PASCAL

UPOTREBA SOFTVERA ZA PASKAL.

Fajl, folder i drajv.

Programi i uopšte podaci potrebni za rad računara drže se na spoljašnjoj memoriji (na disku). Ukupni sadržaj diska podijeljen je na logičke cjeline. Jedna takva cjelina naziva se fajlom, engl. file. Veličina fajla mjeri se u bajtima. Fajl ima svoje ime. Obični primjer za ime fajla je proba.pas. Za "proba" se kaže da je ime fajla u užem smislu. Do 8 karaktera u slučaju DOS, do 32 karaktera u slučaju Windows. Za "pas" se kaže da je ekstenzija. Do tri karaktera. Ekstenzija može i da odsustvuje. Po ekstenziji se poznaje vrsta fajla, po pravilu. Obični primjeri ekstenzija koje se često pojavljuju su: pas – program na paskalu, doc – dokument otkucan pomoću programa Word, htm – za Internet Explorer, itd. Najvažniji slučaj ekstenzije je "exe". To je izvršni fajl tj. program tj. fajl čijim pozivanjem će računar početi nešto da radi. Primjer: proba.exe.

Fajlovi su raspoređeni po grupama. Za jednu grupu se kaže da predstavlja jedan folder ili svejedno direktorijum, directory. Možemo zamisliti da je jedan fajl – jedan list papira, a folder – fascikla. Jedan folder, primjera radi f, sadrži u sebi nekoliko fajlova i još može da sadrži i neke foldere, primjera radi f1, f2 i f3. Tada se za f1, f2 i f3 kaže da su subfolderi od f. Vidimo da folder takođe ima svoje ime.

Obični primjeri spoljašnjih memorija su disketa (označava se kao A odnosno kao A:) i disk ili svejedno hard–disk (označava se kao C odnosno kao C:). Za jednu jedinicu spoljašnje memorije kaže se da predstavlja jedan drajv, engl. drive. Za pojedini drajv, podrazumijeva se njegov tzv. korijeni direktorijum, root directory. Korijeni direktorijum nema svog posebnog imena. Od njega sve "počinje", što se tiče sadržaja posmatranog drajva. To znači da se čitavi sadržaj drajva nalazi u njegovom korijenom direktorijumu.

Upotreba, instalacija i nabavka softvera za Paskal.

Upotreba. Slučaj Borland Turbo Pascal. Program se poziva, iz njegovog foldera, tako što se otkuca turbo, zatim pritisnuti enter. Svi opisani programi rade u msdos promptu ako je riječ o operativnom sistemu Windows 98, odnosno u command promptu ako je riječ o Windows XP. Početi ukucavanje programa, nakon što se uradi File, New. Na kraju ukucavanja, zapisati

program pomoću File, Save. Onda prevođenje programa pomoću Compile. Onda izvršavanje pomoću Run. Tokom rada programa, dobiti user screen pomoću Alt/F5. Takođe, tokom izvršavanja programa (tokom rada programa), unose se ulazni podaci, po potrebi. Iz softvera se izlazi pomoću File, Exit. Neprevedeni oblik programa zove se primjera radi proba.pas, naravno, a izvršni oblik će se naravno onda zvati proba.exe, kao što takođe već znamo.

Slučaj Irie Pascal. Prevođenje pomoću IPC proba onda enter, isto iz njegovog foldera. Izvršavanje pomoću IVM proba, onda enter, ako smo odabrali "proba" kao ime. Svejedno je kucali velikim ili malim slovima. Specifika razmatranog softvera je da izvršni oblik programa ima ekstenziju ivm (umjesto exe), kao proba.ivm (dok ostaje naziv tipa proba.pas).

Instalacija. Slučaj Borland Turbo Pascal. Unzipovati fajl tp55.zip, pomoću programa Winzip ili pomoću programa RAR. To se postiže pomoću File, Open archive i Actions, Extract. Unzipovani fajlovi biće u dva foldera čiji su nazivi disk1 i disk2. Treba staviti da svi fajlovi budu na jednom mjestu tj. u jednom folderu. Zatim pozvati INSTALL i onda enter otkucati. Kreiraće folder TP i ostalo. Iz foldera TP počinje rad, pomoću "turbo", prilikom kasnije upotrebe.

