Laborator 3
Tablouri de pointeri.Pointeri la pointeri.
Pointeri la funcții. Tablou de pointeri la functii

În acest capitol sunt prezentate consideraţii teoretice privind definirea şi utilizarea tablourilor de pointeri, a pointerilor la pointeri şi legătura dintre pointeri şi funcţii: pointeri ca argumente de funcții şi pointeri la funcţii. Sunt prezentate câteva probleme rezolvate cu aceste tipuri de pointeri.

Consideraţii teoretice

Tablourile de pointeri sunt tablouri care conţin ca și elemente pointeri, adică adrese de memorie.

Formatul de declarare al unui tablou de pointeri este:

- Pentru un **tablou unidimensional de pointeri:**

tip *nume_pointer[max]

unde max=dimesiunea maximă a șirului

- Pentru un **tablou bidimensional de pointeri:**

tip *nume_pointer[max1][max2]

unde max1/max2=nr maxim de elemente pe linie /coloană

Un tablou de pointeri se poate transmite unei funcții, în mod similar cu transmiterea numelui tabloului fără indici:

Ex.:

```
void afis_tab(int *q[])
{int i;
for (i=0;i<10;i++)
printf("%d", *q[i]);   }
// q nu este un pointer la intregi! q este pointer catre un tablou de pointeri la intregi
```

Pointeri la pointeri

Declararea unui pointer la pointer se realizează după următorul format:

tip **nume_pointer;

Pointeri ca argumente de funcții

În C transferul parametrilor este efectuat implicit prin valoare și reprezintă transfer sigur pentru că nu modifică parametri de apel. Dacă se dorește modificarea unei variabile parametru atunci trebuie transmisă funcției adresa variabilei, argumente= adrese, parametri formali=pointeri unde se vor copia aceste adrese .

Pointeri la funcții

Formatul de declarare al unui pointer la o funcție este:

tip (*nume_pointer) (lista_param_formali);

unde tip = tipul de bază al pointerului , poate fi void dacă nu returnează nici o valoare, sau unul din tipurile char, int, float, double.
nume = numele pointerului la funcție

Formatul de apel al funcției: **(*nume_pointer) (lista_param_efectivi);**

Chiar dacă o funcție nu este o variabilă, ea are totuși o localizare (adresă) în memorie ce poate fi atribuită unui pointer.

Adresa unei funcții se obține utilizând numele funcției fără paranteze și argumente (în mod analog cu numele tablourilor).

Probleme rezolvate

Ex.1. Programul definește un tablou de pointeri la caracter, care se inițializează cu zilele săptămânii și care afișează pe rând elementele tabloului și șirul de caractere corespunzând elementului al 6-lea din tabloul de pointeri.

Varianta in C

```
#include <stdio.h>
int main()
{char *zile[]={"Luni","Marti","Miercuri","Joi","Vineri","Sambata","Duminica"}; int i;
for(i=0;i<7;i++) printf("zile[%d]=%-10s;zile[%d]=%p\n",i,zile[i], i,zile[i]);
for(i=0;i<7;i++)  printf("*(zile[5]++)=%c\n", *(zile[5]++));
return 0;}
```

Varianta in C++

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{const char *zile[]={"Luni","Marti","Miercuri","Joi","Vineri","Sambata","Duminica"};int i;
for(i=0;i<7;i++) cout<<"zile["<<i<<"]="<<zile[i]<<setw(20)<<"    zile["<<i<<"]="<<(void*)zile[i]<<endl;
for(i=0;i<7;i++)  cout<<"*(zile[5]++)="<<*(zile[5]++)<<endl;
return 0;}
```

Rezultate:

```
zile[0]=Luni      ;zile[0]=00403024
zile[1]=Marti     ;zile[1]=00403029
zile[2]=Miercuri  ;zile[2]=0040302F
zile[3]=Joi       ;zile[3]=00403038
zile[4]=Vineri    ;zile[4]=0040303C
zile[5]=Sambata   ;zile[5]=00403043
zile[6]=Duminica  ;zile[6]=0040304B
*(zile[5]++)=S
*(zile[5]++)=a
*(zile[5]++)=m
*(zile[5]++)=b
*(zile[5]++)=a
*(zile[5]++)=t
*(zile[5]++)=a
```

Aplicație:

Să se modifice programul astfel încât să se afișeze șirul de caractere corespunzător elementului al 3-lea și al 7-lea din tabloul de pointeri.

Ex.2. Să se scrie un program în care se declară și inițializează un tablou de pointeri la șiruri de caractere și se afișează aceste șiruri în diverse moduri prin intermediul tabloului de pointeri.

Varianta in C		Varianta in C++

#include <stdio.h>
int main()
{char *nume[]={"Alex","Ion","Mircea",NULL};
printf("%c\n", *nume[0]);
printf("%c\n", *(nume[0]+1));
printf("%c\n", *(nume[0]+2));
printf("%c\n", *(nume[0]+3)); printf("\n");
printf("%c\n", *nume[1]);
printf("%c\n", *(nume[1]+1));
printf("%c\n", *(nume[1]+2)); printf("\n");
printf("%c\n", *nume[2]);
printf("%c\n", *(nume[2]+1));
printf("%c\n", *(nume[2]+2));
printf("%c\n", *(nume[2]+3));
printf("%c\n", *(nume[2]+4));
printf("%c\n", *(nume[2]+5)); printf("\n");
printf("%c\n", *(nume[1]+3));
printf("%s\n", *nume); printf("%s\n", *nume+1);
printf("%s\n", *(nume+1)); printf("%s\n", *(nume+2));
return 0;}
		#include <iostream>
		using namespace std;
		int main()
		{const char *nume[]={"Alex","Ion","Mircea",NULL};
		cout<<*nume[0]<<endl;
		cout<<*(nume[0]+1)<<endl;
		cout<<*(nume[0]+2)<<endl;
		cout<<*(nume[0]+3)<<endl<<endl;
		cout<<*nume[1]<<endl;
		cout<<*(nume[1]+1)<<endl;
		cout<<*(nume[1]+2)<<endl<<endl;
		cout<<*nume[2]<<endl;
		cout<<*(nume[2]+1)<<endl;
		cout<<*(nume[2]+2)<<endl;
		cout<<*(nume[2]+3)<<endl;
		cout<<*(nume[2]+4)<<endl;
		cout<<*(nume[2]+5)<<endl<<endl;
		cout<<*(nume[1]+3)<<endl;
		cout<<*nume<<endl; cout<<*nume+1<<endl;
		cout<<*(nume+1)<<endl; cout<<*(nume+2)<<endl;
		return 0;}

Rezultate:

```
A
l
e
x

I
o
n

M
i
r
c
e
a


Alex
lex
Ion
Mircea
```

Aplicaţie:

Să se modifice programul astfel încât să se afișeze *toate șirurile de caractere, caracter cu caracter printr-o buclă for, while, sau do while .*

Ex.3: Programul este un exemplu de declarare și afișare a unui pointer la pointer.

Varianta in C		Varianta in C++
#include <stdio.h> int main() { int x,*p,**q; x=10; p=&x; q=&p; printf("x=%d\n", x); printf("*p este pointer la variabila x\n"); printf("\n**q este pointer la variabila pointer *p\n"); printf("\nvaloarea stocata la adresa indicata de \n"); printf("pointerul p este adresa lui x:%p\n", *p); printf("valoarea stocata la adresa indicata de\n"); printf("pointerul q este chiar x:%d", **q); return 0;}		#include <iostream> using namespace std; int main() { int x,*p,**q; x=10; p=&x; q=&p; cout<<"x="<<x<<endl;cout<<"*p este pointer la variabila x"<<endl; cout<<"\n**q este pointer la variabila pointer *p"<<endl; cout<<"\nvaloarea stocata la adresa indicata de "<<endl; cout<<"pointerul p este adresa lui x:"<<(void*)*p<<endl; cout<<"valoarea stocata la adresa indicata de"<<endl; cout<<"pointerul q este chiar x:"<<**q<<endl;return 0;}

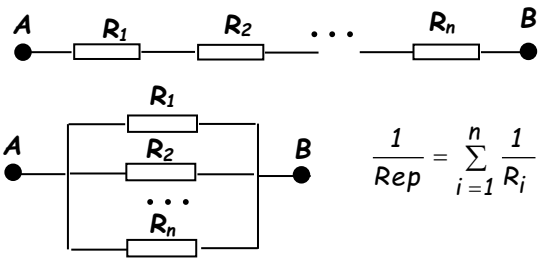
Rezultate:

```
x=10
*p este pointer la variabila x

**q este pointer la variabila pointer *p

valoarea stocata la adresa indicata de
pointerul p este adresa lui x:0000000A
valoarea stocata la adresa indicata de
pointerul q este chiar x:10
```

Aplicaţie:

Să se modifice programul astfel încât să se afişeze rezultatul expresiei $2x^2+1$ utilizând pointerul la pointer **q.

Ex.4: Programul realizează calculul şi afişarea valorii rezistenţei echivalente pentru conectarea în serie şi respectiv în paralel a rezistenţelor, a căror număr şi valori se citesc de la tastatură, utilizând un pointer la o funcţie.



$$Res = \sum_{i=1}^{n} R_i$$

$$\frac{1}{Rep} = \sum_{i=1}^{n} \frac{1}{R_i}$$

Varianta in C		Varianta in C++

#include <stdio.h>
#include <ctype.h>
#include <conio.h>
double calcul(double *, int, double(*)(double*,int));
double serie(double *,int); //calcul rezistenta serie
double paralel(double *,int); //calcul rezistenta paralel
int main()
{int i,n;
double rez[10],rec;
char c;
printf("cate rezistente sunt? n="); scanf("%d", &n);
printf("Valorile rezistentelor sunt:\n");
for (i=0;i<n;i++)
{printf("rez[%d]=", i); scanf("%lf",&rez[i]); }
printf("legare in serie(s) sau paralel(p):");
scanf("\n%c", &c);
if(toupper(c)=='S') rec=calcul(rez,n,serie);
else if(toupper(c)=='P') rec=calcul(rez,n, paralel);
printf("\nRezistenta echivalenta este:%lf", rec);
return 0;}
double calcul(double *rez,int n,double (*p)(double *,int))
{return(p(rez,n));}
double serie(double *rez, int n)
{double s=0.;
while(n) s+=rez[--n];
return(s);}
double paralel(double *rez, int n)
{double s=0.;
while(n)
s+=1./rez[--n];
return(1./s);
}
		#include <iostream>
		using namespace std;
		double calcul(double *, int, double(*)(double*,int));
		double serie(double *,int); //calcul rezistenta serie
		double paralel(double *,int); //calcul rezistenta paralel
		int main()
		{int i,n;
		double rez[10],rec;
		char c;
		cout<<"cate rezistente sunt? n="; cin>>n;
		cout<<"Valorile rezistentelor sunt:"<<endl;
		for (i=0;i<n;i++)
		{cout<<"rez["<<i<<"]="; cin>>rez[i]; }
		cout<<"legare in serie(s) sau paralel(p):";
		cin>>c;
		if(toupper(c)=='S') rec=calcul(rez,n,serie);
		else if(toupper(c)=='P') rec=calcul(rez,n, paralel);
		cout<<endl<<"Rezistenta echivalenta este:"<<rec;
		return 0;}
		double calcul(double *rez,int n,double (*p)(double *,int))
		{return(p(rez,n));}
		double serie(double *rez, int n)
		{double s=0.;
		while(n) s+=rez[--n];
		return(s);}
		double paralel(double *rez, int n)
		{double s=0.;
		while(n)
		s+=1./rez[--n];
		return(1./s);
		}

Rezultate:

```
cate rezistente sunt? n=4
Valorile rezistentelor in ohmi sunt:
rez[0]=1
rez[1]=1
rez[2]=1
rez[3]=1
legare in serie(s) sau paralel(p):s

