Ministerul Educaţiei, Culturii și Cercetării
Universitatea Tehnică a Moldovei

Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator nr. 4
la Structuri de date și algoritmi
Varianta 18

A efectuat:
st. gr. TI-206 Cătălin Pleșu
A verificat:
Lector universitar Vitalie Mititelu

Chișinău – 2021

Cuprins

Tema:

Algoritmi de prelucrare a listelor liniare simplu înlănțuite (listelor unidirecționale)

Scopul:

Obținerea deprinderilor practice de implementare și de utilizare a tipului abstract de date (TAD) „Listă simplu înlănțuită” în limbajul C cu asigurarea operațiilor de prelucrare de bază ale listei.

Sarcina:

Să se scrie 3 fișiere-text în limbajul C pentru implementarea și utilizarea TAD „Listă simplu înlănțuită” cu asigurarea operațiilor de prelucrare de bază ale listei:

1. Fișier antet cu extensia .h, care conține specificarea structurii de date a elementului listei simplu înlănțuite (conform variantelor) și prototipurile funcțiilor de prelucrare de bază ale listei.
2. Fișier cu extensia .c sau .cpp, care conține implementările (codurile) funcțiilor declarate în fișierul antet.
3. Fișier al utilizatorului, funcția main() pentru prelucrarea listei cu afișarea la ecran a următorului meniu de opțiuni de bază:
    1. Crearea listei în memoria dinamică
    2. Introducerea informației despre elementele listei de la tastatură.
    3. Afișarea informației despre elementele listei la ecran.
    4. Căutarea elementului în listă.
    5. Modificarea câmpurilor unui element din listă.
    6. Determinarea adresei ultimului element din listă.
    7. Determinarea lungimii listei (numărul de elemente).
    8. Interschimbarea a două elemente indicate în listă.
    9. Sortarea listei.
    10. Eliberarea memoriei alocate pentru listă.
    0. Ieșire din program.

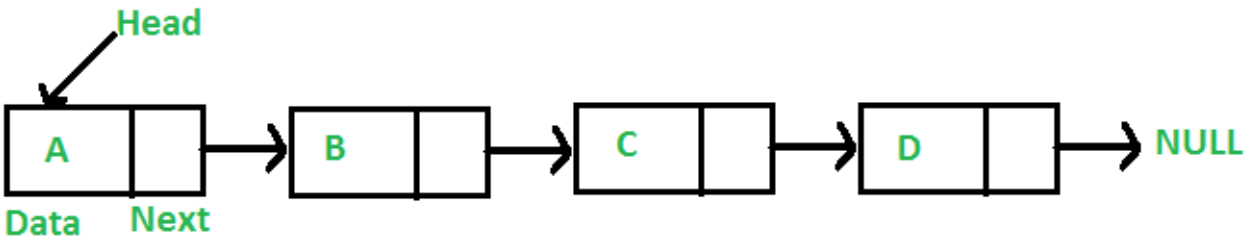
Varianta 18:

Structura Imobil cu câmpurile: proprietarul, tipul, adresa, suprafața, costul.

Rezumat la temă:

Liste liniare:

O listă liniara este o structură de date abstractă, în care elementele nu sunt stocate în locații de memorie continue. Elementele dintr-o listă liniară sunt conectate folosind pointeri, așa cum se arată în imaginea de mai jos:



În cuvinte simple, o listă legată este formată din noduri în care fiecare nod conține un câmp de date și o referință la următorul nod din listă.

Program format din multiple fișiere:

Pentru a compila un program ce conține mai multe fișiere am identificat 2 metode:

1. Includerea fișierului sursă în programul main ceea ce ar fi echivalentul a copierii cofului fin acest fi echivalentul a copierii codului din acest fișier sursă în fișierul ce conține funcția main.
2. Când compilăm utilizând linia de comandă să indicăm toate numele fișierelor .c. De asemenea fișierele .c pot fi compilate separat în fișiere .o apoi de compilat aceste fișiere întrun singur program. Aceasta metodă este convenientă când avem mai multe fișiere care nu sunt modificate deci nu este nevoie ca toate sursele să fie recompilate.

Make:

Am utilizat a doua metodă, iar pentru a fi mai simplu de aplicat am utilizat un fișier make care conține instrucțiunile pentru compilare:

```
gcc -o ./executable/catalin main.c ./sources/implementation.c ./sources/menus.c
```

errno:

Am utilizat libraria errno care detectează erorile apărute la unele operații cum ar fi deschiderea fișierelor sau
alocarea memoriei. Cu ajutorul funcției strerror afișez pe ecran ce înseamnă această eroare. Iar apoi ies din program cu codul de eroare respectiv.

Cod sursă:

./headers/const.h

```
#ifndef HERMINA
#define HERMINA
const char NAME[][30] = {"Catalin", "Marius", "Daniel", "Mirela", "Alex", "Colea", "Sandu", "Ion B",
"Maximo", "Melissa", "Petru", "Stas", "Vlad", "Crstian", "Ion T", "Mihail", "Victor", "Vladislav",
"Maria", "Vitalie", "Nicoleta", "Sam", "Nicu", "Viorel"};
const char TYPE[][30] = {"Apartament", "Birou", "Fabrica", "Magazin", "Mol", "Hotel", "Cladire
Istorica", "Teren gol", "Restaurant"};
const char ADDRESS[][30] = {"Strada Albisoara", "Strada Alexandru Bernardazzi", "Strada
Alexandru cel Bun", "Strada Alexei Mateevici", "Strada Armeneasca", "Strada Bucuresti", "Strada
Calea Iesilor", "Strada Mihail Kogalniceanu"};
const int NAME_COUNT = 24, TYPE_COUNT = 9, ADDRESS_COUNT = 8;
#endif
```

./headers/header.h

```
#ifndef SOBOLAN
#define SOBOLAN
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <errno.h>
typedef struct _imobil
{
// aranjate haotic ar ocupa mai multa memorie(
// structure padding
float suprafata;
float costul;
char *proprietar;
char *tip;
char *adresa;
// adaugat pentru lista liniara
struct _imobil *next;
} imobil;

// functii care returneaza pointeri la imobil
imobil *citeste_element(int ord);
imobil *creaza_lista();
imobil *creaza_lista_demo(int nr);
imobil *determina_ultimul_pointer(imobil *cap);
imobil *interschimba_2_elemente(imobil *cap, int limita);

// functii care nu returneaza nimic
void cauta_structura(imobil *cap);
```

```
void modifica_structura(imobil *cap);
void sorteaza_lista(imobil *cap);
void elibereaza_memoria_listei(imobil **cap);
// int return
int afiseaza_lista(imobil *cap);
int get_struct_info(imobil *struct_ptr);
// meniuri
int afiseaza_meniu();
int afiseaza_submeniu_criterii();
#endif // !SOBOLAN
```