Slučaj Irie Pascal. Samo treba unzipovati fajl irie.zip. Dobiće se 15 fajlova u jednom folderu. Napominje se da, tokom upotrebe, za editovanje (ukucavanje) programa može da posluži edit onda enter (misli se iz prompta). Ili eventualno može da posluži ne onda enter. Bliže, edit proba ili ne proba.

Nabavka. Slučaj Borland Turbo Pascal. Preko Interneta, u smislu pomoću Google ili Yahoo, pronaći fajl tp55.zip i preduzeti ga (slobodno se preduzima). Ili ga preduzeti (presnimiti) sa našeg sajta www.pmf.ac.me sa mjesta Osnovne studije, Matematika i računarske nauke, Godina I, Računari i programiranje (ima li ga tamo). Ili da bi se poslalo preko mejla.

Slučaj Irie Pascal. Slobodno se može preuzeti sa Interneta. Ili se rečeni fajl irie.zip može presnimiti sa našeg sajta, sa maločas navedenog mjesta, istog. Ili iskoristiti disketu, gdje ima 15 fajlova, već su unzipovani, tako da je znači dovoljno da se oni prepišu na disk (u poseban folder).

Isprobati pomoću na primjer sljedećeg programa na paskalu:

```
program proba(input,output);
var n:integer;
begin
n:=12;
writeln('nesto');
writeln(n+n+1)
end.
```

Ponovo o upotrebi softvera Irie: iz command prompta redom:

```
edit proba.pas
ipc proba
ivm proba
```

Ponovo o instalaciji softvera Irie sa diskete: formirati jedan novi folder i u njega prepisati sve fajlove sa diskete.

JEDNOSTAVNI UVOD U PROGRAMSKI JEZIK PASCAL.

1. Osnovni sintaksni elementi paskala.

Osnovni sintaksni elementi su: imena, brojevi, specijalni simboli, rezervisane riječi i niske. Ime može da bude na primjer ime promjenljive. Svejedno je pisali veliko ili malo slovo u imenu. Ime počinje slovom, nakon čega može da slijedi niz alfa–numeričkih znakova (to znači slova ili cifara). Ime ne smije da se poklopi sa nekom rezervisanom riječi. Ako se ime poklopi sa nekim ugrađenim imenom, kakva su recimo SQR i SQRT, onda naravno ugrađeno ime gubi svoj smisao, u datom programu. Primjeri brojeva su 14 -1234 i 1.0E8 Ako učestvuje decimalna tačka onda se mora napisati bar jedna cifra ispred nje i bar jedna cifra poslije nje. Primjeri specijalnih simbola su $+$ $-$ $<$ i $<=$ Rezervisane riječi su AND ARRAY BEGIN ... WITH. Primjer za nisku je 'drugi komplement' Isto je primjer 'two"s complement' (kada sama niska sadrži znak apostrof).

Kaže se blanko ili prazno mjesto, a po potrebi se eksplicitno označava pomoću znaka ␣ kao u primjeru: `prvi␣broj`. Kaže se kraj reda ili prelazak u novi red ili eventualno "enter". Komentar se piše u obliku {objasnjenje} ili svejedno u obliku (∗objasnjenje∗) **Oznake za razdvajanje** su: blanko, kraj reda i komentar. Unutar osnovnog sintaksnog elementa ne smije da bude oznaka za razdvajanje. Ima jedan izuzetak od pravila: niska može da sadrži znake blanko. Vidimo da čitava niska mora da stoji u jednom redu.

Između osnovnih sintaksnih elemenata može da stoji ma koliko oznaka za razdvajanje. Između dva uzastopna imena ili rezervisane riječi mora stajati bar jedna oznaka za razdvajanje.

Navedimo jedan jednostavni primjer programa na paskalu:

```
program prog(input,output);
var i,j,n:integer;
begin
i:=10;
j:=14;
n:=i+j;
write(n);
n:=(i+j+100)div 2;
write(n)
end.
```

Prilikom izvršavanja programa dobićemo na ekranu: 24 62

Naš ukupni rad oko razmatranog jednostavnog programa sastoji se iz koraka:
unošenje ili editovanje teksta programa,
prevođenje programa na mašinski jezik (kompajler) i
izvršavanje programa.

