Laborator 1
Pointeri . Operatii cu pointeri

În acest capitol sunt prezentate consideraţii teoretice privind definirea şi utilizarea pointerilor fiind prezentate câteva probleme rezolvate cu pointeri la date de tip întreg, la tipul caracter şi la şir de caractere.

CONSIDERAŢII TEORETICE

Variabila de tip **pointer** este o variabilă care are ca şi valoare o adresă de memorie.

Pointerii se clasifică astfel:
- **Pointer la date** = conţine adresa unei variabile sau a unei constante
- **Pointer la funcţii** = conţine adresa codului executabil al unei funcţii
- **Pointer la obiecte** = conţine adresa unui obiect în memorie = adrese de date şi funcţii

Pointerii conţin adrese de memorie şi nu valori ca alte tipuri de variabile sau constante.

Dacă la o adresă de memorie se află altă adresă acest lucru are semnificaţia de indirectare ("pointare") (o variabilă indică spre alta).

În Fig.1.1 este prezentat un exemplu de adrese de memorie în care sunt memorate date sau alte adrese ale altor date .

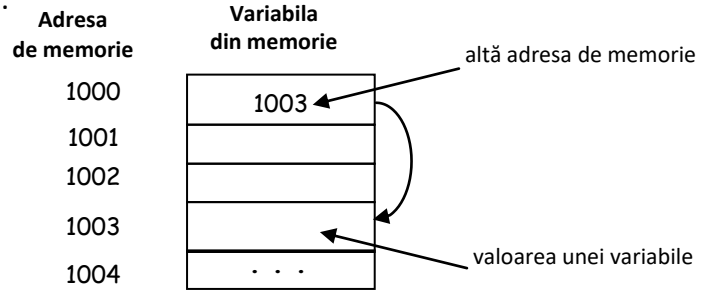

Fig.1.1. Reprezentarea variabilelor în memorie

Formatul de declarare al unei variabile de tip **pointer** este : **tip *nume ;**

unde **tip**= tipul de baza al pointerului, defineşte tipul variabilei la care indică acesta;
poate fi orice tip de variabilă;
nume= numele variabilei pointer

Operatorii specifici pointerilor sunt : **&, *.**

Operatorul de adresare (referentiere): & ("adresa lui") este un operator unar care asociat unei variabile sau obiect, returnează adresa de memorie a acelei variabile sau obiect.

Operatorul de indirectare (dereferentiere): * ("de la adresa") este un operator unar, complementar lui &, returnează valoarea înregistrata la adresa de memorie specificată.
Observatii:

- Operatorul * nu trebuie confundat cu operatorul aritmetic utilizat pentru înmulţire (*). Operatorul & nu trebuie confundat cu operatorul AND pentru biţi (&) .
- Operatorii * şi & au prioritate faţă de toţi operatorii aritmetici (cu exceptia lui –unar cu care au aceeasi precedenţă).

După declararea pointerului trebuie specificat către ce anume indică acesta (o alta variabila de tip data sau un alt pointer) deoarece implicit nu indică nimic.

Ex.1: Instrucţiunile din paragraful INCORECT vor genera eroare , deoarece variabila de tip pointer nu a fost iniţializată.

INCORECT:
```
int *ip;
*ip = 100;
```

CORECT:
```
int *ip,  x;
ip = &x;
*ip = 100;
```

Ex.2: Un alt exemplu de declarare şi iniţializare de pointer

- Prin 2 instrucţiuni distincte:

```
int a=1, *ptoa;
ptoa = &a;       //pointerului ptoa i se atribuie adresa variabilei  a
```

- Printr-o singură instrucţiune:

```
int a=1, *ptoa= &a;  //pointerului ptoa i se atribuie adresa variabilei  a
```

Pentru exemplul anterior în Fig.1.2. este reprezentat modul de definire a pointerului *ptoa într-o zonă de memorie. Adresele de memorie sunt reprezentate prin numere în baza 16 (0x00,…, 0xFF).

Pointerul se numeste *ptoa, este iniţializat cu adresa &a, iar locaţia de memorie care este rezervată lui *ptoa va conţine numărul 1, adicăvaloarea variabilei a.

