1.1. Algoritam i program

Pisanje i osmišljavanje programskog koda prema kojem će raditi neki program posao je **programera**. Prije samog pisanja koda programer prvo treba definirati problem te ulazno/izlazne veličine vezane za zadani problem. Na primjer, ako programira sustav za kontrolu pametne rasvjete u kućanstvu, ulazne veličine mogu biti senzori pokreta i svjetla putem kojih sustav dobiva informacije o prisutnosti ljudi i vanjskoj svjetlosti. Izlazna veličina bila bi upravljanje svjetlima u kući – paljenje, gašenje ili prilagođavanje jačine svjetla ovisno o uvjetima i potrebama korisnika.

Potom kad je definiran problem, programer piše algoritam i tek potom osmišljeni algoritam realizira u nekom od programskih jezika. **Algoritam** predstavlja konačan skup jednoznačno definiranih, logički povezanih koraka. Ima jasno definirane ulazne i izlazne veličine te mora davati točan izlaz za sve njegove dopuštene ulaze.

Algoritme je moguće zapisati na nekoliko načina: **u obliku teksta**, **pseudokodom**, s pomoću **dijagrama toka** ili **u nekom odabranom programskom jeziku**. Započet ćemo s prikazom algoritama s pomoću dijagrama toka u programu **Raptor** te paralelno u pseudokodu, a potom će ti primjeri biti realizirani i u programskom jeziku C.

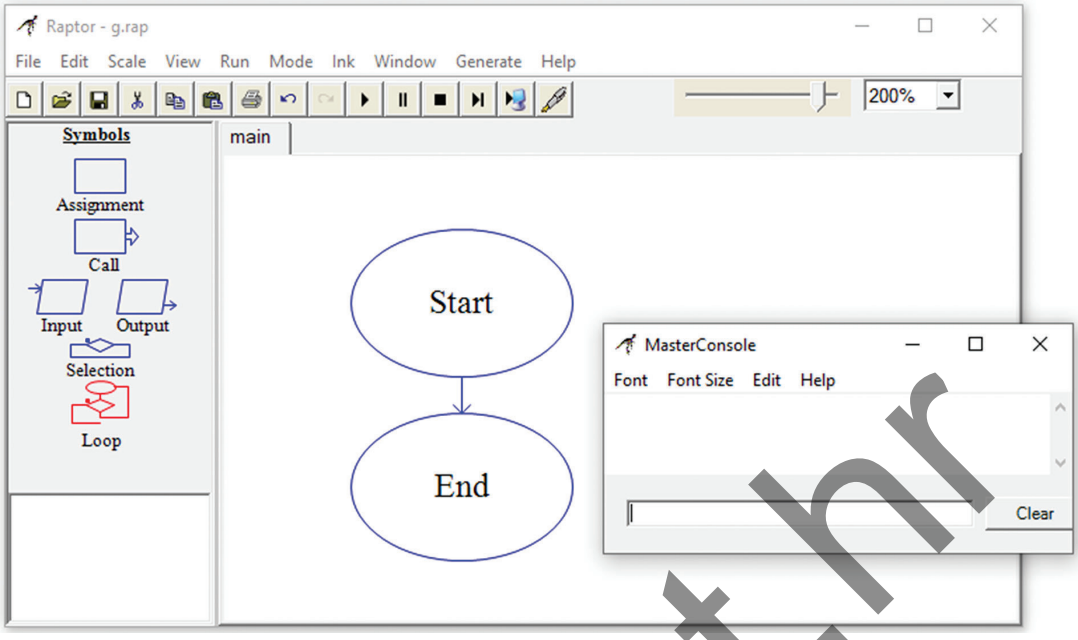
1.2. Pseudokod, dijagram toka i instalacija programa Raptor

U **pseudokodu** se zapisuju naredbe koje čine jednu logičku cjelinu te kojima je predstavljen algoritam. U tim naredbama često su uključeni i izrazi koji podsjećaju na matematičke formule.

Dijagram toka grafički prikazuje kako se odvija algoritam i to korak po korak. Dijagrame toka moguće je crtati u različitim programima kao što su primjerice Word, draw.io i sl. Za simuliranje izvođenja algoritma u ovom je udžbeniku za crtanje dijagrama toka odabran program Raptor.

