Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Departamentul Informatică și Ingineria Sistemelor

# RAPORT

Lucrarea de laborator nr2-3la Programarea Calculatoarelor (SDA)

Varianta 9

A efectuat: st. gr. CR-203 Onica Vitalie

A verificat: Stralutat Ștefan

Chişinău -2020

**Теmа:** Implementarea tipului de date abstract “Lista simplu inlanţuită” in limbajul C

**Scopul lucrării:** obţinerea deprinderilor practice de implementare și de utilizare a unui tip de date abstract (TDA) in limbajul C şi anume TDA “Lista simplu inlanţuită”.

**Sarcină** De scris trei fişiere in limbajul C pentru implementarea (2 fişiere) si utilizarea (1 fişier – program cu functia main) pentru TDA “Lista simplu inlanţuită”:

1. Fişierul antet cu extensia (h) care descrie structura de date a elementului listei simplu inlanţuite (conform variantei din lucrare precedentă) şi prototipurile funcţiilor care asigură operaţiunile de prelucrare a listei simplu inlanţuite.

2. Fişier cu extensia (cpp sau c) care conţine codurile (implementările) tuturor funcţiilor declarate în fişierul antet.

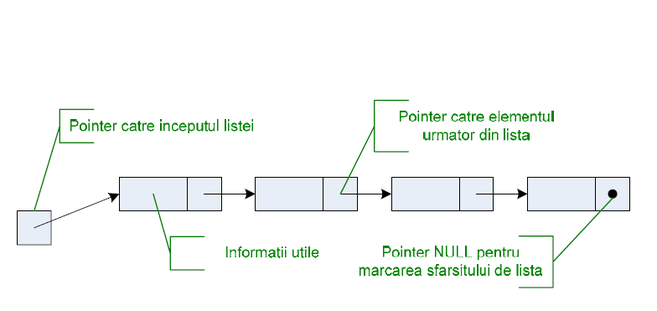
3. Fişierul utilizatorului - programul cu funcţia main () pentru prelucrarea listei simplu inlanţuite cu afişarea meniului de opţiuni pe ecran si anume:

**pentru lucrare de laborator 2** - crearea listei simplu inlanţuite cu n elemente în memoria dinamică, întroducerea informaţiei despre elementele listei de la tastatură, afişarea informaţiei despre elementele listei pe ecran (și a adreselor elementului current și următor), căutarea elementului listei dupa unui cîmp informaţional, modificarea cîmpurilor elementului listei, interschimbarea a 2 elemente listei, determinarea lungimei listei, sortarea elementelor listei după unui cimp, eliberarea memoriei dinamice, ieşire din program;

**pentru lucrare de laborator 3** - adaugarea unui element nou la sfirşitul listei, adaugarea unui element nou la începutul listei, inserarea unui element nou în lista după elementul indicat, inserarea unui element nou în lista înainte elementul indicat, stergerea unui element al listei, divizarea listei simplu inlanţuite in două liste separate, unirea a 2 liste simlu inlanţuite intr-o listă, scrierea informaţiei despre elementele listei în fişier, citirea informatiei despre elementele listei din fişier.

**Note teoretice**

Listele simplu inlantuite sunt structuri de date dinamice omogene. Spre deosebire de masive, listele nu sunt alocate ca blocuri omogene de memorie, ci ca elemente separate de memorie. Fiecare nod al listei contine, in afara ce informatia utila, adresa urmatorului element. Aceasta organizare permite numai acces secvential la elementele listei. Pentru accesarea listei trebuie cunoscuta adresa primului element (numita capul listei); elementele urmatoare sunt accesate parcurgand lista. Lista simplu inlantuita poate fi reprezentata grafic astfel:



**Varianta 9: Casa**

**Analiza datelor**

**Functia: Main**

**a,b –** variabile locale de tip pointer la structura, adresa elementelor ce trebuie interschimbate

**c –** variabila locala de tip pointer la strucura

**head\_2-** variabila locala de tip pointer la structura, adresa capului listei a doua

**n-** variabila locala de tip integer, numarul de elemente a listei simplu inlantuite

**nd** – variabila locala de tip integer, numarul de ordine a elementului pentru stergere

**nm-** variabila locala de tip integer, numarul de orine a casei pentru modificare

**id-** variabila locala de tip integer, numarul de ordine a elementului inaintea(dupa) caruia(care) adaugam un element nou.

