*Mini Project Using Linked List*

**PROPERTY LIST MANAGER** – **Group 07**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

typedef struct property \*link;

struct property {

int propertyId;

int propertyType;

char propertyAddress[100];

char ownerName[50];

char ownerTelNo[20];

int bedrooms;

int bathrooms;

float houseSizeInSqft;

float priceOfProperty;

float noOfPerches;

float pricePerPerch;

float propertySizeInSqft;

int districtNo;

link next;

};

typedef struct{

link head, tail;

}list;

void initPropertyList(list \*l){

l->head = NULL;

l->tail = NULL;

}

link addNewProperty(){

link ptr = (link)malloc(sizeof(struct property));

printf("\n\t\tEnter property id : ");

scanf("%d",&ptr->propertyId);

printf("\n\t\tChoose district number of the property from below list.\n\n");

printf("\n\t\t1.Ampara \t\t2.Anuradhapura \t\t3.Badulla \t\t4.Batticaloa \t\t5.Colombo\n");

printf("\n\t\t6.Galle \t\t7.Gampaha \t\t8.Hambantota \t\t9.Jaffa \t\t10.Kalutara\n");

printf("\n\t\t11.Kandy \t\t12.Kegalle \t\t13.Kilinochchi\t\t14.Kurunegala \t\t15.Mannar\n");

printf("\n\t\t16.Matale \t\t17.Matara \t\t18.Monaragala \t\t19.Mullativu \t\t20.Nuwara-Eliya\n");

printf("\n\t\t21.Polonnaruwa\t\t22.Puttalam \t\t23.Ratnapura \t\t24.Trincomalee\t\t25.Vavuniya\n");

printf("\n\n\t\tEnter district number of the property: ");

scanf("%d",&ptr->districtNo);

fflush(stdin);

printf("\n\t\tEnter address of the property: ");

scanf("%[^\n]s",&ptr->propertyAddress);

fflush(stdin);

printf("\n\t\tEnter property owner's name: ");

scanf("%[^\n]s",&ptr->ownerName);

printf("\n\t\tEnter a contact number of the property owner: ");

scanf("%s",&ptr->ownerTelNo);

printf("\n\t\tChoose property type from below list.\n");

printf("\n\t\t\t1.Residence\n\t\t\t2.Land\n\t\t\t3.Commercial\n\t\t\t4.Industrial\n");

printf("\n\t\tEnter property type no: ");

scanf("%d",&ptr->propertyType);

if(ptr->propertyType == 1){

printf("\n\t\tEnter house size in squarefeet: ");

scanf("%f",&ptr->houseSizeInSqft);

printf("\n\t\tEnter number of perches: ");

scanf("%f",&ptr->noOfPerches);

printf("\n\t\tEnter price of the property: ");

scanf("%f",&ptr->priceOfProperty);

printf("\n\t\tEnter number of bedrooms: ");

scanf("%d",&ptr->bedrooms);

printf("\n\t\tEnter number of bathrooms: ");

scanf("%d",&ptr->bathrooms);

}else if(ptr->propertyType == 2){

printf("\n\t\tEnter number of perches: ");

scanf("%f",&ptr->noOfPerches);

printf("\n\t\tEnter price per perch: ");

scanf("%f",&ptr->pricePerPerch);

}else if(ptr->propertyType == 3){

printf("\n\t\tEnter property size in squarefeet: ");

scanf("%f",&ptr->propertySizeInSqft);

printf("\n\t\tEnter number of perches: ");

scanf("%f",&ptr->noOfPerches);

printf("\n\t\tEnter price of the property: ");

scanf("%f",&ptr->priceOfProperty);

}else if(ptr->propertyType == 4){

printf("\n\t\tEnter property size in squarefeet: ");

scanf("%f",&ptr->propertySizeInSqft);

printf("\n\t\tEnter price of the property: ");

scanf("%f",&ptr->priceOfProperty);

}else{

printf("\n\t\tInvalid property type!");

}

ptr->next = NULL;

return ptr;