./sources/menus.c

```
#include <stdio.h>
#include "./../headers/header.h"

int afiseaza_meniu()
{
printf("\n__Crearea listei in memoria dinamica__");
printf("\n1. Introducerea informatiei despre elementele listei de la tastatura");
printf("\n2. Genereaza o lista cu elemente aleatorii\n");
printf("\n3. Afisarea informatiei despre elementele listei la ecran.");
printf("\n4. Cautarea elementuluiin lista.");
printf("\n5. Modificarea campurilor unui element din lista.");
printf("\n6. Determinarea adresei ultimului element din lista.");
printf("\n7. Determinarea lungimii listei (numarul de elemente).");
printf("\n8. Interschimbareaa doua elemente indicate in lista.");
printf("\n9. Sortarea listei.");
printf("\n10. Eliberarea memoriei alocate pentru lista.");
printf("\n11. Calculeaza memoria ocupata.");
printf("\n0. Iesire din program");
printf("\nOptiunea - ");
int m;
scanf("%d", &m);
return m;
}

int afiseaza_submeniu_criterii()
{
printf("Dupa proprietar%4d\n", 1);
printf("Dupa tip%11d\n", 2);
printf("Dupa adresa%8d\n", 3);
printf("Dupa suprafata%5d\n", 4);
printf("Dupa cost%10d\n", 5);
printf("Anulare%12d\n", 0);
int optiune;
scanf("%d", &optiune);
return optiune;
```

```
}
```