2. Učitavanje i štampanje podataka.

U ovom naslovu daju se definicije ugrađenih potprograma za učitavanje ulaznih podataka, to su read i readln, i za štampanje (prikazivanje na ekran) izlaznih podataka (rezultata), to su write i writeln.

U opštem slučaju, "instrukcija" read zapisuje se kao read(ime,...,ime) gdje su ime,...,ime – imena promjenljivih. Smisao: pročitaj sa standardnog ulaza vrijednosti za nabrojane promjenljive. Slično tome zapisuje se i readln(ime,...,ime) Smisao: pročitaj sa standardnog ulaznog uređaja vrijednosti za nabrojane promjenljive i preskoči eventualne preostale znake do kraja

tekućeg reda. Tastatura predstavlja standardni ulazni uređaj. Drugim riječima, niz otkucanih znakova služi za dodjeljivanje vrijednosti, tokom izvršavanja programa. Takođe, otkucani znaci prikazuju se na ekranu. Primjer: razmotrimo instrukciju read(n1,n2,x1) u programu u kome su n1 i n2 deklarisane kao cjelobrojne promjenljive, a x1 kao realna promjenljiva. Uzmimo da je otkucano 10␣14␣12.34 Još treba otkucati i enter ako želimo da podaci budu prihvaćeni. Kakav je efekat prilikom izvršavanja? Promjenljivoj n1 biće dodijeljena vrijednost 10, n2 14, x1 12.34 Prilikom unošenja (kucanja) pojedine brojeve razdvajamo obično pomoću znakova blanko ili znakova za kraj reda. Drugim riječima, računar pregleda od tekućeg mjesta pa naprijed. Ako nailazi na blanko ili kraj reda, sve to preskače. Zatim preuzima odgovarajuće znake. Sve do znaka koji ne pripada broju (to je opet blanko ili kraj reda, po pravilu). Pomoću read i readln mogu se učitavati cijeli brojevi, realni brojevi i karakteri (podaci tipa char). Slično važi i za štampanje. Prilikom učitavanja jednog karaktera, preuzima se samo jedan znak iz ulaznog niza otkucanih znakova. Slično važi i za slučaj štampanja jednog karaktera.

Opšti oblik poziva potprograma write glasi write(value,...,value) Prikaži navedene veličine preko izlaznog uređaja. Vrsta veličina? Može da bude cijeli izraz kao recimo n+44 Samim tim može da bude i cjelobrojna promjenljiva kao recimo n ili cjelobrojna konstanta kao recimo 44 Isto tako izraz realnog tipa (real) kao 2.0∗x ili realna promjenljiva kao x ili realna konstanta kao 2.0 Slično, tipa char. Još može da bude i niska. Primjer write('n=',n,'x=',x)

Ako se napiše write(n:6) onda to znači da želimo da prikaz cijelog broja n zauzima 6 mjesta (širina polja na ekranu). U slučaju da je predviđeno polje preusko, računar neće ispoštovati našu želju. Broj će zauzeti potreban broj polja i znači prikazaće se pravilno. Slično u slučaju realnog broja kada se napiše primjera radi write(x:12) Instrukcija write(x:12:2) znači da broj treba da bude prikazan sa 2 mjesta iza decimalne tačke.

U slučaju writeln(value,...,value) još se na kraju uradi i radnja: tekuće mjesto se postavi na početak idućeg reda. Moguća je i instrukcija writeln

Primjer programa:

program p(input,output); var i,j:integer; begin read(i); read(j) end.

Ako unesemo 10 (enter) 20 (enter) 30 (enter) onda će postati i = 10, j = 20. A ako unesemo 10␣20␣30 (enter) onda će isto biti i = 10, j = 20. Međutim, neka je sada program malo promijenjen: uzmimo da umjesto read(i) sada stoji readln(i) Ako unesemo 10 (enter) 20 (enter) 30 (enter) onda sve isto kao maločas. A ako unesemo 10␣20␣30 (enter) onda će biti i = 10 a računar čeka da nešto otkucamo, da bi dodijelio vrijednost promjenljivoj j.

3. Naredba IF, sastavljena naredba i naredba CASE.

Uslovna naredba ima oblik IF uslov THEN naredba čiji je smisao ako je uslov tačan onda izvrši naredbu, ili IF uslov THEN naredba ELSE naredba čiji je smisao ako je uslov tačan onda izvrši prvu naredbu a ako nije tačan onda izvrši drugu.