Program je moguće preuzeti s poveznice:
https://raptor.martincarlisle.com/.

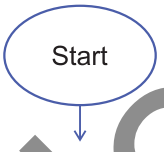
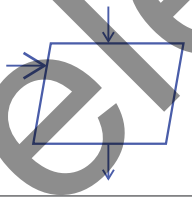
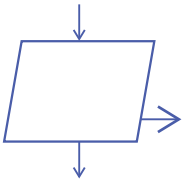
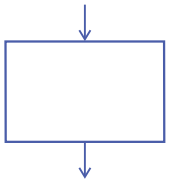
Nakon jednostavne instalacije te pokretanjem programa dobiva se prikaz kao na slici 1.1. U desnom kutu nalazi se padajući izbornik s mogućnošću povećavanja i smanjivanja veličine prikaza dijagrama (trenutačno je odabran prikaz na 200 %).



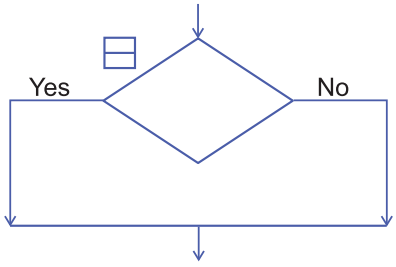
Slika 1.1. Program Raptor

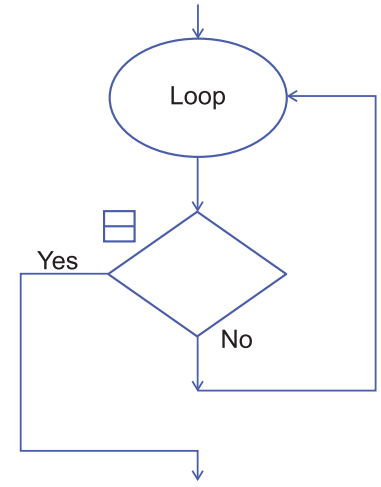
Simboli koji se upotrebljavaju prilikom izrade dijagrama toka te odgovarajuće naredbe u pseudokodu prikazani su u tablici 1.1.

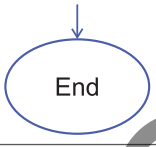
Tablica 1.1. Pregled i opis simbola u programu Raptor

simboli	naredba u pseudokodu	opis
	`početak`	početak algoritma
	`ulaz()`	definiranje ulaznih veličina
	`izlaz()`	definiranje izlaznih veličina
	`varijabla = izraz`	obrada podataka, pisanje naredbi

simboli	zapis u pseudokodu	opis

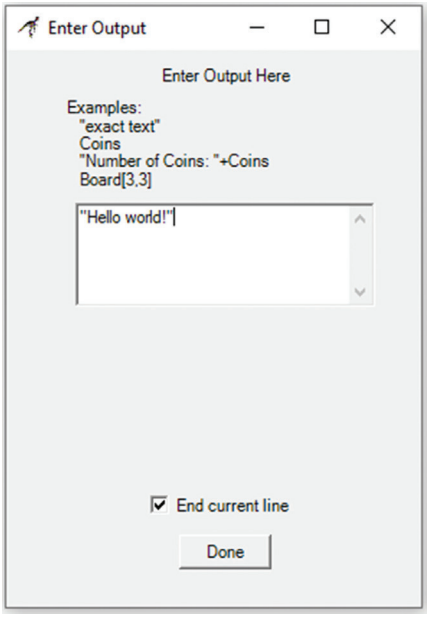
GRANANJE 	ako je uvjet onda
	  naredba
	
	ako je uvjet onda {
	  naredba_1
	  naredba_2
	}
		odluka se donosi na temelju uvjeta unutar romba, a izvršavanje se nastavlja jednim od dvaju mogućih smjerova (unutar romba piše se logički uvjet - primjeri u tablici 1.3)