**l-** variabila locala de tip integer, lungimea primei parti a listeisimplu inlantuite

**com, com1, af-**  variabile locale de tip integer, optiuni pentru alegere din meniuri

**fname[20]-** variabile locala de tip pointer la tipul de date character,adresa unde se pastreaza numele fisierului in care salvam datele sau de unde citim datele.

**denu –** variabila locala de tip character, sirul de caractere introdus de la tastatura

**Functia:** **creates(int n);**

**n-** variabila locala de tip integer, numarul de elemente a listei simplu inlantuite

**c –** variabila locala de tip pointer la strucura

**p-** variabila locala de tip pointer la strucura

**Functia:** **reads() si shows(casa \*head\_2);**

**c –** variabila locala de tip pointer la strucura

1. Variabila locala de tip integer, numarul de ordine a elementului in lista

**head\_2**- variabila locala de tip pointer la structura, adresa capului listei a doua

**Functia: searchs** **( char \*denu)**

**c –** variabila locala de tip pointer la strucura

**denu –** variabila locala de tip pointer la character, adresa sirul de caractere introdus de la tastatura

**Functia:** **sorts(void)**

**a,b –** variabile locale de tip pointer la structura, adresa elementelor ce trebuie interschimbate pentru sortarea listei

**l-** variabila locala returnabila de tip integer, lungimea listei simplu inlantuite

**i,k –**variabile locale de tip integer, indicele elementului in contor

**Functia: appends() si prepends()**

**c-** variabila locala de tip pointer la structura

**p-** variabila locala de tip pointer la structura

**Functia: deletes(int id)**

**nd-** variabila locala de tip integer, numarul de ordine a casei care trebuie sters

**c-** variabila locala de tip pointer la structura

**p-** variabila locala de tip pointer la structura

**Functia: insertbefores(int id) si**

**insertafters(int id)**

**id-** variabila locala de tip integer, numarul de ordine a elementului inaintea(dupa) caruia(care) adaugam un element nou

**c-** variabila locala de tip pointer la structura

**p-** variabila locala de tip pointer la structura

**Functia: modifys(int m)**

**nm-** variabila locala de tip integer, numarul de orine a casei pentru modificare

**i-** variabile locale de tip integer, indicele elementului in contor

**cm,cm1-** variabile locale de tip integer, optiuni pentru alegere din meniuri

**Functia: swaps(casa \*a, case \*b)**

**a,b –** variabile locale de tip pointer la structura, adresa elementelor ce trebuie interschimbate

**na,nb-** variabile locale de tip pointer la structura, adresele elementelor urmatoare a elementelor a si b in lista simplu inlantuita.

**t-** variabila locala de tip pointer la structura, adresa elementului intermediar care ajuta la interschimbarea a 2 case.

**Functia: lengths(void)**

**l-** variabila locala de tip integer, lungimea listei simplu inlantuite

**c-**variabila locala de tip pointer la structura

**Functia:** **saves( char \*fname) si**

**loads( char \*fname)**

**fname[]-** variabile locala de tip pointer la tipul de date character,adresa unde se pastreaza numele fisierului in care salvam datele sau de unde citim datele

**c-** variabila locala de tip pointer la structura

**p-** variabila locala de tip pointer la structura

**Functia: freemems(void)**

**c-** variabila locala de tip pointer la structura

**p-** variabila locala de tip pointer la structura

**Functia: splits(int l,casa \*\*head\_2)**

**l-** variabila locala de tip integer, lungimea primei parti a listeisimplu inlantuite

**head\_2-** variabila locala de tip pointer la pointer la structura, adresa variabilei in memorie unde este salvata adresa capului listei a doua

**c-** variabila locala de tip pointer la structura

**i-** variabile locale de tip integer, indicele elementului in contor

**Functia: joins(casa \*\*head\_2)**

**head\_2-** variabila locala de tip pointer la pointer la structura, adresa variabilei in memorie unde este salvata adresa capului listei a doua

**c-** variabila locala de tip pointer la structura

**Codul programului:**

**Fisierul : “Lab2.1.h”**

#ifndef LAB2\_1\_H\_INCLUDED

#define LAB2\_1\_H\_INCLUDED

typedef struct casa{

char adresa[20];

char telefonul[20];

int suprafata;

int nrdeodai;

float costul;

struct casa \*next;