./sources/implementation.c

```
#include "./../headers/header.h"
#include "./../headers/const.h"

int get_struct_info(imobil *struct_ptr)
{
long int total = sizeof(imobil);
if (struct_ptr)
{
if (struct_ptr->proprietar)
{
total += strlen(struct_ptr->proprietar);
}
if (struct_ptr->adresa)
{
total += strlen(struct_ptr->adresa);
}
if (struct_ptr->tip)
{
total += strlen(struct_ptr->tip);
}
}
return (int)total;
}

imobil *citeste_element(int ord)
{
errno = 0;
imobil *q = (imobil *)malloc(sizeof(imobil));
if (errno)
{
printf("exit code %d: %s\n", errno, strerror(errno));
exit(errno);
}
printf("Citirea elementului %d din lista.\n", ord);
q->next = NULL;

char str[255];
printf("proprietar: ");
scanf("%c", &str[0]); // curata bufferul
scanf("%[^\n]", str);
q->proprietar = strdup(str);
printf("tip: ");
scanf("%c", &str[0]); // curata bufferul
scanf("%[^\n]", str);
q->tip = strdup(str);
```

```
printf("adresa: ");
scanf("%c", &str[0]); // curata bufferul
scanf("%[^\n]", str);
q->adresa = strdup(str);
printf("suprafata : ");
scanf(" %f", &q->suprafata);
printf("costul : ");
scanf(" %f", &q->costul);

return q;
}

imobil *creaza_lista()
{
imobil *cap = NULL, *t = NULL;
errno = 0;
cap = (imobil *)malloc(sizeof(imobil));
if (errno)
{
printf("exit code %d: %s\n", errno, strerror(errno));
exit(errno);
}

int ord = 1;
cap = citeste_element(ord++);
t = cap;

char input = 'c';
printf("continua / termina [c/t]\n");
scanf(" %c", &input);
while (input != 't')
{
t->next = citeste_element(ord++);
t = t->next;
printf("continua / termina [c/t]\n");
scanf(" %c", &input);
};

return cap;
}

imobil *creaza_lista_demo(int nr)
{
srand(time(NULL));
imobil *cap = NULL,
*t = NULL;
errno = 0;
cap = (imobil *)malloc(sizeof(imobil));
if (errno)
```

```
{
printf("exit code %d: %s\n", errno, strerror(errno));
exit(errno);
}

errno = 0;
imobil *q = (imobil *)malloc(sizeof(imobil));
if (errno)
{
printf("exit code %d: %s\n", errno, strerror(errno));
exit(errno);
}

q->next = NULL;
q->proprietar = strdup(NAME[rand() % NAME_COUNT]);
q->tip = strdup(TYPE[rand() % TYPE_COUNT]);
q->adresa = strdup(ADDRESS[rand() % ADDRESS_COUNT]);
q->suprafata = ((float)(rand() % 300) + (float)11 / ((rand() % 9) + 1)) + 16;
q->costul = q->suprafata * (rand() % 1500);
cap = q;
t = cap;
for (int i = 1; i < nr; i++)
{
errno = 0;
q = (imobil *)malloc(sizeof(imobil));
if (errno)
{
printf("exit code %d: %s\n", errno, strerror(errno));
exit(errno);
}

q->next = NULL;
char str[255];

q->proprietar = strdup(NAME[rand() % NAME_COUNT]);
q->tip = strdup(TYPE[rand() % TYPE_COUNT]);
q->adresa = strdup(ADDRESS[rand() % ADDRESS_COUNT]);
q->suprafata = ((float)(rand() % 100) + (rand() % 100) + (rand() % 100) + (float)11 / ((rand() % 9) +
1)) + 16;
q->costul = q->suprafata * (rand() % 21) / ((rand() % 16) + 1) * 1000 + q->suprafata * (rand() %
500);
t->next = q;
t = t->next;
}
return cap;
}

int afiseaza_lista(imobil *cap)
{
```

```
if (cap == NULL)
{
printf("Lista nu este valida sau este NULL-a\n");
return 0;
}
imobil *t = cap;
printf("| Nr.| Proprietar|%15stip|%26sadresa| suprafata|%8scostul|%8s( POINTER | NEXT )\n",
"", "", "", "");
int i = 0;
while (t != NULL)
{

printf("|%-3d |%10s |%17s | %-30s |%9.2f m^2 |%12.2f $| ( %p | %p )\n", ++i, t->proprietar, t-
>tip, t->adresa, t->suprafata, t->costul, t, t->next);
t = t->next;
}
return i;
}

void cauta_structura(imobil *cap)
{
if (cap == NULL)
{
printf("lista este invalida\nincercati sa introduceti o lista inainte de a cauta ceva in ea\n");
return;
}
printf("\ncautarea in lista\n");
int optiune = afiseaza_submeniu_criterii();
if (!optiune)
return;
imobil *t;

char text[25];
if (optiune == 1)
strcpy(text, "proprietarul");
if (optiune == 2)
strcpy(text, "tipul");
if (optiune == 3)
strcpy(text, "adresa");
if (optiune == 4)
strcpy(text, "suprafata");
if (optiune == 5)
strcpy(text, "cost");
char str[50];
float min, max;
int gasit = 0, i = 1;
t = cap;
switch (optiune)
```

```
{
case 1:
case 2:
case 3:
printf("Introduceti %s imobilului cautat: ", text);
// gets(str);
// gets(str);
scanf("%c", &str[0]); // curata bufferul
scanf("%[^\n]", str);
printf("Lista imobilelor %s \"%s\":\n", text, str);
break;
case 4:
case 5:
printf("Introduceti intervalul de %s pentru imobil\n", text);
printf("%s minim: ", text);
scanf("%f", &min);
printf("%s maxim: ", text);
scanf("%f", &max);
if (min > max)
{
min += max;
max = min - max;
min = min - max;
}
break;
}

printf("| Nr.| Proprietar|%15stip|%26sadresa| suprafata|%8scostul|%8s( POINTER | NEXT )\n",
"", "", "", "");
while (t)
{
switch (optiune)
{
case 1:
if (!strcasecmp(t->proprietar, str))
{
printf("|%-3d |%10s |%17s | %-30s |%9.2f m^2 |%12.2f $| ( %p | %p )\n", i, t->proprietar, t->tip,
t->adresa, t->suprafata, t->costul, t, t->next);
gasit = 1;
}
break;
case 2:
if (!strcasecmp(t->tip, str))
{
printf("|%-3d |%10s |%17s | %-30s |%9.2f m^2 |%12.2f $| ( %p | %p )\n", i, t->proprietar, t->tip,
t->adresa, t->suprafata, t->costul, t, t->next);
gasit = 1;
}
```

```
break;
case 3:
if (!strcasecmp(t->adresa, str))
{
printf("|%-3d |%10s |%17s | %-30s |%9.2f m^2 |%12.2f $| ( %p | %p )\n", i, t->proprietar, t->tip,
t->adresa, t->suprafata, t->costul, t, t->next);
gasit = 1;
}
break;
case 4:
if (t->suprafata > min && t->suprafata < max)
{
printf("|%-3d |%10s |%17s | %-30s |%9.2f m^2 |%12.2f $| ( %p | %p )\n", i, t->proprietar, t->tip,
t->adresa, t->suprafata, t->costul, t, t->next);
gasit = 1;
}
break;
case 5:
if (t->costul > min && t->costul < max)
{
printf("|%-3d |%10s |%17s | %-30s |%9.2f m^2 |%12.2f $| ( %p | %p )\n", i, t->proprietar, t->tip,
t->adresa, t->suprafata, t->costul, t, t->next);
gasit = 1;
}
break;
}

t = t->next;
i++;
}
if (gasit == 0)
if (optiune == 1 || optiune == 2 || optiune == 3)
printf("Imobilele cu %s \"%s\" nu exista in lista data.\n", text, str);
if (optiune == 4 || optiune == 5)
printf("Imobilele cu %s cu valori pe intervalul %f - %f nu exista in lista data.\n", text, min, max);
}

void modifica_structura(imobil *cap)
{
if (cap == NULL)
{
printf("lista este invalida\nincercati sa introduceti o lista inainte de a modifica ceva\n");
return;
}
int id = 0;
printf("\nDati id-ul imobilului pe care doriti sa-l modificati: ");
scanf("%d", &id);
imobil *t = cap;
```

```
int i = 0, modificat = 0;
while (t)
{
i++;
if (i == id)
{
char str[50];
float val;
modificat = 1;
printf("daca nu doriti sa modificati un camp dati [SPACE]+[ENTER]\n");
printf("proprietarul vechi: %s\nproprietarul nou: ", t->proprietar);
scanf("%c", &str[0]); // curata bufferul
scanf("%[^\n]", str);
if (strcmp(str, " "))
{
t->proprietar = realloc(t->proprietar, strlen(str));
strcpy(t->proprietar, str);
}
else
printf("ramane nemodificat\n");
printf("tipul vechi: %s\ntipul nou: ", t->tip);
scanf("%c", &str[0]); // curata bufferul
scanf("%[^\n]", str);
if (strcmp(str, " "))
{
t->tip = realloc(t->tip, strlen(str));
strcpy(t->tip, str);
}
else
printf("ramane nemodificat\n");
printf("adresa veche: %s\nadresa noua: ", t->adresa);
scanf("%c", &str[0]); // curata bufferul
scanf("%[^\n]", str);
if (strcmp(str, " "))
{
t->adresa = realloc(t->adresa, strlen(str));
strcpy(t->adresa, str);
}
else
printf("ramane nemodificat\n");
printf("suprafata veche: %f\nsuprafata noua: ", t->suprafata);
scanf("%c", &str[0]); // curata bufferul
scanf("%[^\n]", str);
if (strcmp(str, " "))
{
t->suprafata = atof(str);
}
else
```

```
printf("ramane nemodificat\n");
printf("costul vechi: %f\ncostul nou: ", t->costul);
scanf("%c", &str[0]); // curata bufferul
scanf("%[^\n]", str);
if (strcmp(str, " "))
{
t->costul = atof(str);
}
else
printf("ramane nemodificat\n");
break;
}
t = t->next;
}
if (!modificat)
printf("Nu exista un imobil cu asa id deci nu poate fi modificat\n");
else
printf("|%-3d |%10s |%17s | %-30s |%9.2f m^2 |%12.2f $| ( %p | %p )\n", id, t->proprietar, t-
>tip, t->adresa, t->suprafata, t->costul, t, t->next);
}

imobil *determina_ultimul_pointer(imobil *cap)
{
if (cap == NULL)
{
printf("lista este invalida\nincercati sa introduceti o lista pentru a avea un ultim pointer\n");
return NULL;
}
imobil *t = cap;
while (t->next)
{
t = t->next;
}
printf("ultima adresa: %p\n", t);
return t;
}

imobil *interschimba_2_elemente(imobil *cap, int limita)
{
printf("__INTERSCHIMBAREA__\n");
if (cap == NULL)
{
printf("lista este invalida\nincercati sa introduceti o lista inainte de a modifica ceva\n");
return cap;
}
printf("dati id-urile a doua elemente pe care doriti sa le interschimbati: ");
int a, b;
scanf("%d %d", &a, &b);
```

```
if (a < 1 || a > limita || b < 1 || b > limita)
{
printf("ati iesit din limitele listei :/\n");
return cap;
}
if (a == b)
{
printf("numerele sunt egale, nu avem ce schimba\n");
return cap;
}
if (a > b)
{
int temp = b;
b = a;
a = temp;
}
imobil *t = cap, *a0 = NULL, *a1, *b0 = NULL, *b1;
int i = 0;
while (t)
{
i++;
if (i == a - 1)
a0 = t;
if (i == a)
a1 = t;
if (i == b - 1)
b0 = t;
if (i == b)
b1 = t;
t = t->next;
}
if (!a0)
{
if (a + 1 == b)
{
a1->next = b1->next;
b1->next = a1;
cap = b1;
}
else
{
imobil *t = b1->next;
b1->next = a1->next;
cap = b1;
a1->next = t;
b0->next = a1;
}
}
```

```
else
{
if (a + 1 == b)
{
a1->next = b1->next;
b1->next = a1;
a0->next = b1;
}
else
{
imobil *t = b1->next;
b1->next = a1->next;
a0->next = b1;
a1->next = t;
b0->next = a1;
}
}
return cap;
}

void sorteaza_lista(imobil *cap)
{
printf("__SORTARE__\n");
if (cap == NULL)
{
printf("lista este invalida\nincercati sa introduceti o lista inainte de a modifica ceva\n");
return;
}
int optiune = afiseaza_submeniu_criterii();
imobil *t = cap;
imobil *p, *q = (imobil *)malloc(sizeof(imobil));
int sortat; //bool
printf("1. sortarea crescator\n2.sortarea descrescator\n");
int s;
scanf("%d", &s);
do
{
sortat = 1;
p = cap;
int i = 0;

while (p->next)
{
i++;
if (s == 1)
{
if (optiune == 1 && strcmp(p->proprietar, p->next->proprietar) > 0 ||
optiune == 2 && strcmp(p->tip, p->next->tip) > 0 ||
```

```
optiune == 3 && strcmp(p->adresa, p->next->adresa) > 0 ||
optiune == 4 && p->suprafata > p->next->suprafata ||
optiune == 5 && p->costul > p->next->costul)
{
char str[255];
strcpy(str, p->next->proprietar);
p->next->proprietar = (char *)realloc(p->next->proprietar, strlen(p->proprietar) + 1);
strcpy(p->next->proprietar, p->proprietar);
p->proprietar = (char *)realloc(p->proprietar, strlen(str) + 1);
strcpy(p->proprietar, str);

strcpy(str, p->next->tip);
p->next->tip = (char *)realloc(p->next->tip, strlen(p->tip) + 1);
strcpy(p->next->tip, p->tip);
p->tip = (char *)realloc(p->tip, strlen(str) + 1);
strcpy(p->tip, str);

strcpy(str, p->next->adresa);
p->next->adresa = (char *)realloc(p->next->adresa, strlen(p->adresa) + 1);
strcpy(p->next->adresa, p->adresa);
p->adresa = (char *)realloc(p->adresa, strlen(str) + 1);
strcpy(p->adresa, str);

float num;
num = p->suprafata;
p->suprafata = p->next->suprafata;
p->next->suprafata = num;

num = p->costul;
p->costul = p->next->costul;
p->next->costul = num;

sortat = 0;
}
}
if (s == 2)
{
if (optiune == 1 && strcmp(p->proprietar, p->next->proprietar) < 0 ||
optiune == 2 && strcmp(p->tip, p->next->tip) < 0 ||
optiune == 3 && strcmp(p->adresa, p->next->adresa) < 0 ||
optiune == 4 && p->suprafata < p->next->suprafata ||
optiune == 5 && p->costul < p->next->costul)
{
char str[255];
strcpy(str, p->next->proprietar);
p->next->proprietar = (char *)realloc(p->next->proprietar, strlen(p->proprietar) + 1);
strcpy(p->next->proprietar, p->proprietar);
p->proprietar = (char *)realloc(p->proprietar, strlen(str) + 1);
strcpy(p->proprietar, str);
```

```
strcpy(str, p->next->tip);
p->next->tip = (char *)realloc(p->next->tip, strlen(p->tip) + 1);
strcpy(p->next->tip, p->tip);
p->tip = (char *)realloc(p->tip, strlen(str) + 1);
strcpy(p->tip, str);

strcpy(str, p->next->adresa);
p->next->adresa = (char *)realloc(p->next->adresa, strlen(p->adresa) + 1);
strcpy(p->next->adresa, p->adresa);
p->adresa = (char *)realloc(p->adresa, strlen(str) + 1);
strcpy(p->adresa, str);

float num;
num = p->suprafata;
p->suprafata = p->next->suprafata;
p->next->suprafata = num;

num = p->costul;
p->costul = p->next->costul;
p->next->costul = num;

sortat = 0;
}
}

p = p->next;
}

} while (sortat == 0);
}

void elibereaza_memoria_listei(imobil **cap)
{
while (*cap)
{
imobil *next = (*cap)->next;
free((*cap)->proprietar);
free((*cap)->tip);
free((*cap)->adresa);
free((*cap));
*cap = next;
}
}
```

