U nastavku je predstavljen tutorijal za pisanje programa u programskom jeziku definisanom za projekat iz predmeta Programski prevodioci.

# Sadržaj programskog jezika

Programski jezik za koji će biti implementiran kompajler sastoji se od različitih tipova podataka, direktiva i ostalih reči.

1.Tipovi podataka

**Funkcionalnost: num\_int**

**Ova funkcionalnost predstavlja**: deklaraciju prirodnih brojeva

**Ova funkcionalnost se piše na sledeći način**: sadrži prirodne(cele) brojeve koji nemaju tačku, u rasponu od -2,147,483,648(-2^31) do 2,147,483,648(2^31). Promenljivu kojoj želimo da dodelimo vrednost prirodnog broja označavamo sa num\_int ‘nazivpromenljive’.Vrednost promenljive je pozitivan ili negativan ceo broj.Ovi brojevi se mogu predstaviti i u oktalnom I heksadecimalnom zapisu.U programskom kodu to izgleda:

num\_int broj;

broj := 15 ;

**Funkcionalnost: num\_double**

**Ova funkcionalnost predstavlja**: deklaraciju realnih brojeva

**Ova funkcionalnost se piše na sledeći način**: sadrži brojeve sa pokretnim zarezom(decimalne,realne brojeve) u rasponu od 1.7E – 308 do 1,7E + 308.Promenljivu kojoj želimo da dodelimo vrednost decimalnog broja označavamo sa num\_double ‘nazivpromenljive’.Vrednost koju želimo da dodelimo realnoj promenljivoj mora da sadrži tačku.U programskom kodu to izgleda:

num\_double broj ;

broj := 101.55 ;

**Funkcionalnost: char**

**Ova funkcionalnost predstavlja**: deklaraciju karakternih promenljivih

**Ova funkcionalnost se piše na sledeći način**: sadrži pojedinačan karakter(mala i velika slova,specijalne karaktere poput ‘!’ , ‘@’ , ‘#’ , ‘$’ , ‘%’ , ‘^’ , ‘\*’ , ‘(‘ , ‘) , ‘+’ . ‘-‘ , ‘\’, ‘?’,itd.).Promeljivu za koju želimo da nam bude karakter označavamo sa char ‘nazivpromenljive’.Vrednost koju želimo da dodelimo karakternoj promenljivoj mora da se nalazi izmedju jednostrukih navodnik.U programskom kodu to izgleda:

char slovo = ‘a’ ;

char karakter = ‘@’ ;

**Funkcionalnost: strng**

**Ova funkcionalnost predstavlja**: deklaraciju tekstualnih promenljivih

**Ova funkcionalnost se piše na sledeći način**: sadrži niz karaktera promenljive dužine.Promenljivu koju želimo da označimo kao strng predstavljamo sa strng ‘nazivpromenljive’ . Vrednost koju želimo da dodelimo strng promenljivoj mora da se nalazi izmedju dvostrukih navodnika.Ukoliko se string sadrzi od vise reci,te reci su razdvojene crticom radi lakseg citanja programa. U programskom kodu to izgleda:

strng ime;

ime := “programski prevodioci” ;

char slovoO := ime[2] ;

**Funkcionalnost: array**

**Ova funkcionalnost predstavlja**: deklaraciju nizovnih promenljivih.

**Ova funkcionalnost se piše na sledeći način**: sadrži kolekciju elemenata istog tipa podatka,rasporedjenih u memoriji uzastupno.Promenljivi za koju želimo da nam bude niz,mora da se dodeli tip podatka(num\_int,num\_double) koji će se nalaziti u nizu,otvorena i zatvorena uglasta zagrada koja označava da je struktura niz i naziv promenljive gde želimo da čuvamo članove niza.Vrednosti koje želimo da dodamo nizovnoj promenljivoj moraju da se nalaze izmedju vitičastih zagrada({}) i da su medjusobno razdvojene zarezom.Odredjenom elementu niza pristupamo preko naziva niza,i njegovog indeksa,navedenim u uglastoj zagradi.U programskom kodu to izgleda:

num\_int[] niz ;

niz := {1,2,3,4,5} ;

prints niz[1] ;

2.Osnovne funkcije

**Funkcionalnost: main\_body**

**Ova funkcionalnost predstavlja**:glavno telo programa

**Ova funkcionalnost se piše na sledeći način**: sadrži sve naredbe koje želimo da se izvrše u našem programu.Ima svoj početak(komandu main\_body i otvorenu vitičastu zagradu ’{‘) i kraj(zatvorenu vitičastu zagradu ‘}’). U programskom kodu to izgleda:

main\_body{

num\_int broj := 10 ;

char karakter := ‘A’ ;

