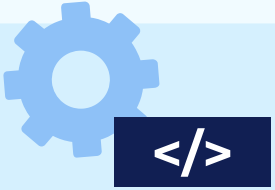
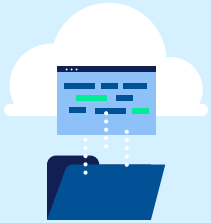
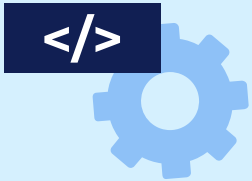
UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI
FAKULTET



TESTIRANJE BAZIRANO NA TOKU PODATAKA

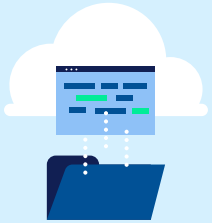


TESTIRANJE SOFTVERA



01

TOK PODATAKA



DODELA I UPOTREBA VREDNOSTI

- Testiranje bazirano na toku podataka ispituje tok prenosa podataka u programu, odnosno mesta gde promenljiva dobija neku vrednost i gde se vrednost promenljive upotrebljava.
- Za svaki iskaz **S** u programu se definišu dva skupa:
  - **DEF(S)** – Skup svih promenljivih kojima se dodeljuje vrednost u iskazu S.
  - **USE(S)** – Skup svih promenljivih čija se vrednost upotrebljava u iskazu S.
- Upotrebe mogu biti predikatske (*p-use*) ili računske (*c-use*):
  - **P-use** – Promenljiva se koristi u uslovu naredbe kontrole toka (*if*, *for*, *while*, *switch*).
  - **C-use** – Promenljiva se koristi u svrhu računanja nekog izraza.
  - Ako upotreba nije predikatska, onda je računska.

DU LANCI

- **Lanac dodele-upotrebe** (DU lanac) za neku promenljivu definiše putanju od dodele vrednosti toj promenljivoj do upotrebe te dodeljene vrednosti u nekom drugom iskazu:

  [X, S_1, S_2]

  - **X** – Promenljiva
  - **S_1** – Iskaz u kom se dodeljuje vrednost promenljivoj X
  - **S_2** – Iskaz u kom se upotrebljava vrednost promenljive X koja je dodeljena u S1

- Na putanji od iskaza **S_1** do iskaza **S_2** u programu ne sme da postoji nova dodela vrednosti promenljivoj **X**.
  - Tada se kaže da je dodela vrednosti u iskazu S_1 **živa** u iskazu S_2.

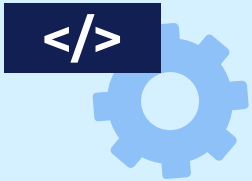


KRITERIJUMI POKRIVENOSTI

- Kriterijumi pokrivenosti definicija i upotreba definišu kakvi DU lanci treba da budu izvršeni tokom testiranja:
  - **Pokrivenost svih definicija** (engl. *all-defs coverage*)
    - Svaka definicija svake promenljive mora da dostigne jednu upotrebu te promenljive.
  - **Pokrivenost svih upotreba** (engl. *all-uses coverage*)
    - Svaka definicija svake promenljive mora da dostigne svaku upotrebu te promenljive.
  - **Pokrivenost svih DU lanaca** (engl. *all-DU-paths coverage*)
    - Svaka prosta putanja od svake definicije do svake upotrebe mora da bude izvršena.

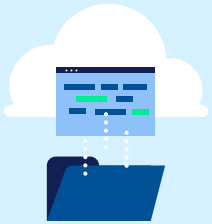
KRITERIJUMI POKRIVENOSTI

- **Pokrivenost svih c-upotreba** (engl. *all-c-uses coverage*)
  - Svaka definicija svake promenljive mora da dostigne svaku c-upotrebu te promenljive.
- **Pokrivenost svih p-upotreba** (engl. *all-p-uses coverage*)
  - Svaka definicija svake promenljive mora da dostigne svaku p-upotrebu te promenljive.
- **Pokrivenost svih c, nekih p-upotreba** (engl. *all-c-uses/some-p-uses coverage*)
  - Svaka definicija svake promenljive mora da dostigne svaku c-upotrebu te promenljive ako ona postoji, a ako ne onda da dostigne jednu p-upotrebu.
- **Pokrivenost svih p, nekih c-upotreba** (engl. *all-p-uses/some-c-uses coverage*)
  - Svaka definicija svake promenljive mora da dostigne svaku p-upotrebu te promenljive ako ona postoji, a ako ne onda da dostigne jednu c-upotrebu.