}casa;

casa \*head;

int creates(int n);

void reads(void);

void shows(casa \*head\_2);

int searchs(char \*denu);

void sorts(void);

void appends();

int prepends();

void deletes(int nd);

casa\* insertbefores(int id);

casa\* insertafters(int id);

void modifys(int nm);

void swaps(casa \*a, casa \*b);

int lenghts(void);

void saves( char \*frame);

void loads (char \*frame);

void freemems(void);

void splits(int l, casa \*\*head\_2);

void joins(casa \*\*head\_2);

#endif // LAB2\_1\_H\_INCLUDED

**Fisierul : “functii lab2.cpp”**

#include "Lab2.1.h"

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

//crearea listei simplu inrdeodaiantuite

int creates(int n)

{ casa \*c,\*p;

int i;

for(i=0;i<n;i++)

{c=(casa\*)malloc(sizeof(\*c));

if(!c)

{return 0;

}

if(i==0)

{head =c;

}

else

{p->next=c;

}

p=c;

}

c->next=NULL;

return 1;

}

//introducere de la tacasaura

void reads(void)

{casa \*c;

int i=0;

puts("Scriti informatia despre casa:");

c=head;

while(c)

{

printf("casa %d:\n",i+1);

printf("adresa: ");

scanf("%s",&c->adresa);

printf("telefonul: ");

scanf("%s",&c->telefonul);

printf("suprafatarafata: ");

scanf("%d",&c->suprafata);

printf("Numarul de odai: ");

scanf("%d",&c->nrdeodai);

printf("costul: ");

scanf("%f",&c->costul);

c=c->next;

i++;

if(c==head) break;

}

}

//interschimbarea a 2 casae

void swaps(casa \*a,casa \*b)

{casa \*na,\*nb;

casa t;

na=a->next;

nb=b->next;

t=\*a;

\*a=\*b;

\*b=t;

a->next=na;

b->next=nb;

}

//afisarea listei

void shows(casa \*head\_2)

{

casa \*c;

int i=0;

if(head\_2 == NULL){

c=head;

} else {

c=head\_2;

}

puts("Informatia despre casa:");

while(c)

{printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("Datele casei: %d\n",i+1);

printf("Adresa: %s\n",c->adresa);

printf("telefonul: %s\n",c->telefonul);

printf("suprafatarafata: %d\n",c->suprafata);

printf("Numarul de odai: %d\n",c->nrdeodai);

printf("Costul: %.2f\n\n",c->costul);

printf ("Adresa elementului curent: ");

printf ("%p\n", c);

printf ("Adresa elementului urmator: ");

printf ("%p", c->next);

printf ("\n");

c=c->next;

i++;

}

}

//lungimea listei

int lengths(void)

{int l;

casa \*c;

l=0;

c=head;

while(c)

{l++;

c=c->next;

if(c==head) break;

}

return l;

}

//cautarea

int searchs(char \*denu)

{casa \*c=head;

int i=0,k=0;

while(c){

if(stricmp(c->adresa, denu)==0){

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

printf("Datele casaului: %d\n",i+1);

printf("adresaa: %s\n",c->adresa);

printf("telefonul: %s\n",c->telefonul);

printf("suprafatarafata: %d\n",c->suprafata);

printf("Numarul de odai: %d\n",c->nrdeodai);

printf("Costul: %.2f\n\n",c->costul);

k++;

}

c=c->next;

i++;

}

if(k){

return 1;

}

return 0;

}

//sortarea

void sorts(void)

{casa \*a,\*b;

int l;

int i,k;

l=lengths();

for(i=0;i<l-1;i++)

{a=head;

b=a->next;

for(k=0;k<l-1-i;k++)

{if(stricmp (a->adresa, b->adresa)>0)

{swaps(a,b);

}

a=a->next;

b=b->next;

}

}

}

//modificare

void modifys(int nm)

{casa \*c=head;

int i=0,cm,cm1;

while(c){

if(i+1 == nm){

asd:system("cls");

