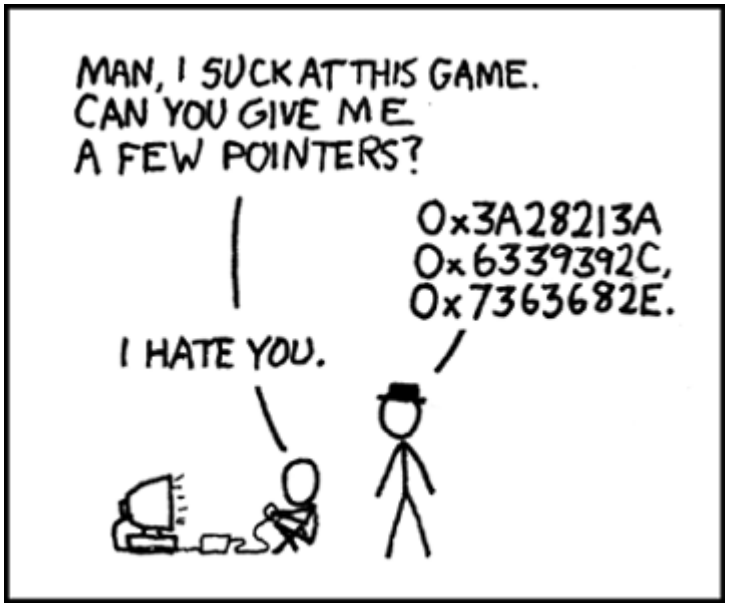
Uvod u pokazivače



Sadržaj

- Uvod
- Operatori referenciranja i dereferenciranja
- Deklaracija pokazivača
- Inicijalizacija pokazivača
- Rad s pokazivačima
  - Preusmjeravanje pokazivača
  - Pridruživanje pokazivača
  - Primjeri
- null-pokazivač
- Pokazivači i polja
- Pokazivači i strukture

Uvod

- Pokazivač (engl. pointer)
  - varijabla koja sadrži **memorijsku adresu** objekta na kojeg "pokazuje"
- Podaci se u pohranjuju na memorijskoj lokaciji koja ima svoju adresu:
  - 1 bajt
  - 1 memorijski blok: adrese definirane slijedno
  - za adrese je uobičajen heksadekadski zapis:
    - Primjer: 0013FF78

Uvod

- Primjena pokazivača
  - implementacija nekih složenih struktura
    - povezane liste, stog, red, stabla, višedimenzionalna polja promjenljivih veličina,...
  - dinamička alokaciju memorije
  - učinkovito rukovanje s poljima
  - dijeljenje istog memorijskog prostora od strane različitih dijelova koda
  - promjena vrijednosti parametara funkcije

Operator referenciranja

- *engl. reference operator*
- Operator referenciranja (adrese): **&**
- Daje adresu varijable na kojoj je ona pohranjena u memoriji
- Adresa na kojoj je pohranjena varijabla naziva se referencom te varijable

Operator referenciranja

```
int x = 10;
int &ref = x;
cout << "Vrijednost x = "<<x<<
  ", a ref = "<<ref<<endl;
cout << "Adresa x = "<< &x <<
  ", a adresa ref = "<< &ref <<endl;
cout << "Promjena vrijednosti varijable x.\n";
x=11;
cout << "Vrijednost x = "<<x<<
  ", a ref = "<<ref<<endl;
cout << "Adresa x = "<< &x <<
  ", a adresa ref = "<< &ref <<endl;
```

Operator referenciranja

- **Ispis nakon pokretanja programa:**

Vrijednost x = 10, a ref = 10
Adresa x = 0026F738, a adresa ref = 0026F738

Promjena vrijednosti varijable x.

