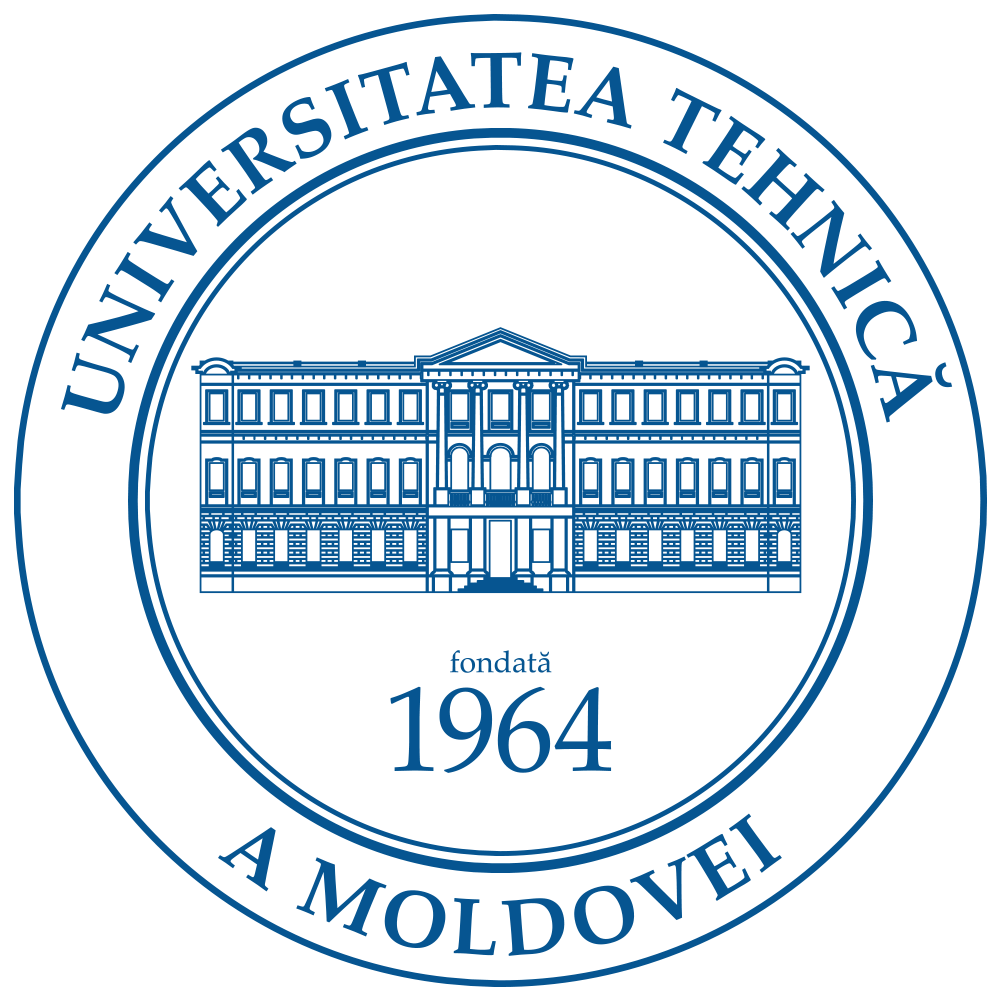
Ministerul Educaţiei, Culturii și Cercetării

al Republicii Moldova   
  
Universitatea Tehnică a Moldovei   
  
  
  
  
  
  
  
RAPORT

Lucrarea de laborator nr. 1  
la Structuri de date si algoritmi   
  
  
A efectuat: Costin Ion

st. gr. TI-231  
  
A verificat: Ștefan Stratulat



UTM, Chișinău 2024

**Tema:**

Implementarea tipului de date abstract “Tablou de structuri” în limbajul C.

**Scopul lucrarii**

Scopul lucrării este de a familiariza studentul cu mecanismul de creare a tablourilor statice și dinamice de structuri și operații elementare asupra acestuia, utilizînd pentru aceasta limbajul C.

**Problema**

1. **Să se elaboreze un program ce va aloca static un tablou unidimensional de structuri și va realiza următoarele funcții:**

* citirea de la tastatură a elementelor tabloului;
* afișarea la consolă a elementelor tabloului;
* căutarea unui element după o valoare a unui cîmp din structură și returnarea poziției unde se află primul element găsit.

