




PROGRAMIRANJE
Predavanje 11 – Pokazivači

Ishod učenja 5

1

POKAZIVAČI

Strana • 2



2

Introduction

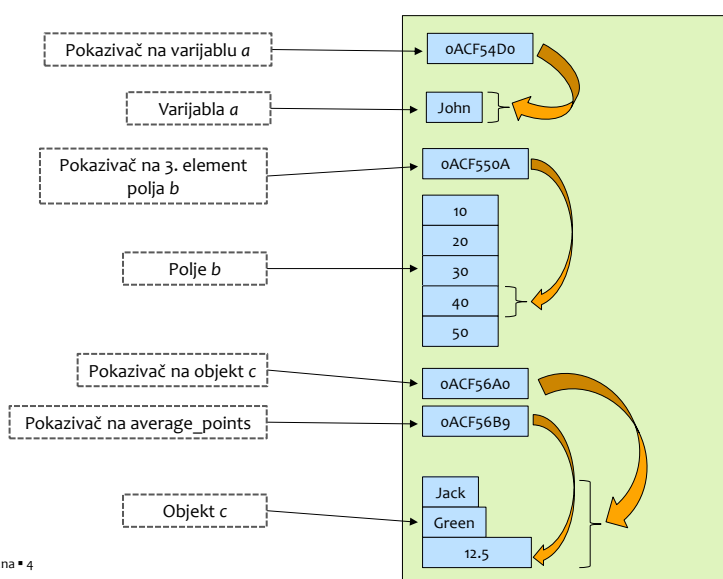
- **Pokazivač** je varijabla koja sadrži memorijsku adresu
 - Veličina adrese je:
 - 32-bitni programi: 4 bajta
 - 64-bitni programi: 8 bajtova
 - Kažemo da pokazivač pokazuje na tu adresu
 - Na toj adresi se nalazi neki podatak
 - Taj podatak ima svoj tip podataka koji se onda naziva ciljni tip
 - Kad deklariramo pokazivač, moramo definirati ciljni tip podataka
 - Nakon toga se ciljni tip podataka više ne može mijenjati
 - Pokazivač obično pokazuje na varijablu, element polja, objekt ili člana objekta

Strana • 3



3

Vizualizacija



Strana • 4



4

Deklariranje pokazivača

- Pokazivač je tip podataka
 - Koristimo ga tako da deklariramo varijablu tog tipa
- Moramo specificirati sljedeće:
 - Ciljni tip
 - Operator*
 - Naziv varijable (pokazivača)
- Primjer:


```
string* pfirst_name;
```

 - Kreirali smo varijablu pfirst_name koja je pokazivač na string
 - Ali na koji string (na kojoj adresi)?
 - Pokazivač je trenutno neinicijaliziran – prije nego ga koristimo , moramo ga usmjeriti na nešto (inicijalizirati ga)

Strana * 5



5

Inicijalizacija pokazivača

- Pokazivač inicijaliziramo upisivanjem memorijske adrese u njega
- Operator& uzima adresu onoga što piše s desne strane


```
int number = 37;
int* p1 = &number;
```
- Posebna vrsta pokazivača je **null-pokazivač**
 - Kažemo da ne pokazuje nikuda („uzemlje” pokazivač)
 - Definira se ključnom riječju nullptr:


```
int* p2 = nullptr;
```
 - Kad ne znamo kamo usmjeriti pokazivač, najbolje je usmjeriti ga u nullptr

Strana * 6




6

Vizualizacija

```
int number = 42;
int* pnumber = &number;
```

The diagram illustrates the memory layout for the provided code. It shows a vertical stack of memory addresses from 0xF001018 down to 0xF001005. The address 0xF001007 is highlighted in orange and contains the value 42. A bracket labeled 'pnumber' spans from 0xF001016 to 0xF001007, indicating that the pointer variable pnumber holds the address 0xF001007. Another bracket labeled 'number' spans from 0xF001010 to 0xF001007, indicating that the variable number holds the value 42 at the address 0xF001007.

Strana * 7




7

Usporedba tipova double i double*

- Usporedimo tipove double i double* na primjeru:


```
double a = 89.5;
double* p = &a;
```
- Koja je veličina memorije na koju pokazuju varijable?
 - Varijabla a: 8 bajtova (jer je to double)
 - Varijabla p: 4 bajta (jer je to adresa na 32-bitnom sustavu)
- Što sadržavaju varijable?
 - Varijabla a sadržava 89.5
 - Varijabla p sadržava adresu varijable a
- Uvijek treba imati na umu: što sadržava varijabla koja je pokazivač? **Samo adresu!**

Strana * 8



8

Pokazivač i objekt na koji pokazuje

- Potrebno je **razlikovati pokazivač** (engl. *pointer*) **od objekta koji pokazuje** (engl. *pointee*)
- Da bismo preko pokazivača došli do vrijednosti na adresi na koju on pokazuje, koristimo operator **dereferenciranja**, a to je također '*'
 - Isti operator koristimo na dva različita načina, prevoditelj u ovisnosti o kontekstu zna na što mislimo

```
int broj = 37;
```

```
int* pbroj = &broj;
```

```
cout << pbroj;
```

```
cout << *pbroj;
```

Ispisuje adresu
(primjerice, 0012FF70)

Ispisuje 37

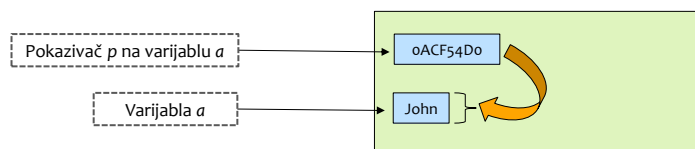
Strana • 9



9

Upotreba pokazivača

- Kad kreiramo pokazivač, napravili smo još jedan način za pristupanje istoj memorijskoj lokaciji