02

ZADACI



ZADATAK 1 – ZALIHE ČOKOLADA

POSTAVKA

- Dat je program napisan na programskom jeziku Python koji učitava podatke o čokoladama i njihovim zalihama jednog trgovinskog lanca. Na osnovu učitanih podataka program određuje trgovačku proviziju.

1. Odrediti sve DU lance za promenljive *cokolade*, *zalihe* i *popust*.
2. Napisati minimalni skup test primera kojim će biti pokriveni svi DU lanci iz prethodne tačke. Za svaki test primer navesti sekvencu ulaznih podataka i očekivanu vrednost provizije.

```
S0:  broj_cokolada = broj_zaliha = 0
S1:  cokolade = int(input())

S2:  while cokolade > 0:
S3:     zalihe = int(input())
S4:     broj_cokolada += cokolade
S5:     broj_zaliha += zalihe
S6:     cokolade = int(input())

S7:  popust = 45 * broj_cokolada + 30 * broj_zaliha
S8:  if popust > 1000.0:
S9:     provizija = 100.0
S10:    provizija = provizija + 0.2 * (popust - 1000)
     else:
S11:    provizija = 0.1 * popust
```

ZADATAK 1 – ZALIHE ČOKOLADA

REŠENJE – DU LANCI (1)

- Promenljivoj *cokolade* se dodeljuje vrednost u iskazima S_1 i S_6:

  DEF(cokolade) = {S_1, S_6}

- Vrednost promenljive *cokolade* se koristi u iskazima S_2 i S_4:

  USE(cokolade) = {S_2, S_4}

- Kandidat DU lanci su:

  DU_1 = [cokolade, S_1, S_2]

  DU_2 = [cokolade, S_1, S_4]

  DU_3 = [cokolade, S_6, S_2]

  DU_4 = [cokolade, S_6, S_4]

```
S0:  broj_cokolada = broj_zaliha = 0
S1:  cokolade = int(input())

S2:  while cokolade > 0:
S3:     zalihe = int(input())
S4:     broj_cokolada += cokolade
S5:     broj_zaliha += zalihe
S6:     cokolade = int(input())

S7:  popust = 45 * broj_cokolada + 30 * broj_zaliha
S8:  if popust > 1000.0:
S9:     provizija = 100.0
S10:    provizija = provizija + 0.2 * (popust - 1000)
     else:
S11:    provizija = 0.1 * popust
```

ZADATAK 1 – ZALIHE ČOKOLADA

REŠENJE – DU LANCI (2)

- Lanac **[cokolade, S_1, S_2]** će uvek biti izvršen jer se nakon unosa čokolada u iskazu **S_1** uvek provera uslov u iskazu **S_2**.
- Lanac **[cokolade, S_1, S_4]** će biti izvršen ukoliko se barem jednom uđe u telo petlje. Takođe, na putu od **S_1** do **S_4** nema ponovne definicije.
- Lanac **[cokolade, S_6, S_2]** će biti izvršen ukoliko se barem jednom uđe u telo petlje. Nakon kraja prve iteracije se sa iskaza **S_6** direktno prelazi na proveru uslova u iskazu **S_2**.
- Lanac **[cokolade, S_6, S_4]** će biti izvršen ukoliko se barem dva puta uđe u telo petlje.

```
S0:  broj_cokolada = broj_zaliha = 0
S1:  cokolade = int(input())

S2:  while cokolade > 0:
S3:     zalihe = int(input())
S4:     broj_cokolada += cokolade
S5:     broj_zaliha += zalihe
S6:     cokolade = int(input())

S7:  popust = 45 * broj_cokolada + 30 * broj_zaliha
S8:  if popust > 1000.0:
S9:     provizija = 100.0
S10:    provizija = provizija + 0.2 * (popust - 1000)
     else:
S11:    provizija = 0.1 * popust
```

ZADATAK 1 – ZALIHE ČOKOLADA

REŠENJE – DU LANCI (3)

- Promenljivoj *zalihe* se dodeljuje vrednost samo u iskazu S_3:

  DEF(zalihe) = {S_3}

- Vrednost promenljive *zalihe* se koristi samo u iskazu S_5:

  USE(zalihe) = {S_5}

- Jedini kandidat DU lanac je:

  DU_5 = [zalihe, S_3, S_5]

- Ovaj lanac će biti izvršen ukoliko se barem jednom uđe u telo petlje. U telu petlje ne postoji nijedna redefinicija promenljive *zalihe* na putu od iskaza **S_3** do iskaza **S_5**.

```
S0:  broj_cokolada = broj_zaliha = 0
S1:  cokolade = int(input())

S2:  while cokolade > 0:
S3:     zalihe = int(input())
S4:     broj_cokolada += cokolade
S5:     broj_zaliha += zalihe
S6:     cokolade = int(input())

S7:  popust = 45 * broj_cokolada + 30 * broj_zaliha
S8:  if popust > 1000.0:
S9:     provizija = 100.0
S10:    provizija = provizija + 0.2 * (popust - 1000)
     else:
S11:    provizija = 0.1 * popust
```

ZADATAK 1 – ZALIHE ČOKOLADA

REŠENJE – DU LANCI (4)

- Promenljivoj *popust* se dodeljuje vrednost samo u iskazu S_7:

$$\mathbf{DEF(popust) = \{S_7\}}$$

- Vrednost promenljive *popust* se koristi u iskazima S_8, S_{10} i S_{11}:

$$\mathbf{USE(popust) = \{S_8, S_{10}, S_{11}\}}$$

- Kandidat DU lanci su:

$$\mathbf{DU_6 = [popust, S_7, S_8]}$$
$$\mathbf{DU_7 = [popust, S_7, S_{10}]}$$
$$\mathbf{DU_8 = [popust, S_7, S_{11}]}$$

```
S0:  broj_cokolada = broj_zaliha = 0
S1:  cokolade = int(input())

S2:  while cokolade > 0:
S3:     zalihe = int(input())
S4:     broj_cokolada += cokolade
S5:     broj_zaliha += zalihe
S6:     cokolade = int(input())

S7:  popust = 45 * broj_cokolada + 30 * broj_zaliha
S8:  if popust > 1000.0:
S9:     provizija = 100.0
S10:    provizija = provizija + 0.2 * (popust - 1000)
     else:
S11:    provizija = 0.1 * popust
```

ZADATAK 1 – ZALIHE ČOKOLADA

REŠENJE – DU LANCI (5)

- Lanac **[popust, S_7, S_8]** će uvek biti izvršen jer se nakon izračunavanja popusta u iskazu **S_7** odmah proverava uslov u iskazu **S_8**.
- Lanac **[popust, S_7, S_{10}]** će biti izvršen samo ukoliko je zadovoljen uslov u iskazu **S_8**, odnosno ako je popust veći od 1000.
- Lanac **[popust, S_7, S_{11}]** će biti izvršen samo ukoliko nije zadovoljen uslov u iskazu **S_8**, odnosno ako je popust manji ili jednak od 1000.

```
S0:  broj_cokolada = broj_zaliha = 0
S1:  cokolade = int(input())

S2:  while cokolade > 0:
S3:     zalihe = int(input())
S4:     broj_cokolada += cokolade
S5:     broj_zaliha += zalihe
S6:     cokolade = int(input())

S7:  popust = 45 * broj_cokolada + 30 * broj_zaliha
S8:  if popust > 1000.0:
S9:     provizija = 100.0
S10:    provizija = provizija + 0.2 * (popust - 1000)
     else:
S11:    provizija = 0.1 * popust
```



ZADATAK 1 – ZALIHE ČOKOLADA

REŠENJE – DU LANCI (6)

- Utvrdili smo da su svi DU lanci izvršivi, eventualno pod određenim uslovima. Dakle, nema nijednog DU lanca koji nije izvršiv.
- Konačan skup DU lanaca je:

Br.	DU lanac
DU_1	[cokolade, S_1, S_2]
DU_2	[cokolade, S_1, S_4]
DU_3	[cokolade, S_6, S_2]
DU_4	[cokolade, S_6, S_4]
DU_5	[zalihe, S_3, S_5]
DU_6	[popust, S_7, S_8]
DU_7	[popust, S_7, S_{10}]
DU_8	[popust, S_7, S_{11}]

ZADATAK 1 – ZALIHE ČOKOLADA

REŠENJE – TEST PRIMERI

- Dovoljna su dva test primera kako bi se pokrili svi definisani DU lanci:

TP	Ulazni podaci	Provizija	Pokriveni DU lanci
1	15, 15, 5, 5, 0	200	DU_1, DU_2, DU_3, DU_4, DU_5, DU_6, DU_7
2	10, 10, 0	75	DU_1, DU_2, DU_3, DU_5, DU_6, DU_8

```
S0:  broj_cokolada = broj_zaliha = 0
S1:  cokolade = int(input())

S2:  while cokolade > 0:
S3:     zalihe = int(input())
S4:     broj_cokolada += cokolade
S5:     broj_zaliha += zalihe
S6:     cokolade = int(input())

S7:  popust = 45 * broj_cokolada + 30 * broj_zaliha
S8:  if popust > 1000.0:
S9:     provizija = 100.0
S10:    provizija = provizija + 0.2 * (popust - 1000)
     else:
S11:    provizija = 0.1 * popust
```

ZADATAK 2 – OCENE

POSTAVKA (1)

- Data je implementacija glavnog programa u programskom jeziku Java koji računa ocene studenata na ispitu na osnovu podataka o studentima i osvojenih poena na domaćim zadacima, projektnom zadatku i ispitu.

1. Odrediti sve definicije, p-upotrebe i c-upotrebe za promenljive *domaci*, *poeni*, *ukupno* i *brojac_studenata*.



ZADATAK 2 – OCENE

POSTAVKA (2)

```
  public static void main(String args[]) {
1.  int ukupno_studenata = args.length;
2.  int brojac_studenata = 0;
3.  while (brojac_studenata < ukupno_studenata) {
4.    System.out.println(args[brojac_studenata]);
5.    int broj_domacih = 4;
6.    int domaci = 0, ukupno = 0;
7.    while (domaci < broj_domacih) {
8.      int poeni = citajDomaci(args[brojac_studenata],domaci);
9.      if (poeni < 0)
10.       System.out.println("Nekorektan unos");
11.     else {
12.       ukupno += poeni;
13.       domaci++;
14.     }
15.   }
16.   int poeni = citajPoeneIspit(args[brojac_studenata]);
17.   if (poeni < 0)
18.     System.out.println("Nekorektan unos");
19.   else if (poeni >= 31 && poeni <= 50)
20.     ukupno += poeni;
21.   poeni = citajPoeneProjekat(args[brojac_studenata]);
22.   if (poeni >= 10 && poeni <= 30)
23.     ukupno += poeni;
24.   System.out.println("Ukupno poena " + ukupno);
25.   if (ukupno >= 91)
26.     System.out.println("Ocena 10");
27.   else if (ukupno >= 81)
28.     System.out.println("Ocena 9");
29.   else if (ukupno >= 71)
30.     System.out.println("Ocena 8");
31.   brojac_studenata++;
32. }
33. }
```

ZADATAK 2 – OCENE

REŠENJE – DEFINICIJE I UPOTREBE (1)

- Skup iskaza u kojima se vrši dodela vrednosti promenljivoj *domaci*:

  DEF(domaci) = {S_6, S_{13}}

- Skup iskaza u kojima se vrši p-upotreba promenljive *domaci*:

  P-USE(domaci) = {S_7}

- Skup iskaza u kojima se vrši c-upotreba promenljive *domaci*:

  C-USE(domaci) = {S_8, S_{13}}

```
  public static void main(String args[]) {
1.  int ukupno_studenata = args.length;
2.  int brojac_studenata = 0;
3.  while (brojac_studenata < ukupno_studenata) {
4.    System.out.println(args[brojac_studenata]);
5.    int broj_domacih = 4;
6.    int domaci = 0, ukupno = 0;
7.    while (domaci < broj_domacih) {
8.      int poeni = citajDomaci(args[brojac_studenata],domaci);
9.      if (poeni < 0)
10.       System.out.println("Nekorektan unos");
11.     else {
12.       ukupno += poeni;
13.       domaci++;
14.     }
15.   }
16.   int poeni = citajPoeneIspit(args[brojac_studenata]);
```

ZADATAK 2 – OCENE

REŠENJE – DEFINICIJE I UPOTREBE (2)

