Universiteti i Prishtinës

Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Algoritmet dhe struktura e të dhënave

Vehbi Neziri

FIEK, Prishtinë 2015/2016

- Pointerët
  - Deklarimi
  - Adresat dhe vlerat e variablave
  - Operime përmes pointerëve
  - Pointerët në fusha
  - Pointerët në funksione
  - Pointerët në klasa dhe struktura

Pointerët

- Secili lokacion i memories i cili përdoret për të ruajtur vlera të të dhënave e ka një adresë.

- Pra, pointeri është një variabël që mban adresën e një variable tjetër.

- Emri i variablës i referohet vlerës direkt, kurse pointeri i referohet vlerës indirekt.

- Pointeri mund të mbajë adresën e
  - të dhënave primitive si: char, int, float, double …
  - të dhënave të përcaktuara nga përdoruesi, si funksione, pointer …
  - të dhëna të derivuara si fusha, struktura, grupe...

Deklarimi

- Pointerët, si edhe variablat e tipave të tjerë, para se të shfrytëzohen duhet të deklarohen.
- Deklarimi i pointerëve është pothuajse i njëjtë sikurse i variablave të zakonshme.
- Për t'i dalluar pointerët nga variablat e zakonshme, para pointerëve shënohet simboli *:

```
t  *p;
```

```
int  *a;
```

t – tipi i variablës p
p – emri i variablës së tipit pointer

- Tipi **t** i variablës **p** duhet të përputhet me tipin e variablës adresa e së cilës do të ruhet në të.

Shembull 3.1

- Brenda funksionit main të deklarohen disa variante të variablave të tipit pointer.

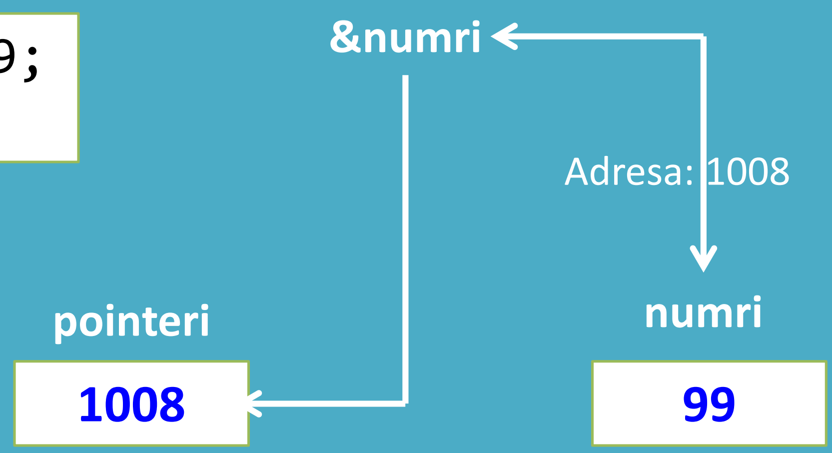
```cpp
#include <iostream>
using namespace std;

int main()
{
    //simboli * mund të shkruhet kudo
    //në mes tipit dhe emrit të pointerit
    int *a;
    int * b, *b1;
    int* c;
    float *d;
    float * e;
    double* f;
    char *g;
    int x, y, *h, *k;
    long* pointer, number=99;
    //lejohet inicimi vetem me zero dhe NULL
    int *y=0;
    int *z=NULL;
    //...
}
```

- Duke e shfrytëzuar operatorin **&**, përmes shprehjes së formës

```
p = &v;
```

- adresa e variablës së zakonshme **v** vendoset të variabla **p** e tipit pointer.

```
long* pointeri, numri=99;
pointeri=&numri;
```



Shembull 3.2

- Brenda funksionit main të deklarohet një variabël dhe një pointer. Më pas të merret dhe të shtypet adresa e variablës **v** duke e shfrytëzuar pointerin **p.**

```cpp
#include <iostream>
using namespace std;

int main()
{
    int v=5;
    int *p;
    cout << "\nVlera e variables v= "<< v
    << "\nAdresa e variables v ="<<&v;
    p=&v; //krijon nje reference tek v
    cout << "\nAdresa e variables v "
        <<"permes pointerit p: "
    << p
    << "\n\n";
    system("pause");
    return 0;
}
```

Vlera në adresën e variablës

- Nëse dihet adresa e një variable, përmes procesit të ***deadresimit*** ose ***dereferimit*** mund të merret vlera e variablës.