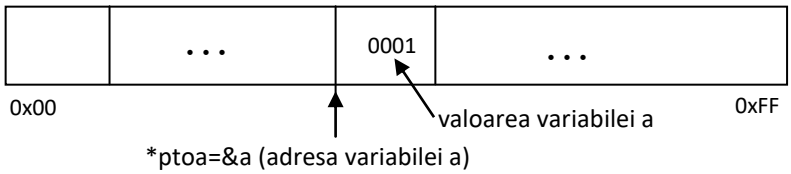

Fig.1.2. Reprezentarea unui pointer în memorie prin adresă şi conţinut

Operațiile permise asupra pointerilor sunt:

1. Comparare
2. Iniţializare. Atribuire
3. Adunarea/Scăderea unui nr. întreg la/dintr-un pointer
4. Scăderea a 2 pointeri
5. Incrementare/Decrementare

Operațiile nepermise asupra pointerilor sunt:

- Adunarea a 2 sau mai mulţi pointeri
- Adunarea/scăderea tipului float sau double la/din pointeri
- Înmulţirea a 2 sau mai mulţi pointeri
- Împărţirea a 2 sau mai mulţi pointeri
- Aplicarea operatorilor pe bit pointerilor

1.Compararea pointerilor se poate realiza în două moduri:

a) Compararea a doi pointeri

Operatorii relaţionali permit compararea a 2 pointeri într-o expresie numai dacă aceştia indică spre obiecte de acelaşi tip.

Ex.:

```
int *p, *q, k,j; p=&k;    q=&j;
if (p<q) printf ("p indica o adresa de memorie mai mica decit q");
printf("p=%p q=%p\n",p,q );
```

b) Comparare pointeri cu NULL

Operatorii de egalitate = = și ! = permit compararea pointerilor cu constanta NULL (definită în <stdio.h>)

Ex.:

```
#define NULL 0
void *p //p=pointer generic,nu este asociat nici unui tip de date
p==NULL;
p!=NULL //verifica daca lui p ii este asociata sau nu o valoare
```

În C++ se recomandă utilizarea comparației cu 0 a pointerilor nu cu NULL : **Ex.:** p= =0 si p!=0

2. Iniţializarea pointerilor se realizează utilizând următorul format: **tip *nume =const;**

unde **tip** = tipul de bază al pointerului, defineşte tipul variabilei la care indică acesta;
nume = numele variabilei pointer
const = expresie cu valoare constantă

Operatorul de atribuire "=" permite inițializarea și atribuirea de valori unui pointer, precum în exemplul de mai jos.

Ex.:

```
p=NULL;          //initializare
*p=2;            //atribuire
 p=&nr; //atribuire
```

3.Adunarea/scăderea unui nr. întreg la/dintr-un pointer

Se considera un pointer *p declarat astfel: tip *p;
Adunarea/scăderea unui număr întreg la/din acest pointer , p+n și p-n reprezintă adunarea/scăderea la adresa indicată de p a valorii **n*sizeof(tip).**

Ex.: p=p+10; // p va indica la al 10-lea element de acelasi tip cu p

4.Scăderea a doi pointeri

Se considera doi pointeri declarați astfel: tip *p1,*p2; Scăderea acestor doi pointeri, p1-p2 este permisă numai pentru elemente de același tip, rezultatul reprezentând nr. de obiecte dintre cei doi pointeri..

Ex.:

```
int q; float j[3], *p1=&j[1],*p2=&j[3];
q=p2-p1; //variabilei q i se atribuie nr.de obiecte dintre cele doua adrese:2
```

5.Incrementarea/decrementarea pointerilor

Incrementarea/decrementarea nu reprezintă adunarea/ scăderea propriuzisă cu 1 (nici a adresei nici a valorii spre care indică pointerul respectiv). Incrementarea/decrementarea semnifică adunarea/ scăderea cu sizeof (tip), unde tip este tipul pointerului respectiv. După incrementarea/decrementarea unui pointer

acesta va indica spre elementul următor/anterior de același tip cu tipul său de bază.

În exemplele prezentate mai jos este ilustrată incrementarea și decrementarea unui pointer .
Ex.1: Dacă p1 are valoarea 2000, atunci p1++ are valoarea 2004 (nu 2001, pentru că int se reprezintă pe 4 octeți) și deci va indica spre următorul întreg .
int *p1; p1++;
Ex.2 : Daca p1 are valoarea 2000, p1-- are valoarea 1996
int *p1; p1--;

PROBLEME REZOLVATE

Ex.1: Programul este un exemplu de declarare a 2 pointeri la date de tip int si respectiv real (float). Programul citeste de la tastatura o valoare intreaga x si o valoare de tip float corespunzatoare lui y si apoi afiseaza adresele variabilelor x si y si valorile lor si ale patratelor lor prin pointeri.