Vrijednost x = 11, a ref = 11
Adresa x = 0026F738, a adresa ref = 0026F738

Operator dereferenciranja

- *engl. dereference operator*
- Operator dereferenciranja (indirekcije): *****
- Daje vrijednost varijable koja je pohranjena na adresi na koju pokazuje
- Dvije namjene:
  - u definiciji pokazivača
  - u postupku dohvata sadržaja varijable na koju pokazuje pokazivač

Deklaracija pokazivača

- Pokazivač se može deklarirati tako da pokazuje na bilo koji tip podatka
- Deklaracija:
  - navodi se tip podatka na koji pokazivač pokazuje
  - ispred imena se stavlja znak (*asterisk,* zvjezdica) *

```
tip_podatka *ime_pokazivača;
```

- Primjer:

```
//pokazivač na varijablu tipa int
int *pok;
```

Inicijalizacija pokazivača

- Nakon deklaracije pokazivač je moguće inicijalizirati
- Dva načina:
  1. Usmjeravanje pokazivača na objekt koji već postoji u memoriji
     - pokazivaču se pridruži adresa objekta

       pok = &n;
  2. Inicijalizacija pokazivača pomoću operatora **new**
     - dinamička alokacija memorije: memorijski prostor zauzima se u trenutku kada je potreban i oslobađa se kada više nije potreban
     - pokazivaču se pridruži adresa prostora koji mu je dodijeljen

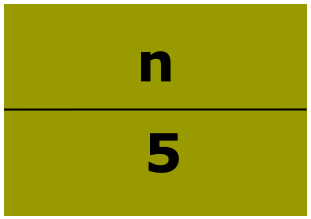
Inicijalizacija pokazivača

- **Primjer:**

```
//deklaracija i definicija varijable n
int n=5;
//deklaracija pokazivača
int *pok;
//usmjeravanje pokazivača
pok= &n;
```