Rezistenta echivalenta este:4.000000
```

```
cate rezistente sunt? n=4
Valorile rezistentelor in ohmi sunt:
rez[0]=1
rez[1]=1
rez[2]=1
rez[3]=1
legare in serie(s) sau paralel(p):p

Rezistenta echivalenta este:0.250000
```

Aplicaţie:

Să se modifice programul astfel încât să se calculeze și afișeze valorile capacităților unor condensatoare conectate în serie .

Ex.5: Programul afişează valorile a 3 funcţii trigonometrice: sin(π/6), cos(π/6), tg(π/4) utilizând pointeri la funcţiile sin(), cos() si tan() (din biblioteca <math.h>).

Varianta in C		Varianta in C++

#include <stdio.h>
#include <math.h>
#define PI 3.1415926
int main()
{double (*ps)(double), (*pc)(double), (*pt)(double);
ps=sin; pc=cos; pt=tan;
printf("\n\t pointer la sin = %p", ps);
printf("\n\t pointer la cos = %p", pc);
printf("\n\t pointer la tg = %p", pt);
printf("\n\napelul functiilor trigonometrice\nprin pointeri la aceste functii\n");
printf("\n\t sin(pi/6) = %.2lf", (*ps)(PI/6));
printf("\n\t cos(pi/6) = %.2lf", (*pc)(PI/6));
printf("\n\t tg(pi/4) = %.2lf\n", (*pt)(PI/4));
return 0;}
		#include <iostream>
		#include <math.h>
		#define PI 3.1415926
		using namespace std;
		int main()
		{double (*ps)(double), (*pc)(double), (*pt)(double);
		ps=sin; pc=cos; pt=tan;
		cout<<"\n\t pointer la sin = "<<(void*)ps<<endl;
		cout<<"\t pointer la cos = "<<(void*)pc<<endl;
		cout<<"\t pointer la tg = "<<(void*)pt<<endl;
		cout<<"\napelul functiilor trigonometrice"<<endl<<"prin pointeri la aceste functii"<<endl;
		cout<<"\t sin(pi/6) = "<<(*ps)(PI/6)<<endl;
		cout<<"\t cos(pi/6) = "<<(*pc)(PI/6)<<endl;
		cout<<"\t tg(pi/4) = "<<(*pt)(PI/4)<<endl;
		return 0;}

Rezultate:

```
        pointer la sin = 00401C40
        pointer la cos = 00401C48
        pointer la tg  = 00401C50

apelul functiilor trigonometrice
prin pointeri la aceste functii

        sin(pi/6) = 0.50
        cos(pi/6) = 0.87
         tg(pi/4) = 1.00
```

Aplicaţie:

Să se modifice programul astfel încât să se utilizeze și pointeri la alte funcții din biblioteca <math.h>

Ex.6: Programul compară 2 şiruri de caractere utilizând un pointer la funcţia strcmp din biblioteca <string.h>

Varianta in C		Varianta in C++
#include <stdio.h> #include <string.h> void compara(char *a,char *b, int(*comp)(const char *, const char *));		#include <iostream> #include<stdio.h> #include<string.h> using namespace std;

```
int main()
{char s1[80],s2[80];
int (*p) (const char *, const char *);
p=strcmp;
printf("Introduceti sirul 1:");  gets(s1);
printf("Introduceti sirul 2:");  gets(s2);
compara(s1,s2,p);
return 0;}
void compara(char *a,char *b, int(*comp)(const char *,
const char *))
{printf("Testeaza egalitatea dintre siruri:\n");
if(!(*comp)(a,b)) printf("SIRURILE SUNT EGALE!\n");
else printf("SIRURILE SUNT DIFERITE!\n");
}
```