Varianta in C		Varianta in C++

#include <stdio.h>
#include <stdlib.h>
int main()
{int x, *p1;float y,*p2;
p1=&x; p2=&y;
printf("Introduceti o valoare intreaga x=");
scanf("%d", &x);
printf("Introduceti o valoare reala y="); scanf("%f", &y);
printf("\nAdresa lui x este:\n");
printf("p1=%p\n", p1);printf("\nValoarea lui x este:\n");printf("x=*p=%d\n",*p1);
printf("\nValoarea lui x^2 este:\n");
printf("*p1^2=%d\n", *p1* *p1);
printf("\nAdresa lui y este:\n");
printf("p1=%p\n", p2);
printf("\nValoarea lui y este:\n");
printf("x=*p2=%.2f\n",*p2);
printf("\nValoarea lui y^2 este:\n");
printf("*p1^2=%.2f\n", *p2* *p2);
return 0;}
		#include <iostream>
		using namespace std;
		int main()
		{int x, *p1;float y,*p2;
		p1=&x; p2=&y;
		cout<<"Introduceti o valoare intreaga x=";
		cin>>x ;
		cout <<"\nIntroduceti o valoare reala y=";
		cin>>y;
		cout<<"\nAdresa lui x este:"<<p1;
		cout<<"\nValoarea lui x este:"<<*p1;
		cout<<"\nValoarea lui x^2 este:"<<*p1**p1;
		cout<<"\nAdresa lui y este:"<<p2;
		cout<<"\nValoarea lui y este:"<<*p2;
		cout<<"\nValoarea lui y^2 este:"<<*p2* *p2;
		return 0;}

Rezultate:

```
Introduceti o valoare intreaga x=2
Introduceti o valoare reala y=5.5

Adresa lui x este:
p1=0060FF04
Valoarea lui x este:
x=*p=2
Valoarea lui x^2 este:
*p1^2=4
Adresa lui y este:
p1=0060FF00
Valoarea lui y este:
x=*p2=5.50
Valoarea lui y^2 este:
*p1^2=30.25
```

Aplicatie:

Să se modifice programul de mai sus astfel încât, utilizând pointeri, să se afişeze rezultatul calculului expresiei: $x^2 + 3y^2$.

Ex.2 Programul este un exemplu de utilizare a pointerilor la date de tip caracter.

Varianta in C		Varianta in C++

#include <stdio.h>
int main()
{char c = 'A'; char *p = &c;
printf("Initial variabila c este %c\n",c);
printf("Continutul variabilei c se poate afisa utilizand\n");
printf("a)variabila c sau b) pointerul *p=&c\n");
printf("astfel a) c=%c b)*p=%c\n", c, *p);
printf("\nSchimbarea valorii initiale se poate realiza:\n");
printf("a) prin variabila sau b)prin pointer\n");
printf("Introduceti un caracter:"); scanf("%c", p);
printf("astfel c=%c *p=%c\n", c, *p);
return 0;}
		#include <iostream>
		using namespace std;
		int main()
		{char c = 'A'; char *p = &c;
		cout<<"Initial variabila c este "<<c<<endl;
		cout<<"Continutul variabilei c se poate afisa utilizand" <<endl;
		cout<<"a)variabila c sau b) pointerul *p=&c"<<endl;
		cout<<"astfel a)c="<<c<< "b)*p="<<*p<<endl;
		cout<<"\nSchimbarea valorii initiale se poate realiza:"<<endl;
		cout<<"a) prin variabila sau b)prin pointer"<<endl;
		cout<<"Introduceti un caracter:";cin>>*p;
		cout<<"astfel c="<<c<<" *p="<<*p;return 0;}

Rezultate:

```
Initial variabila c este A
Continutul variabilei c se poate afisa utilizand
a)variabila c sau b) pointerul *p=&c
astfel a) c=A b)*p=A

Schimbarea valorii initiale se poate realiza:
a) prin variabila sau b)prin pointer
Introduceti un caracter:B
astfel c=B *p=B
```

Aplicatie:

*Sa se tipareasc̆a şi adresa variabilei c utilizand pointerul *p.*

Ex.3. Programul calculeaza si afiseaza rezultatele unor expresii cu numere intregi prin intermediul pointerilor. Initial se considera i_1=8 si se cere sa se afiseze $i_2=i_1/2+1$ direct si prin intermediul pointerului *p1, apoi se va vor afisa rezultatele operatiilor aritmetice i_1+i_2, i_1-i_2, i_1*i_2, i_1/i_2 prin pointeri.