- Skup iskaza u kojima se vrši dodela vrednosti promenljivoj *poeni*:

$$\text{DEF(poeni)} = \{S_8, S_{16}, S_{21}\}$$

- Skup iskaza u kojima se vrši p-upotreba promenljive *poeni*:

$$\text{P-USE(poeni)} = \{S_9, S_{17}, S_{19}, S_{22}\}$$

- Skup iskaza u kojima se vrši c-upotreba promenljive *poeni*:

$$\text{C-USE(poeni)} = \{S_{12}, S_{20}, S_{23}\}$$

```
7.  while (domaci < broj_domacih) {
8.      int poeni = citajDomaci(args[brojac_studenata],domaci);
9.      if (poeni < 0)
10.       System.out.println("Nekorektan unos");
11.     else {
12.       ukupno += poeni;
13.       domaci++;
14.     }
15.   }
16.   int poeni = citajPoeneIspit(args[brojac_studenata]);
17.   if (poeni < 0)
18.     System.out.println("Nekorektan unos");
19.   else if (poeni >= 31 && poeni <= 50)
20.     ukupno += poeni;
21.   poeni = citajPoeneProjekat(args[brojac_studenata]);
22.   if (poeni >= 10 && poeni <= 30)
23.     ukupno += poeni;
24.   System.out.println("Ukupno poena " + ukupno);
```

ZADATAK 2 – OCENE

REŠENJE – DEFINICIJE I UPOTREBE (3)

- Skup iskaza u kojima se vrši dodela vrednosti promenljivoj *ukupno*:

  DEF(ukupno) = {S_6, S_{12}, S_{20}, S_{23}}

- Skup iskaza u kojima se vrši p-upotreba promenljive *ukupno*:

  P-USE(ukupno) = {S_{25}, S_{27}, S_{29}}

- Skup iskaza u kojima se vrši c-upotreba promenljive *ukupno*:

  C-USE(ukupno) = {S_{12}, S_{20}, S_{23}, S_{24}}




```
6.  int domaci = 0, ukupno = 0;
7.    while (domaci < broj_domacih) {
8.      int poeni = citajDomaci(args[brojac_studenata],domaci);
9.      if (poeni < 0)
10.       System.out.println("Nekorektan unos");
11.     else {
12.       ukupno += poeni;
13.       domaci++;
14.     }
15.   }
16.   int poeni = citajPoeneIspit(args[brojac_studenata]);
17.   if (poeni < 0)
18.     System.out.println("Nekorektan unos");
19.   else if (poeni >= 31 && poeni <= 50)
20.     ukupno += poeni;
21.   poeni = citajPoeneProjekat(args[brojac_studenata]);
22.   if (poeni >= 10 && poeni <= 30)
23.     ukupno += poeni;
24.   System.out.println("Ukupno poena " + ukupno);
25.   if (ukupno >= 91)
26.     System.out.println("Ocena 10");
27.   else if (ukupno >= 81)
28.     System.out.println("Ocena 9");
29.   else if (ukupno >= 71)
30.     System.out.println("Ocena 8");
```

ZADATAK 2 – OCENE

REŠENJE – DEFINICIJE I UPOTREBE (4)

- Skup iskaza u kojima se vrši dodela vrednosti promenljivoj *brojac_studenata*:

  DEF(brojac_studenata) = {S_2, S_{31}}

- Skup iskaza u kojima se vrši p-upotreba promenljive *brojac_studenata*:

  P-USE(brojac_studenata) = {S_3}

- Skup iskaza u kojima se vrši c-upotreba promenljive *brojac_studenata*:

  C-USE(brojac_studenata) =

  {S_4, S_8, S_{16}, S_{21}, S_{31}}

```
  public static void main(String args[]) {
1.  int ukupno_studenata = args.length;
2.  int brojac_studenata = 0;
3.  while (brojac_studenata < ukupno_studenata) {
4.    System.out.println(args[brojac_studenata]);
5.    int broj_domacih = 4;
6.    int domaci = 0, ukupno = 0;
7.    while (domaci < broj_domacih) {
8.      int poeni = citajDomaci(args[brojac_studenata],domaci);
9.      if (poeni < 0)
10.       System.out.println("Nekorektan unos");
11.     else {
12.       ukupno += poeni;
13.       domaci++;
14.     }
15.   }
16.   int poeni = citajPoeneIspit(args[brojac_studenata]);
17.   if (poeni < 0)
18.     System.out.println("Nekorektan unos");
19.   else if (poeni >= 31 && poeni <= 50)
20.     ukupno += poeni;
21.   poeni = citajPoeneProjekat(args[brojac_studenata]);
22.   // ...
31.   brojac_studenata++;
32. }
33. }
```

ZADATAK 2 – OCENE

REŠENJE – DEFINICIJE I UPOTREBE (5)