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <stdbool.h>

typedef struct library

{int book\_size\_as\_in\_pages;

char author\_surname[200];

char title[200];

char author\_name[200];

int number\_of\_books;

} library;

void read\_vector (library b[100], int n)

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

{printf("Insert some information about the book: \n");

printf("The title> ");

scanf("%s", &b[i].title);

printf("The name of the author> ");

scanf("%s", &b[i].author\_name);

printf("The surname of the author> ");

scanf("%s", &b[i].author\_surname);

printf("Number of pages> ");

scanf("%d", &b[i].book\_size\_as\_in\_pages); }}

void show\_vector (library b[100], int n)

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

{printf("The book '%s' is written by %s %s, and has %d pages !\n", b[i].title, b[i].author\_name, b[i].author\_surname,b[i].book\_size\_as\_in\_pages ); }}

void find\_a\_book(library b[100], int n)

{char key[200];

printf("Insert the name of the book you are looking for> \n");

scanf("%s", &key);

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

{if(strcmp(key, b[i].title) == 0)

{printf ("The book named '%s' is located on the %d position.", b[i].title, i+1); }} }

int main ()

{int number\_of\_books;

printf ("Insert the number of books you want to have in the system > ");

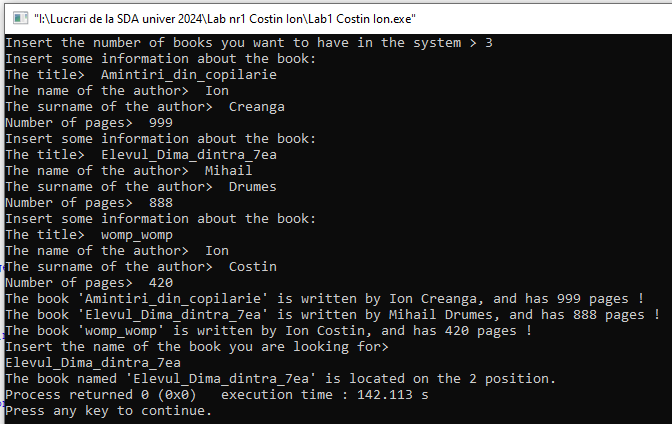
scanf ("%d", &number\_of\_books);

library b[number\_of\_books];

read\_vector( b, number\_of\_books);

show\_vector (b, number\_of\_books);

find\_a\_book(b ,number\_of\_books); }



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

* 1. citirea de la tastatură a elementelor tabloului;
* 2. afișarea la consolă a elementelor tabloului;
* 3. căutarea unui element după o valoarea a unui cîmp din structură și returnarea poziției unde se află primul element găsit;
* 4. eliberarea memoriei tabloului;
* 5. sortarea elementelor după un cîmp a structurii;
* 6. inserarea unui element nou la capătul tabloului;
* 7. inserarea unui element nou la începutul tabloului;
* 8. insearea unui element la o poziție X (introdusă de la tastatură) în cadrul tabloului;
* 9. ștergerea unui element de pe o poziție X ( introdusă de la tastatură) din cadrul tabloului.

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <stdbool.h>

struct element

{int value;

};

void read(struct element \*\*table, int \*n)

{printf("Insert the number of elements you want in your table: ");

scanf("%d", n);

\*table = (struct element \*)malloc(\*n \* sizeof(struct element));

for (int i = 0; i < \*n; i++)

{printf("Insert the value of the %d element: ", i + 1);

scanf("%d", &(\*table)[i].value);}}

void show(struct element \*table, int n)

{printf("The table's elemets are: ");

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

{

printf("%d ", table[i].value);}

printf("\n");}

int search(struct element \*table, int n, int val)

{

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

{if (table[i].value == val)

{return i; }} return -1;}

void free\_memory(struct element \*table) {

free(table);

}

void sort\_element(struct element \*table, int n)

{for (int i = 0; i < n - 1; i++)

{for (int j = i + 1; j < n; j++)

{if (table[i].value > table[j].value)

{struct element temp = table[i];

table[i] = table[j];

table[j] = temp; }}} }

void insert\_at\_the\_end(struct element \*\*table, int \*n, int val)

{(\*n)++;

\*table = (struct element \*)realloc(\*table, \*n \* sizeof(struct element));