- Për këtë qëllim shfrytëzohet operatori *****.

```
v = *p;
```

- Vlera a variablës, adresa e së cilës është tek pointeri **p** do të ruhet në variablën **v.**

- Operatorët **&** dhe ***** mes vete llogariten si *operatorë inverzë*

Shembull 3.3

- Brenda funksionit main të tregohet shtypja e vlerës së variablës **a** dhe variablës **b**. Vlera e variablës **b** të shtypet duke shfrytëzuar adresën e cila ruhet tek pointeri **p**.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int *p;
    int a=200,b;
    cout << "\nVlera e variablës a: "
    << a << "\n";

    //adresa e variablës a ruhet tek pointeri p
    p=&a;

    cout<< "\nAdresa te variabla a: "
        << p << "\n";

    /*vlera e variablës, adresa e së cilës
    ruhet të pointeri p është ruajur tek
    variabla b.*/

    b=*p;

    cout << "\nVlera e variablës b: "
    << b << "\n\n";

    system("pause");
    return 0;
}
```

Shoqërimi i vlerave

- Përmes pointerëve, variablave të tipave të zakonshëm, mund të u shoqërohen vlera konstante ose vlera të variablave të tjera.

- Variablës, adresa e së cilës ruhet te pointeri **p**, mund t'i shoqërohet një vlerë konstante **k**.

```
*p = k;
```

- Variablës, adresa e së cilës ruhet te pointeri **p**, mund t'i shoqërohet vlera e një variable tjetër.

```
*p = v;
```

Shembull 3.4

- Brenda funksionit main të tregohet ndryshimi i vlerës së variablës **x** në mënyrë indirekte duke e shfrytëzuar adresën e pointerit **p1**.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int x = 2013;
    int *p1 = &x;

    cout<<"&x = "<< &x << "\t\tx   = "<< x
        <<"\np1 = "<< p1 << "\t\t*p1 = "<< *p1;

    *p1 += 25;

    cout << "\n&x = " << &x << "\t\tx   = " << x
        << "\np1 = " << p1 << "\t\t*p1 = " << *p1
        << "\n\n";

    system("pause");
    return 0;
}
```

Shembull 3.5

- Brenda funksionit main të tregohet vendosja e vlerës së variablës **x** tek variabla **y** në mënyrë indirekte duke e shfrytëzuar adresën e pointerit **p.**

```
#include <iostream>
using namespace std;

int main()
{
    int x=1999,y;
    int *p;

    cout<< "\nVlera e variablës x = "
        << x;

    /*Adresa e variablës y është ruajtur
    tek pointeri p.*/
    p=&y;

    cout << "\nAdresa e variablës x = "
        <<&x
        <<"\nAdresa e variablës y = "
        << p;
    /*Vlera e x është ruajtur në adresën
    që e cila është ruajtur tek pointeri p*/
    *p=x;

    cout << "\nVlera e variablës y = "
    << y << "\n\n";

    system("pause");
    return 0;
}
```

Llogaritjet përmes pointerëve

- Përveç variablave, mund të shfrytëzohen edhe pointerët për llogaritje të ndryshme.

```
if (x > y)
{
   z = *a + *b;
}
else
{
   z = *a - *b;
}
```

```
void rrethi (int r, float *p)
{
   *p = 3.14 * r * r ;
}
```

Shembull 3.6

- Të modifikohet programi i mëposhtëm dhe operacionet e mbledhjes dhe zbritjes të kryhen duke shfrytëzuar pointerët.

```cpp
int main()
{
    int x,y,z;

    cout << "\nx = ";
    cin >> x;
    cout << "\ny = ";
    cin >> y;

    if (x>y)
    {
        z=x+y;
    }
    else
    {
        z=x-y;
    }
    cout<< "\nz="<< z
        << "\n\n";

    system("pause");
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    int x,y,z,*a,*b;
    cout << "\nx = ";
    cin >> x;
    cout << "\ny = ";
    cin >> y;

    a=&x;
    b=&y;

    if (x>y)
    {
        z=*a+*b;
    }
    else
    {
        z=*a-*b;
    }

    cout << "\nz = "
    << z
    << "\n\n";

    system("pause");
    return 0;
}
```

Operimi me pointerë 1/2

- Vlerat e pointerëve mund të rriten, të zvogëlohen, të krahasohen, ose përmes operatorit të barazimit, ato vlera edhe mund t'u shoqërohen pointerëve të tjerë.