Primjer programa:

```
program p(input,output);
var n:integer;
begin
readln(n);
if (n div 2)*2=n then writeln('paran')
else writeln('neparan');
if (10<=n)and(n<=99) then writeln('dvocifren')
end.
```

U opštem slučaju, program na paskalu sastoji se od dvije sekcije: dio koji sadrži deklaracije (promjenljivih i slično) i dio koji sadrži instrukcije (naredbe, radnje). Od program do begin i

od begin do end. Uzmimo da se druga sekcija sastoji od 4 naredbe. Tada ta sekcija ima oblik n1;n2;n3;n4 Vidimo da znak ; služi za razdvajanje naredbi. Imamo ekvivalentan program ako bismo napisali n1;n2;n3;n4; Samo što sada ima 5 naredbi. Drugim riječima, nakon posljednjeg znaka ; napisali smo jednu tzv. praznu naredbu. Slično ako bismo napisali n1;;n2;n3;n4 Vidjećemo da slične okolnosti važe u slučaju sastavljene naredbe.

Opšti oblik sastavljene naredbe glasi begin naredba;...;naredba end Drukčije se kaže složena naredba ili engl. compound statement. Prosto rečeno, sastavljena naredba je – jedna naredba. Koristi se kada treba da uradimo više radnji na mjestu gdje je dozvoljeno da se samo jedna naredba napiše. Primjer takvog mjesta je poslije riječi THEN u uslovnoj naredbi. Isto i poslije riječi ELSE.

Primjer programa:

```
program p(input,output); var n:integer; begin readln(n);
if (n div 2)*2=n then begin writeln('paran'); writeln('even') end
else begin writeln('neparan'); writeln('odd') end;
if (10<=n)and(n<=99) then begin writeln('dvocifren broj');
writeln('two digits number') end end.
```

Kada ima više slučajeva. Na engleskom se kaže izraz, vrijednost, naredba – expression, value, instruction. Opšti oblik naredbe CASE glasi

CASE izraz OF vrijednost: naredba;...;vrijednost: naredba END

Primjer programa:

```
program p(input,output);
var n:integer;
begin
write('o astronomiji');
write('kazi koliko ima planeta');
read(n);
case n of
7: write('to je malo');
8: write('dodaj jedan');
9: write('tacno');
10: write('smanji za jedan');
11: write('to je puno')
end;
write('nema vise')
end.
```

U definiciji, dozvoljeno je kao vrijednost da bude broj ili ime konstante (eventualno i sa predznakom −).

Takođe je dozvoljeno da na pojedinom mjestu umjesto jedne vrijednosti piše nekoliko vrijednosti, preko znaka ,

Mimo Wirth–ovog standarda programskog jezika Pascal, postoji i tzv. prošireni oblik naredbe CASE i glasi: neposredno ispred riječi END može da stoji OTHERWISE: naredba Da prevedemo: otherwise – u ostalim slučajevima.

```
case n of 9: writeln('tacno');
8,10: writeln('omasio za jedan');
otherwise: writeln('pogresno') end
```

4. Petlje FOR, WHILE i REPEAT.

Sintaksa (gramatika) FOR i:=e1 TO e2 DO n gdje je i – ime promjenljive, e1 i e2 – izrazi i n – naredba. Semantika (smisao) izvršiti n za i = e1, . . ., i = e2. U primjeru for i:=5 to 10 do write(i) biće 6 puta urađeno write. Moguće je da se n ne izvrši nijednom – kada je e1 > e2. Ili FOR i:=e1 DOWNTO e2 DO n Naniže.

WHILE uslov DO n ima smisao: ponavljati naredbu n sve dok je uslov ispunjen. Kada prestane da važi uslov, onda je i kompletirano izvršavanje napisane WHILE naredbe. Može se desiti da se n ne izvrši nijednom – kada u startu uslov nije ispunjen.

REPEAT n_1;. . .;n_k UNTIL uslov ima smisao: ponavljaj niz naredbi n_1;. . .;n_k sve dok uslov nije ispunjen. Kada uslov postane tačan, onda je i kompletirano izvršavanje navedene naredbe REPEAT. U svakom slučaju, niz naredbi n_1;. . .;n_k biće izvršen bar jednom, jer kontrola dolazi nakon naredbi. Vidimo da između rezervisanih riječi REPEAT i UNTIL može da bude više naredbi.