Inicijalizacija pokazivača

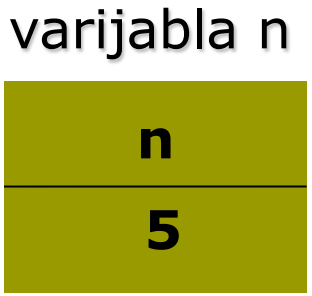
varijabla n




```
int n = 5;
```

deklaracija i definicija varijable

Inicijalizacija pokazivača



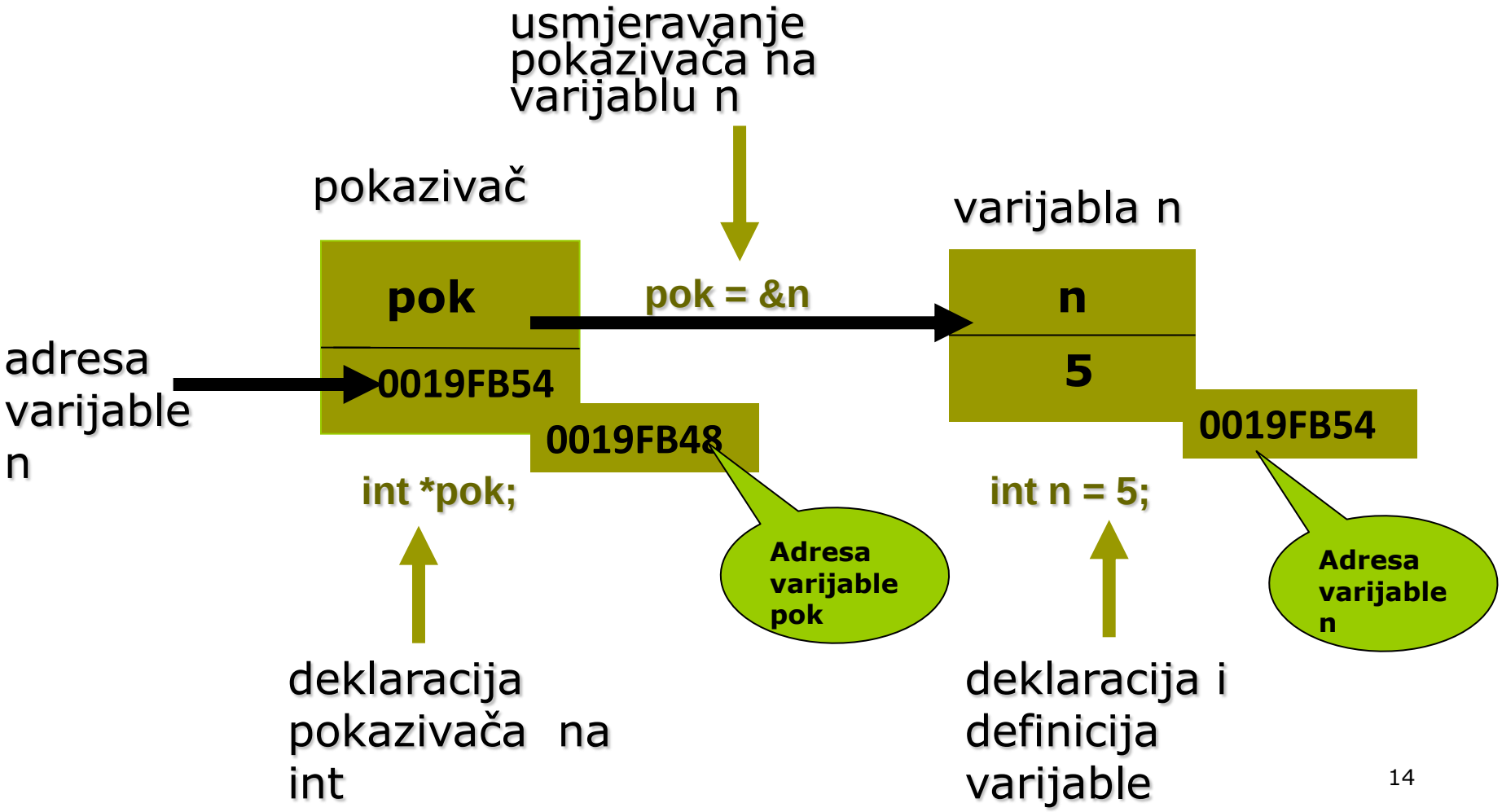

```
int *pok;
```

```
int n = 5;
```

deklaracija pokazivača na int

deklaracija i definicija varijable

Inicijalizacija pokazivača



Inicijalizacija pokazivača

- **Pokazivač se može usmjeriti i prilikom deklaracije:**

```
//deklaracija i definicija varijable n
int n=5;
//deklaracija i usmjeravanje pokazivača
int *pok = &n;
```

Inicijalizacija pokazivača

- **Što će se ispisati nakon pokretanja programa?**

```
int n=5;
int *pok=&n;
cout<<"Ispis podataka:\n";
cout<<"n="<<n<<endl;
cout<<"&n="<<&n<<endl;
cout<<"*pok="<<*pok<<endl;
cout<<"pok="<<pok<<endl;
cout<<"&pok="<<&pok<<endl;
```

Inicijalizacija pokazivača

- **Ispis nakon pokretanja programa:**

Ispis podataka:

n=5

&n=0019FB54

***pok=5**

pok=0019FB54

&pok=0019FB48

Preusmjeravanje pokazivača

```
int *pok;    //pokazivač na int
int n=8;     //varijabla
pok=&n;      //usmjeravanje na varijablu n
int m=3;     // uvodimo novu varijablu
pok =&m;
*pok = 10    // preusmjeravanje na novu varijablu m
```

- Što se ispisuje na ekranu ako nakon napisanog koda dodamo naredbu **cout<<*pok<<endl;** ?

Primjer (pitanje)

- Što će se ispisati izvršavanjem slijedećih naredbi?

```
/*1*/        int i=1;
/*2*/        int j=10;
/*3*/        int *p=&j; //i=1; j=10; *p=10
/*4*/        *p *= *p; // i=1; j=100; *p=100
/*5*/        i=i+j; //i=101; j=100; *p=100
/*6*/        p=&i; //i=101; j=100; *p=101
/*7*/        cout<<i<<endl<<j<<endl<<*p<<endl;
```

Primjer greške

- Dereferencirati se može samo usmjereni pokazivač
- Nakon deklaracije pokazivač se mora prvo usmjeriti:
  - ne može se pridružiti vrijednost pokazivaču dok on nije usmjeren na varijablu
  - nije još zauzeta memorijska adresa za pohranjivanje vrijednosti
- Primjer:

```
int *pok; // deklaracija pokazivača
*pok=100;          // greška
```

- Run-time error:
  - Run-Time Check Failure #3 - The variable 'pok' is being used without being initialized.

