****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**Calancea Cătălin**

**MI-222**

**Raport**

**pentru lucrarea de laborator Nr.2**

***la cursul de “Structuri de date şi algoritmi”***

Verificat:

**Guțu Maria,** *doctor, conf. univ.*

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

**Chișinău – 2023**

**Scopul lucrării:** Programarea algoritmilor de prelucrare a tablourilor 1D & 2D, aplicând tehnicile și metodele de sortare prin utilizarea funcțiilor, pointerilor, alocării dinamice a memoriei în limbajul C.

**Nr. variantei:** 4

**Condiția problemei ( sarcinii de lucru ) :**

1. Elaborați un program C care va crea un meniu recursiv. Acesta trebuie să cuprindă următoarele funcții cu apelare ulterioară ale acestora în funcția main:

1. Alocarea dinamică a memoriei ;

2. Introducerea valorilor tabloului de la tastatură ;

3. Completarea tabloului cu valori random ;

4. Afisarea tabloului;

5. Bubble Sort ;

6. Selection Sort ;

7. Insertion Sort ;

0. Eliberarea memoriei și ieșirea din program.

**Codul programului:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

void alloc\_memory (int \*\*arr, int n);

void input\_values (int \*arr, int n);

void random\_values (int \*arr, int n);

void print\_array (int \*arr, int n);

void bubble\_sort (int \*arr, int n);

void selection\_sort (int \*arr, int n);

void insertion\_sort (int \*arr, int n);

void free\_memory (int \*arr);

int

main ()

{

int \*arr = NULL;

int n;

int choice;

printf ("Enter the size of the array: ");

scanf ("%d", &n);

while (1)

{

printf ("\n1. Allocate memory.\n");

printf ("2. Input values\n");

printf ("3. Random values\n");

printf ("4. Print array\n");

printf ("5. Bubble Sort\n");

printf ("6. Selection Sort\n");

printf ("7. Insertion Sort\n");

printf ("8. Free memory\n");

printf ("0. Exit\n");

printf ("Enter your choice: ");

scanf ("%d", &choice);

switch (choice)

{

case 1:

alloc\_memory (&arr, n);

break;

case 2:

input\_values (arr, n);

break;

case 3:

random\_values (arr, n);

break;

case 4:

print\_array (arr, n);

break;

case 5:

bubble\_sort (arr, n);

break;

case 6:

selection\_sort (arr, n);

break;

case 7:

insertion\_sort (arr, n);

break;

case 8:

free\_memory (arr);

break;

case 0:

exit (0);

default:

printf ("Invalid choice!\n");

}

}

return 0;

}

void

alloc\_memory (int \*\*arr, int n)

{

\*arr = (int \*) malloc (n \* sizeof (int));

if (\*arr == NULL)

{

printf ("Memory allocation failed!\n");

exit (1);

}

else

{

printf ("Memory allocated succesfuly.\n");

}

}

void

input\_values (int \*arr, int n)

{

printf ("Enter %d values:\n", n);

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

{

scanf ("%d", (arr + i));

}

}

void

random\_values (int \*arr, int n)

{

srand (time (NULL));

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

{

\*(arr + i) = rand () % 100;

}

printf ("Random values generated successfully!\n");

}

void

print\_array (int \*arr, int n)

{

printf ("The array is:\n");

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

{

printf ("%d ", \*(arr + i));

}

printf ("\n");

}

void

bubble\_sort (int \*arr, int n)

{

bool sorted;

do

{

sorted = true;

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

{

if (\*(arr + i) > \*(arr + i + 1))

{

int temp = \*(arr + i);

\*(arr + i) = \*(arr + i + 1);

\*(arr + i + 1) = temp;

sorted = false;

}

}

}

while (!sorted);

printf ("Array sorted successfully using Bubble Sort!\n");

}

void

selection\_sort (int \*arr, int n)

{

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

{

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

{

if (\*(arr + i) > \*(arr + j))

{

int temp = \*(arr + i);

\*(arr + i) = \*(arr + j);

\*(arr + j) = temp;

}

}

}

printf ("Array sorted successfully using Selection Sort!\n");

}

void

insertion\_sort (int \*arr, int n)