./main.c

```
#include "./headers/header.h"

int main()
```

```
{
imobil *cap = NULL;
int nr = 0;
printf("__main_program__\n");

int meniu = 999;
do
{
meniu = afiseaza_meniu();
switch (meniu)
{
case 1:
{
elibereaza_memoria_listei(&cap);
cap = creaza_lista();
}
break;
case 2:
{
elibereaza_memoria_listei(&cap);
printf("dati numarul de elemente in lista: ");
int elemente;
scanf("%d", &elemente);
cap = creaza_lista_demo(elemente);
}
break;
case 3:
{
nr = afiseaza_lista(cap);
}
break;
case 4:
{
cauta_structura(cap);
}
break;
case 5:
{
modifica_structura(cap);
}
break;
case 6:
{
determina_ultimul_pointer(cap);
}
break;
case 7:
{
```

```
nr = afiseaza_lista(cap);
printf("numarul de elemente din lista de mai sus este: %d\n", nr);
}
break;
case 8:
{
nr = afiseaza_lista(cap);
cap = interschimba_2_elemente(cap, nr);
}
break;
case 9:
{
sorteaza_lista(cap);
}
break;
case 10:
{
elibereaza_memoria_listei(&cap);
}
break;
case 11:
{
imobil *temp = cap;
int B = 0;
while (temp)
{
B += get_struct_info(temp);
temp = temp->next;
}
int KB = B / 1024;
B -= KB * 1024;
int MB = KB / 1024;
KB -= MB * 1024;
printf("Memoria ocupata %d MB %d KB %d B\n", MB, KB, B);
}
break;
case 0:
{
exit(0);
}
break;
default:
break;
}
} while (meniu);
return 0;
}
```

Testarea programului:

Directoria in care lucrez:

```
catalin@catalin-ThinkPad-E595:~/UTM/SDA/ll/ll4$ tree
.
├── CMakeLists.txt
├── executable
├── exper
│   └── main x.c
├── headers
│   ├── const.h
│   └── header.h
├── Lab 4.pdf
├── Lucrarea de laborator nr 4 la SDA Plesu Catalin.odt
├── main.c
├── makefile
└── sources
    ├── implementation.c
    └── menus.c
```

Utilizarea make:

```
catalin@catalin-ThinkPad-E595:~/UTM/SDA/ll/ll4$ make
gcc -o ./executable/catalin main.c ./sources/implementation.c ./sources/menus.c
```

Meniul programului:

```
1. Introducerea informației despre elementele listei de la tastatură
2. Genereaza o lista cu elemente aleatorii

3.  Afișarea informației despre elementele listei la ecran.
4.  Căutarea elementuluiîn listă.
5.  Modificarea câmpurilor unui element din listă.
6.  Determinarea adresei ultimului element din listă.
7.  Determinarea lungimii listei (numărul de elemente).
8.  Interschimbareaa două elemente indicate în listă.
9.  Sortarea listei.
10. Eliberarea memoriei alocate pentru listă.
11. Calculeaza memoria ocupata.
0.  Iesire din program
```