Primjer greške

- Dereferencirati se može samo usmjereni pokazivač
- Nakon deklaracije pokazivač moramo prvo usmjeriti:
  - ne može se upisivati vrijednost pokazivača ako on nije usmjeren na varijablu
  - nije još zauzeta memorijska adresa za pohranjivanje vrijednosti
- Primjer:

```
int *pok;          // deklaracija pokazivača
cin>>*pok;         // dereferenciranje uzrokuje grešku
```

- Run-time error:
  - Run-Time Check Failure #3 - The variable 'pok' is being used without being initialized.

Pridruživanje pokazivača

- Kada pokazivaču pridružimo drugi pokazivač oni nakon toga sadrže istu adresu
  - pokazuju na isti objekt
- Pridruživanje pokazivača ne mijenja sadržaj na danoj adresi.
- Primjer:

```
int n = 10;
int *pok1 = &n,*pok2;
pok2 = pok1; //p1 i p2 pokazuju na isti objekt, n
```

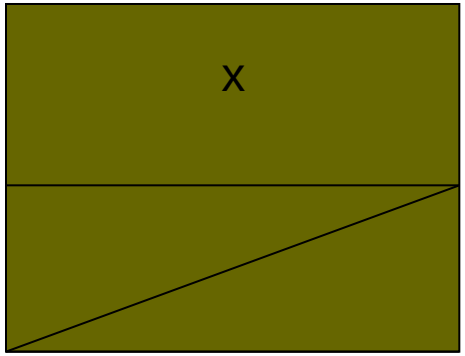
null-pokazivač

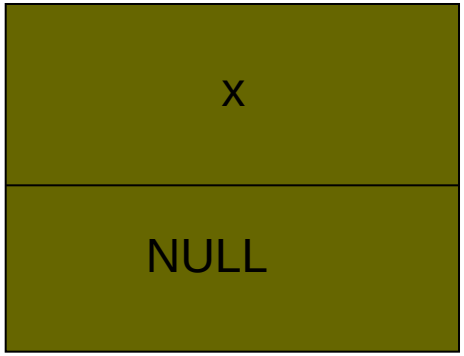
- engl. *null-pointer*
- Posebna vrijednost koju pokazivač može imati je **NULL (nul-pokazivač)**.
- Preporučuje se pokazivač pri deklaraciji **inicijalizirati** na NULL vrijednost, da bi se kasnije mogla vršiti provjera njegove vrijednosti.
- Takav pokazivač **nije usmjeren**, pa ako pokušamo pristupiti njegovom sadržaju (dereferenciranje) dolazi do greške (run-time error).

NUL-pokazivač

- Inicijalizacija na NULL vrijednost:

```
int *x=0; //ili
int *x=NULL;  //  konstanta NULL
```





Pokazivači i polja

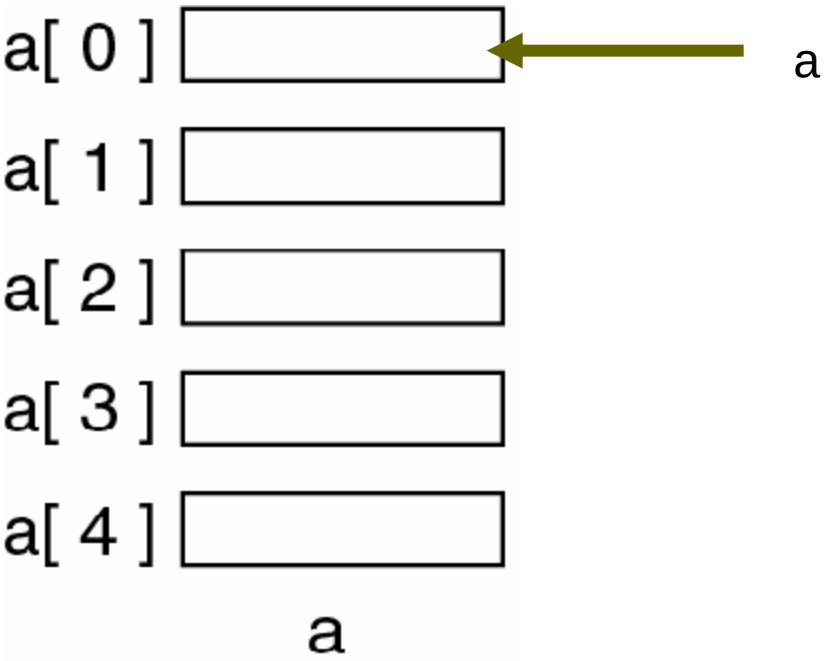
- U programskom jeziku C++ polja i pokazivači su povezani na način da se elementi polja dohvaćaju preko pokazivača
- Ime polja daje adresu prvog elementa polja:

  polje = &polje[0]

  polje = &polje

- Za pristupanje elementima polja možemo koristiti operator dereferenciranja (*)
  - 1. Element polja: ***polje**

