SVEUČILIŠTE U ZAGREBU
Fakultet prometnih znanosti
Zavod za inteligentne transportne sustave
Vukelićeva 4, Zagreb, HRVATSKA

Računalstvo

Pseudokôd i uvod u dijagrame toka

Izv. prof. dr. sc. Edouard Ivanjko, dipl.ing.

Sadržaj

- Uvod
- Pseudokôd
- Dijagram toka
- Primjeri

Uvod

- Program predstavlja pamet u računalu
  - Izvodi se tijekom rada računala
- Programi za različite namjene
  - Obradu teksta
    - MS Word, Notepad, OpenOffice, GoogleDocs
  - Tablični proračuni
    - MS Excel, OpenOffice, GoogleDocs
  - Simulatori
    - VISSIM/VISUM, Matlab, MatSim, SciLab
  - Izradu slika i nacrta
    - MS Visio, AutoCad, MS Paint, CorelDRAW
  - Ostali programi

Uvod

- Prije izrade složenih programa potrebna skica programa uz popis varijabli
- Skica idejnog rješenja se radi pomoću pseudokôda i dijagrama toka
  - Pseudokôd univerzalan i tekstualan pristup izradi skice programa
    - Pogodan za opis algoritama i složenih programa
  - Dijagram toka grafički pristup izradi skice programa
    - Pogodan za jednostavnije programe

Pseudokôd – Grananje

- U programima se često donose odluke
  - Ovisno o vrijednosti varijable program mijenja tijek izvođenja
- Postoji naredba grananja **ako je** (engl. „if”)
  - Ako je uvjet ispunjen (istinit) izvodi se naredba1
  - Ako uvjet nije ispunjen (neistina) izvodi se naredba2
- Uvjet može biti aritmetički ili logički izraz
- Moguće zadati više naredbi
  - Blok naredbi

```
ako je uvjet onda
        naredba1
inače
        naredba2
```

Pseudokôd – Grananje

- Moguće uzastopno ispitivanje više vezanih uvjeta
  - Koristi se grana „inače ako je uvjet onda” za provjeru dodatnih uvjeta
  - Može biti po volji mnogo dodatnih uvjeta
    - Uvijek će se izvršiti samo jedan programski blok strukture grananja

```
ako je prvi uvjet onda
   naredbe prvi uvjet ispunjen
inače ako je drugi uvjet onda
   naredbe drugi uvjet ispunjen
inače ako je treći uvjet onda
   naredbe treći uvjet ispunjen
inače
   naredbe niti jedan uvjet nije ispunjen
```

Pseudokôd – Grananje

- Ukoliko su uvjeti kod grananja jednostavni i svode se na usporedbu s konstantom koristi se skretnica (engl. „switch”)
  - Provjerava se vrijednost varijable s unaprijed zadanim konstantnim vrijednostima
    - Jedna konstantna vrijednost predstavlja jedan slučaj
      - Nije moguće koristiti izraz za definiciju slučajeva
    - Svaki slučaj (engl. „case) ima svoje pridružene naredbe
    - Slučaj „zadano” će se izvršiti ako je vrijednost ispitivane varijable različita od svih slučajeva
  - U svakom slučaju se provjerava uvijek ista zadana varijabla

Pseudokôd – Grananje

- Slučajeva kod skretnice može biti po volji mnogo
- Pseudokôd skretnice

```
skretnica (varijabla)
        slučaj prva vrijednost:
                naredbe za prvu vrijednost
        slučaj druga vrijednost:
                naredbe za drugu vrijednost
        zadano:
                naredbe za nedefiniranu vrijednost
```

Pseudokôd – Grananje

- Primjer programa s grananjem za ispitivanje je li broj veći od unesenog praga
  - Za ovaj program su potrebne dvije varijable
    - Prva za spremanje iznosa praga
    - Druga za spremanje iznosa podatka koji će se usporediti s pragom

Ime varijable	Tip varijable	Značenje varijable
prag	cijeli broj	Iznos praga s kojim će se uneseni podatak usporediti
vrijednost	cijeli broj	Uneseni podatak koji će se usporediti s pragom

Pseudokôd – Grananje

- Primjer programa s grananjem za ispitivanje je li broj veći od unesenog praga

```
Početak programa
    Unos
        prag
        vrijednost
    ako je vrijednost > prag onda
        Ispis
            "Unesena vrijednost je veća od praga!"
    inače
        Ispis
            "Unesena vrijednost nije veća od praga!"
Kraj programa
```