Introducerea informației despre elementele listei de la tastatură:

```
Optiunea - 1
Citirea elementului 1 din lista.
proprietar: Catalin
tip: casa
adresa: Livezilor 12
suprafata : 70
costul : 10000
continua / termina [c/t]
c
Citirea elementului 2 din lista.
proprietar: Alex
tip: apartament
adresa: undeva in botanica
suprafata : 50
costul : 50000
continua / termina [c/t]
c
Citirea elementului 3 din lista.
proprietar: Nicu
tip: Casa
adresa: Hanaseni
suprafata : 1000
costul : 10000
continua / termina [c/t]
t
```

Afișarea informației despre elementele listei la ecran:

```
Optiunea - 3
| Nr.| Proprietar|              tip|                        adresa|     suprafata|        costul|
|1   |   Catalin |             casa | Livezilor 12                 |    70.00 m^2 |    10000.00 $|
|2   |      Alex |       apartament | undeva in botanica           |    50.00 m^2 |    50000.00 $|
|3   |      Nicu |             Casa | Hanaseni                     |  1000.00 m^2 |    10000.00 $|
```

Generarea unei liste mai mari utilizând opțiunea 2:

```
0.  Ieșire din program
Optiunea - 2
dati numarul de elemente in lista: 40
```

Nr.	Proprietar	tip	adresa	suprafata	costul
1	Stas	Birou	Strada Calea Iesilor	34.22 m^2	15400.00 $
2	Stas	Restaurant	Strada Alexandru cel Bun	141.57 m^2	120639.08 $
3	Colea	Fabrica	Strada Armeneasca	37.83 m^2	156024.67 $
4	Vladislav	Restaurant	Strada Bucuresti	210.67 m^2	259822.23 $
5	Ion T	Restaurant	Strada Calea Iesilor	190.00 m^2	224362.86 $
6	Vitalie	Birou	Strada Bucuresti	199.83 m^2	370557.59 $
7	Daniel	Fabrica	Strada Alexandru cel Bun	180.75 m^2	73048.83 $
8	Colea	Fabrica	Strada Albisoara	196.38 m^2	280096.22 $
9	Petru	Hotel	Strada Albisoara	220.00 m^2	358960.00 $
10	Petru	Hotel	Strada Alexei Mateevici	178.22 m^2	143231.28 $
11	Stas	Teren gol	Strada Bucuresti	80.22 m^2	251817.56 $
12	Vlad	Cladire Istorica	Strada Alexandru cel Bun	144.22 m^2	143089.06 $
13	Maria	Cladire Istorica	Strada Alexei Mateevici	171.57 m^2	342399.38 $
14	Marius	Apartament	Strada Alexandru Bernardazzi	128.57 m^2	100607.14 $
15	Maximo	Mol	Strada Bucuresti	107.57 m^2	436309.75 $
16	Vlad	Birou	Strada Armeneasca	213.20 m^2	77178.40 $
17	Sandu	Restaurant	Strada Bucuresti	171.00 m^2	458622.00 $
18	Vlad	Cladire Istorica	Strada Alexandru Bernardazzi	161.75 m^2	149457.00 $
19	Sam	Birou	Strada Armeneasca	162.75 m^2	117397.00 $
20	Mihail	Cladire Istorica	Strada Alexandru Bernardazzi	208.50 m^2	226848.00 $
21	Victor	Restaurant	Strada Alexandru Bernardazzi	115.67 m^2	393035.34 $
22	Maria	Apartament	Strada Armeneasca	202.83 m^2	82321.36 $

Căutarea elementului în listă:

```
cautarea in lista
Dupa proprietar   1
Dupa tip          2
Dupa adresa       3
Dupa suprafata    4
Dupa cost         5
Anulare           0
```

După proprietar „Viorel”:

```
1
Introduceti proprietarul imobilului cautat: viorel
Lista imobilelor proprietarul "viorel":
```

Nr.	Proprietar	tip	adresa	suprafata	costul	(POINTER	NEXT)
1	Viorel	Cladire Istorica	Strada Alexei Mateevici	313.00 m^2	79815.00 $	(0x56121c821af0	0x56121c821b80)
20	Viorel	Fabrica	Strada Albisoara	207.22 m^2	702068.94 $	(0x56121c822620	0x56121c8226b0)

După suprafață:

```
cautarea in lista
Dupa proprietar   1
Dupa tip          2
Dupa adresa       3
Dupa suprafata    4
Dupa cost         5
Anulare           0
4
Introduceti intervalul de suprafata pentru imobil
suprafata minim: 100
suprafata maxim: 125
```

Nr.	Proprietar	tip	adresa	suprafata	costul
10	Stas	Fabrica	Strada Alexei Mateevici	106.20 m^2	119085.60 $
22	Alex	Mol	Strada Alexei Mateevici	110.20 m^2	140044.16 $

Modificarea câmpurilor unui element din listă**:**

```
Dati id-ul imobilului pe care doriti sa-l modificati: 15
daca nu doriti sa modificati un camp dati [SPACE]+[ENTER]
proprietarul vechi: Mihail
proprietarul nou: Mihail Danilenco
tipul vechi: Birou
tipul nou: Cazino
adresa veche: Strada Calea Iesilor
adresa noua: Strada Calea Iesilor din Sangera
suprafata veche: 123.500000
suprafata noua: 500
costul vechi: 169236.171875
costul nou: 1234567
|15  |Mihail Danilenco |          Cazino | Strada Calea Iesilor din Sangera |   500.00 m^2 |  1234567.00 $| ( 0x56121c822340 | 0x56121c8223d0 )
```

Determinarea adresei ultimului element din lista:

```
Optiunea - 6
ultima adresa: 0x56121c8231b0
```

Acest lucru poate fi observat in tabelul afișat de opțiunea 3:

```
|35  |    Mihail |           Birou | Strada Armeneasca                |   224.38 m^2 |  1007443.75 $| ( 0x56121c822ee0 | 0x56121c822f70 )
|36  |   Vitalie |             Mol | Strada Bucuresti                 |   198.22 m^2 |   913209.75 $| ( 0x56121c822f70 | 0x56121c823000 )
|37  |     Petru |         Fabrica | Strada Calea Iesilor             |   208.57 m^2 |   294363.81 $| ( 0x56121c823000 | 0x56121c823090 )
|38  |    Mihail |       Teren gol | Strada Alexei Mateevici          |   150.22 m^2 |   720315.56 $| ( 0x56121c823090 | 0x56121c823120 )
|39  |   Crstian |           Hotel | Strada Calea Iesilor             |   244.57 m^2 |   696784.00 $| ( 0x56121c823120 | 0x56121c8231b0 )
|40  |    Mihail |           Birou | Strada Mihail Kogalniceanu       |   198.20 m^2 |  3378517.25 $| ( 0x56121c8231b0 | (nil) )
```