PROGRAMSKA PETLJA 	dok je uvjet činiti
	  naredba
	
	dok je uvjet činiti {
	  naredba_1
	  naredba_2
	}
		naredbe se ponavljaju dok uvjet u rombu ne postane zadovoljen; kada postane zadovoljen, izlazi se iz petlje

	kraj	kraj algoritma

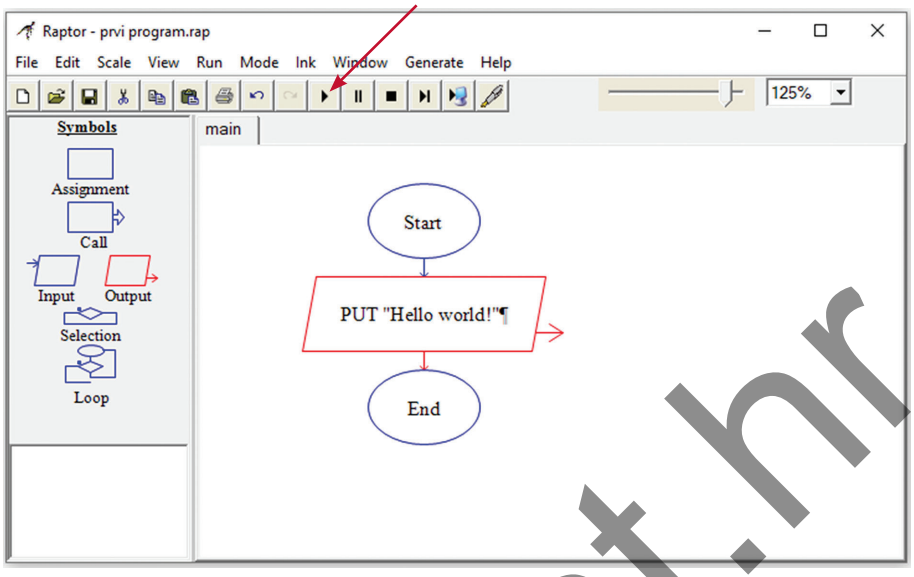
1.3. Pokretanje prvog programa u Raptoru

Slika 1.1 prikazuje izgled programa nakon pokretanja. Za prvo testiranje upotrebljava se jednostavan algoritam koji ispisuje poruku: **“Hello world!”**. U lijevom izborniku, među dostupnim simbolima potrebno je odabrati simbol za definiranje izlaznih podataka (*Output*). Taj simbol povuče se i postavi između eliptičnih simbola za početak (*Start*) i kraj (*End*) programa. Nakon što je simbol postavljen na radnu površinu, dvaput se klikne na njega kako bi se otvorilo polje za unos. U to polje unosi se željena poruka, unutar dvostrukih navodnika, koja će se prikazati na zaslonu računala (slika 1.2).



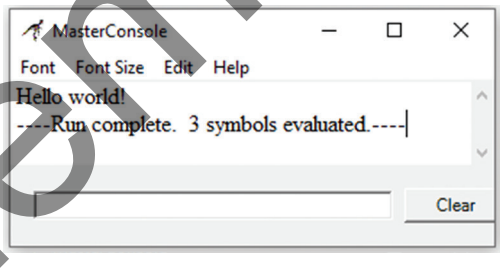
Slika 1.2. Upis teksta unosi se unutar dvostrukih navodnika

Klikom na *Done* te spremanjem programa pod nazivom *prvi program.rap* dobiva se prikaz kao na slici 1.3.



Slika 1.3. Izrada prvog dijagrama toka

Klikom na *Execute to Completion* (na taj simbol pokazuje strelica) pokreće se simulacija. Rezultat simulacije vidljiv je unutar prozora *MasterConsole* (slika 1.4). S pomoću klizača u gornjem desnom kutu radnog prozora moguće je definirati brzinu izvođenja simulacije. Prilikom izvođenja programa svjetlozelenom bojom uvijek je označen trenutačni simbol unutar kojeg se izvršavaju naredbe. Tako se lakše prati tok izvođenja algoritma.