```
void compara(char *a,char *b, int(*comp)(const char *,
const char *));
int main()
{char s1[80],s2[80];
int (*p) (const char *, const char *);p=strcmp;
cout<<"Introduceti sirul 1:";          gets(s1);
cout<<"Introduceti sirul 2:";  gets(s2);
compara(s1,s2,p);
return 0;}
void compara(char *a,char *b, int(*comp)(const char *,
const char *))
{cout<<"Testeaza egalitatea dintre siruri:"<<endl;
if(!(*comp)(a,b)) cout<<"SIRURILE SUNT EGALE!"<<endl;
else cout<<"SIRURILE SUNT DIFERITE!"<<endl; }
```

Rezultate:

```
Introduceti sirul 1:Programare C/C++
Introduceti sirul 2:Programare C
Testeaza egalitatea dintre siruri:
SIRURILE SUNT DIFERITE!
```

```
Introduceti sirul 1:Programare C/C++
Introduceti sirul 2:Programare C/C++
Testeaza egalitatea dintre siruri:
SIRURILE SUNT EGALE!
```

Aplicaţie:

Să se modifice programul astfel încât să se utilizeze pointeri şi la alte funcţii din biblioteca <string.h>, de ex: la strcat(), strlen(), strchr(), strstr(), etc.

Ex.7. Programul realizează sortarea în ordine alfabetică a unui tablou de pointeri la şiruri de caractere. Şirurile de caractere reprezintă numele şi prenumele unor persoane.

Varianta in C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void sort(char*a[],int n);
int main()
{char *sir[10]={"Popescu Alin","Morar Calin","Arcan Silviu","Zidane Zinedin"};
int i;
sort(sir,4);
for (i=0;i<4;i++)  printf("%s\n",sir[i]);
return 0;}
void sort(char*a[],int n)
{int i,j; char *aux;
for (i=0;i<n;i++)
for (j=0;j<n;j++)
  if(strcmp(a[i],a[j])<0)   {aux=a[i];   a[i]=a[j]; a[j]=aux;}}
```

Varianta in C++

```
#include <iostream>
#include<string.h>
using namespace std;
void sort(const char*a[],int n);
int main()
{ const char *sir[10]={"Popescu Alin","Morar Calin","Arcan Silviu","Zidane Zinedin"};
int i;
sort(sir,4);
for (i=0;i<4;i++) cout<<sir[i]<<endl;
return 0;}
```

```
void sort(const char*a[],int n)
{int i,j; const char *aux;
for (i=0;i<n;i++)
for (j=0;j<n;j++)
  if(strcmp(a[i],a[j])<0)   {aux=a[i];   a[i]=a[j]; a[j]=aux;}}
```

Rezultate:

```
Arcan Silviu
Morar Calin
Popescu Alin
Zidane Zinedin
```

Aplicaţie:

Să se modifice programul astfel încât iniţializarea şirurilor să se realizeze prin citire de la tastatură .

Ex.8. Programul calculeaza si afiseaza valoarea sin(x), unde x real citit de la tastatura prin apelul functiei sin() din <math.h> si prin descompunerea in serie Taylor:

$$sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^{2i+1}}{(2i+1)!}$$

Varianta in C

```
#include <stdio.h>
#include <math.h>
double fact(int j)
{double t; int k; t=1;
for (k=1;k<=j;k++)t=t*k;return t;}
int main()
{int i,n; double x,fx;
double (*p)(double, double); p=pow;
double (*q) (int); q=fact;
printf("x=");scanf("%lf",&x);
printf("n=");scanf("%d",&n);
fx=x;
for (i=1;i<=n;i++)
 //fx+=pow(-1,i)*pow(x,(2*i+1))/fact(2*i+1);
 fx+=(*p)(-1,i)*(*p)(x,(2*i+1))/(*q)(2*i+1);
printf("Val functiei sin(x) calculata\ncu descompunerea in
serie este:\n");
printf("sin1(%lf)=%.20lf\n", x, fx);
printf("\nval functiei sin(x) calculata\ncu functia sin din
<math.h> este:\n");
printf("sin2(%lf)=%.20lf\n", x ,sin(x));
return 0;}
```