}

**Funkcionalnost: deklaracija**

**Ova funkcionalnost predstavlja**: definisanje promenljive

**Ova funkcionalnost se piše na sledeći način**: sadrži tip podatka koji želimo da dodelimo promenljivoj.Navodi se tip podatka a zatim posle jednog empty space razmaka se navodi naziv promenljive.U programskom kodu to izgleda:

num\_int broj ;

char karakter ;

**Funkcionalnost: scans**

**Ova funkcionalnost predstavlja**: naredbu za unos sa standardnog ulaza.

**Ova funkcionalnost se piše na sledeći način**: sadrži naziv promenljive(koja je prethodno deklarisana) koju će korisnik uneti sa standardnog ulaza.Promeljiva se nalazi posle ključne reči scans.U jednoj liniji se može učitati samo jedan tip podatka.U programskom kodu to izgleda:

num\_int broj ;

scans broj ;

**Funkcionalnost: prints**

**Ova funkcionalnost predstavlja**: naredbu za ispis na standardni izlaz.

**Ova funkcionalnost se piše na sledeći način**: sadrži naziv promenljive koju korisnik želi da ispiše na standardni izlaz.Naziv promeljive ili neceg drugog sto zelimo da ispisemo se nalazi posle kljucne reci prints.Osim prethodno deklarisanog tipa podatka,mogu da se ispisuju i pojedinačni karakteri pod jednostrukim navodnicima,kao i string pod dvostrukim navodnicima.U programskom kodu to izgleda:

num\_double broj ;

broj = 45.675 ;

prints broj ;

prints “programski-prevodioci” ;

3.Operatori

**Funkcionalnost: operator dodele(:=)**

**Ova funkcionalnost predstavlja**: dodeljivanje vrednosti nekoj promenljivoj.

**Ova funkcionalnost se piše na sledeći način**: sa leve strane operatora nalazi se naziv promenljive kojoj dodeljujemo vrednost,a sa njegove desne strane nalazi se vrednost koju želimo da dodelimo toj promenljivoj.Vrednost se može dodeliti i u istom redu gde se promenljiva deklariše. U programskom kodu to izgleda:

num\_int broj ;

broj := 5 ;

strng predmet := “prevodioci” ;

num\_int a := 45 ;

**Funkcionalnost: aritmetički operatori(+,-,/,%)**

**Ova funkcionalnost predstavlja**: matematičke operacije nad prirodnim i realnim brojevima.

**Ova funkcionalnost se piše na sledeći način**: sa leve I desne strane od operatora,sadrže promenljive nad kojima želimo da radimo aritmetičke operacije,ili konkretne brojeve s kojima želimo da operiramo.Ovi operatori mogu da se koriste samo nad istim tipom podatka.U programskom kodu to izgleda:

num\_int broj1 := 5 ;

num\_int broj2 := 15 ;

num\_int sabirak := broj1 + broj2 ;

num\_int razlika := 25 – 18 ;

num\_int proizvod := 15.55 \* 28.56 ;

**Funkcionalnost: relacioni operatori(<,>,<=,>=,==,!=)**

**Ova funkcionalnost predstavlja**: poredjenje vrednosti izraza.

**Ova funkcionalnost se piše na sledeći način**: sa leve I desne strane sadrži promenljive koje želimo da poredimo,(najčešće brojne promeljive).Kao rezultat se dobija vrednost 1(ako je izraz tačan) ili 0

(ako je izraz netačan).Najviše se koriste u kontrolnim strukturama.U programskom kodu to izgleda:

num\_int a := 15;

num\_int b := 20;

if (a>b) {

prints a ;

} elif (a<b) {

prints b ;

} elif (a==b) {

prints a ;

prints b ;

}

**Funkcionalnost: logički operatori(&,||,!)**

**Ova funkcionalnost predstavlja**: logičku obradu tačnosti dva ili više izraza.

**Ova funkcionalnost se piše na sledeći način**: sa leve i desne strane od logičkog operatora se nalaze promenljive čiju tačnost želimo da proverimo.U programskom kodu to izgleda:

a & b – operator & vraća vrednost 1 ako su oba uslova tačna,u suprotnom vraća 0

a || b – operator || vraća vrednost 1 ukoliko je bar jedan od uslova tačan,u suprotnom vraća 0