Pseudokôd – Grananje

- Primjer programa s grananjem za ispitivanje je li broj veći od unesenog praga
  - Analiza pseudokôda testnim podacima
    - Bitno je pokriti sve karakteristične slučajeve
  - Ulazni podaci: prag = 3, vrijednost = 2
    - Uvjet nije ispunjen i ispisuje se druga poruka
  - Ulazni podaci: prag = 3, vrijednost = 3
    - Uvjet nije ispunjen i ispisuje se druga poruka
  - Ulazni podaci: prag = 3, vrijednost = 4
    - Uvjet je ispunjen i ispisuje se prva poruka

Pseudokôd – Petlje

- Računala često obrađuju velike količine podataka na isti način
  - Isti skup naredbi se izvršava na drugoj vrijednosti podatka
  - Npr. naplata cestarine, usmjeravanje pošiljke, …
- U tu svrhu se koriste petlje
- Sastavni dijelovi petlje
  - Uvjet radi ispitivanja dali su sve vrijednosti podataka obrađene
  - Tijelo petlje koje sadrži jednu ili više naredbi
    - Obavezna naredba za povećanje količine obrađenih vrijednosti podataka

Pseudokôd – Petlje

- Tri vrste petlji
  - Ispitivanje uvjeta prije izvođenja tijela petlje
    - Moguće da se tijelo petlje nikad neće izvršiti
  - Ispitivanje uvjeta nakon izvođenja tijela petlje
    - Tijelo petlje se sigurno izvrši barem jednom
  - Petlja s unaprijed poznatim brojem izvođenja
    - Inicijalizacija i povećanje količine obrađenih podataka se izvodi automatski
    - Dobra ako je količina podataka za obradu unaprijed poznata
- Uz prilagođen uvjet se može koristiti bilo koja petlja za rješavanje istog problema
  - Ovisi o preferenciji programera

Pseudokôd – Petlje

- Petlje s ispitivanjem uvjeta prije izvođenja

```
dok je uvjet činiti
    naredba
```

- Ukoliko tijelo petlje sadrži više naredbi programski blok se ograničava s „{" i „}"

```
dok je uvjet činiti
{
  naredbe
}
```

- Ukoliko je uvjet jednak logičkoj jedinici govorimo o beskonačnoj petlji
  - Za uređaje koji rade 24/7

Pseudokôd – Petlje

- Primjer programa koji računa zbroj prvih **n** cijelih pozitivnih brojeva
  - Vrijednost podatka **n** nije unaprijed poznata programu
    - Unosi se tijekom izvođenja programa
  - Za izračun navedenog zbroja moguće iskoristiti posebnosti načina rada računala
    - Sve promjene se odvijaju mijenjanjem sadržaja određene memorijske lokacije gdje je varijabla spremljena

broj = broj + 1

  - Matematički neispravno
  - U računalu su to naredbe koje povećavaju vrijednost spremljenu u varijabli „**broj**” za **1**

Pseudokôd – Petlje

- Primjer programa koji računa zbroj prvih **n** cijelih pozitivnih brojeva
  - Matematički izražen zbroj prvih n cijelih brojeva

$$zbroj = \sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n$$

  - Ako se razloži za pojedine vrijednosti varijable **n**
    - n = 1

$$zbroj(1) = \sum_{i=1}^{1} i = 1 = 1$$

    - n = 2

$$zbroj(2) = \sum_{i=1}^{2} i = 1 + 2 = zbroj(1) + 2 = 3$$

    - n = 3

$$zbroj(3) = \sum_{i=1}^{3} i = 1 + 2 + 3 = zbroj(2) + 3 = 6$$

Pseudokôd – Petlje

- Primjer programa koji računa zbroj prvih **n** cijelih pozitivnih brojeva
  - Računala rade slijedno i obrađuju samo dvije varijable u aritmetičkoj operaciji
    - Koristimo zakonitost

      zbroj(n) = zbroj(n-1) + n
    - U programu za računalo
      - Inicijalizacija prije pokretanja petlje

        zbroj = 0
      - Iteracija izvršavanja tijela petlje

        zbroj = zbroj + i

        » Varijabla „**i**” raste od **1** do **n** za korak **1**

Pseudokôd – Petlje

- Primjer programa koji računa zbroj prvih **n** cijelih pozitivnih brojeva
  - Za ovaj program su potrebne tri varijable

