windows.h>**, les fonctions clear_screen () et Sleep () ont été utilisées pour offrir une expérience utilisateur plus fluide et agréable.

9-searchStudenT ()

Cette fonction est appelée lorsque l'utilisateur décide de rechercher un étudiant. Elle lui demande d'entrer soit le nom, soit le bus de l'étudiant, puis effectue les comparaisons nécessaires pour le trouver. Les variables **found_name** et **found_bus** sont utilisées pour indiquer, par un incrément (1 ou 0), si la donnée recherchée (nom ou bus) a été trouvée. En fonction de leur valeur, le programme détermine si l'étudiant existe ou non. Si l'étudiant n'est pas trouvé, il demande à l'utilisateur s'il souhaite l'ajouter. Si oui, il appelle la fonction addStudent_intrv (), sinon il retourne au menu. Si l'étudiant est trouvé, la fonction vérifie si la donnée correspond à un nom ou à un bus. Selon le cas, l'utilisateur reçoit un message affichant respectivement le bus de l'étudiant ou son nom, en utilisant un pointeur **info** qui stocke l'opposé du champ recherché (le nom si l'utilisateur a entré un bus, ou le bus si l'utilisateur a entré un nom). Enfin, si l'étudiant est trouvé, la fonction permet à l'utilisateur de modifier ou de supprimer cet étudiant en appelant la fonction mdf_sup (). Après cela, elle libère l'espace mémoire alloué pour les pointeurs info et to_find.

```
10-menu ()
```

Cette fonction est appelée après l'affichage du menu à l'utilisateur, et elle enregistre son choix dans la variable x. Ensuite, elle vérifie la valeur de x :

- Si x est égal à 1, elle appelle la fonction **addStudent ()** et incrémente le nombre d'étudiants de 1.
- Si x est égal à 2, elle appelle la fonction **displayAllStudents ()**.
- Si x est égal à 3, elle appelle la fonction **searchStudent ()**.
- Si x est égal à 0, elle affiche le message "**HAVE A NICE DAY !**" et termine le programme.

```
11-lib_esp ()
```

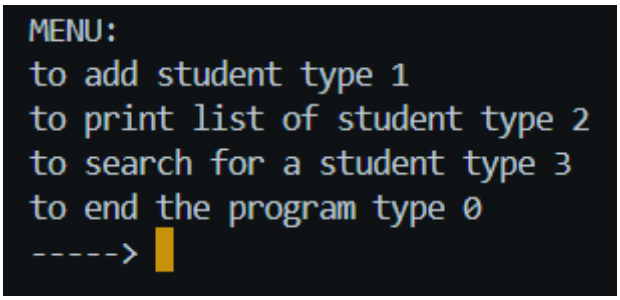
Comme il est essentiel de libérer l'espace mémoire alloué avec **malloc ()**, cette fonction est appelée à la fin du programme pour parcourir toute la liste et libérer l'espace alloué pour chaque nœud, un par un, en partant de la tête jusqu'à la queue de la liste. Elle vérifie si le nœud est égal à **NULL** à chaque étape afin d'éviter les fuites de mémoire.

```
12-print_menu ()
```

La fonction stocke le choix de l'utilisateur dans une variable **x**, puis le renvoie à la fonction **menu ()**. Ensuite, elle efface l'écran pour garantir une expérience utilisateur agréable. La fonction **menu ()** est ensuite appelée pour exécuter toutes les instructions précédentes. La fonction est intégrée dans une boucle **while** afin de permettre à l'utilisateur de réafficher le menu et de ne pas limiter son expérience à une seule instruction en vérifiant la valeur de la variable **x**. Si l'utilisateur entre un nombre qui ne figure pas dans le menu, le menu se relance, et s'il entre un caractère ou chaine de caractère un message d'erreur s'affiche et lui redemande d'entrer un nouveau un entrer valide.

Programme principale :

La fonction principale **main ()** affiche tout d'abord le menu en appelant la fonction print_main () **:**



Enfin, l'espace mémoire est libéré en appelant la fonction **lib_esp ()**.