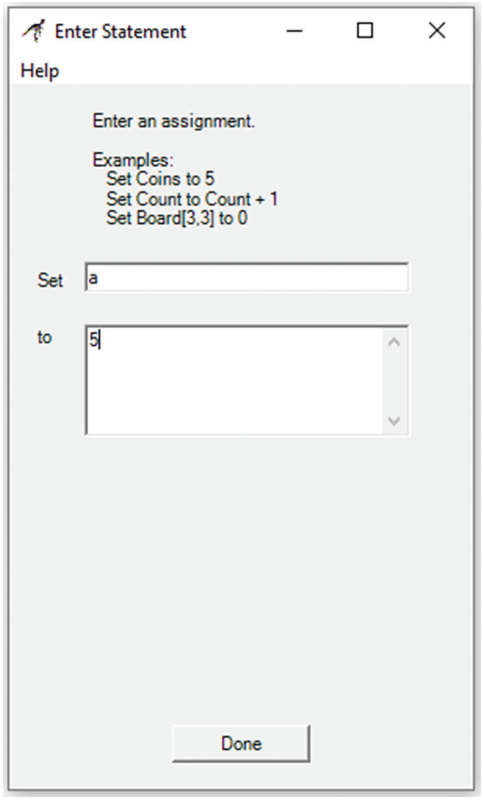
Slika 1.4. Prikaz rezultata izvođenja programa

1.4. Osnovna pravila prilikom pisanja algoritama u programu Raptor

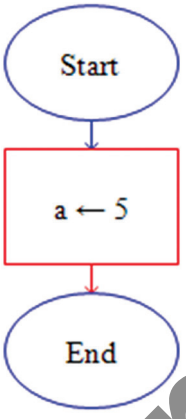
1.4.1. Varijable, naredba pridruživanja i ispis vrijednosti na zaslonu računala

Prilikom pisanja programa u pravilu uvijek treba **definirati i inicijalizirati varijable**. Svaka varijabla ima svoju memorijsku lokaciju, a sadržava vrijednost koja joj se pridruži.

Naredba pridruživanja radi se s pomoću bloka *Assignement*. Povlačeći blok na željenu poziciju u dijagramu toka, dva se puta klikne te se napiše naziv varijable koja se inicijalizira na željenu vrijednost (slika 1.5). Rezultat pridruživanja prikazuje slika 1.6.

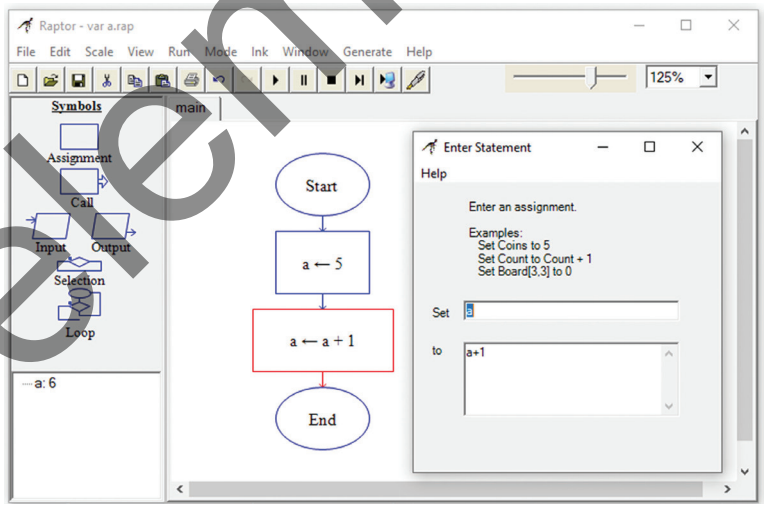


Slika 1.5. Definiranje i inicijaliziranje varijable



Slika 1.6. Prikaz definirane i inicijalizirane varijable *a* (naredba pridruživanja)