Ime varijable	Tip varijable	Značenje varijable
n	cijeli broj	Količina brojeva koju je potrebno zbrojiti
i	cijeli broj	Količina do sada zbrojenih brojeva
zbroj	cijeli broj	Zbroj do sada obrađenih brojeva

Pseudokôd – Petlje

- Primjer programa koji računa zbroj prvih **n** cijelih pozitivnih brojeva

```
Početak programa
    Unos
        n
    Inicijaliziraj
        zbroj := 0
        i := 1
    dok je i <= n činiti
        zbroj := zbroj + i
        i := i + 1
    Ispis
        ”Zbroj prvih ” + n + ” cijelih brojeva iznosi ” + zbroj
Kraj programa
```

Pseudokôd – Petlje

- Petlje s provjerom uvjeta nakon izvođenja tijela petlje
  - Tijelo petlje će se izvršiti barem jednom

```
činiti
   naredba
dok je uvjet
```

- Ukoliko tijelo petlje sadrži više naredbi programski blok se ograničava s „{" i „}"

```
činiti
{
   naredbe
}
dok je uvjet
```

Pseudokôd – Petlje

- Primjer programa koji računa umnožak prvih **n** cijelih pozitivnih brojeva
  - Vrijednost podatka **n** nije unaprijed poznata programu
    - Unosi se tijekom izvođenja programa
  - Za izračun umnoška opet koristimo posebnosti načina rada računala
    - Matematički

$$umnožak = \prod_{i=1}^{n} i = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

    - U programu za računalo
      - Inicijalizacija prije pokretanja petlje

        umnožak = 1
      - Iteracija izvršavanja tijela petlje

        umnožak = umnožak * i
        - » Varijabla „**i**” raste od **1** do **n** za korak **1**

Pseudokôd – Petlje

- Primjer programa koji računa umnožak prvih **n** cijelih pozitivnih brojeva
  - Za ovaj program su potrebne tri varijable

Ime varijable	Tip varijable	Značenje varijable
n	cijeli broj	Količina brojeva koju je potrebno pomnožiti
i	cijeli broj	Količina do sada pomnoženih brojeva
umnožak	cijeli broj	Umnožak do sada obrađenih brojeva

Pseudokôd – Petlje

- Primjer programa koji računa umnožak prvih **n** cijelih pozitivnih brojeva

```
Početak programa
    Unos
        n
    Inicijaliziraj
        umnožak := 1
        i := 1
    činiti
        umnožak := umnožak * i
        i := i + 1
    dok je i <= n
    Ispis
        ”Umnožak prvih ” + n + ”cijelih brojeva iznosi ” + umnožak
Kraj programa
```

Pseudokôd – Petlje

- Petlje s unaprijed poznatim brojem izvođenja tijela petlje
  - U definiciji petlje tri dijela
    - Inicijalizacija varijabli
    - Definicija uvjeta
    - Definicija koraka

```
za b := p do n korak k činiti
    naredba
```

  - Ukoliko tijelo petlje sadrži više naredbi programski blok se ograničava s „{" i „}"

```
za b := p do n korak k činiti
{
    naredbe
}
```

Pseudokôd – Petlje

- Primjer programa koji će ispisati vrijednosti prvih **n** elemenata aritmetičkog niza
  - Aritmetički niz je niz brojeva u kojem je razlika svakog člana i njegovog prethodnika stalan broj (konstanta)
    - Izračun novog člana aritmetičkog niza

$$a(n) = a(n-1) + k$$

    - Izračun n-tog člana aritmetičkog niza

$$a(n) = a(1) + (n-1) \cdot k$$

  - Potrebno je poznavati iznos prvog člana aritmetičkog niza **a(1)** te razliku između dva člana **k**
    - Aritmetički niz može biti rastući i padajući

Pseudokôd – Petlje

- Primjer programa koji će ispisati vrijednosti prvih **n** elemenata aritmetičkog niza
  - Implementacija u programu za računalo
    - Koristimo petlju s unaprijed poznatim brojem izvršavanja
      - Inicijalizacija prije pokretanja petlje

        korak

        niz = početnaVrijednost

        n
      - Iteracija izvršavanja tijela petlje

        niz = niz + korak

Pseudokôd – Petlje

- Primjer programa koji će ispisati vrijednosti prvih **n** elemenata aritmetičkog niza
  - Za ovaj program je potrebno pet varijabli