{

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

{

int p = i;

while (p > 0 && \*(arr + p - 1) >= \*(arr + p))

{

int temp = \*(arr + p - 1);

\*(arr + p - 1) = \*(arr + p);

\*(arr + p) = temp;

p--;

}

}

printf ("Array sorted successfully using Insertion Sort!\n");

}

void

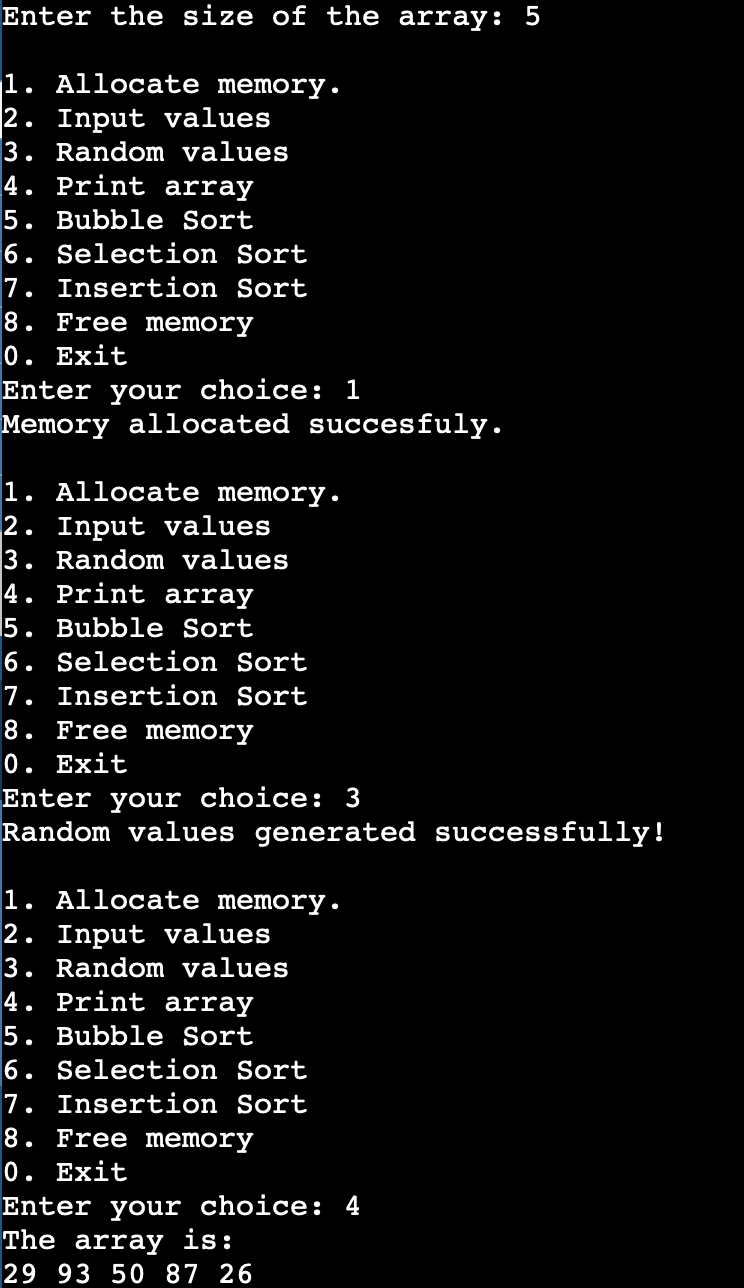
free\_memory (int \*arr)