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

printf("Datele casaului: %d\n",i+1);

printf("[1] adresa: %s\n",c->adresa);

printf("[2] telefonul: %s\n",c->telefonul);

printf("[3] suprafatarafata: %d\n",c->suprafata);

printf("[4] Numarul de odai: %d\n",c->nrdeodai);

printf("[5] Costul: %.2f\n\n",c->costul);

printf("Ce doriti sa modificati?\n\nComanda << ");

scanf("%d",&cm);

switch(cm){

case 1:

printf("adresaa: ");

scanf("%s",&c->adresa);

break;

case 2:

printf("telefonul: ");

scanf("%s",&c->telefonul);

break;

case 3:

printf("suprafatarafata: ");

scanf("%d",&c->suprafata);

break;

case 4:

printf("Numarul de odai: ");

scanf("%d",&c->nrdeodai);

break;

case 5:

printf("Costul: ");

scanf("%f",&c->costul);

break;

default:

printf("\n\aAti introdus o comanda gresita! { %d }\nIncercati din nou.\n");

system("pause");

goto asd;

break;

}

printf("\n\nMai doriti sa modificati ceva? 1 / 0 \n");

scanf("%d",&cm1);

if(cm1){

goto asd;

} else {

return;

}

}

c=c->next;

i++;

}

}

//elierarea memoriei

void freemems(void)

{casa \*c,\*p;

c=head;

while(c)

{p=c;

c=c->next;

free(p);

if(c==head) break;

}

return;

}

//adaugarea dupa

casa\* insertafters(int id)

{

casa \*c=head,\*p;

int i=0;

while(c){

if(i == id-1){

p=c;

c=(casa\*)malloc(sizeof(casa));

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

printf("Introdu datele casaului\n");

printf("adresaa: ");

scanf("%s",&c->adresa);

printf("telefonul: ");

scanf("%s",&c->telefonul);

printf("suprafatarafata: ");

scanf("%d",&c->suprafata);

printf("Numarul de odai: ");

scanf("%d",&c->nrdeodai);

printf("Costul: ");

scanf("%f",&c->costul);

c->next=p->next;

p->next=c;

break;

}

i++;

c=c->next;

}

}

//adaugarea inainte

casa \*insertbefores(int id){

casa \*c=head,\*p;

int i=0;

while(c){

if(i == id-1){

p=c;

c=(casa\*)malloc(sizeof(casa));

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

printf("Introdu datele casaului\n");

printf("adresaa: ");

scanf("%s",&c->adresa);

printf("telefonul: ");

scanf("%s",&c->telefonul);

printf("suprafatarafata: ");

scanf("%d",&c->suprafata);

printf("Numarul de odai: ");

scanf("%d",&c->nrdeodai);

printf("Costul: ");

scanf("%f",&c->costul);

if(p == head){

c->next=head;

head=c;

break;

} else {

c->next=p;

i=0;

p=head;

while(p){

if(i == id-2){

p->next=c;

}

i++;

p=p->next;

}

break;

}

}

i++;

c=c->next;

}

}

//adaugarea la sfirsit

void appends(){

casa \*c=head,\*p;

while(c){

if(c->next == NULL){

p=c;

c=(casa\*)malloc(sizeof(casa));

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

printf("Introdu datele casaului\n");

printf("adresaa: ");

scanf("%s",&c->adresa);

printf("telefonul: ");

scanf("%s",&c->telefonul);

printf("suprafatarafata: ");

scanf("%d",&c->suprafata);

printf("Numarul de odai: ");

scanf("%d",&c->nrdeodai);

printf("Costul: ");

scanf("%f",&c->costul);

p->next=c;

c->next=NULL;

break;

}

c=c->next;

}

}

//adaugarea la inceput

int prepends(){

casa \*c,\*p;

p=(casa\*)malloc(sizeof(casa));

if(!p)

{return 0;

}

if(head==NULL)

{p->next=NULL;

p=head;

}

else

{p->next=head;

head=p;

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

printf("Introdu datele casaului\n");

printf("adresaa: ");

scanf("%s",&p->adresa);

printf("telefonul: ");

scanf("%s",&p->telefonul);

printf("suprafatarafata: ");

scanf("%d",&p->suprafata);

printf("Numarul de odai: ");

scanf("%d",&p->nrdeodai);

printf("Costul: ");

scanf("%f",&p->costul);

}

}

//stergerea casaului

void deletes(int nd)

{

casa \*c=head,\*p;

int i=0;

while(c){

if(i == nd-1){

if(c == head){

head=c->next;

free(c);

break;

} else if(c->next == NULL){

p=head;

while(p){

if(p->next == c){

free(c);

p->next=NULL;

break;