}

void insertFront(list \*l,link ptr){

if(ptr->propertyType <= 4){

if(l->head == NULL){

l->head = l->tail = ptr;

printf("\n\t\t\"Property inserted at the front successfully!\"\n\n");

}else{

ptr->next = l->head;

l->head = ptr;

printf("\n\t\t\"Property inserted at the front successfully!\"\n\n");

}

}else{

printf("\n\t\t\"Property insertion failed!\"\n\n");

}

}

void insertRear(list \*l,link ptr){

if(ptr->propertyType <= 4){

if(l->head == NULL){

l->head = l->tail = ptr;

printf("\n\t\t\"Property inserted at the end successfully!\"\n\n");

}else{

l->tail->next = ptr;

l->tail = ptr;

printf("\n\t\t\"Property inserted at the end successfully!\"\n\n");

}

}else{

printf("\n\t\t\"Property insertion failed!\"\n\n");

}

}

void insertNext(list \*l, link ptr){

link temp;

int found = 0, position;

if(ptr->propertyType <= 4){

if(l->head == NULL){

l->head = l->tail = ptr;

}else{

temp = l->head;

printf("\n\t\tEnter the property id after which you want to insert the new property: ");

scanf("%d",&position);

while(temp != NULL && (!found)){

if(temp->propertyId == position){

found = 1;

}else{

temp = temp->next;

found = 0;

}

}

if(found){

ptr->next = temp->next;

temp->next = ptr;

printf("\n\t\t\"Property inserted at the required position successfully!\"\n");

}else{

ptr->next = l->head;

l->head = ptr;

printf("\n\t\t\"Given property id is not found!\"\n");

}

}

}else{

printf("\n\t\t\"Property insertion failed!\"\n");

}

}

void searchProperty(list \*l){

int position, found = 0;

link temp;

temp = l->head;

if(temp == NULL){

printf("\n\t\t\"Property List is empty!\"\n");

}else{

printf("\n\t\tEnter the property id which you want to search: ");

scanf("%d",&position);

while(temp != NULL && (!found)){

if(temp->propertyId == position){

found = 1;

}else{

temp = temp->next;

found = 0;

}

}

if(found){

printf("\n\t\t\"Property is found at the position %d.\"\n ",position);

}else{

printf("\n\t\t\"Property is not found!\"\n");

}

}

}

void deleteProperty(list \*l){

link ptr1, ptr2;

int found = 0, position;

ptr1 = l->head;

ptr2 = ptr1;

printf("\n\t\tEnter the property id which you want to delete from the list: ");

scanf("%d",&position);

while((!found) && (ptr1 != NULL)){

if(ptr1->propertyId == position){

found = 1;

}else{

ptr2 = ptr1;

ptr1 = ptr1->next;

}

}

if(found){

if(ptr2 == ptr1){

l->head = ptr2->next;

printf("\n\t\t\"Property is deleted successfully!\"\n");

}else if(ptr1 == l->tail){

l->tail = ptr2;

ptr2->next = NULL;

printf("\n\t\t\"Property is deleted successfully!\"\n");

}else{

ptr2->next = ptr1->next;

printf("\n\t\t\"Property is deleted successfully!\"\n");

}

}else{

printf("\n\t\t\"Property does not exists!\"\n");

}

}

void displayProperty(link ptr){

printf("\n\t\t\t\t``````````````````````````````````````````````\n");

if(ptr == NULL){

printf("\n\t\t\t\t\"Property doesn't exist!\"\n");

}else{

printf("\n");

printf("\t\t\t\tProperty Id = %d\n",ptr->propertyId);

printf("\t\t\t\tAddress of Property = %s\n",ptr->propertyAddress);

printf("\t\t\t\tProperty district id = %d\n",ptr->districtNo);

printf("\t\t\t\tOwner's Name = %s\n",ptr->ownerName);

printf("\t\t\t\tOwner Telephone No = %s\n",ptr->ownerTelNo);

printf("\t\t\t\tProperty Type = %d\n",ptr->propertyType);

if(ptr->propertyType == 1){

printf("\t\t\t\tHouse size in squarefeet = %.2f\n",ptr->houseSizeInSqft);

printf("\t\t\t\tNumber of perches = %.2f\n",ptr->noOfPerches);

printf("\t\t\t\tPrice of the property = %.2f\n",ptr->priceOfProperty);

printf("\t\t\t\tNumber of bedrooms = %d\n",ptr->bedrooms);

printf("\t\t\t\tNumber of bathrooms = %d\n",ptr->bathrooms);