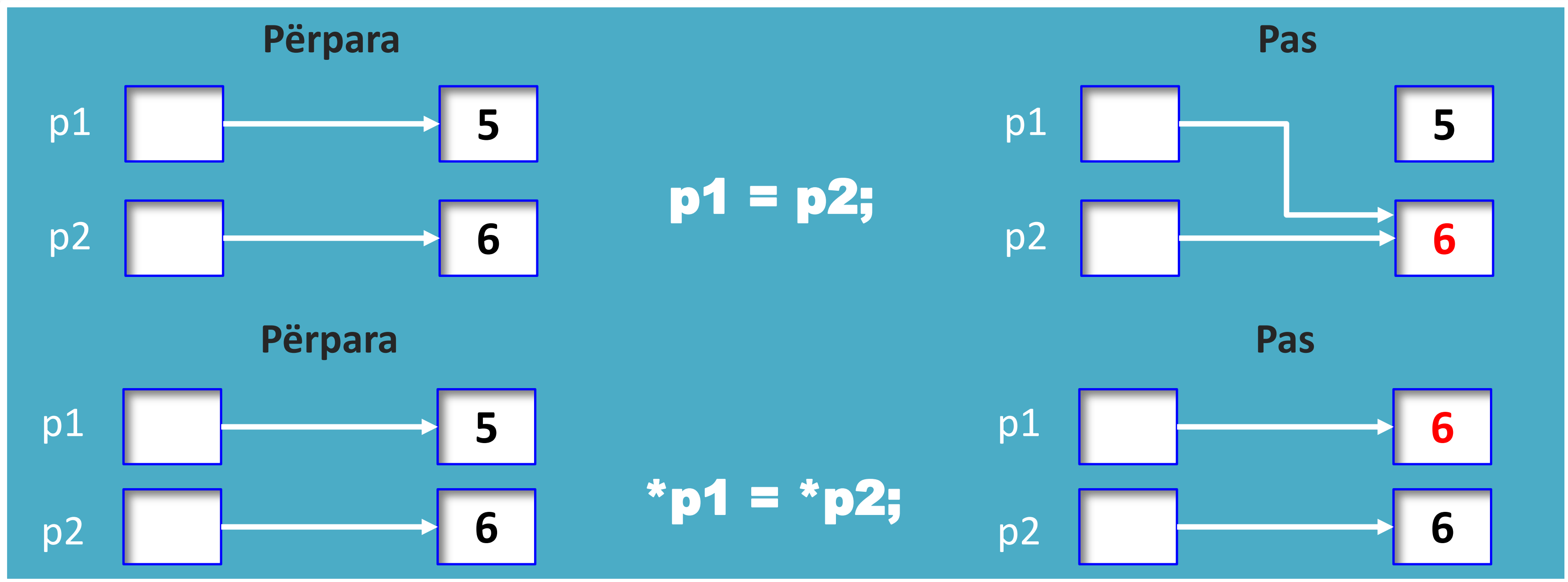
- Operacione të vlefshme me pointerë janë:

  - Mbledhja dhe zbritja e pointerit me një integer **($p \pm i$)**
  - Shoqërimi i pointerit të tipit të njëjtë (**$p = q$**)
  - Zbritja ose krahasimi i dy pointerëve në fushën e njëjtë **($p = q$)**
  - Caktimi ose krahasimi me zero **($p = 0$)**

Operimi me pointerë 2/2

- Nëse **p1** dhe **p2** janë variabla pointer, atëherë

p1 = p2;	ndryshon vlerën e **p1** dhe ajo tani është adresë e memories (pointer) në **p2**.
***p1 = *p2;**	ndryshon vlerën në adresën e memories e cila është e ruajtur në pointerin **p1**.



Shembull 3.7

- Brenda funksionit main të përdoret një unazë nga 1 deri në 10. Brenda unazës të rritet vlera e pointerit për 1 dhe të shtypet adresa e pointerit.