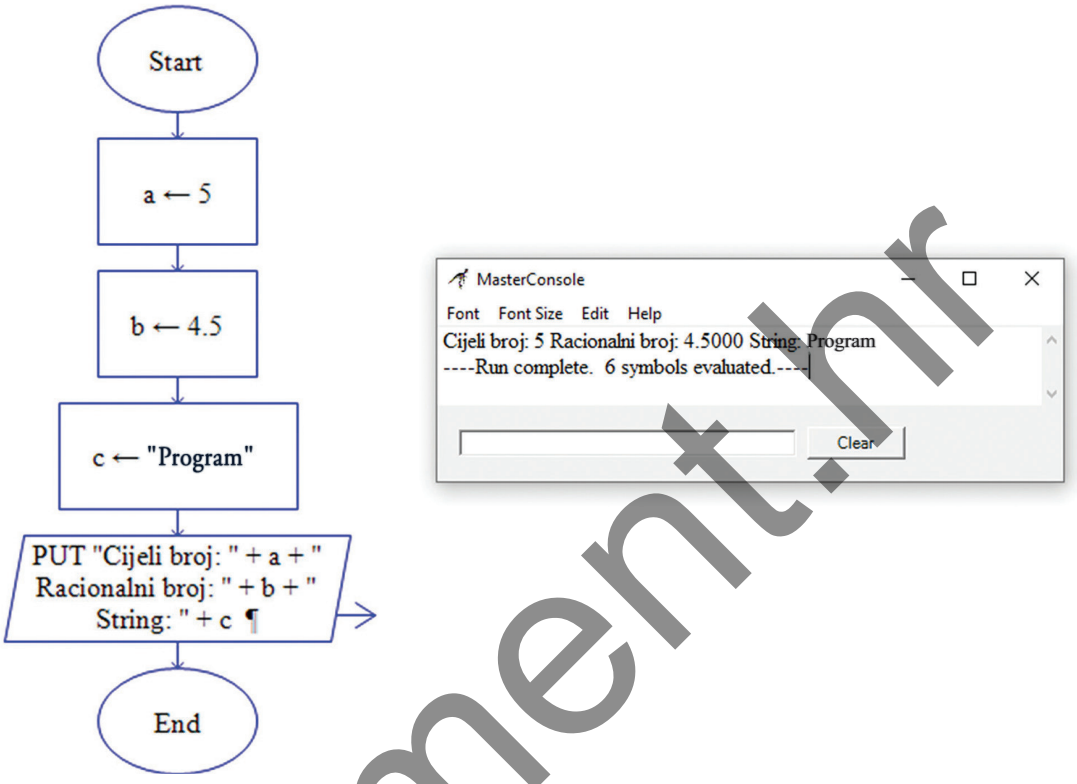
Ako želimo vrijednost varijable *a* uvećati za 1, može se dodati i crvenom bojom označen blok *Assignment* kao što prikazuje slika 1.7.



Slika 1.7. Primjer naredbe pridruživanja

Osim cjelobrojnih vrijednosti mogu se rabiti i numeričke vrijednosti s decimalnom točkom (racionalni brojevi) te nizovi znakova. Za ispis željenih vrijednosti upotrebljava se blok *Output*, što prikazuje slika 1.8. Nakon što dodamo blok *Output* na radnu površinu, dvaput se klikne na njega kako bi se otvorilo polje za unos. U to polje upisuje se tekst koji se želi

prikazati, unutar dvostrukih navodnika, a zatim se dodaje znak + i naziv varijable čija se vrijednost treba ispisati. Ako je potrebno upisati dodatni tekst, ponovno se upiše znak +, nakon čega se nastavlja unositi željeni tekst unutar dvostrukih navodnika. Ovim postupkom moguće je kombinirati tekst i vrijednosti varijabli u jednom ispisu.



Slika 1.8. Ispis svih vrijednosti varijabli na zaslonu računala

Algoritam prikazan na slici 1.8. s pomoću pseudokoda može se zapisati na sljedeći način:

```
početak
    a = 5
    b = 4.5
    c = "Program"

    izlaz("Cijeli broj: ", a)
    izlaz("Racionalni broj: ", b)
    izlaz("String: ", c)
kraj
```

Napomena

U pseudokodu se upotrebljava znak = za pridruživanje vrijednosti varijabli, dok se u aplikaciji Raptor za istu operaciju upotrebljava strelica (←).

1.4.2. Operatori, konstante i funkcije u Raptoru

Neki od osnovnih matematičkih operatora i funkcija koji se upotrebljavaju prilikom izrade početnih dijagrama prikazani su tablicom 1.2.