Pokazivači i polja



Pokazivači i polja

```
#include <iostream.h>
void main(){
   int a[5];
   cout << "Adresa elementa a[0]: " << &a[0] << endl
        << "Ime je pokazivac: " << a << endl
         <<&a << endl;
}

/* ispis:

Adresa elementa a[0]: 0x00427D50

Ime je pokazivac: 0x00427D50

0x00427D50
*/
```

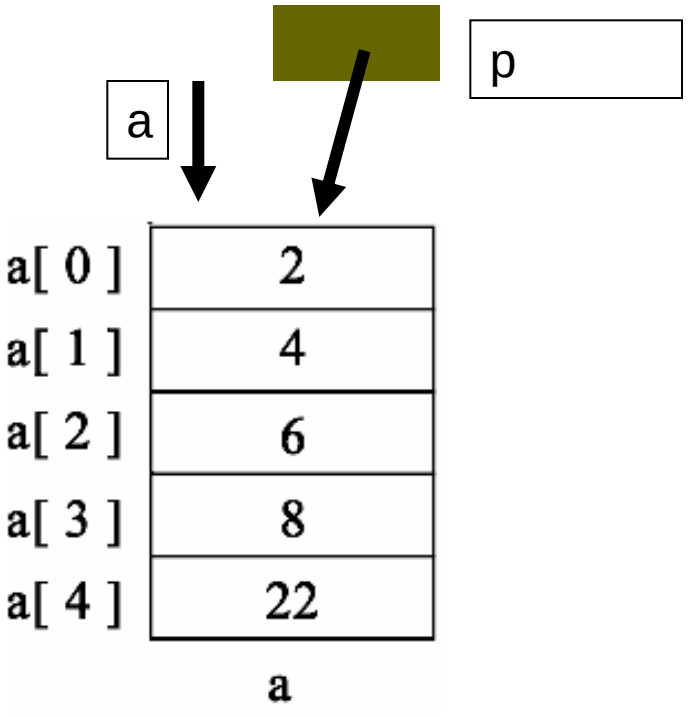
Pokazivači i polja

- Preostalim elementima polja možemo pristupiti tako da dodajemo 1
  - zato što su elementi polja pohranjeni u memoriji u nizu, svaki elementi je na adresi za 1 većoj od prethodnog…

- Elementi polja: *a, *(a+1), *(a+2),…*(a+n-1)
- Daje isto što i: a[0], a[1], a[2],…a[n-1]

Pokazivači i polja

Da bi pristupili polju, možemo koristiti bilo koji pokazivač na prvi element, umjesto imena polja.



Pokazivači i polja

```
#include <iostream.h>
int main(){
  int a[5]={2, 4, 6, 8, 22};
  int *p=a;
  int i=0;
  cout<<a[i]<<"  "<<*p;
  return 0;
}
```

Pokazivači i polja

- Pokazivač možemo koristiti za dohvaćanje uzastopnih članova polja:

```
int a[]={2,4,6,8,22};
int *p=a;
for(int i=0;i<5;i++)
    cout<<*(p+i)<<endl; // *(p+i) = p[i]
```

Pokazivači i polja

- Ime polja je (samo) sinonim za pokazivač na prvi element polja:
  - sam pokazivač nigdje nije alociran u memoriji

```
int x[5];
cout<<"x="<<x<<endl;
cout<<"&x="<<&x<<endl;
//ispisuje istu adresu
```

  - ne može se mijenjati vrijednost pokazivača na polje koje je definirano preko imena pokazivača (u smislu korištenja aritmetike pokazivača)

Aritmetika pokazivača

- Na pokazivačima su definirane neke aritmetičke operacije kao i usporedbe pokazivača
- Moguće je:
  - pokazivaču pribrojiti ili oduzeti cijeli broj,
  - računati razliku između dva pokazivača,
  - uspoređivati pokazivače
- Rezultat aritmetičkih operacija s pokazivačima je nova memorijska adresa