- Skupovi definicija, p-upotreba i c-upotreba za sve zadate promenljive:

Promenljiva	Definicija	P-upotreba	C-upotreba
domaci	S_6, S_{13}	S_7	S_8, S_{13}
poeni	S_8, S_{16}, S_{21}	S_9, S_{17}, S_{19}, S_{22}	S_{12}, S_{20}, S_{23}
ukupno	S_6, S_{12}, S_{20}, S_{23}	S_{25}, S_{27}, S_{29}	S_{12}, S_{20}, S_{23}, S_{24}
brojac_studenata	S_2, S_{31}	S_3	S_4, S_8, S_{16}, S_{21}, S_{31}

ZADATAK 3 – BINARNA PRETRAGA

POSTAVKA

- Data je implementacija funkcije **bin_search** u programskom jeziku C koja vrši binarnu pretragu ključa **k** u neopadajuće uređenom nizu **arr** dužine **n**. Povratna vrednost funkcije je pozicija ključa u nizu, odnosno –1 ukoliko ključ ne postoji u nizu.

1. Odrediti sve DU lance za promenljive *low*, *mid*, *high* i *k*.
2. Napisati minimalni skup test primera kojim će biti pokriveni svi DU lanci iz prethodne tačke. Za svaki test primer navesti ulazne podatke i očekivanu povratnu vrednost funkcije.

```
   int bin_search(int k, int arr[], int n) {
1.     int low = 0;
2.     int high = n - 1;
3.     while (low <= high) {
4.        mid = (low + high) / 2;
5.        if (k == arr[mid]) return mid;
6.        else if (k < arr[mid]) high = mid - 1;
7.        else low = mid + 1;
8.     }
9.     return -1;
10. }
```

ZADATAK 3 – BINARNA PRETRAGA

REŠENJE – DU LANCI (1)

- Promenljivoj *low* se dodeljuje vrednost u iskazima S_1 i S_7:

$$\mathbf{DEF(low) = \{S_1, S_7\}}$$

- Vrednost promenljive *low* se koristi u iskazima S_3 i S_4:

$$\mathbf{USE(low) = \{S_3, S_4\}}$$

- Kandidat DU lanci su:

$$\mathbf{DU_1 = [low, S_1, S_3]}$$
$$\mathbf{DU_2 = [low, S_1, S_4]}$$
$$\mathbf{DU_3 = [low, S_7, S_3]}$$
$$\mathbf{DU_4 = [low, S_7, S_4]}$$

```
  int bin_search(int k, int arr[], int n) {
1.      int low = 0;
2.      int high = n - 1;
3.      while (low <= high) {
4.         mid = (low + high) / 2;
5.         if (k == arr[mid]) return mid;
6.         else if (k < arr[mid]) high = mid - 1;
7.         else low = mid + 1;
8.      }
9.      return -1;
10. }
```

ZADATAK 3 – BINARNA PRETRAGA

REŠENJE – DU LANCI (2)

- Promenljivoj *high* se dodeljuje vrednost u iskazima S_2 i S_6:

$$\mathbf{DEF(high) = \{S_2, S_6\}}$$

- Vrednost promenljive *high* se koristi u iskazima S_3 i S_4:

$$\mathbf{USE(high) = \{S_3, S_4\}}$$

- Kandidat DU lanci su:

$$\mathbf{DU_5 = [high, S_2, S_3]}$$
$$\mathbf{DU_6 = [high, S_2, S_4]}$$
$$\mathbf{DU_7 = [high, S_6, S_3]}$$
$$\mathbf{DU_8 = [high, S_6, S_4]}$$

```
  int bin_search(int k, int arr[], int n) {
1.      int low = 0;
2.      int high = n - 1;
3.      while (low <= high) {
4.         mid = (low + high) / 2;
5.         if (k == arr[mid]) return mid;
6.         else if (k < arr[mid]) high = mid - 1;
7.         else low = mid + 1;
8.      }
9.      return -1;
10. }
```

ZADATAK 3 – BINARNA PRETRAGA

REŠENJE – DU LANCI (3)