Tablica 1.2. Pregled operatora i matematičkih funkcija

opis		primjer pridruživanja	vrijednost sadržana u varijabli x
osnovna matematika	zbrajanje +	`x ← 2 + 3`	5
	oduzimanje –	`x ← 2 - 3`	–1
	množenje *	`x ← 2 * 3`	6
	dijeljenje /	`x ← 2 / 3`	0.6667
	potenciranje ** ili ^	`x ← 2 ** 3` `x ← 2 ^ 3`	8
	cjelobrojno dijeljenje s ostatkom `mod`	`x ← 2 mod 3`	2
	korjenovanje `sqrt(argument)`	`x ← sqrt(2)`	1.4142
	logaritmiranje po bazi *e* `log(argument)`	`x ← log(2)`	0.6931
	apsolutna vrijednost `abs(argument)`	`x ← abs(-2)` `x ← abs(2)`	2
	zaokruživanje na najbliži veći cijeli broj `ceiling(argument)`	`x ← ceiling(3.14)` `x ← ceiling(4)`	4
	zaokruživanje na najbliži manji cijeli broj `floor(argument)`	`x ← floor(3.14)` `x ← floor(3)`	3
	zaokruživanje na najbliži cijeli broj `round(x)`	`x ← round(3.6)`	4
trigonometrijske funkcije	sinus `sin(argument)` kosinus `cos(argument)` tangens `tan(argument)` kotangens `cot(argument)` *argument je kut u radijanima	`x ← sin(1.57)`	1.0000
konstante	broj pi – `pi`	`x ← pi`	3.1416
	broj e – `e`	`x ← e`	2.7183
	`true`	`x ← true`	1
	`false`	`x ← false`	0
	`yes`	`x ← yes`	1
	`no`	`x ← no`	0

Prilikom pisanja uvjeta u razgranatoj i cikličkoj algoritamskoj strukturi u simbolu romba upotrebljavaju se **relacijski** i **logički operatori**. S pomoću relacijskih operatora određuje se odnos između dviju vrijednosti, a kao rezultat dobiva se logička istina (*true*) ili logička laž (*false*). S pomoću logičkih operatora moguće je povezati više izraza koji sadržavaju relacijske operatore. Na taj se način pišu složeni logički izrazi. Pregled i opis relacijskih i logičkih operatora nalazi se u tablici 1.3. Navedeni operatori upotrebljavaju se kod pi-

sanja **uvjeta** kod razgranate i cikličke algoritamske strukture. U tablici 1.4. prikazan je redoslijed izvođenja operatora.

Tablica 1.3. Relacijski i logički operatori

opis		primjer	rezultat
relacijski operatori	jednako ==	`2 == 3`	`false`
	različito != ili /=	`2 != 3` `2 /= 3`	`true`
	manje od <	`2 < 3`	`true`
	veće od >	`2 > 3`	`false`
	veće od ili jednako >=	`2 >= 3`	`false`
	manje od ili jednako <=	`2 <= 3`	`true`

logički operatori	I and	`false and false`
		`false and true`
		`true and false`
		`true and true`
			`false`
			`false`
			`false`
			`true`

	ILI or	`false or false`
		`false or true`
		`true or false`
		`true or true`
			`false`
			`true`
			`true`
			`true`

	NE not	`not(true)` `not(false)`	`false` `true`

Tablica 1.4. Redoslijed izvođenja operatora

	operator
1.	zagrade ( )
2.	potenciranje (** ili ^)
3.	množenje i dijeljenje (*, /, mod)
4.	zbrajanje i oduzimanje (+, −)
5.	relacijski operatori (==, !=, <, >, <=, >=)
6.	logički operatori (not → and → or)

1.5. Osnovne algoritamske strukture zapisane u pseudokodu te prikazane dijagramom toka u programu Raptor

Program Raptor primjenjuje **tri osnovne algoritamske strukture**:

a) **linijska** algoritamska struktura (slijed)

b) **razgranata** algoritamska struktura (grananje)

c) **ciklička** algoritamska struktura (petlja).

Kod **linijske algoritamske strukture** naredbe u obliku blokova navode se jedna za drugom te se algoritam izvršava **slijedno**.

Kod **razgranate algoritamske strukture** algoritam se **grana** te se izvršava blok naredbi u onoj grani kako je definirano uvjetom koji se nalazi unutar simbola romba. Ako je uvjet istinit, izvršava se grana *Yes*, a ako nije, onda se izvršava grana *No*. Uvjet se upisuje u romb tako da se dva puta klikne na njega. U praznom polju pišu se uvjeti s pomoću aritmetičkih, relacijskih, logičkih i ostalih operatora. U *Yes* i *No* grane dodaju se blokovi naredbi. Slika 1.9 prikazuje dio algoritma koji sadržava blok za grananje programa. Pored prikaza grananja u programu dijagramom toka dan je i odgovarajući zapis u pseudokodu.