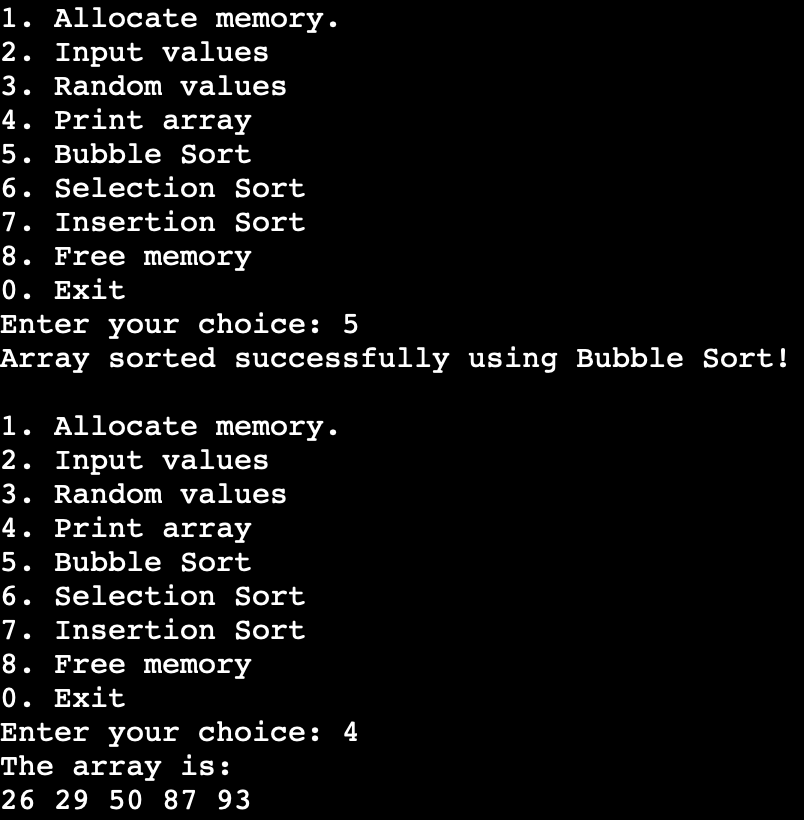
{

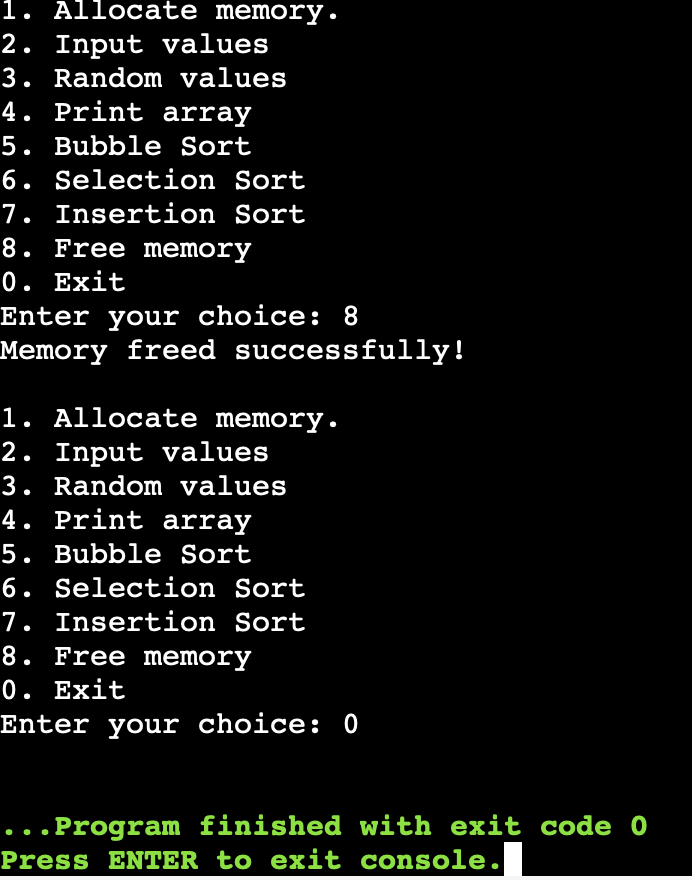
free (arr);

printf ("Memory freed successfully!\n");

}







**Concluzie:**În urma executarii lucrării de laborator nr.1 , am facut cunostinta cu algoritmilor de prelucrare a tablourilor 1D și 2D, sortarea este o tehnică importantă pentru a organiza și manipula datele într-o ordine specifică. În limbajul C, există mai multe funcții de sortare disponibile, cum ar fi inserion sort , selection sort și bubble sort, care pot fi utilizate pentru a sorta un tablou. Alocarea dinamică a memoriei poate fi folosită pentru a crea tablouri cu dimensiuni variabile sau pentru a permite modificarea dimensiunilor tablourilor în timpul rulării programului.