- Sljedeće linije su potpuno jednake

```
getline(cin, a);
```

```
getline(cin, *p);
```

Strana • 10

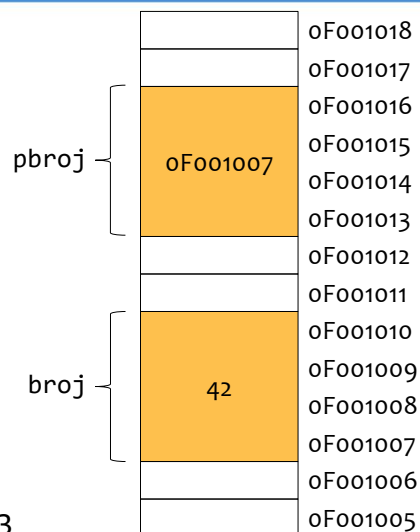


10

Vizualizacija (jako važan slajd)

```
int broj = 42;
int* pbroj = &broj;

cout << broj; // 42
cout << &broj; // 0F001007
cout << pbroj; // 0F001007
cout << *pbroj; // 42
cout << &pbroj; // 0F001013
```



Strana • 11



11

Primjeri

1. Riješite sljedeće:
 - a. Napravite string varijablu i ispišite njenu adresu.
 - b. Napravite pokazivač na nju.
 - c. Pomoću pokazivača učitajte u nju vrijednost od korisnika.
 - d. Ispišite vrijednost varijable direktno i preko pokazivača.

Strana • 12



12

Pokazivač na pokazivač na pokazivač na ...

- Po istom principu možemo koristiti i pokazivač na pokazivač na pokazivač na ...

```
int broj = 42;
```

```
int* p1 = &broj;
```

```
int** p2 = &p1;
```

```
int*** p3 = &p2;
```

Pokazivač na int

Pokazivač na pokazivač na int

Pokazivač na pokazivač na pokazivač na int

```
cout << p3 << endl; // Ispisuje adresu od p2.
```

```
cout << *p3 << endl; // Ispisuje adresu od p1.
```

```
cout << **p3 << endl; // Ispisuje adresu od broj.
```

```
cout << ***p3 << endl; // Ispisuje 42.
```

Strana * 13



13

Dodatni operatori

- Osim navedenih, pokazivač sadrži i sljedeće operatore:
 - Operatori '+' i '-' između pokazivača i cijelog broja
 - Rezultat je nova memorijska adresa
 - Adresa je veća/manja za **n puta veličina tipa podataka**
 - Operatori '++' i '--' na pokazivaču
 - Rezultat je nova memorijska adresa
 - Adresa je veća/manja za **veličinu tipa podataka!**
 - Operator '=' između dva pokazivača
 - Kopira adresu

Strana * 14



14

POKAZIVAČI, POLJA I OBJEKTI

Strana • 15



15

Pokazivači i polja

- **Naziv polja je pokazivač na nulti element polja**

- Ako deklariramo:

```
int brojevi[] = { 10, 20, 30, 40, 50 };
```

- **brojevi** sadržava **adresu nultog** elementa polja
- **brojevi + 1** sadržava **adresu prvog** elementa polja
- **brojevi + 2** sadržava **adresu drugog** elementa polja
- **brojevi + 3** sadržava **adresu trećeg** elementa polja
- **brojevi + 4** sadržava **adresu četvrtog** elementa polja

Dakle, **brojevi[n]** je ekvivalentno ***(brojevi + n)**

Strana • 16



16

Primjeri

2. Napravite polje od 5 cijelih brojeva i u njega učitajte brojeve od korisnika koristeći operator []. Ispišite polje obrnutim redoslijedom koristeći operator [].
3. Promijenite prethodni program tako da umjesto operatora [] koristite pokazivače.
4. Zadajte polje od nekoliko brojeva i pronađite najmanji. Koristite operator [].
5. Riješite prethodni zadatak korištenjem pokazivača umjesto operatora [].

Strana • 17



17

Pokazivači i objekti

- Objektu nastalom na osnovu strukture također možemo pristupiti pomoću pokazivača
- Pri dohvaćanju članova treba pokazivač dereferencirati, paziti na redoslijed primjene operatora

- '.' je jača od '*' pa moramo koristiti okrugle zagrade

```
Osoba susjed;
susjed.ime = "Marko";
Osoba* p = &susjed;
cout << (*p).ime << endl;
```

- Alternativa je korištenje operatora '->' koji prvo radi dereferenciranje, a onda pristupa članu

```
cout << p->ime << endl;
```

Strana • 18



18

Primjeri

6. Napravite strukturu za čuvanje podataka o knjigama. Napravite objekt te strukture, učitajte vrijednosti članova od korisnika te ih ispišite. Nemojte koristiti pokazivače.
7. Promijenite prethodni zadatak tako da koristite pokazivače.
8. Napravite polje od 3 knjige i dodijelite im vrijednosti. Pronađite najstariju i ispišite joj naslov. Koristite samo pokazivače bez operatora '->'.
 9. Riješite prethodni zadatak korištenjem operatora '->'.

Strana * 19



19

Zadaci za sljedećih 7 dana

▪ Prije sljedećeg predavanja trebate:

1. Pročitati iz *Demistificirani C++*:
 - 6.1 Pokazivači
2. Pogledati sljedeće:
 - W11-1 Pointers
 - https://youtu.be/_lhWobFIHUo

Strana * 20



20