MINISTERUL EDUCAȚIEI AL REPUBLICII MOLDOVA

UNIVERSITATEA TEHNICĂ A MOLDOVEI

**RAPORT**

LUCRAREA DE LABORATOR NR. 6

LA PROGRAMAREA CALCULATOARELOR

Elaborat: st. gr. TI – 211, Popa Ecaterina

Verificat: Stratulat Ștefan

UTM, Chișinău, 2021

**Probleme pentru pointeri:**

1. Scrieți un program C pentru a crea, inițializa și utiliza pointeri pentru printarea a diferitor variabile.

#include <stdio.h>

#include <stdlib.h>

void main ()

{

int \*p, \*ptr, \*y;

int a, b, c;

p = &a;

ptr = &b;

y = &c;

printf ("a = ");

scanf("%d", &a);

printf ("Value = %d\n", \*p);

printf ("Address = %d\n", p);

printf ("\nb = ");

scanf("%d", &b);

printf ("Value = %d\n", \*ptr);

printf ("Address = %d\n", ptr);

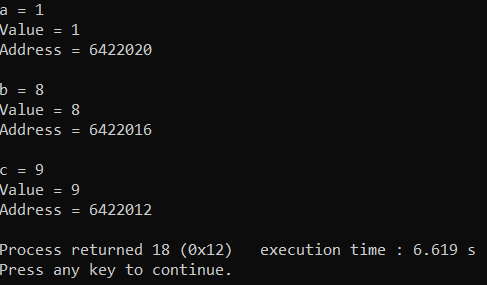
printf ("\nc = ");

scanf("%d", &c);

printf ("Value = %d\n", \*y);

printf ("Address = %d\n", y);

}



1. Scrieți un program C pentru a aduna două numere folosind pointerii.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void main ()

{

srand(time(NULL));

int \*p, \*ptr, \*sum, c;

int a = rand()%100;

int b = rand()%100;

p = &a;

ptr = &b;

sum = &c;

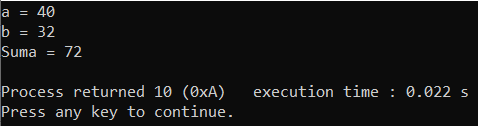
printf ("a = %d\n", a);

printf ("b = %d\n", b);

\*sum = \*p + \*ptr;

printf ("Suma = %d\n", \*sum);

}



1. Scrieți un program C pentru a interschimba valorile a două variabile folosind pointerii.

#include <stdio.h>

#include <stdlib.h>

void main ()

{

int \*p, \*ptr, temp;

int a = rand()%100;

int b = rand()%100;

p = &a;

ptr = &b;

printf ("Inainte de schimbare:\n");

printf ("a = %d\n", a);

printf ("b = %d\n", b);

temp = \*p;

\*p = \*ptr;

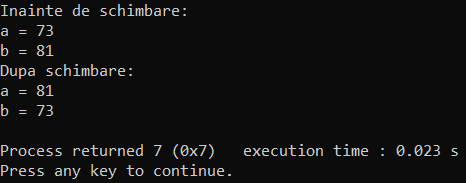
\*ptr = temp;

printf ("Dupa schimbare:\n");

printf ("a = %d\n", a);

printf ("b = %d\n", b);

}



1. Scrieți un program C pentru a inițializa și afișa elementele unei matrici folosind pointerii.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void main ()

{

srand (time(NULL));

int row, col;

printf ("Numarul de randuri: ");

scanf("%d", &row);

printf ("Numarul de coloane: ");

scanf("%d", &col);

int matrix[row][col];

int \*p=&matrix;

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

{

for (int j = 0; j < col; j++)

{

\*((p+i)+j) = rand()%100;

}

}

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

{

for (int j = 0; j < col; j++)

{

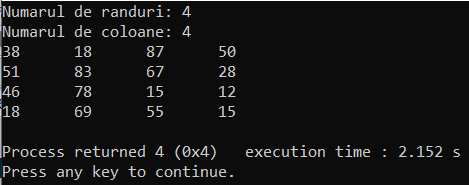
printf ("%d\t", \*((p + i)+ j));