Determinarea lungimei listei:

```
|33  |   Melissa |      Restaurant | Strada Bucuresti                 |   223.67 m^2 |   228885.58 $| ( 0x56121c822db0 | 0x56121c822e40 )
|34  |     Ion B |         Magazin | Strada Alexandru cel Bun         |   265.22 m^2 |   176107.56 $| ( 0x56121c822e40 | 0x56121c822ee0 )
|35  |    Mihail |           Birou | Strada Armeneasca                |   224.38 m^2 |  1007443.75 $| ( 0x56121c822ee0 | 0x56121c822f70 )
|36  |   Vitalie |             Mol | Strada Bucuresti                 |   198.22 m^2 |   913209.75 $| ( 0x56121c822f70 | 0x56121c823000 )
|37  |     Petru |         Fabrica | Strada Calea Iesilor             |   208.57 m^2 |   294363.81 $| ( 0x56121c823000 | 0x56121c823090 )
|38  |    Mihail |       Teren gol | Strada Alexei Mateevici          |   150.22 m^2 |   720315.56 $| ( 0x56121c823090 | 0x56121c823120 )
|39  |   Crstian |           Hotel | Strada Calea Iesilor             |   244.57 m^2 |   696784.00 $| ( 0x56121c823120 | 0x56121c8231b0 )
|40  |    Mihail |           Birou | Strada Mihail Kogalniceanu       |   198.20 m^2 |  3378517.25 $| ( 0x56121c8231b0 | (nil) )
numarul de elemente din lista de mai sus este: 40
```

rezultatul obținut este 40.

Interschimbarea a două elemente indicate în listă:

```
| Nr.| Proprietar|              tip|                         adresa|     suprafata|        costul|        ( POINTER | NEXT )
|1   |     Viorel | Cladire Istorica | Strada Alexei Mateevici          |   313.00 m^2 |    79815.00 $| ( 0x56121c821af0 | 0x56121c821b80 )
|2   |     Marius |      Restaurant | Strada Mihail Kogalniceanu       |   130.75 m^2 |   241887.50 $| ( 0x56121c821b80 | 0x56121c821c20 )
|3   |     Mihail | Cladire Istorica | Strada Bucuresti                 |   206.38 m^2 |   218688.70 $| ( 0x56121c821c20 | 0x56121c821cb0 )
|4   |      Colea |       Apartament | Strada Alexei Mateevici          |   217.20 m^2 |    95640.40 $| ( 0x56121c821cb0 | 0x56121c821d40 )
|5   |       Alex |       Teren gol | Strada Alexandru cel Bun         |   141.38 m^2 |   418281.50 $| ( 0x56121c821d40 | 0x56121c821de0 )
|6   |     Daniel |             Mol | Strada Alexei Mateevici          |   234.50 m^2 |   177399.25 $| ( 0x56121c821de0 | 0x56121c821e70 )
|7   |  Vladislav |             Mol | Strada Mihail Kogalniceanu       |   186.83 m^2 |   307901.31 $| ( 0x56121c821e70 | 0x56121c821f10 )
|8   |     Mirela |       Apartament | Strada Alexandru cel Bun         |   272.00 m^2 |   522451.53 $| ( 0x56121c821f10 | 0x56121c821fb0 )
|9   |     Marius |           Birou | Strada Bucuresti                 |   100.20 m^2 |   779656.19 $| ( 0x56121c821fb0 | 0x56121c822040 )
|10  |      Colea | Cladire Istorica | Strada Alexandru cel Bun         |   137.67 m^2 |    59885.00 $| ( 0x56121c822040 | 0x56121c8220e0 )
|11  |      Maria |         Magazin | Strada Alexandru Bernardazzi     |   257.50 m^2 |   228917.50 $| ( 0x56121c8220e0 | 0x56121c822180 )
|12  |       Stas |       Apartament | Strada Alexei Mateevici          |   116.22 m^2 |   166048.36 $| ( 0x56121c822180 | 0x56121c822210 )
|13  |       Stas |       Teren gol | Strada Alexandru Bernardazzi     |   261.83 m^2 |   724642.44 $| ( 0x56121c822210 | 0x56121c8222b0 )
|14  |   Nicoleta |         Magazin | Strada Bucuresti                 |    87.50 m^2 |   142712.50 $| ( 0x56121c8222b0 | 0x56121c822340 )
|15  |Mihail Danilenco |          Cazino | Strada Calea Iesilor din Sangera |   500.00 m^2 |  1234567.00 $| ( 0x56121c822340 | 0x56121c
|16  |       Vlad |         Fabrica | Strada Albisoara                 |   134.67 m^2 |   572333.31 $| ( 0x56121c8223d0 | 0x56121c822460 )
|17  |  Vladislav |           Hotel | Strada Bucuresti                 |   169.57 m^2 |    44446.55 $| ( 0x56121c822460 | 0x56121c8224f0 )
|18  |     Daniel |           Hotel | Strada Mihail Kogalniceanu       |    94.67 m^2 |   300566.66 $| ( 0x56121c8224f0 | 0x56121c822590 )
|19  |       Alex |           Birou | Strada Bucuresti                 |    78.50 m^2 |    93975.71 $| ( 0x56121c822590 | 0x56121c822620 )
|20  |     Viorel |         Fabrica | Strada Albisoara                 |   207.22 m^2 |   702068.94 $| ( 0x56121c822620 | 0x56121c8226b0 )
|21  |    Catalin |             Mol | Strada Bucuresti                 |   185.67 m^2 |   103416.34 $| ( 0x56121c8226b0 | 0x56121c822740 )
|22  |  Vladislav |       Teren gol | Strada Alexandru Bernardazzi     |   140.83 m^2 |   194913.34 $| ( 0x56121c822740 | 0x56121c8227e0 )
|23  |        Sam |       Teren gol | Strada Alexei Mateevici          |   224.57 m^2 |   180882.08 $| ( 0x56121c8227e0 | 0x56121c822870 )
|24  |     Daniel | Cladire Istorica | Strada Alexandru Bernardazzi     |   124.22 m^2 |  2417861.25 $| ( 0x56121c822870 | 0x56121c822910 )
|25  |      Maria |         Fabrica | Strada Calea Iesilor             |   151.20 m^2 |   358545.62 $| ( 0x56121c822910 | 0x56121c8229a0 )
|26  |      Petru | Cladire Istorica | Strada Alexandru cel Bun         |    51.75 m^2 |   134239.50 $| ( 0x56121c8229a0 | 0x56121c822a40 )
|27  |      Petru |      Restaurant | Strada Bucuresti                 |   178.67 m^2 |  3615320.25 $| ( 0x56121c822a40 | 0x56121c822ad0 )
|28  |     Ion T |             Mol | Strada Bucuresti                 |   122.67 m^2 |   141472.41 $| ( 0x56121c822ad0 | 0x56121c822b60 )
|29  |    Catalin |      Restaurant | Strada Bucuresti                 |   241.75 m^2 |   153994.75 $| ( 0x56121c822b60 | 0x56121c822bf0 )
|30  |     Ion T |      Restaurant | Strada Alexandru cel Bun         |   157.75 m^2 |   413620.50 $| ( 0x56121c822bf0 | 0x56121c822c90 )
|31  |       Stas |         Fabrica | Strada Armeneasca                |   230.50 m^2 |   238798.00 $| ( 0x56121c822c90 | 0x56121c822d20 )
|32  |   Crstian |      Restaurant | Strada Albisoara                 |   178.38 m^2 |   176056.12 $| ( 0x56121c822d20 | 0x56121c822db0 )
|33  |   Melissa |      Restaurant | Strada Bucuresti                 |   223.67 m^2 |   228885.58 $| ( 0x56121c822db0 | 0x56121c822e40 )
|34  |     Ion B |         Magazin | Strada Alexandru cel Bun         |   265.22 m^2 |   176107.56 $| ( 0x56121c822e40 | 0x56121c822ee0 )
|35  |    Mihail |           Birou | Strada Armeneasca                |   224.38 m^2 |  1007443.75 $| ( 0x56121c822ee0 | 0x56121c822f70 )
|36  |   Vitalie |             Mol | Strada Bucuresti                 |   198.22 m^2 |   913209.75 $| ( 0x56121c822f70 | 0x56121c823000 )
|37  |     Petru |         Fabrica | Strada Calea Iesilor             |   208.57 m^2 |   294363.81 $| ( 0x56121c823000 | 0x56121c823090 )
|38  |    Mihail |       Teren gol | Strada Alexei Mateevici          |   150.22 m^2 |   720315.56 $| ( 0x56121c823090 | 0x56121c823120 )
|39  |   Crstian |           Hotel | Strada Calea Iesilor             |   244.57 m^2 |   696784.00 $| ( 0x56121c823120 | 0x56121c8231b0 )
|40  |    Mihail |           Birou | Strada Mihail Kogalniceanu       |   198.20 m^2 |  3378517.25 $| ( 0x56121c8231b0 | (nil) )
__INTERSCHIMBAREA__
dati id-urile a doua elemente pe care doriti sa le interschimbati: 21 1
```