}else if(ptr->propertyType == 2){

printf("\t\t\t\tNumber of perches = %.2f\n",ptr->noOfPerches);

printf("\t\t\t\tPrice per perch = %.2f\n",ptr->pricePerPerch);

}else if(ptr->propertyType == 3){

printf("\t\t\t\tProperty size in squarefeet = %.2f\n",ptr->propertySizeInSqft);

printf("\t\t\t\tNumber of perches = %.2f\n",ptr->noOfPerches);

printf("\t\t\t\tPrice of the property = %.2f\n",ptr->priceOfProperty);

}else{

printf("\t\t\t\tProperty size in squarefeet = %.2f\n",ptr->propertySizeInSqft);

printf("\t\t\t\tPrice of the property = %.2f\n",ptr->priceOfProperty);

}

}

}

void showProperties(list \*l){

link ptr;

ptr = l->head;

if(ptr == NULL){

printf("\n\t\t\t\t\"Property List is empty!\"\n");

}else{

printf("\n\t\t\t\t\"Show Property List!\"\n");

while(ptr!= NULL){

displayProperty(ptr);

ptr=ptr->next;

}

}

}

void filterPropertyByLocation(list \*l){

link ptr;

ptr = l->head;

int location;

printf("\n\t\t\t\tEnter the district id which you want to filter:");

scanf("%d",&location);

if(ptr == NULL){

printf("\n\t\t\t\t\"Property List is empty!\"\n");

}else{

while(ptr!= NULL){

if(ptr->districtNo == location){

displayProperty(ptr);

}

ptr = ptr->next;

}

}

}

void main (){

system("COLOR B0");

list \*l;

l = (list\*)malloc(sizeof(list));

initPropertyList(l);

int choice =0;

while(choice != 9){

printf("\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*WELCOME\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PROPERTY LIST MANAGER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n\t\t\tWhat do you want to do? Choose one option from the following list.\n");

printf("\t\t\t==================================================================\n");

printf("\n\t\t\t\t1.Insert Property To The Front\n\t\t\t\t2.Insert Property To The End\n\t\t\t\t3.Insert Property To Any Random Location\n"

"\t\t\t\t4.Search for a Property\n\t\t\t\t5.Delete Property\n\t\t\t\t6.Show all properties\n\t\t\t\t7.Filter Properties by desired location\n\t\t\t\t8.Exit\n");

printf("\n\t\tEnter your choice : ");

scanf("%d",&choice);

printf("\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

switch(choice)

{

case 1:

insertFront(l, addNewProperty());

break;

case 2:

insertRear(l,addNewProperty());

break;

case 3:

insertNext(l,addNewProperty());

break;

case 4:

searchProperty(l);

break;

case 5:

deleteProperty(l);

break;

case 6:

showProperties(l);

break;

case 7:

filterPropertyByLocation(l);

break;

case 8:

exit(0);

break;

default:

printf("\n\t\t\t\t\tPlease enter valid choice");

}

}

}

Group Members & Individual Contribution:

|  |  |  |
| --- | --- | --- |
| INDEX NO | NAME | CONTRIBUTION |
| 17APP3049 | A.A.R.D.PERERA | addNewProperty() , deleteProperty() , filterPropertyByLocation() , Finilazing Programming code, Presenting |
| 17APP3056 | M.L.M.DHARANI | insertNext(), displayProperty(),main(),Presentation Prepared, Presenting |
| 17APP3064 | W.A.D.H.SANDUNIKA | searchProperty(), showProperty(), Presentation Prepared |
| 17APP3275 | E.O.S.EDIRIWEERA | insertFront(), Finalized presentation Prepared |
| 17APP3058 | N.P.L.P.DHARMARATHNE | insertRear(), Presentation & word document prepared |