}

printf ("\n");

}

}



1. Scrieți un program C pentru a copia o matrice în alta folosind pointeri.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void main ()

{

srand (time(NULL));

int row, col;

printf ("Numarul de randuri: ");

scanf("%d", &row);

printf ("Numarul de coloane: ");

scanf("%d", &col);

int matrix[row][col];

int \*p=&matrix;

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

{

for (int j = 0; j < col; j++)

{

\*((p+i)+j) = rand()%100;

}

}

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

{

for (int j = 0; j < col; j++)

{

printf ("%d\t", \*((p + i)+ j));

}

printf ("\n");

}

printf("\n");

int copy[row][col];

int \*ptr = &copy;

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

{

for (int j = 0; j < col; j++)

{

copy[i][j] = matrix[i][j];

}

}

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

{

for (int j = 0; j < col; j++)

{

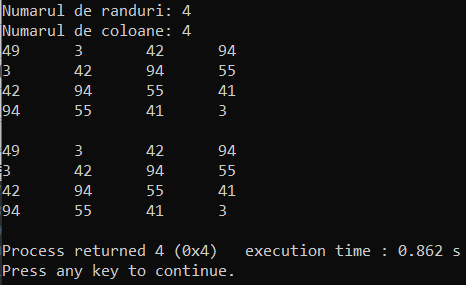
printf ("%d\t", \*((ptr + i)+ j));

}

printf ("\n");

}

}



1. Scrieți un program C pentru a interschimba valorile a două matrice folosind pointeri.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void main ()

{

srand (time(NULL));

int row, col;

printf ("Numarul de randuri: ");

scanf("%d", &row);

printf ("Numarul de coloane: ");

scanf("%d", &col);

int matrix[row][col];

int \*p=&matrix;

printf("Matricele inainte de schimbare: \n");

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

{

for (int j = 0; j < col; j++)

{

\*((p+i)+j) = rand()%100;

}

}

printf("\n");

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

{

for (int j = 0; j < col; j++)

{

printf ("%d\t", \*((p + i)+ j));

}

printf ("\n");

}

printf("\n");

int copy[row][col];

int \*ptr = &copy;

int temp;

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

{

for (int j = 0; j < col; j++)

{

\*((ptr+i)+j) = rand()%100;

}

}

printf ("\n");

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

{

for (int j = 0; j < col; j++)

{

printf ("%d\t", \*((ptr + i)+ j));

}

printf ("\n");

}

printf ("\n");

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

{

for (int j = 0; j < col; j++)

{

temp = p;

p = ptr;

ptr = temp;

}

printf ("\n");

}

printf("Matricele dupa schimbare: \n");

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

{

for (int j = 0; j < col; j++)

{

printf ("%d\t", \*((p + i)+ j));

}

printf ("\n");

}

printf("\n");

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

{

for (int j = 0; j < col; j++)

{

printf ("%d\t", \*((ptr + i)+ j));

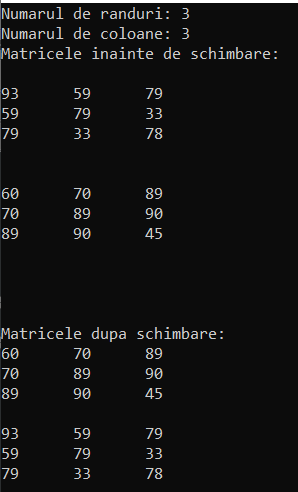
}

printf ("\n");

}

printf ("\n");

}



**Probleme pentru alocarea dinamică:**

1. Scrieți un program C pentru alocarea unui șir de caractere și inițializarea acestuia.

#include <stdio.h>

#include <stdlib.h>

void main ()

{

char \*str = (char\*)malloc(40\*sizeof(char));

printf ("Introduceti un sir de caractere: ");

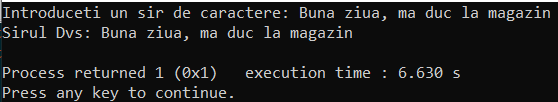
gets(str);

printf ("Sirul Dvs: ");

puts (str);

free(str);

}



1. Scrieți un program C pentru alocarea, inițializarea și afișarea a unui tablou unidimensional de matrici, ce va fi gestionat de un pointer triplu(ex: int \*\*\*ptr).