Aritmetika pokazivača

- Ako se pokazivaču pribroji ili oduzme cijeli broj, rezultat takve operacije bit će pokazivač koji pokazuje na memorijsku lokaciju udaljenu za navedeni broj objekata tog tipa
- Na primjer, ako se od pokazivača na objekt int oduzme broj 3, dobit će se pokazivač koji pokazuje na memorijeksu lokaciju koja je za 3*sizeof(int) manja od početne:

```
int a;
int *poka=&a;
cout<<poka<<endl<<poka-3<<endl;
```

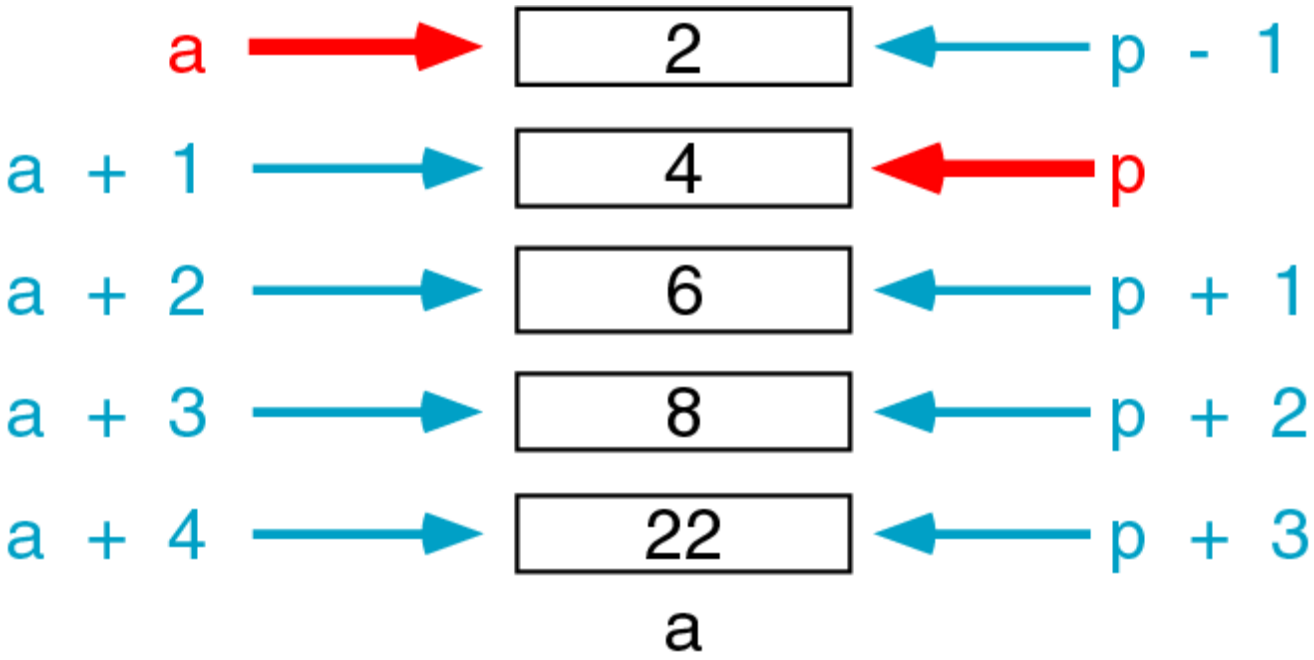
Aritmetika pokazivača

- Općenito, aritmetiku pokazivača nije poželjno koristiti ako ne možemo predvidjeti rezultat promjene adrese
- Aritmetika pokazivača ima prednosti kod dohvaćanja uzastopnih članova polja:

```
int a[]={2,4,6,8,22};
int *p=a;
for(int i=0;i<5;i++){
    cout<<*p<<endl;
    p++; //adresa pokazivača p uveća se za 1,
    //prelazi na idući element polja}
```

Aritmetika pokazivača

- Za zadani pokazivač p, izraz p+n označava element koji je udaljen od p za n mjesta u polju.