Varianta in C		Varianta in C++

#include <stdio.h>
int main()
{int i1=8, i2, *p1, *p2;
p1 = &i1; p2 = &i2;
printf("daca i1=8 si p1=&i1\n");
printf("atunci operatiile sunt echivalente:\n");
printf("i2=i1/2+1=%d <=>i2=*p1/2+1=%d\n", i1/2+1,*p1/2+1);i2=*p1/2+1;
printf("i1=%d, i2=%d\n", *p1, *p2);
printf("operatiile aritmetice se pot realiza in 2 moduri:\n");
printf("a)i1+i2=%d,i1-i2=%d,i1*i2=%d, i1/i2=%d\n", i1+i2,i1-i2,i1*i2,i1/i2);
printf("b)*p1+*p2=%d,*p1-*p2=%d, *p1**p2=%d, *p1/*p2=%d\n", *p1+*p2, *p1-*p2, *p1**p2,*p1/(*p2));
return 0;}
		#include <iostream>
		using namespace std;
		int main()
		{int i1=8, i2, *p1, *p2;
		p1 = &i1; p2 = &i2;
		cout<<"daca i1=8 si p1=&i1"<<endl;
		cout<<"atunci operatiile sunt echivalente:"<<endl;
		cout<<"i2=i1/2+1="<<i1/2+1<<" <=> i2=*p1/2+1="<<*p1/2+1<<endl;i2=*p1/2+1;
		cout<<"i1="<<*p1<<", i2="<<*p2<<endl;
		cout<<"operatiile aritmetice se pot realiza in 2 moduri:"<<endl;
		cout<<"a)i1+i2="<<*p1+*p2<<", i1-i2="<<i1+i2<<", i1*i2="<<i1*i2<<", i1/i2="<<i1/i2<<endl;
		cout<<"b)*p1+*p2="<<*p1+*p2<<", *p1-*p2="<< *p1-*p2<<", *p1**p2="<<*p1**p2<<", *p1/*p2="<<*p1/(*p2)<<endl;return 0;}

Rezultate:

```
daca i1=8 si p1=&i1
atunci operatiile sunt echivalente:
i2=i1/2+1=5 <=>i2=*p1/2+1=5
i1=8, i2=5
operatiile aritmetice se pot realiza in 2 moduri:
a)i1+i2=13,i1-i2=3,i1*i2=40, i1/i2=1
b)*p1+*p2=13,*p1-*p2=3, *p1**p2=40, *p1/*p2=1
```

Aplicatie:

Să se modifice programul astfel încât sa se afişeze şi adresele variabilelor i1 şi i2

Ex.4. In program se considera 4 variabile: de tip int, float, double si char carora le corespunde cate un pointer:*a,*c,*d, *e. Programul realizeaza diverse operatii cu pointeri.