}

p=p->next;

}

} else {

p=head;

while(p){

if(p->next == c){

p->next=c->next;

free(c);

break;

}

p=p->next;

}

}

}

i++;

c=c->next;

}

}

//divizarea

void splits(int l,casa \*\*head\_2){

casa \*c=head;

int i=0;

while(c){

if(i == l-1){

\*head\_2=c->next;

c->next=NULL;

break;

}

i++;

c=c->next;

}

}

//unirea

void joins(casa \*\*head\_2){

casa \*c=head;

while(c){

if(c->next == NULL){

c->next=\*head\_2;

break;

}

c=c->next;

}

\*head\_2=NULL;

}

//salvarea

void saves(char \*fname){

casa \*c=head;

FILE \*demo=fopen(fname,"w");

while(c){

fprintf(demo,"%s %s %d %d %.2f",c->adresa,c->telefonul,c->suprafata,c->nrdeodai,c->costul);

if(c->next != NULL){

fprintf(demo,"\n");

}

c=c->next;

}

fclose(demo);

}

//citirea din fisier

void loads(char \*fname){

casa \*c,\*p;

int i=0;

FILE \*demo=fopen(fname,"r");

if(!demo){

printf("\aFisierul nu exista!\n");

system("pause");

exit(1);

}

while(!feof(demo)){

c=(casa\*)malloc(sizeof(casa));

if(c == NULL){

printf("\n\aEroare la alocarea memoriei!\n");

system("pause");

}

if(i == 0){

head=c;

p=c;

} else {

p->next=c;

p=c;

}

fscanf(demo,"%s %s %d %d %f",&c->adresa,&c->telefonul,&c->suprafata,&c->nrdeodai,&c->costul);

i++;

}

p->next=NULL;

fclose(demo);

}

**Fisierul: “Main2.cpp”**

**#include "functii lab2.cpp"**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<conio.h>

#include "functii lab2.cpp"

int main(){

casa \*head2=NULL;

casa \*a,\*b,\*c,\*head\_2=NULL ;

int n,m=1,nd,nm,id,l,af;

int com,com1;

char fname[20];

char denu;

while(1){

fflush(stdin);

while(m){

fflush(stdin);

m=0;

system("cls");

printf("Introduceti datele caselor|\n-------------------------------\n");

printf("\n[ 1 ] Manual\n[ 2 ] Fisier\n\n[ 0 ] Exit\n\nComanda << ");

scanf("%d",&com);

system("cls");

switch(com){

case 0:

if(head){

freemems();

}

exit(0);

break;

case 1:

printf("Introdu numarul de case: ");

scanf("%d",&n);

if(creates(n)){

reads();

} else {

printf("\nEroare la alocarea memoriei!\a\n");

system("pause");

}

break;

case 2:

printf("Introdu numele fisierului pentru citire:\n");

scanf("%s",&fname);

strcat(fname,".txt");

loads(fname);

break;

default:

printf("\aAti introdus o comanda gresita!\n");

system("pause");

m=1;

break;

}

saves(fname);

}

qaz: system("cls");

printf("\n[ 1 ] Lista cu case");

printf("\n[ 2 ] Cautarea caselor dupa denumire");

printf("\n[ 3 ] Lista in ordine descrescatoare dupa suprafata");

printf("\n[ 4 ] Adaugarea casei la sfirsit");

printf("\n[ 5 ] Adaugarea casei la inceput");

printf("\n[ 6 ] Stergerea unei case ");

printf("\n[ 7 ] Modifica datele casei");

printf("\n[ 8 ] Sortarea caselor dupa denumire");

printf("\n[ 9 ] Salvarea datelor in fisier");

printf("\n[ 10 ] Interschimbarea a 2 elemente");

printf("\n[ 11 ] Adaugarea a unei case inaintea casei indicat");

printf("\n[ 12 ] Adaugarea a unei case dupa casa indicata");

printf("\n[ 13 ] Desparte lista in 2");

printf("\n[ 14 ] Unirea listei");

printf("\n[ 15 ] Meniul precedent");

printf("\n[ 0 ] Exit");

printf("\n\nComanda << ");

scanf("%d",&com1);

system("cls");

switch(com1){

case 0:

if(head){

freemems();