Ime varijable	Tip varijable	Značenje varijable
n	cijeli broj	Količina članova niza koju je potrebno ispisati
i	cijeli broj	Količina do sada ispisanih članova niza
niz	cijeli broj	Vrijednost trenutnog člana niza
korak	cijeli broj	Vrijednost koraka aritmetičkog niza
početnaVrijednost	cijeli broj	Iznos prvog člana niza

Pseudokôd – Petlje

- Primjer programa koji će ispisati vrijednosti prvih **n** elemenata aritmetičkog niza

```
Početak programa
    Unos
        n
        korak
        početnaVrijednost
    Inicijaliziraj
        niz := početnaVrijednost
    Ispis
        ”Vrijednost 1. elementa niza iznosi ” + niz
    za i := 2 do n korak 1 činiti
        niz := niz + korak
        Ispis
            ”Vrijednost ” + i + ”. elementa niza iznosi ” + niz
Kraj programa
```

Dijagram toka

- Dijagram toka omogućuje kreiranje koncepta programa grafički
- Naredbe su prikazane grafičkim simbolom
  - Simboli standardizirani
- Moguće prikazati slijed naredbi, operacija, upute za razne procedure, …
  - Univerzalan koncept
  - Omogućuje laganu komunikaciju između inženjera iz različitih polja i zemalja
- Preporuča se korištenje engleskog jezika za oznake i imena varijabli

Dijagram toka

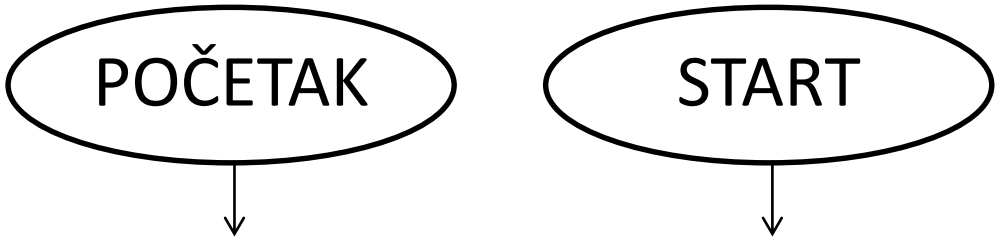
- Prednosti dijagrama toka
  - Univerzalno kreiranje koncepta za bilo koji programski jezik
    - Pojedini blok se samo zamjeni naredbom pripadnog programskog jezika
  - Lako kreiranje i provjera koncepta programske logike
    - Grafički simboli lakši za praćenje u analizi, nego tekstualan programski kôd
  - Omogućuje povezivanja više različitih struka
    - Bitno kod današnjeg timskog rada i interdisciplinarnih projekata
  - Moguće automatsko kreiranje programskog kôda
    - Ubrzavanje učenja te izradu proizvoda (aplikacija)

Dijagram toka

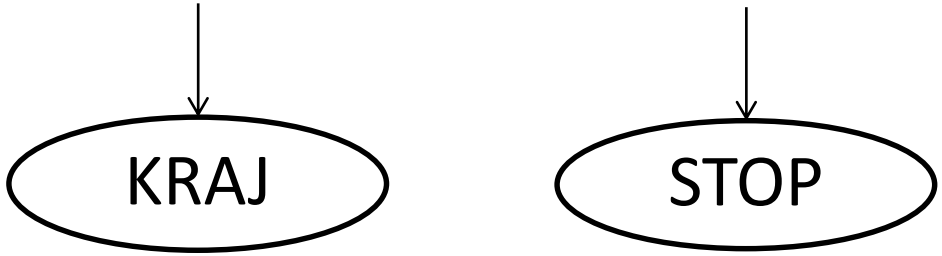
- Pravila kreiranja dijagrama toka
  - Naredbe (blokovi) se izvršavaju slijedno od početka prema završetku dijagrama toka
    - Smjer označen strelicom
    - Nije moguća promjena smjera izvršavanja
  - Svaka vrsta naredbe ima svoj pripadni blok
    - Nije moguće iskoristiti blok za neku drugu namjenu
  - Grananje programa omogućuje kreiranje više smjerova izvršavanja programa
    - Moguće odabrati samo jedan smjer
    - Svi smjerovi se na kraju spajaju prije bloka završetka
  - Dijelovi dijagrama toka se spajaju poveznicama