Rezultat:

Nr.	Proprietar	tip	adresa	suprafata	costul	(POINTER	NEXT)
1	Catalin	Mol	Strada Bucuresti	185.67 m^2	103416.34 $	(0x56121c8226b0	0x56121c821b80)
2	Marius	Restaurant	Strada Mihail Kogalniceanu	130.75 m^2	241887.50 $	(0x56121c821b80	0x56121c821c20)
3	Mihail	Cladire Istorica	Strada Bucuresti	206.38 m^2	218688.70 $	(0x56121c821c20	0x56121c821cb0)
4	Colea	Apartament	Strada Alexei Mateevici	217.20 m^2	95640.40 $	(0x56121c821cb0	0x56121c821d40)
5	Alex	Teren gol	Strada Alexandru cel Bun	141.38 m^2	418281.50 $	(0x56121c821d40	0x56121c821de0)
6	Daniel	Mol	Strada Alexei Mateevici	234.50 m^2	177399.25 $	(0x56121c821de0	0x56121c821e70)
7	Vladislav	Mol	Strada Mihail Kogalniceanu	186.83 m^2	307901.31 $	(0x56121c821e70	0x56121c821f10)
8	Mirela	Apartament	Strada Alexandru cel Bun	272.00 m^2	522451.53 $	(0x56121c821f10	0x56121c821fb0)
9	Marius	Birou	Strada Bucuresti	100.20 m^2	779656.19 $	(0x56121c821fb0	0x56121c822040)
10	Colea	Cladire Istorica	Strada Alexandru cel Bun	137.67 m^2	59885.00 $	(0x56121c822040	0x56121c8220e0)
11	Maria	Magazin	Strada Alexandru Bernardazzi	257.50 m^2	228917.50 $	(0x56121c8220e0	0x56121c822180)
12	Stas	Apartament	Strada Alexei Mateevici	116.22 m^2	166048.36 $	(0x56121c822180	0x56121c822210)
13	Stas	Teren gol	Strada Alexandru Bernardazzi	261.83 m^2	724642.44 $	(0x56121c822210	0x56121c8222b0)
14	Nicoleta	Magazin	Strada Bucuresti	87.50 m^2	142712.50 $	(0x56121c8222b0	0x56121c822340)
15	Mihail Danilenco	Cazino	Strada Calea Iesilor din Sangera	500.00 m^2	1234567.00 $	(0x56121c822340	0x56121c8
16	Vlad	Fabrica	Strada Albisoara	134.67 m^2	572333.31 $	(0x56121c8223d0	0x56121c822460)
17	Vladislav	Hotel	Strada Bucuresti	169.57 m^2	44446.55 $	(0x56121c822460	0x56121c8224f0)
18	Daniel	Hotel	Strada Mihail Kogalniceanu	94.67 m^2	300566.66 $	(0x56121c8224f0	0x56121c822590)
19	Alex	Birou	Strada Bucuresti	78.50 m^2	93975.71 $	(0x56121c822590	0x56121c822620)
20	Viorel	Fabrica	Strada Albisoara	207.22 m^2	702068.94 $	(0x56121c822620	0x56121c821af0)
21	Viorel	Cladire Istorica	Strada Alexei Mateevici	313.00 m^2	79815.00 $	(0x56121c821af0	0x56121c822740)

Sortarea listei**:**

```
Dupa proprietar   1
Dupa tip          2
Dupa adresa       3
Dupa suprafata    4
Dupa cost         5
Anulare           0
1
1. sortarea crescator
2.sortarea descrescator
1
```