}

exit(0);

break;

case 1:

if(head\_2){

printf("[ 1 ] Afisare head\_1\n");

printf("[ 2 ] Afisare head\_2\n\n");

printf("[ 0 ] Iesire din acest meniu\n");

printf("\nComanda << ");

scanf("%d",&af);

switch(af){

system ("pause");

case 0:

goto qaz;

system ("pause");

break;

case 1:

shows(NULL);

system ("pause");

getch();

break;

case 2:

shows(head\_2);

system ("pause");

getch();

break;

default:

printf("\aAti introdus o comanda gresita!\n");

system("pause");

goto qaz;

break;

}

} else {

shows(NULL);

}

break;

case 2:

printf("Introduceti denumirea casei: ");

scanf("%s",&denu);

if(!searchs(&denu)){

printf("\nNu sa gasit casa cu asa denumire{ %s }\a\n\n",denu);

}

break;

case 3:

printf("Lista in ordine descrescatoare dupa suprafata: \n");

sorts();

shows(NULL);

break;

case 4:

appends();

break;

case 5:

prepends();

break;

case 6:

printf("Introdu numarul de ordine a casei pentru stergere: ");

scanf("%d",&nd);

if(nd > 0 && nd <= lengths()){

deletes(nd);

printf("Casa a fost sters cu succes.\n");

} else {

printf("\aAti introdus un numar iesit din intervalul existent\n");

system("pause");

goto qaz;

}

break;

case 7:

printf("Introdu numarul de ordine a casei: ");

scanf("%d",&nm);

modifys(nm);

break;

case 8:

system ("cls");

sorts ();

shows (head\_2);

system ("pause");

break;

case 9:

printf("Introdu numele fisierului pentru salvare: ");

scanf("%s",&fname);

strcat(fname,".txt");

saves(fname);

break;

case 10:

system ("cls");

printf ("Alegeti primul element care trebuie de interschimbat: ");

scanf ("%casa", &a);

printf ("Alegeti al doilea element care trebu de interschimbat: ");

scanf ("casa", &b);

swaps (a, b);

getch ();

break;

case 11:

system ("cls");

printf ("Alegeti numarul de ordine a casei inaintea caruia doriti sa inserati noua casa: ");

scanf ("%i", &id);

c=insertbefores(id);

system ("pause");

break;

case 12:

system ("cls");

printf ("Alegeti numarul de ordine a casei dupa care doriti sa inserati noua casa: ");

scanf ("%i", &id);

c=insertafters(id);

system ("pause");

break;

case 13:

printf("Introdu lungimea primei parti, ( partea a 2 va obtine restul)\n");

scanf("%d",&l);

if(lengths() > 1 && l>0 && l < lengths()){

splits(l,&head\_2);

} else {

printf("\aListele trebuie sa aiba macar cite un element!\n");

}

break;

case 14:

if(head\_2){

joins(&head\_2);

printf("Unirea a avut loc cu succes\n");

} else {

printf("\aMai intii lista trebuie despartita!\n");

}

break;

case 15:

m=1;

break;

default:

printf("\aAti introdus o comanda gresita!\n");

break;

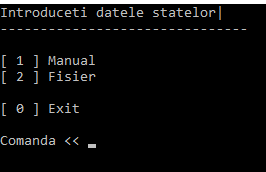
}

}

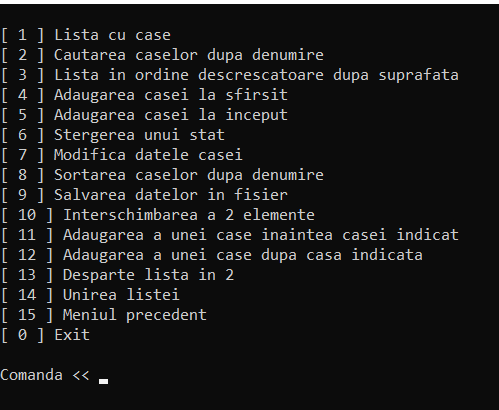
}

**Rezultatele obtinute:**

**Meniul de introducere a datelor de la tastatura si din fisier**

****

**Meniu**



**Concluzie**

In urma acestei lucrari de laborator am ajuns la ideia ca e destul de usor de lucrat cu structuri de date liste, ele sunt usor de modificat si lucrul dat nu necesita mult timp.

De asemenea verificarea rezultatelor demonstreaza ca valorile primite sunt adevarate ceea ce arata ca programul lucreaza.