Pokazivači i strukture

- Primjer strukture:

```
struct student{
    string ime;
    int id;
    int ocjene[3];
};
    ....
    int main() {
    student stu; // deklaracija instance strukture
    ...
}
```

Pokazivači i strukture

- Možemo definirati pokazivač na instancu:

```
student *pokStu = &stu;
```

- Općenito:

ime_strukture * ime_pokazivaca = & ime_instance;

- Pristupanje komponenti strukture preko pokazivača:

```
(*pokStu).id = 123;
```

- Drugi način:

```
pokStu->id = 123;
```

Ponavljanje: Pokazivači i strukture

1) Dane su deklaracije:

```
struct TipProizvoda {
   int kolicina;
   float cijena;};
TipProizvoda Proizvod1 = {17, 4.99};
TipProizvoda* pokazivac = &Proizvod1;
```

Koji od slijedećih izraza predstavlja ispravni pristup sadržaju podatkovnog člana strukture Proizvod1?

1. (*pokazivac).cijena

2. *pokazivac.cijena

3. pokazivac -> cijena

Pitanja

- Definirajte što je pokazivač
- Objasnite što omogućavaju operator referenciranja i operator dereferenciranja
- Navedite primjene pokazivača.
- Napišite dio koda kojim se deklarira i usmjerava pokazivač na varijablu realnog tipa.
- Što će se dogoditi ako pridružujemo vrijednost pokazivaču koji nije usmjeren?
- Što je null-pokazivač?

Litertura

- Demistificirani C++
- ...

Ponavljanje

- Koje od navedenih deklaracija pokazivača su dozvoljene:

```
int *pok1;
int* pok2;
int * pok3;
int*pok4;
```

Ponavljanje

- Na koje varijable se odnosi zvjezdica u sljedećoj naredbi (jesu li obje pokazivači na int ili ne?):

```
int* x, y;
```

Ponavljanje

- Primjer. Neka je zadan dio koda, odredite što će se ispisati (koje od vrijednosti su izjednačenje):

```
int a = 10, *pok1;
pok1 = &a;
int *pok2 = pok1;
cout<<"a = "<<a<<"\t&a = "<<&a<<endl;
cout<<"pok1 = "<<pok1<<"\t*pok1 = "
    <<*pok1<<"\t&pok1="<<&pok1<<endl;
cout<<"pok2 = "<<pok2<<"\t*pok2 = "
    <<*pok2<<"\t&pok2="<<&pok2<<endl;
```

Ponavljanje

- Primjer. Neka je zadan dio kod, odredite što će se ispisati:

```
int a = 10, b=2;
int *pok = &a;
*pok = a+b;
b = a+b;
pok = &b;
cout<<"a = "<<a<<"\tb = "
    <<b<<"\t*pok="<<*pok<<endl;
```

Ponavljanje

- Koja je razlika između zadane dvije naredbe ako znamo da je varijabla pok pokazivač:

```
...
*pok = 0;
...
```

i

```
...
pok = 0; // NULL pokazivač!!!
...
```

Ponavljanje

- Što će se ispisati na ekranu:

```
int a = 30;
int *pok = &a;
*pok = 0;
if(pok)
 cout<<*pok<<endl;
else
 cout<<"null-pokazivac.\n";
```

Ponavljanje

- Što će se ispisati na ekranu:

```
int a = 30;
int *pok = &a;
pok = 0;
if(pok)
 cout<<*pok<<endl;
else
 cout<<"null-pokazivac.\n";
```

Ponavljanje

- Komentirajte zadani isječak koda:

```
int polje[]={7,4,12,3,5};
for(int i=0;i<5;i++)
      cout<<*(polje+i)<<endl;
```

Ponavljanje

- Komentirajte zadani isječak koda:

```
int polje[]={7,4,12,3,5};
int *pok = polje;
for(int i=0;i<5;i++)
      cout<<*(pok+i)<<endl;
```

Ponavljanje

- Komentirajte zadani isječak koda:

```
int polje[]={7,4,12,3,5};
int *pok = polje;
for(int i=0;i<5;i++,pok++)
     cout<<*pok<<endl;
```

Ponavljanje

- Komentirajte zadani isječak koda:

```
int polje[]={7,4,12,3,5};
for(int i=0;i<5;i++,polje++)
      cout<<*polje<<endl;
```


MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A
0x6339392C,
0x7363682E.
I HATE YOU.