Ministerul Educaţiei, Culturii și Cercetării

al Republicii Moldova   
  
Universitatea Tehnică a Moldovei   
  
  
  
  
  
  
  
RAPORT

Lucrarea de laborator nr. 6   
la Programarea Calculatoarelor

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

UTM, Chișinău 2021

**Scopul lucrării**

Scopul lucrării este de a familiariza studentul cu utilizarea pointerilor și a alocării dinamice de memorie, pentru procesarea șirurilor de caractere, tablourilor unidimensionale și bidimensionale, utilizînd limbajul C.

**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>

int main()

{

int a = 10;

int \*p = &a;

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

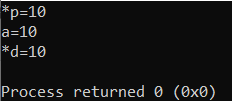
printf("a=%d\n",a);

int \*d = &\*p;

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

return 0;

}



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

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n,m,\*p,\*b,suma;

printf("Dati doua numere: \n");

scanf("%d%d",&n,&m);

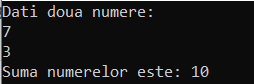
p=&n;

b=&m;

printf("Suma numerelor este: %d\n",\*p + \*b);

return 0;

}



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

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n,m,\*p,\*b,cons;

printf("Dati doua valori:\n");

scanf("%d%d",&n,&m);

p = &n;

b = &m;

printf("Inainte de schimbare:\n n=%d\n m=%d\n",n,m);

cons = \*p;

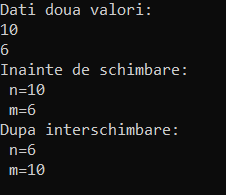
\*p = \*b;

\*b = cons;

printf("Dupa interschimbare:\n n=%d\n m=%d\n",n,m);

return 0;

}



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

#include <stdio.h>

#include <stdlib.h>

int main()

{

int arr[3][3] = {{2,4,6},{6,4,3},{6,9,7}};

int \*p = &arr;

printf("Matricea este:\n");

for(int i=0; i<3; i++)

{

for(int j=0; j<3; j++)

{

printf("%d ",\*((p + i\*3)+j));

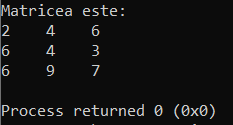
}

printf("\n");

}

return 0;

}



10. Scrieți un program C pentru a aduna două matrice folosind pointeri.

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i,j,ran,col,pos;

int arr[100][100],arr2[100][100],sum[100][100];

printf("Dati numarul de randuri:\n");

scanf("%d",&ran);

printf("Dati numrul de coloane:\n");

scanf("%d",&col);

printf("Dati prima matrice:\n");

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

{

for(j=0; j<= col - 1; j++)

{

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

}

}

printf("Dati a doua matrice:\n");

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

{

for(j=0; j<= col - 1; j++)

{

scanf("%d",(\*(arr2+i)+j));

}

}

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

{

for(j=0; j<=col - 1; j++)

{

\*(\*(sum + i)+j) = \*(\*(arr + i)+j) + \*(\*(arr2 + i)+j);

}

}

printf("Suma matricelor este:\n");

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

{

for(j=0; j<= col -1; j++)

{

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

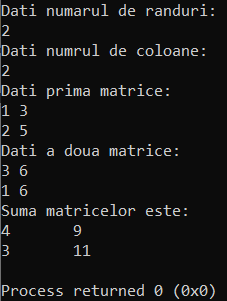
}

printf("\n");

}

return 0;

}



**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 str1(str2)

{

char \*str = malloc(sizeof(char)\*100);

strcpy(str,str2);

printf("\n%s\n",str);

free(str);

}

int main()

{

char str[100];

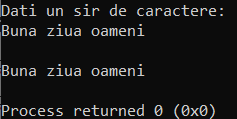
printf("Dati un sir de caractere:\n");

gets(str);

str1(str);

return 0;

}



4. Scrieți un program C pentru extinderea unei matrici de la dimensiunea NxM la dimensiunea KxP, unde N,M,K,P sunt variabile inițializate de la tastatură.

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

void \*\* create\_matrix(int n,int m,size\_t size)

{

void \*\* a = malloc(sizeof(void\*)\*n);

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

{

a[i] = malloc(size\*m);

}

return a;

}

void delete\_matrix(void \*\*a,int n)

{

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

{

free(a[i]);

}

free(a);

}

void init\_matrix(int \*\*a,int n,int m)

{

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

{

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

{

a[i][j] = rand()%50;

}

}

}

void extindere(int \*\*a,int \*\*b,int n,int m)

{

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

{

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

{

if(a[i][j]!=" ")

{

b[i][j] = a[i][j];

}

}

}

}

void extindere2(int \*\*b,int n, int m, int k,int p)

{

for(int i=0; i>n && i<k; i++)

{

for(int j=0; j>m && i<p; j++)

{

b[i][j] = 0;

}

}

}

void print\_matrix(int \*\*a,int n,int m)

{

printf("\nElementele matricii sunt:\n");

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

{

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

{

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

}

printf("\n");

}

}

int main()

{

int i,j,n,m,k,p;

printf("Dati numarul de randuri:\n");

scanf("%d",&n);

printf("Dati numrul de coloane:\n");

scanf("%d",&m);

int \*\*a = create\_matrix(n,m,sizeof(int));

init\_matrix(a,n,m);

print\_matrix(a,n,m);

printf("Dati numarul de randuri matricei 2:\n");

scanf("%d",&k);

printf("Dati numrul de coloane matricei 2:\n");

scanf("%d",&p);

int \*\*b = create\_matrix(k,p,sizeof(int));

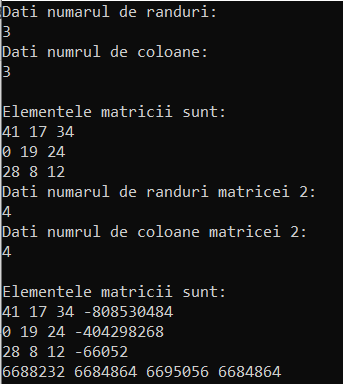
extindere(a,b,n,m);

extindere2(b,n,m,k,p);

print\_matrix(b,k,p);

return 0;

}



**Rezultatul lucrării:**

* ce reprezintă tipul de date pointer în limbajul C?

Tipul de date pointeri reprezinta o valoare de memorie in care se memoreaza o adresa de memorie.

* cum se accesează adresa unei variabile în limbajul C?

\*ptr = &variable

* ce operații se poate de aplicat asupra pointerilor?

Asupra pointelor se poate de aplicat o multimea de operatii:

1. Referentiere

2.Dereferentiere

3.Adunare,scadere

4.Incrementare,decrementare

5.Scaderea a doi pointeri

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

void \*(malloc(size\_t size);

void \*(calloc(size\_t memb, size\_t size);

* cu ce funcție se poate de reallocat memorie dinamică în limbajul C?

void \*realloc(void \*ptr, size\_t size);

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

void free(void \*ptr);

* în ce cazuri este utilizată alocarea dinamică?

Cand avem memorie limitata, astfel prin alocarea memoriei ne ramane mai multa memorie disponibila.

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

Da este adevarat, deoarece aici a este un pointer catre elementul i al matricei si a[i] este valoarea unui element care este la I elemente mai departe de a, care este acceasi ca \*(a+i).

Iar a[i] == i[a] deoarece aici este proprietatea comutativa.