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void main()

{

int mat = 2; // numarul de matrici

int row = 3; // numarul de randuri

int col = 4; // numarul de coloane

int\*\*\* matrix = (int\*\*\*)malloc(mat \* sizeof(int\*\*));

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

{

matrix[i] = (int\*\*)malloc(row \* sizeof(int\*));

for (int j = 0; j < row; j++)

{

matrix[i][j] = (int\*)malloc(col \* sizeof(int));

}

}

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

{

for (int j = 0; j < row; j++)

{

for (int k = 0; k < col; k++)

{

matrix[i][j][k] = rand()%100;

}

}

}

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

{

for (int j = 0; j < row; j++)

{

for (int k = 0; k < col; k++)

{

printf("%d ", matrix[i][j][k]);

}

printf("\n");

}

printf("\n");

}

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

{

for (int j = 0; j < row; j++)

{

free(matrix[i][j]);

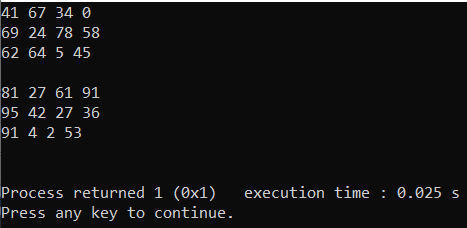
}

free(matrix[i]);

}

free(matrix);

}



**Rezultatul lucrării:**

* Ce reprezintă tipul de date pointer în limbajul C?

În limbajul C, pointerul reprezintă o variabilă, ce conţine adresa altei variabile.

Sintaxa de declarare a unui indicator este următoarea:

***tip \*nume;***

unde:

*tip* – este tipul de date a variabilei la care se poate referi pointerul;

*“\*”* - indică faptul că identificatorul ce urmează după el este un pointer;

*nume* – este un identificator ce desemnează numele variabilei de tip pointer.

* Cum se accesează adresa unei variabile în limbajul C?

Adresa unei variabile în limbajul C se accesează prin denumirea pointerului, fără a utiliza astericsul (fără a-l depointa).

* Ce operații se poate de aplicat asupra pointerilor?

Pointerul poate fi incrementat, decrementat, depointat, adunat, scăzut, comparat.

* Cu ce funcții se poate de alocat memorie dinamică în limbajul C?

În limbajul C este posibilă alocarea dinamică a memoriei utilizând funcțiile malloc, realloc, calloc.

* Cu ce funcție se poate de realocat memorie dinamică în limbajul C?

În limbajul C este posibilă realocarea dinamică a memoriei utilizând funcția realloc.

* Cu ce funcție se poate de eliberat memoria dinamică în limbajul C?

În limbajul C este posibilă eliberarea memoriei dinamice utilizând funcția free.

* În ce cazuri este utilizată alocarea dinamică?

Alocarea dinamică este utilizată în cazul în care dimensiunea fișierului este prea mare și este necesar de micșorat spațiul pentru o funcționalitate mai eficientă a programului și gestionarea eficientă a memoriei calculatorului.

* Rezultatul expresiei date ”a[i]==i[a]==\*(a+i)==\*(i+a)” este adevăr? Să se argumenteze răspunsul.

Rezultatul expresiei este adevăr, deoarece a[i] ne indică valoarea elementului i al matricei, iar i[a] este elementul la distanța de a elemente față de i; \*(a+i) și, respectiv, \*(i+a) sunt echivalentele expresiilor utilizând pointeri, iar compilatorul le citește la fel.