Zadatak. Računanje približne vrijednosti broja π, po Arhimedovom postupku. Razmotrimo krug poluprečnika $r = 1$, čiji je obim očito jednak 2π. Neka je $n \geq 3$. Razmotrimo pravilni n–tougao upisan u krug. Označimo njegovu stranicu sa a_n. Njegov obim očito je jednak na_n. Vidimo da je $\lim_{n\to\infty} na_n = 2\pi$. Uzeti broj $\frac{1}{2}na_n$, gdje je broj n dovoljno velik, kao aproksimaciju za π.

Lako se vidi da je $a_6 = 1$. Važi formula $a_{2n} = \sqrt{2 - \sqrt{4 - a_n^2}}$.

Napisati program na paskalu za računanje i štampanje veličine $\frac{1}{2}na_n$ za $n = 6$, $n = 12$, $n = 24$, $n = 48$.

Rješenje:

```
program brojpi(output);
var i,n:integer;
a,p:real;
begin
n:=6;
a:=1;
for i:=1 to 4 do
begin
p:=n*a/2;
writeln(i,n,a,p);
a:=sqrt(2-sqrt(4-a*a));
n:=2*n
end
end.
```

Biće odštampano:

```
 1    6   1.00000   3.00000
 2   12   0,51763   3.10582
 3   24   0.26105   3.13262
 4   48   0.13080   3.13935
```

Zadatak. Brojanje slova u riječi. Ulazni podatak programa treba da bude niz slova. Kao oznaka kraja, neka posluži znak =. Izbrojati koliko puta se pojavljuje slovo A. Isto pitanje za slovo B.

Rješenje:

```
program slova(input,output);
var broja,brojb:integer;
ch:char;
```

```
begin
broja:=0; brojb:=0;
read(ch);
while ch<>'=' do
begin
if ch=chr(65) then broja:=broja+1;
if ch=chr(66) then brojb:=brojb+1;
read(ch)
end;
writeln(broja,brojb)
end.
```

Ako unesemo MATEMATIKA= onda ćemo dobiti 3 0

5. Rad sa nizom (ARRAY).

Niz je primjer promjenljive (vrste podataka) koja se sastoji iz više komponenti ili se svejedno kaže da je to jedna tzv. složena promjenljiva. Treba reći o deklaraciji niza i o oznaci za člana niza. Poslužimo se primjerom. Razmotrimo deklaraciju var x:array[1..15]of integer; Navedena deklaracija govori da će se u programu pojaviti niz čije je ime x, čiji indeksi mogu da budu od 1 do 15 i da je pojedini član niza – cio broj. U programu, vrše se radnje sa pojedinim članovima niza. Prema tome. potrebno je uvesti oznaku za člana niza, da bi neka naredba mogla da pristupi članu. U našem primjeru, član niza se označava kao x[i] ili x[1] ili x[2] i slično.

U našem primjeru radi se o nizu ili jednodimenzionom nizu – pojedini član ima jedan indeks. Moguć je i slučaj dvodimenzionog niza tj. matrice.

Zapaziti da je bolje kada bi se umjesto niz govorilo konačan niz.

Zadatak. Neka je $1 \leq n \leq 2^{15} - 1$. Učitava se n, a štampa se njegov prikaz u binarnom brojnom sistemu.

Rješenje:

```
program b(input,output); var i,j,n:integer; x:array[1..15]of integer; begin read(n);
for i:=15 downto 1 do begin x[i]:=n mod 2; n:=n div 2 end;
j:=1; while x[j]=0 do j:=j+1; for i:=j to 15 do write(x[i]:1) end.
```

6. Matrice.

Primjer deklaracije je var matr:array[1..10,1..10]of real; Tada je pojedini član matrice realan broj i označava se kao (recimo) matr[i,j]

Još treba 7. O potprogramima.

Šta je to potprogram? Jedan dio procesa obrade može da se izdvoji u posebnu cjelinu, da čini jedan potprogram. U paskalu postoje dvije vrste potprograma: opšti i funkcijski.

Treba objasniti: opšti mehanizam pozivanja potprograma, parametri po vrijednosti i po adresi, lokalne i globalne promjenljive i slučaj FUNCTION.