- Promenljivoj *mid* se dodeljuje vrednost u iskazu S_4:

$$\mathbf{DEF(mid) = \{S_4\}}$$

- Vrednost promenljive *mid* se koristi u iskazima S_5 (na dva mesta), S_6 (na dva mesta) i S_7:

$$\mathbf{USE(mid) = \{S_{5,1}, S_{5,2}, S_{6,1}, S_{6,2}, S_7\}}$$

- Kandidat DU lanci su:

$$\mathbf{DU_9 = [mid, S_4, S_{5,1}]}$$
$$\mathbf{DU_{10} = [mid, S_4, S_{5,2}]}$$
$$\mathbf{DU_{11} = [mid, S_4, S_{6,1}]}$$
$$\mathbf{DU_{12} = [mid, S_4, S_{6,2}]}$$
$$\mathbf{DU_{13} = [mid, S_4, S_7]}$$

```
  int bin_search(int k, int arr[], int n) {
1.      int low = 0;
2.      int high = n - 1;
3.      while (low <= high) {
4.         mid = (low + high) / 2;
5.         if (k == arr[mid]) return mid;
6.         else if (k < arr[mid]) high = mid - 1;
7.         else low = mid + 1;
8.      }
9.      return -1;
10. }
```

ZADATAK 3 – BINARNA PRETRAGA

REŠENJE – DU LANCI (4)

- Vrednost promenljive *k* se koristi u iskazima S_5 i S_6:

  USE(k) = {S_5, S_6}

- Gde se dodeljuje vrednost promenljivoj *k*?

```
  int bin_search(int k, int arr[], int n) {
1.      int low = 0;
2.      int high = n - 1;
3.      while (low <= high) {
4.         mid = (low + high) / 2;
5.         if (k == arr[mid]) return mid;
6.         else if (k < arr[mid]) high = mid - 1;
7.         else low = mid + 1;
8.      }
9.      return -1;
10. }
```

ZADATAK 3 – BINARNA PRETRAGA

REŠENJE – DU LANCI (5)

- Promenljivoj *k* se dodeljuje vrednost kroz parametar funkcije:

$$DEF(k) = \{S_0\}$$

- Vrednost promenljive *k* se koristi u iskazima S_5 i S_6:

$$USE(k) = \{S_5, S_6\}$$

- Kandidat DU lanci su:

$$DU_{14} = [k, S_0, S_5]$$

$$DU_{15} = [k, S_0, S_6]$$

```
  int bin_search(int k, int arr[], int n) {
1.      int low = 0;
2.      int high = n - 1;
3.      while (low <= high) {
4.         mid = (low + high) / 2;
5.         if (k == arr[mid]) return mid;
6.         else if (k < arr[mid]) high = mid - 1;
7.         else low = mid + 1;
8.      }
9.      return -1;
10. }
```

ZADATAK 3 – BINARNA PRETRAGA

REŠENJE – DU LANCI (6)

- Svi DU lanci su izvršivi pod određenim uslovima. Konačan skup DU lanaca je:

Br.	DU lanac
DU_1	[low, S_1, S_3]
DU_2	[low, S_1, S_4]
DU_3	[low, S_7, S_3]
DU_4	[low, S_7, S_4]
DU_5	[high, S_2, S_3]
DU_6	[high, S_2, S_4]
DU_7	[high, S_6, S_3]
DU_8	[high, S_6, S_4]

Br.	DU lanac
DU_9	[mid, S_4, $S_{5,1}$]
DU_{10}	[mid, S_4, $S_{5,2}$]
DU_{11}	[mid, S_4, $S_{6,1}$]
DU_{12}	[mid, S_4, $S_{6,2}$]
DU_{13}	[mid, S_4, S_7]
DU_{14}	[k, S_0, S_5]
DU_{15}	[k, S_0, S_6]

ZADATAK 3 – BINARNA PRETRAGA

REŠENJE – TEST PRIMERI

- Dovoljan je samo jedan test primer kako bi se pokrili svi definisani DU lanci:

TP	Ulazni podaci	Pozicija	Pokriveni DU lanci
1	k = 5 arr = {1,2,3,4,5,6,7} n = 7	4	Svi DU lanci

```
  int bin_search(int k, int arr[], int n) {
1.     int low = 0;
2.     int high = n - 1;
3.     while (low <= high) {
4.        mid = (low + high) / 2;
5.        if (k == arr[mid]) return mid;
6.        else if (k < arr[mid]) high = mid - 1;
7.        else low = mid + 1;
8.     }
9.     return -1;
10. }
```

UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI
FAKULTET

PITANJA?

https://si3ts.etf.bg.ac.rs/

nikolas@etf.bg.ac.rs