Varianta in C++

```
#include <iostream>
#include <math.h>
using namespace std;
double fact(int j)
{double t; int k; t=1;
for (k=1;k<=j;k++)t=t*k;return t;}
int main()
{int i,n; double x,fx;
double (*p)(double, double); p=pow;
double (*q) (int); q=fact;
cout<<"x=";cin>>x;
cout<<"n=";cin>>n;
fx=x;
for (i=1;i<=n;i++)
 //fx+=pow(-1,i)*pow(x,(2*i+1))/fact(2*i+1);
 fx+=(*p)(-1,i)*(*p)(x,(2*i+1))/(*q)(2*i+1);
cout<<"Val functiei sin(x) calculata\ncu descompunerea
in serie este:"<<endl;
cout<<"sin1("<<x<<")="<<fx<<endl;
cout<<endl<<"val functiei sin(x) calculata"<<endl<<"cu
functia sin din <math.h> este:"<<endl;
cout<<"sin2("<<x<<")="<<sin(x)<<endl;
return 0;}
```

Rezultate:

```
x=0.5
n=5
Val functiei sin(x) calculata
cu descompunerea in serie este:
sin1(0.500000)=0.47942553860418341000

val functiei sin(x) calculata
cu functia sin din <math.h> este:
sin2(0.500000)=0.47942553860420301000
```

Aplicaţie:

Să se modifice programul astfel încât sa se calculeze cos(x), cu descompunerea in serie Taylor:

$$\cos(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!}$$

Ex.9. Programul utilizeaza un tablou de pointeri la functii predefinite din <math.h> prin care

se calculeaza si afiseaza valorile: e^3, log(10), $\sqrt{9}$, sin(π/6), cos(π/6):

Varianta in C		Varianta in C++

#include <stdio.h>
#include <math.h>
#define pi 3.14
int main ()
{double (*tabfun[])(double) = {sin, cos, exp, log, sqrt};
printf("exp(3)=%lf", (*tabfun[2])(3));
printf("\nlog(10)=%lf", (*tabfun[3])(3));
printf("\nsqrt(9)=%lf", (*tabfun[4])(9));
printf("\nsin((pi/6)=%lf", (*tabfun[0])(pi/6));
printf("\ncos((pi/6)=%lf", (*tabfun[1])(pi/6));
return 0;}
		#include <iostream>
		#include <math.h>
		#define pi 3.14
		using namespace std;
		int main ()
		{double (*tabfun[])(double) = {sin, cos, exp, log, sqrt};
		cout<<"exp(3)="<<(*tabfun[2])(3);
		cout<<"\nlog(10)="<<(*tabfun[3])(3);
		cout<<"\nsqrt(9)="<<(*tabfun[4])(9);
		cout<<"\nsin((pi/6)="<<(*tabfun[0])(pi/6);
		cout<<"\ncos((pi/6)="<<(*tabfun[1])(pi/6);
		return 0;}

Rezultate:

```
exp(3)=20.085537
log(10)=1.098612
sqrt(9)=3.000000
sin((pi/6)=0.499770
cos((pi/6)=0.866158
```

Aplicaţie:

Să se modifice programul astfel încât sa se calculeze valoarea absoluta a lui x cu functia abs(x)

Probleme propuse

1. ***Să se scrie programul în care se definește un tablou de pointeri la șiruri de caractere, care se inițializează cu numele lunilor anului și se afișează pe rând elementele tabloului precum și numărul de zile corespunzătoare fiecărei luni calendaristice.***

2. ***Să se scrie un program în care se utilizează pentru calculul valorilor unor funcţii matematice, un tablou de pointeri la aceste funcţii predefinite: exp(), abs(), log(), pow(), sqrt(),round(), etc.***

3. ***Să se scrie un program care realizează sortarea mai multor șiruri de numere reale prin intermediul unui tablou de pointeri.***

4. ***Să se scrie un program care declara un tablou de pointeri la functii predefinite din <math.h> prin care se calculeaza si afiseaza valorile unor functii pentru cateva valori constante: asin(), acos(), atan(), atan2(), sinh(), cosh(), tanh().***