Varianta in C		Varianta in C++

#include <stdio.h>
#include <stdlib.h>
int main()
{int i=1, *a,*a1;
float x=1.,*c,*c1;
double pi=3.14, *d,*d1;
char ch='A',*e,*e1;a=&i;
printf("i=variabila de tip int,i=%d\n",i);
printf("a=variabila de tip pointer la int,a=&i=%p\n",a);
printf("a-1=%p a+1=%p\n", a-1,a+1);
printf("sizeof(int):%d\n",sizeof(int));
printf("comparare a si a+1: ");
if (a<(a+1)) printf("%p < %p\n",a,a+1);
else printf("%p > %p\n",a,a+1);
a1=a+3;
printf("a1=a+3=%p\n", a1);
printf("a1-a=%d\n", a1-a);
printf("----------\n");
c=&x;
printf("x=variabila de float,x=%f\n",x);
printf("c=variabila de tip pointer la float,
c=&x=%p\n",c);
printf("c-1=%p c+1=%p\n", c-1,c+1);
printf("sizeof(float):%d\n",sizeof(float));
c1=c+2;
printf("c+2=%p\n", c1);
printf("c1=c+2;c1-c=%d\n", c1-c);
printf("----------\n");
d=π
printf("pi=variabila de tip double,pi=%lf\n",pi);
printf("d=variabila de tip pointer la
double,d=&pi=%p\n",d);
printf("d-1=%p d+1=%p\n", d-1,d+1);
printf("sizeof(double):%d\n",sizeof(double));
d1=d-2;
printf("d-2=%p\n", d1);
printf("d1=d-2;d1-d=%d\n", d1-d);
printf("----------\n");
e=&ch;
		#include <iostream>
		using namespace std;
		int main()
		{int i=1, *a,*a1;
		float x=1.,*c,*c1;
		double pi=3.14, *d,*d1;
		char ch='A',*e,*e1;a=&i;
		cout<<"i=variabila de tip int,i="<<i<<endl;
		cout<<"a=variabila de tip pointer la int,a=&i="<<a<<endl;
		cout<<"a-1="<<a-1<<", a+1="<<a+1<<endl;
		cout<<"sizeof(int):"<<sizeof(int)<<endl;
		cout<<"comparare a si a+1: ";
		if (a<(a+1)) cout<<a<<"<"<<a+1<<endl;
		else cout<<a<<">"<<a+1<<endl;a1=a+3;
		cout<<"a1=a+3="<<a1<<endl;
		cout<<"a1-a="<<a1-a<<endl;
		cout<<"----------"<<endl;
		c=&x;
		cout<<"x=variabila de float,x="<<x<<endl;
		cout<<"c=variabila de tip pointer la float,
		c=&x="<<c<<endl;
		cout<<"c-1="<<c-1<<" c+1="<<c+1<<endl;
		cout<<"sizeof(float):"<<sizeof(float)<<endl;c1=c+2;
		cout<<"c+2="<<c1<<endl;
		cout<<"c1=c+2;c1-c="<<c1-c<<endl;
		cout<<"----------"<<endl;
		d=π
		cout<<"pi=variabila de tip double,pi="<<pi<<endl;
		cout<<"d=variabila de tip pointer la
		double,d=&pi="<<d<<endl;
		cout<<"d-1="<<d-1<<" d+1="<<d+1<<endl;
		cout<<"sizeof(double):"<<sizeof(double)<<endl;d1=d-2;
		cout<<"d-2="<<d1<<endl;
		cout<<"d1=d-2;d1-d="<<d1-d<<endl;
		cout<<"----------"<<endl;
		e=&ch;
		cout<<"ch=variabila de tip char,ch="<<ch<<endl;
		cout<<"e=variabila de tip pointer la char,
		e=&ch="<<(void *)e<<endl;
		cout<<"e-1="<<(void *)(e-1)<<" e+1="<<(void
		*)(e+1)<<endl;
		cout<<"sizeof(char):"<<sizeof(char)<<endl;
		e1=e+3;

```
printf("ch=variabila de tip char,ch=%c\n",ch);
printf("e=variabila de tip pointer la char,
e=&ch=%p\n",e);
printf("e-1=%p e+1=%p\n", e-1,e+1);
printf("sizeof(char):%d\n",sizeof(char) );
e1=e+3;
printf("e+3=%p\n", e1);
printf("e1=e+3,e1-e=%d\n", e1-e);
return 0;}
```

```
cout<<"e+3="<<(void *)e1<<endl;
cout<<"e1=e+3,e1-e="<<e1-e<<endl;
return 0;}
```

Rezultate:

```
a-1=0060FEE8 a+1=0060FEF0
sizeof(int):4
comparare a si a+1: 0060FEEC < 0060FEF0
a1=a+3=0060FEF8
a1-a=3
--------------------------------------------------
x=variabila de float,x=1.000000
c=variabila de tip pointer la float,  c=&x=0060FEE8
c-1=0060FEE4 c+1=0060FEEC
sizeof(float):4
c+2=0060FEF0
c1=c+2;c1-c=2
--------------------------------------------------
pi=variabila de tip double,pi=3.140000
d=variabila de tip pointer la double,d=&pi=0060FEE0
d-1=0060FED8 d+1=0060FEE8
sizeof(double):8
d-2=0060FED0
d1=d-2;d1-d=-2
--------------------------------------------------
ch=variabila de tip char,ch=A
e=variabila de tip pointer la char,  e=&ch=0060FEDF
e-1=0060FEDE e+1=0060FEE0
sizeof(char):1
e+3=0060FEE2
e1=e+3,e1-e=3
```

Aplicație:

Să se modifice programul de mai sus astfel încât, utilizând pointeri, să se afişeze adresele: a+2, c1, d+2 si e+2.

PROBLEME PROPUSE

1. ***Sa se scrie un program care afiseaza solutia ecuatiei de gradul I unde coeficientii se citesc de la tastatura utilizand pointeri si operatii prin pointeri***

2. ***Sa se scrie un program care calculeaza si afiseaza valoarea functiei f(x) = x^2-1 utilizand pointeri si operatii prin pointeri.***