Nr.	Proprietar	tip	adresa	suprafata	costul	(POINTER	NEXT)
1	Alex	Teren gol	Strada Alexandru cel Bun	141.38 m^2	418281.50 $	(0x56121c8226b0	0x56121c821b80)
2	Alex	Birou	Strada Bucuresti	78.50 m^2	93975.71 $	(0x56121c821b80	0x56121c821c20)
3	Catalin	Mol	Strada Bucuresti	185.67 m^2	103416.34 $	(0x56121c821c20	0x56121c821cb0)
4	Catalin	Restaurant	Strada Bucuresti	241.75 m^2	153994.75 $	(0x56121c821cb0	0x56121c821d40)
5	Colea	Apartament	Strada Alexei Mateevici	217.20 m^2	95640.40 $	(0x56121c821d40	0x56121c821de0)
6	Colea	Cladire Istorica	Strada Alexandru cel Bun	137.67 m^2	59885.00 $	(0x56121c821de0	0x56121c821e70)
7	Crstian	Restaurant	Strada Albisoara	178.38 m^2	176056.12 $	(0x56121c821e70	0x56121c821f10)
8	Crstian	Hotel	Strada Calea Iesilor	244.57 m^2	696784.00 $	(0x56121c821f10	0x56121c821fb0)
9	Daniel	Mol	Strada Alexei Mateevici	234.50 m^2	177399.25 $	(0x56121c821fb0	0x56121c822040)
10	Daniel	Hotel	Strada Mihail Kogalniceanu	94.67 m^2	300566.66 $	(0x56121c822040	0x56121c8220e0)
11	Daniel	Cladire Istorica	Strada Alexandru Bernardazzi	124.22 m^2	2417861.25 $	(0x56121c8220e0	0x56121c822180)
12	Ion B	Magazin	Strada Alexandru cel Bun	265.22 m^2	176107.56 $	(0x56121c822180	0x56121c822210)
13	Ion T	Mol	Strada Bucuresti	122.67 m^2	141472.41 $	(0x56121c822210	0x56121c8222b0)
14	Ion T	Restaurant	Strada Alexandru cel Bun	157.75 m^2	413620.50 $	(0x56121c8222b0	0x56121c822340)
15	Maria	Magazin	Strada Alexandru Bernardazzi	257.50 m^2	228917.50 $	(0x56121c822340	0x56121c8223d0)
16	Maria	Fabrica	Strada Calea Iesilor	151.20 m^2	358545.62 $	(0x56121c8223d0	0x56121c822460)
17	Marius	Restaurant	Strada Mihail Kogalniceanu	130.75 m^2	241887.50 $	(0x56121c822460	0x56121c8224f0)
18	Marius	Birou	Strada Bucuresti	100.20 m^2	779656.19 $	(0x56121c8224f0	0x56121c822590)
19	Melissa	Restaurant	Strada Bucuresti	223.67 m^2	228885.58 $	(0x56121c822590	0x56121c822620)
20	Mihail	Cladire Istorica	Strada Bucuresti	206.38 m^2	218688.70 $	(0x56121c822620	0x56121c821af0)
21	Mihail	Birou	Strada Armeneasca	224.38 m^2	1007443.75 $	(0x56121c821af0	0x56121c822740)
22	Mihail	Teren gol	Strada Alexei Mateevici	150.22 m^2	720315.56 $	(0x56121c822740	0x56121c8227e0)
23	Mihail	Birou	Strada Mihail Kogalniceanu	198.20 m^2	3378517.25 $	(0x56121c8227e0	0x56121c822870)
24	Mihail Danilenco	Cazino	Strada Calea Iesilor din Sangera	500.00 m^2	1234567.00 $	(0x56121c822870	0x56121c
25	Mirela	Apartament	Strada Alexandru cel Bun	272.00 m^2	522451.53 $	(0x56121c822910	0x56121c8229a0)

Sortarea după suprafață descrescătoare:

Nr.	Proprietar	tip	adresa	suprafata	costul	(POINTER | NEXT)
1	Mihail Danilenco	Cazino	Strada Calea Iesilor din Sangera	500.00 m^2	1234567.00 $	(0x56121c8226b0	0x56121c
2	Viorel	Cladire Istorica	Strada Alexei Mateevici	313.00 m^2	79815.00 $	(0x56121c821b80	0x56121c821c20
3	Mirela	Apartament	Strada Alexandru cel Bun	272.00 m^2	522451.53 $	(0x56121c821c20	0x56121c821cb0
4	Ion B	Magazin	Strada Alexandru cel Bun	265.22 m^2	176107.56 $	(0x56121c821cb0	0x56121c821d40
5	Stas	Teren gol	Strada Alexandru Bernardazzi	261.83 m^2	724642.44 $	(0x56121c821d40	0x56121c821de0
6	Maria	Magazin	Strada Alexandru Bernardazzi	257.50 m^2	228917.50 $	(0x56121c821de0	0x56121c821e70
7	Crstian	Hotel	Strada Calea Iesilor	244.57 m^2	696784.00 $	(0x56121c821e70	0x56121c821f10
8	Catalin	Restaurant	Strada Bucuresti	241.75 m^2	153994.75 $	(0x56121c821f10	0x56121c821fb0
9	Daniel	Mol	Strada Alexei Mateevici	234.50 m^2	177399.25 $	(0x56121c821fb0	0x56121c822040
10	Stas	Fabrica	Strada Armeneasca	230.50 m^2	238798.00 $	(0x56121c822040	0x56121c8220e0
11	Sam	Teren gol	Strada Alexei Mateevici	224.57 m^2	180882.08 $	(0x56121c8220e0	0x56121c822180
12	Mihail	Birou	Strada Armeneasca	224.38 m^2	1007443.75 $	(0x56121c822180	0x56121c822210
13	Melissa	Restaurant	Strada Bucuresti	223.67 m^2	228885.58 $	(0x56121c822210	0x56121c8222b0
14	Colea	Apartament	Strada Alexei Mateevici	217.20 m^2	95640.40 $	(0x56121c8222b0	0x56121c822340
15	Petru	Fabrica	Strada Calea Iesilor	208.57 m^2	294363.81 $	(0x56121c822340	0x56121c8223d0
16	Viorel	Fabrica	Strada Albisoara	207.22 m^2	702068.94 $	(0x56121c8223d0	0x56121c822460
17	Mihail	Cladire Istorica	Strada Bucuresti	206.38 m^2	218688.70 $	(0x56121c822460	0x56121c8224f0
18	Vitalie	Mol	Strada Bucuresti	198.22 m^2	913209.75 $	(0x56121c8224f0	0x56121c822590
19	Mihail	Birou	Strada Mihail Kogalniceanu	198.20 m^2	3378517.25 $	(0x56121c822590	0x56121c822620
20	Vladislav	Mol	Strada Mihail Kogalniceanu	186.83 m^2	307901.31 $	(0x56121c822620	0x56121c821af0
21	Catalin	Mol	Strada Bucuresti	185.67 m^2	103416.34 $	(0x56121c821af0	0x56121c822740
22	Petru	Restaurant	Strada Bucuresti	178.67 m^2	3615320.25 $	(0x56121c822740	0x56121c8227e0
23	Crstian	Restaurant	Strada Albisoara	178.38 m^2	176056.12 $	(0x56121c8227e0	0x56121c822870
24	Vladislav	Hotel	Strada Bucuresti	169.57 m^2	44446.55 $	(0x56121c822870	0x56121c822910
25	Ion T	Restaurant	Strada Alexandru cel Bun	157.75 m^2	413620.50 $	(0x56121c822910	0x56121c8229a0
26	Maria	Fabrica	Strada Calea Iesilor	151.20 m^2	358545.62 $	(0x56121c8229a0	0x56121c822a40

Memoria ocupată de această listă de 40 elemente:

```
Optiunea - 11
Memoria ocupata 0 MB 2 KB 996 B
```

Eliberarae memoriei:

```
Optiunea - 3
Lista nu este valida sau este NULL-a
```

Concluzii :

1. Programele la SDA devin din ce în ce mia interesante.
2. Am obținut deprinderile practice de a utiliza TAD „listă simplu înlănțuită”.
3. Am aflat cum pot separa un program în mai multe fișiere.
4. Am obșinut deprindera de a utiliza un instrument de automatizare a compilării (make).
5. Am fost în stare să scriu un cod mai succint decât m-aș fi așteptat.
6. M-am pregătit să lucrez și cu alte TAD.
7. Am utilizat errno ceea ce pare o practica bună.
8. În loc de camelCase am utilizat “_” (underscore) ceea ce face citirea codului mai ușoară.