Dijagram toka – Blokovi

- Oznaka početka dijagrama toka
  - Obavezno na početku dijagrama toka
  - Samo jedna izlazna strelica

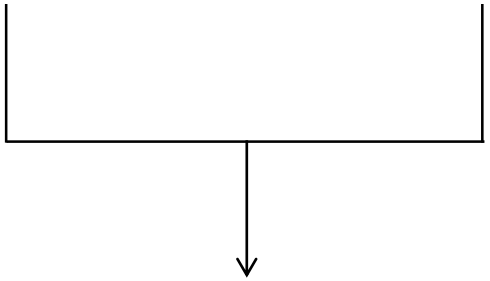


- Oznaka završetka dijagrama toka
  - Obavezno na kraju dijagrama toka
  - Samo jedna ulazna strelica



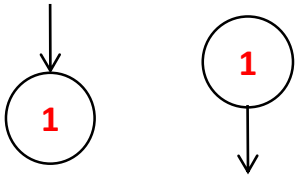
Dijagram toka – Blokovi

- Za povezivanje blokova se koriste strelice
  - Uvijek jednosmjerne radi jednoznačnosti
  - Zbog jasnoće nije dozvoljeno križanje strelica
  - Smjer strelice definira smjer izvođenja dijagrama toka
  - Izlazi iz više blokova se mogu spajati u jednu poveznicu



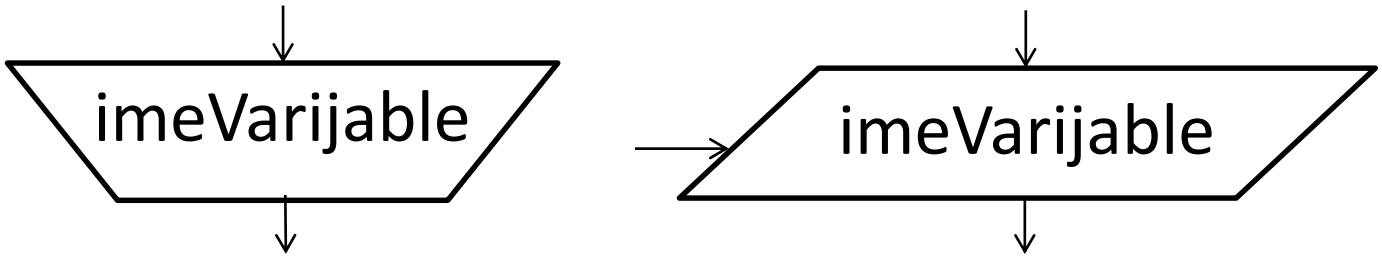
Dijagram toka – Blokovi

- Veći dijelovi dijagrama toka se međusobno spajaju poveznicama
  - Logička poveznica pojedinih dijelova
  - U simbol se stavlja jednoznačna oznaka
    - Iste oznake predstavljaju isto mjesto u dijagramu toka
  - Početak razdvajanja ima jednu ulaznu strelicu
  - Nastavak dijagrama toka ima jednu izlaznu strelicu



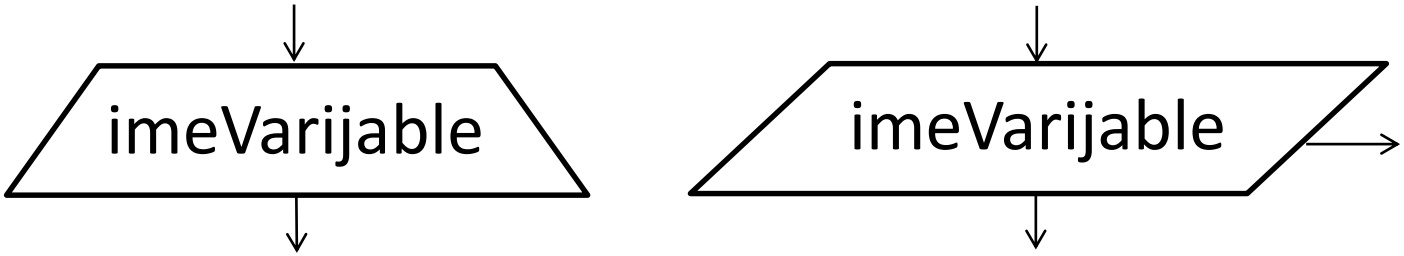
Dijagram toka – Blokovi

- Unos podatka u dijagram toka
  - Nekoliko inačica simbola
    - Neki programi za izradu dijagrama toka koriste svoje simbole
  - Jedna ulazna i jedna izlazna strelica
  - U blok se upisuje ime varijable
    - Vrijednost koju unosi operater se sprema u navedenu varijablu
    - Moguće unijeti više varijabli
      - Vrijednosti se spremaju prema redoslijedu unosa i redoslijedu varijabli u bloku