Opšti potprogram ima tekst oblika PROCEDURE PROC(X;Y;Z) deklaracije BEGIN naredbe END; gdje je uzeto da ima tri fiktivna parametra. U tekstu glavnog programa pojavljuje se naredba recimo PROC(A,B,C) gdje vidimo da su stvarni parametri označeni kao A, B i C. Prilikom izvršavanja navedene naredbe, vrši se pozivanje potprograma. Vrijednosti A, B i C predaju se promjenljivima X, Y i Z, redom. Sada radi potprogram. Povratak u glavni program ostvaruje se kada potprogram uradi svoje. Kaže se da se tri parametra predaju potprogramu

po vrijednosti (by value). Izmjene vrijednosti X, Y i Z do kojih dolazi tokom rada potprograma neće se odraziti na A, B i C.

Uzmimo sada da umjesto PROC(X;Y;Z) piše PROC(VAR X;Y;Z). Kaže se da se prvi parametar prenosi po adresi (by reference). Izmjena X u potprogramu u potpunosti mijenja vrijednosti promjenljive A.

Napominje se da imena fiktivnih i stvarnih parametara mogu da se poklapaju (ne utiče).

Uzmimo da je u okviru potprograma deklarisana neka promjenljiva recimo L. Za L se kaže da je jedna lokalna promjenljiva tog potprograma. Njena deklaracija važi samo u okviru njenog potprograma. Van njega, ona ne postoji. Suprotan slučaj imamo ako je neka promjenljiva recimo G deklarisana u programu (u glavnom programu, PROGRAM). Tada ona važi i u potprogramu. Za G se kaže da je jedna globalna promjenljiva. Vidimo da se globalne promjenljive mogu upotrebljavati za iznošenje rezultata iz potprograma, kao i za predaju vrijednosti potrebnih potprogramu. Zamišljamo da glavni program obuhvata potprograma, da je potprogram "podskup".

Primjer zaglavlja funkcijskog potprograma je FUNCTION F(X;Y;Z). Razlikuje se od opšteg potprograma u jednom elementu: posredstvom imena F predaje se jedna vrijednost programu. Za F se kaže da predstavlja rezultat. U tekstu potprograma, treba dodijeliti vrijednost promjenljivoj F. U programu, u okviru izraza (aritmetičkog izraza) može se pisati recimo F(A,B,C).

Na kraju, ako se primopredaja parametara vrši po vrijednosti onda je dozvoljeno da na mjestima stvarnih parametara stoje izrazi (aritmetički izrazi), bilo da se radi o opštem bilo o funkcijskom potprogramu. Primjer PROC(A,B1+B2,3.0) A ako se vrši po adresi onda to očito nije moguće.

Treba ustvari PROCEDURE PROC(X:REAL;Y:REAL;Z:REAL)

Treba ustvari function f(x:integer;y:integer;z:integer):integer; ili slično.

Primjer: funkcijski potprogram za računanje najvećeg zajedničkog djelioca dva broja (parametri po vrijednosti):

```
program gcd(output);
var m,n,rez:integer;
    function nzd(x,y:integer):integer;
    var pom,p,q:integer;
    begin
    while x<>y do
    begin
    if x<y then begin pom:=x; x:=y; y:=pom end;
    p:=y; q:=x mod y;
    x:=p; y:=q
    end;
    nzd:=x
    end;
begin
m:=100; n:=350; rez:=nzd(m,n); write(rez);
m:=100; n:=8; rez:=nzd(m,n); write(rez);
m:=100; n:=7; rez:=nzd(m,n); write(rez)
end.
```

Biće odštampano 50 4 1

Potprogram se temelji na formuli: ako je x>y onda je NZD(x,y) = NZD(y,x mod y).

PASCAL DA NAUČIMO

1. Uvod

Programski jezik PASCAL sastavili su C. A. R. Hoare i N. Wirth iz ETH u Cirihu. PASCAL možemo smatrati školskim primjerom lijepo izrađenog programskog jezika. Jezik sadrži niz korisnih načina koje možemo naći u drugim programskim jezicima i prirodno pomaže programeru da svoje programe piše koliko je moguće strukturno. Zlonamjerni kritičar bi pak podsjetio da u PASCALU nema ovog ili onog načina, ali je zato prevodilac mali i efikasan, isto kao i prevedeni programi.