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    int i,x;
    int *p;

    p=&x;
    cout << "\n i  adresa (hex)   adresa(dec)"
    << "\n-----------------------------\n";
    for (int i=1;i<=10;i++)
    {
        cout << setw(2)
        << i
        << setw(10)
        << p
        << setw(15)
        << (long)p //adresa në decimal
        << "\n";
        p++;
    }
    cout << "-----------------------------\n";
    system("pause");
    return 0;
}
```

Pointerët në fusha 1/2

- Fushat numerike janë të lidhura ngushtë me pointerët dhe janë sinonim në aspektin se si i qasen memories.

- Për të iu qasur anëtarit të **i**-të vektorit **v** me ndërmjetësimin e pointerit **p** mund të shfrytëzohet shprehja:

```
p = &v[i];
```

- Në fusha numerike, emri i fushës tregon gjithmonë për elementin e parë. Adresa e anëtarit të parë të fushës numerike është **&v[0]**.

Pointerët në fusha 2/2

- Nëse janë të deklaruara: `int a[m];` dhe `int* p;` atëherë **p=a** dhe **a=p** janë të vlefshme dhe

  a[i] ≡ *(a+i)

  p[i] ≡ *(p+i)

- Dallimi i vetëm në mes tyre është se vlera e "**p**" mund të ndryshojë tek secila adresë e variablës integer, ndërsa "**a**" gjithmonë do të tregojë një fushë integer me madhësi **m**.

```
int  i;
int  array[10];

for (i = 0; i < 10; i++)
{
  array[i] = …;
}
```

≡

```
int *p;
int  array[10];

for (p = array; p < &array[10]; p++)
{
  *p = …;
}
```

Shembull 3.8

- Brenda funksionit main të tregohet shfrytëzimi i pointerit **p** për shtypjen e adresave të vektorit **v(5)**.

```
#include <iostream>
using namespace std;

int main()
{
    int v[5], *p;

    for(int i=0;i<5;++i)
    {
        p=&v[i];
        cout<<"Adresa e v["<<i<<"]="
            <<p<<"\n";
    }
    system("pause");
    return 0;
}
```

Shembull 3.9

- Brenda funksionit main të shtypen anëtarët e vektorit **vek[]** në menyrë direkte dhe përmes pointerit **pvek** duke pasur parasysh që vlejnë shprehjet e mëposhtme:

a[i] ≡ *(a+i)

p[i] ≡ *(p+i)

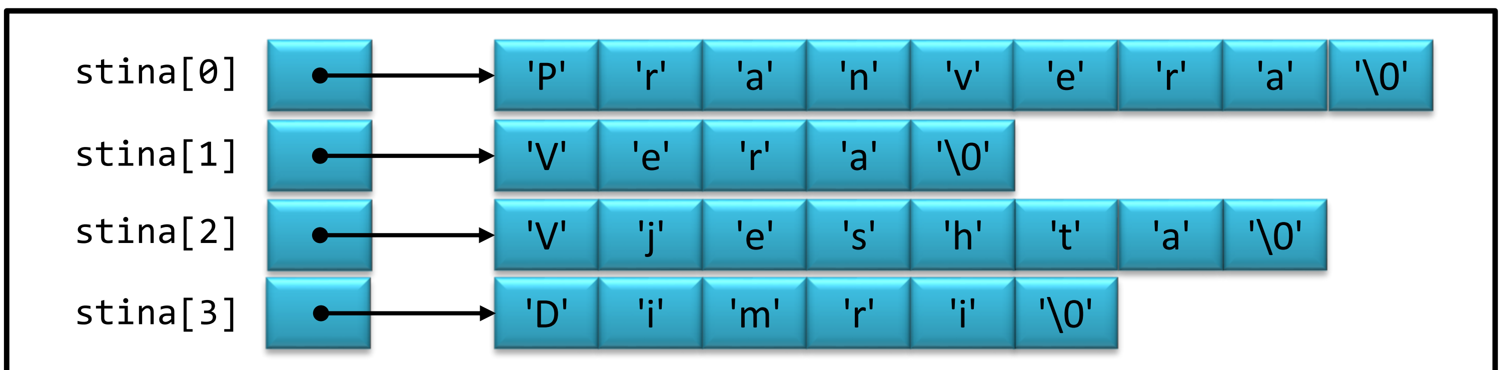
```
#include <iostream>
using namespace std;

int main()
{
    int vek[] = { 10, 20, 30, 40 };
    int *pvek = vek;

    cout << "Printimi përmes emrit të vektorit\n";
    for ( int i = 0; i < 4; ++i )
        cout << "vek[" << i << "] = " <<vek[i] << '\n';

    cout << "\nPrintimi përmes emrit të vektorit dhe operatorit *\n";
    for ( int j = 0; j < 4; ++j )
        cout << "*(vek + " << j << ") = " <<*(vek + j ) << '\n';

    cout << "\nAnëtarët përmes pointerit \n";
    for ( int k = 0; k < 4; ++k )
        cout << "pvek[" << k << "] = " << pvek[k] << '\n';

    cout << "\nAnëtarët përmes pointerit me zhvendosje\n";
    for ( int m = 0; m < 4; ++m )
        cout << "*(pvek + " << m << ") = "
            << *( pvek + m ) << '\n';

    system("pause");
    return 0;
}
```

Fusha pointerësh 1/2

- Për të vendosur stringje brenda fushave, p.sh., siç janë emrat e ndryshëm, fushat duhet të deklarohen si dydimensionale, përkatësisht si matrica.