(\*table)[\*n - 1].value = val; }

void insert\_at\_start(struct element \*\*table, int \*n, int val)

{(\*n)++;

\*table = (struct element \*)realloc(\*table, \*n \* sizeof(struct element));

for (int i = \*n - 1; i > 0; i--)

{(\*table)[i] = (\*table)[i - 1]; }

(\*table)[0].value = val;}

void insert\_at\_position\_given(struct element \*\*table, int \*n, int val, int position)

{(\*n)++;

\*table = (struct element \*)realloc(\*table, \*n \* sizeof(struct element));

for (int i = \*n - 1; i > position; i--)

{(\*table)[i] = (\*table)[i - 1]; }

(\*table)[position].value = val;}

void delete\_position\_given(struct element \*\*table, int \*n, int position)

{for (int i = position; i < \*n - 1; i++)

{(\*table)[i] = (\*table)[i + 1]; }

(\*n)--;

\*table = (struct element \*)realloc(\*table, \*n \* sizeof(struct element));}

int main()

{int option, position, val;

int n = 0;

struct element \*table = NULL;

do

{

printf("\nMenu:\n");

printf("1. I want to insert the element(s) of the table\n");

printf("2. I want to see all the elemets inserted in the table so far\n");

printf("3. I want to search an element with the value of a field for the structure\n");

printf("4. I want to free the table's memory (delete all elements)\n");

printf("5. I want to sort the elements within the table after a structure's field\n");

printf("6. I want to insert a new element at the end of the table\n");

printf("7. I want to insert a new element at the start of the table\n");

printf("8. I want to insert an element at an X position within the table\n");

printf("9. I want to delete an element from an X position from within the table\n");

printf("10. I want to end it all\n\n");

printf("What now? ");

scanf("%d", &option);

switch (option)

{case 1:

read(&table, &n);

break;

case 2:

show(table, n);

break;

case 3:

printf("What's the value of the element you want to search> ");

scanf("%d", &val);

position = search(table, n, val);

if (position == -1)

{printf("The element with the %d value was not found in the table\n", val);}

else

{printf("The element with the %d value was found on the %d position\n", val, position+1);}

break;

case 4:

free\_memory(table);

break;

case 5:

sort\_element(table, n);

break;

case 6:

printf("What's the value of your new element? ");

scanf("%d", &val);

insert\_at\_the\_end(&table, &n, val);

break;

case 7:

printf("What's the value of your new element? ");

scanf("%d", &val);

insert\_at\_start(&table, &n, val);

break;

case 8:

printf("What's the value of your new element? ");

scanf("%d", &val);

printf("Where do you want to insert/add your new element? ");

scanf("%d", &position);

insert\_at\_position\_given(&table, &n, val, position);

break;

case 9:

printf("What's the position of the element you want to delelete (!starting with position 0!): ");

scanf("%d", &position);

delete\_position\_given(&table, &n, position);

break;

case 10:

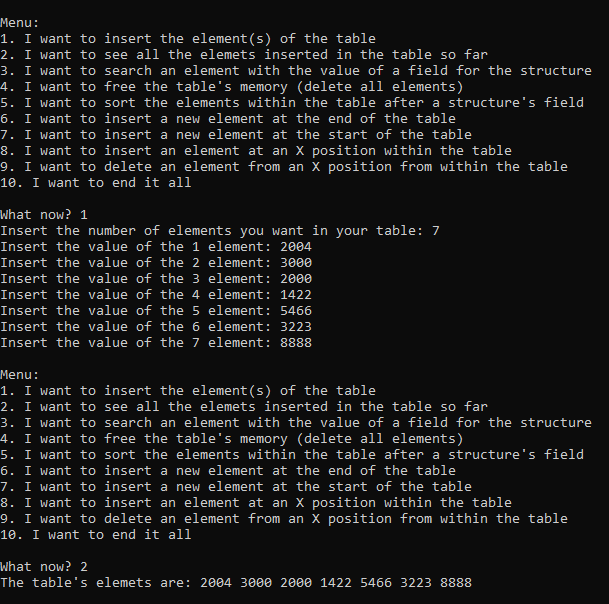
free\_memory(table);

printf("The End.\n");

break;

default:

printf("Whatever you inserted - not a valid input!\n");

 break; }} while (option != 10); return 0;}