!a – operator ! se koristi za negiranje logičkog uslova

**Funkcionalnost: ternarni operator**

**Ova funkcionalnost predstavlja**: obrada uslovnih iskaza u jednom redu.

**Ova funkcionalnost se piše na sledeći način**: sadrži uslov(nalazi se izmedju obične otvorene i zatvorene zagrade),nakon čega se nalazi upitnik(?),koji označava proveru tog uslova.Nakon znaka pitanja,nalazi se izraz koji želimo da se izvrši ukoliko je uslov ispunjen,a nakon njega slede dve tačke(:),iza kojih se nalazi izraz koji želimo da se izvrši ukoliko je uslov netačan.U programskom kodu to izgleda:

(a>=b)?prints(a):prints(b) ;

4.Kontrole toka

**Funkcionalnost: if**

**Ova funkcionalnost predstavlja**: postavljanje nekih uslova kako bi se odredjene naredbe izvršile.

**Ova funkcionalnost se piše na sledeći način**: sadrži uslove(nalaze se izmedju običnih zagrada) koji moraju biti zadovoljeni kako bi se dalji blok naredbi izvršio.Početak bloka naredbi se označava sa otvorenom ivičastom zagradom({),a njihov kraj sa zatvorenom ivičastom zagradom(}).U programskom kodu to izgleda:

If (a<b) {

prints a ;

}

**Funkcionalnost: else**

**Ova funkcionalnost predstavlja**: izvršavanje naredbi za koje ne važe uslovi u if-u.

**Ova funkcionalnost se piše na sledeći način**: sadrži blok naredbi čiji se početak označava otvorenom ivičastom zagradom({),a završava se zatvorenom ivičastom zagradom(}).Prethodno mora da postoji if petlja na koju se else nastavlja.U programskom kodu to izgleda:

If (a<b) {

prints a ;

} else {

prints b ;

}

**Funkcionalnost: elif**

**Ova funkcionalnost predstavlja**: višestruke uslove sa zasebnim naredbama.

**Ova funkcionalnost se piše na sledeći način**: sadrži uslove(nalaze se izmedju običnih zagrada) za izvršenje bloka naredbi(nalaze se izmedju ivičastih zagrada).Prethodno mora da postoji if petlja na koju se elif nastavlja.Može da se nadje proizvoljan broj elif petlji,ali prethodno mora da postoji if petlja na koju se elif petlje nastavljaju.U programskom kodu to izgleda:

If (a<b) {

prints “A je manje od B” ;

} elif (a>b) {

prints “A je veće od B” ;

}else{

prints “A je jednako B” ;

**Funkcionalnost: while**

**Ova funkcionalnost predstavlja**: cikličnu petlju sa zadatim uslovom.

**Ova funkcionalnost se piše na sledeći način**: sadrži blok naredbi koje će se izvršavati ciklično sve dok je uslov ispunjen.Uslovi se nalaze izmedju običnih zagrada odmah nakon ključne reči while,a blok naredbi se nalazi izmedju ivičastih zagrada.U programskom kodu to izgleda:

while (a<b) {

a := a + 1 ;

b := b + 1 ;

}

**Funkcionalnost: for**

**Ova funkcionalnost predstavlja**: petlju za ponavljanje odredjenog bloka koda više puta.

**Ova funkcionalnost se piše na sledeći način**: sadrži blok naredbi koje će se izvršavati dok je uslov ispunjen.Nakon ključne reči for,nalazi se obična zagrada u kojoj će se nalaziti:

inicijalizacija-dodeljivanje početne vrednosti kontrolnoj promenljivoj

uslov-uslov koji se proverava u svakom koraku petlje,dok je uslov tačan petlja će se izvršavati

inkrementacija-povećavanje ili smanjivanje kontrolne promenljive nakon svakog koraka petlje,

koje su medjusobno razdvojene karakterom ; .Blok naredbi počinje sa otvorenom ivičastom zagradom ({),a završava se sa zatvorenom(}).U programskom kodu to izgleda:

for ( i:=0 ; i< a ; i++ ) {

prints i ;

}

**Funkcionalnost: do-while**

**Ova funkcionalnost predstavlja**: petlja sa ponovljenim prolaskom pod odredjenim uslovom.

**Ova funkcionalnost se piše na sledeći način**: sadrži naredbu do na početku iza koje stoji otvorena vitičasta zagrada koja označava početak petlje tj bloka naredbi.Zatvorena vitičasta zagrada označava kraj bloka naredbi,iza koje stoji ključna reč while,posle koje u običnoj zagradi stoj uslov za dalje izvršavanje petlje.U programskom kodu to izgleda:

num\_int broj ;

broj := 15 ;

do {

broj := broj – 1 ;

prints broj ;

} while(broj>0) ;