```
char matrix[m][m];
```

- Por, nëse fushat e tilla ruhen në një vektor të deklaruar si *fushë pointerësh* (ang. array of pointers), hapësira memoruese do të jetë minimale.

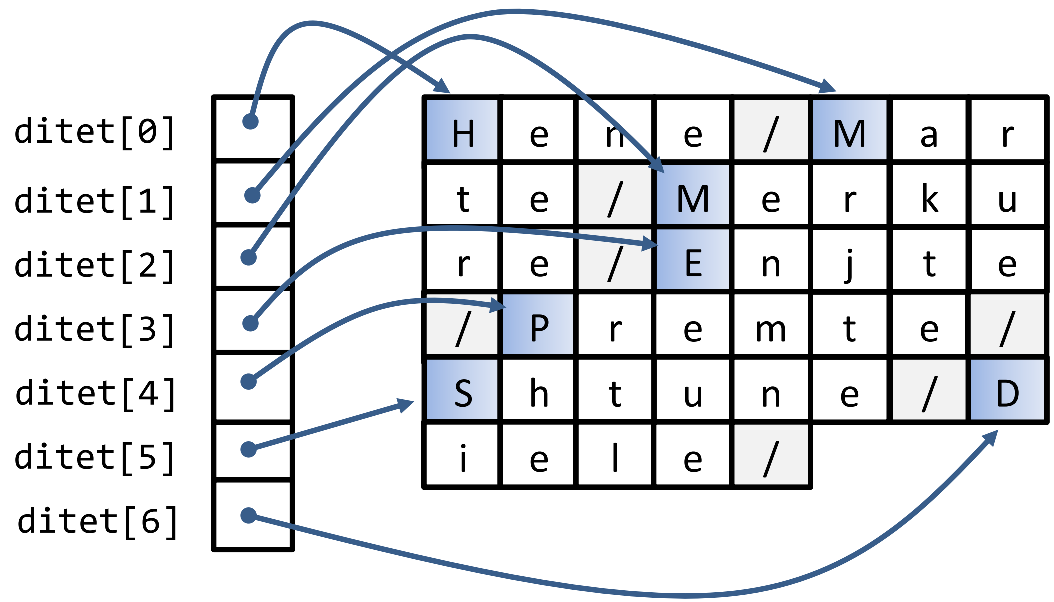


Fusha pointerësh 2/2

```c
char ditet[][8]=
{
        "Hene","Marte","Merkure",
        "Enjte", "Premte",
        "Shtune", "Diele"
};
```

```c
char *ditet[]=
{
        "Hene","Marte","Merkure",
        "Enjte", "Premte",
        "Shtune", "Diele"
};
```

H	e	n	e	/			
M	a	r	t	e	/		
M	e	r	k	u	r	e	/
E	n	j	t	e	/		
P	r	e	m	t	e	/	
S	h	t	u	n	e	/	
D	i	e	l	e	/		



Shembull 3.10

- Brenda funksionit main të ruhen emrat e katër studentëve duke shfrytëzuar vektorin **studentet** i cili është deklaruar si pointer.

```
#include <iostream>
using namespace std;
const int m = 4;

int main ()
{
    char *studentet[m] ={
                        "Filan Fisteku",
                        "Kosovare Shkodra",
                        "Saranda Prizreni",
                        "Milot Mitrovica",
                        };

    for (int i = 0; i < m; i++)
    {
        cout<< "Studenti[" << i << "] = "
            << studentet[i] << endl;
    }

    system("pause");
    return 0;
}
```

Pointerët si parametra të funksioneve

- Pointerët :
  - Mund të jenë si parametra të funksioneve,
  - Mund të kthehen nga funksionet,
  - Mund të ruhen në fusha,
  - Mund të shoqërohen në pointerë të funksioneve tjera.

```
void funksioni(int *p1, double *p2)
{
   //...
}
```

```
void shuma(double *x, int *n, double *y);

int main()
{
   //...
   shuma(&x,&n,&y);
}
```

Shembull 3.11

- Brenda funksionit main të bëhet konvertimi prej inçëve në centimetra. Për këtë të shfrytëzohet funksioni **inchNeCm** i cili ka parametrin formal **p** të tipit pointer.

