Ministerul Educaţiei, Culturii și Cercetării

al Republicii Moldova   
  
Universitatea Tehnică a Moldovei   
  
  
  
  
  
  
  
RAPORT

Lucrarea de laborator nr. 2   
la Structuri de date si algoritmi

A efectuat: st. gr. TI-211 Popa Cătălin   
  
A verificat: Stratulat Ștefan

UTM, Chișinău 2021

**Tema:**

Implementarea tipului de date abstract “Listă simplu înlățuită” în limbajul C.

**Scopul lucrării**

Scopul lucrării este de a familiariza studentul cu mecanismul de creare a listeleor simplu înlățuite și operații elementare asupra acestuia, utilizînd pentru aceasta limbajul C.

**Problema**

1. **Să se elaboreze un program ce va aloca dinamic o listă simplu înlățuită de structuri și va realiza următoarele funcții, unde funcțiile date reprezintă opțiuni organizate într-un meniu în cadrul programului:**

* citirea de la tastatură a elementelor listei;
* afișarea la consolă a elementelor listei;
* căutarea unui element după o valoarea a unui cîmp din structură și returnarea poziției unde se află primul element găsit;
* eliberarea memoriei listei;
* sortarea elementelor după un cîmp a structurii;
* inserarea unui element nou la capătul listei;
* inserarea unui element nou la începutul listei;
* insearea unui element la o poziție X (introdusă de la tastatură) în cadrul listei;
* ștergerea unui element de pe o poziție X ( introdusă de la tastatură) din cadrul listei;
* inversarea ordinei elementelor din listă;
* curățirea listei.

#include <stdio.h>

#include <stdlib.h>

typedef struct Node

{

int data;

struct Node \* next;

} Node;

typedef struct List

{

Node \* head;

} List;

List \* create\_list()

{

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

list->head = NULL;

return list;

}

Node \* create\_node(int data)

{

Node \* node = malloc(sizeof(Node));

node->data = data;

return node;

}

void push\_front(List \*list, int data)

{

Node \* new\_node = create\_node(data);

new\_node->next = list->head;

list->head = new\_node;

}

int pop\_front(List \*list)

{

Node \* old\_node = list->head;

int return\_value = 0;

if(old\_node)

{

list->head = list->head->next;

return\_value = old\_node->data;

free(old\_node);

}

return return\_value;

}

int search\_list(List \*list, int data)

{

int counter = 0;

for(Node \* current\_node = list->head; current\_node; current\_node=current\_node->next)

{

if(current\_node->data==data)

{

return counter;

}

counter++;

}

return -1;

}

void print\_list(List \*list)

{

printf("\nElementele listei:\n");

if(list->head==NULL)

{

printf("\nLista este vida!\n");

return;

}

for(Node \* current\_node = list->head; current\_node; current\_node=current\_node->next)

{

printf("%d ",current\_node->data);

}

printf("\n");

}

void clear\_list(List \*list)

{

Node \* current\_node = list->head, \* back\_node =NULL;

while(current\_node)

{

back\_node = current\_node;

current\_node=current\_node->next;

free(back\_node);

}

list->head = NULL;

}

void delete\_list(List \*list)

{

clear\_list(list);

free(list);

}

int is\_empty(List \*a)

{

return (a->head)?0:1;

}

int menu()

{

int error\_input\_found = 0;

menu\_start:

system("cls");

printf("---------Menu---------\n");

printf("1) Creaza un sir de numere\n");

printf("2) Afiseaza sirul\n");

printf("3) Elibereaza memoriei\n");

printf("4) Cauta un element\n");

printf("5) Adaugare element la inceput\n");

printf("6) Stergerea elementului de pe o pozitie\n");

printf("7) Curatarea listei\n");

int command = 100;

printf("\n\nSelectati commanda: ");

scanf("%d",&command);

if((command>10)||(command<0))

{

error\_input\_found = 1;

goto menu\_start;

}

return command;

}

void main()

{

int values[100];

List \* a = create\_list();

int command=100;

while(command)

{

command = menu();

switch(command)

{

case 1:

{

printf("\nDati 6 numere: \n");

for(int i = 0; i < 6; ++i)

{

scanf("%d,", &values[i]);

}

for(int i=0; i<6; i++)

{

push\_front(a,values[i]);

}

break;

}

case 2:

{

print\_list(a);

break;

}

case 3:

{

free(a);

printf("Memoria a fost eliberata");

List \* a = create\_list();

break;

}

case 4:

{

int n;

printf("Dati un numar pe care doriti sa il gasiti:");

scanf("%d",&n);

int found\_position = search\_list(a,n);

if(found\_position!=-1)

{

printf("\nNumarul [%d] a fost gasit pe pozitia %d!\n",n,found\_position);

}

else

{

printf("\nNumarul [%d] nu a fost gasit in lista!\n",n);

}

break;

}

case 5:

{

int b;

printf("\nDati un numar care doriti sa adaugati in fata: \n");

scanf("%d",&b);

push\_front(a,b);

break;

}

case 6:

{

pop\_front(a);

break;

}

case 7:

{

clear\_list(a);

break;

}

}

printf("\n\nApasati orice buton pentru continuare!\n");

\_getch();

}

return 0;

}