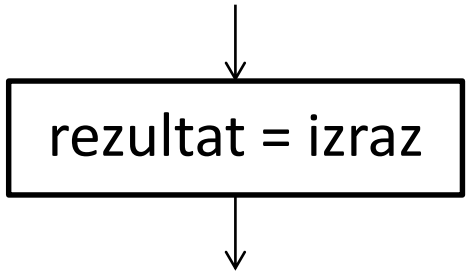
Dijagram toka – Blokovi

- Ispis podataka u dijagramu toka
  - Nekoliko inačica simbola
    - Neki programi za izradu dijagrama toka koriste svoje simbole
  - Jedna ulazna i jedna izlazna strelica
  - U blok se upisuje ime varijable
    - Blok ispisuje vrijednost spremljenu u varijabli na zaslon računala
    - Moguće unijeti više varijabli
      - Vrijednosti se ispisuju prema redoslijedu varijabli u bloku



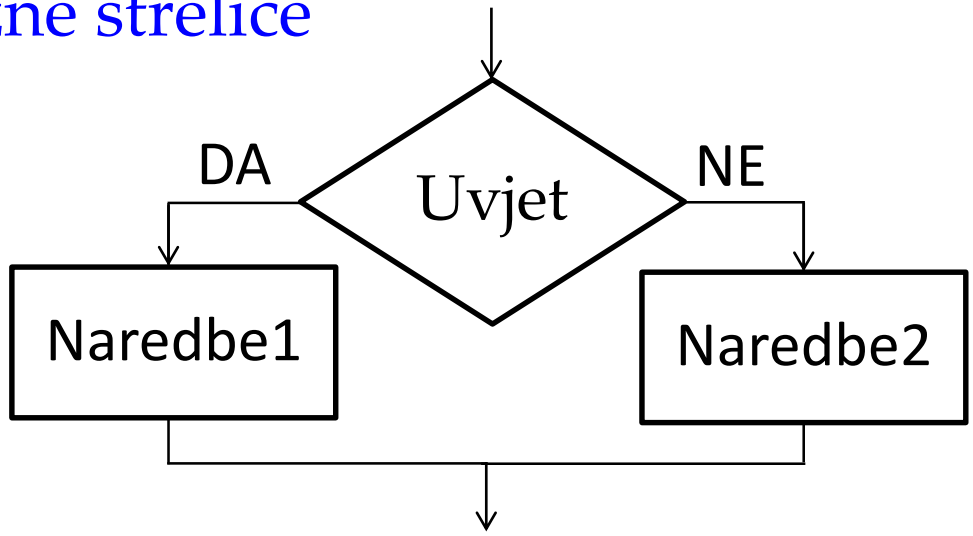
Dijagram toka – Blokovi

- Blok za izvršavanje aritmetičko-logičkih operacija
  - Jedna ulazna i jedna izlazna strelica
  - U blok se unosi izraz
    - Samo jedan izraz po bloku
    - Obavezno spremiti rezultat operacije u varijablu
  - Varijable u izrazu imaju biti postavljene na neku vrijednost prije izvršavanja bloka
  - Varijabla za rezultat se može koristiti prvi puta u ovom bloku



Dijagram toka – Blokovi

- Blok za grananje omogućuje donošenje odluke za odabir daljnjeg tijeka izvođenja programa
  - Blok za grananje omogućuje kreiranje petlje
  - Grananje ovisno o uvjetu
    - Dva moguća smjera -> uvjet ispunjen / nije ispunjen
    - Izvršava se samo jedan smjer odnosno grana
  - Jedna ulazna i dvije izlazne strelice
  - Nakon grananja se linije izvođenja spajaju ponovno u jedan smjer



Dijagram toka – Raptor