```cpp
#include <iostream>
using namespace std;

double inchNeCm(double*); //prototipi

int main()
{

    double y,x = 5.0;
    cout << "x = " << x << " inch"
        << endl;

    y=inchNeCm(&x); //dergo adresen e var

    cout << "y = " << y << " centimetra"
        << endl;

    system("pause");
    return 0;
}

double inchNeCm(double* p)
{
    double y;
    y =*p * 2.54;
    return y;
}
```

Pointerët në funksione

- Funksionet gjatë ekzekutimit të programit vendosën në hapësira të caktuara memoruese.
- Adresa ku fillon vendosja e një funksioni në memorien e kompjuterit quhet adresë e funksionit.
- Për ruajtjen e adresës së funksionit shfrytëzohet *pointeri në funksion,* i cili njihet edhe si *pointer funksioni.*
- Pointeri i funksionit mund të shfrytëzohet për thirrjen e funksionit adresa e së cilit është ruajtur tek ku pointer.

```
void shuma(int, int);
```

```
void (*pointeri)(int, int);
```

- Të shkruhet programi përmes së cilit llogaritet vlera e shprehjes së mëposhtme duke shfrytëzuar funksionin **shuma**. Funksioni shuma të thirret përmes pointerit **p**.

$$y = \sum_{i=1}^{n} (i)$$

```cpp
#include <iostream>
using namespace std;
void shuma(int n);

int main()
{
    void (*p)(int);
    p=shuma;
    int n=10;

    (*p)(n);

    system("pause");
    return 0;
}

void shuma(int n)
{
    int y=0;
    for (int i=1; i<=n;i++)
        y=y+i;

    cout << "\ny="
        << y
        << "\n\n";
}
```

Pointerët në struktura

- Pointerët, përveç te të dhënat e tipave të zakonshëm, mund të shfrytëzohen edhe gjatë operimit me struktura.

- Deklarimi i variablave të tipit pointer për strukturat dhe operimin me anëtarët e tyre nuk dallon nga mënyra e shfrytëzimit të pointerëve për tipat e zakonshëm të të dhënave.

```
struct studenti {int studentID;};

int main()
{
   studenti *stPtr;

   (*stPtr).studentID= 122012;
   stPtr->studentID = 122012;
}
```

Shembull 3.13

- Brenda funksionit main të deklarohet struktura **vija** e cila ruan koordinatat e pikës A dhe B. Në funksionin **main** të deklarohet një pointer i strukturës dhe qasja tek anëtarët e strukturës të bëhet përmes pointerit.

```
#include <iostream>
using namespace std;

struct vija
{
	double x1,y1;
	double x2, y2;
};

int main()
{
	vija v;
	vija *p;
	p=&v;
	(*p).x1=0;
	(*p).y1=0;
	(*p).x2=5;
	(*p).y2=5;
	cout<< "\n\nPika A = ("
		<< (*p).x1
		<<", "
		<<(*p).y1
		<<")";
	cout << "\nPika B = ("
		<< (*p).x2
		<<", "
		<<(*p).y2
		<<")"
		<< "\n\n";
	system("pause");
	return 0;
}
```

Pointerët në objekte

- Pointerët mund të shfrytëzohen gjatë operimit me objekte të klasave, njëlloj sikurse edhe tek strukturat.

```
...
data dt;
dt.vendos(17,2,2008);
dt.shfaq();

data *dtp;
dtp=new data;
dtp->vendos(17,2,2008);
dtp->shfaq();
...
```

Shembull 3.14

- Të shkruhet programi, përmes së cilit definohet klasa rrethi. Brenda klasës të definohet **rrezja** si private dhe funksionet publike **vendos_rrezen** dhe **siperfaqja.** Objekti i klasës të jetë pointeri **rrp**.

```cpp
#include <iostream>
using namespace std;

const double pi=3.1415;
class rrethi
{
    private:
        double rrezja;
    public:
        void vendos_rrezen(double r)
        {
            rrezja = r;
        }
        double siperfaqja()
        {
            return pi * rrezja * rrezja ;
        }
};

int main()
{
    rrethi rr, *rrp;
    rrp=&rr;

    rrp->vendos_rrezen(5.6);
    cout<<"Siperfaqja e rrethit = "
        <<rrp->siperfaqja()
        << "\n\n";
    system("pause");
    return 0;
}
```