Da bismo stekli predstavu o PASCALU, npišimo sljedeći program.

```
{Program inflacija računa koliko se smanji vrijednost neke valute za 1, 2, . . .,
10 godina, ako godišnja stopa inflacije iznosi 22%.}
program inflacija(outpit);
const n=10;
var i: integer; w: real;
begin i:=0; w:=1.0;
repeat i:=i+1;
w:=w/1.22; writeln(i,w)
until i=n end.
```

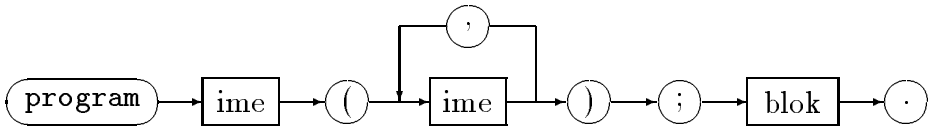
U svakom programskom jeziku, svaki program je u suštini sastavljen od dva dijela, koji ne moraju da budu vidno razdvojeni. Jedan dio opisuje događanja u programu i u PASCALU ga opisujemo naredbama, drugi dio opisuje strukture podataka koje naredbe obrađuju. U PASCALU je to razlikovanje veoma očito, pa je zato tako i realizovano.

Program je sastavljen iz dva dijela koje nazivamo "glava programa" i "blok". Glava programa samo ima zadatak da programu da ime i da možemo navesti imena datoteka koje će program upotrebljavati. Među tim imenima, ime "output" je obavezno.

Ostatak programa predstavlja blok. Blok je sastavljen od šest odjeljaka, čiji je redosljed propisan. Svaki odjeljak, sa izuzetkom zadnjeg, može biti prazan. Prvih pet odjeljaka predstavljaju "deklaracije" koje opisuju strukture podataka, dok šesti odjeljak, koji sintaksno gledano predstavlja "sastavljenu naredbu" sadrži sve naredbe.

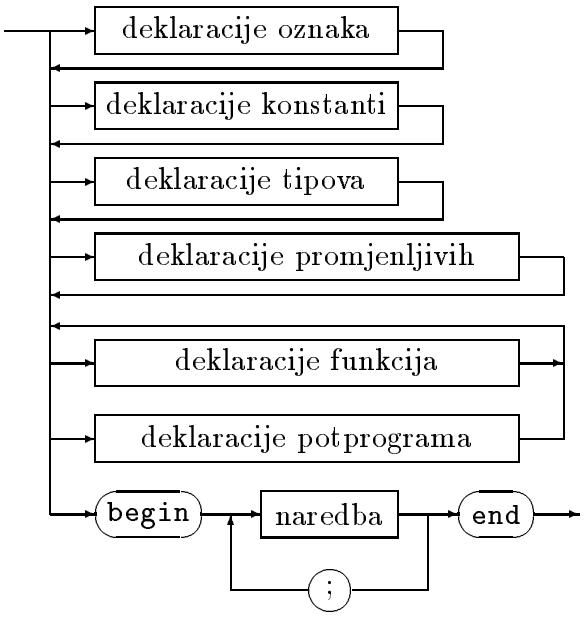
Sintaksu programskog jezika najljepše možemo opisati pomoću sintaksnih dijagrama, na primjer

program



Svaki put u smjeru strelice kroz sintaksni dijagram predstavlja sintaksno pravilnu konstrukciju, koju dijagram opisuje. Simboli napisani u ovalnim likovima su terminalni simboli i upravo tako se pojavljuju u programu. U pravougaonicima su upisana imena neterminalnih konstrukcija, za koje su sintaksna pravila napisana negdje drugo. Tako važi

blok



2. Lokalnost

Kasnije ćemo vidjeti da funkcije i potprogrami imaju skoro istu sintaksu kao i program. Pored ostalog, funkcije i potprogrami mogu imati i svoje deklaracije. Detaljnije, i funkcije i potprogrami su sastavljeni tako da za glavom slijedi blok, u bloku lako može biti deklaracija. Pri tome važi sljedeće. Imena, deklarisana u jednom bloku, lokalna su bloku. To znači dvoje. Te veličine dostupne su samo bloku u kome su deklarisane, kao i blokovima unutar toga bloka. Ako u nekom bloku upotrebljavamo neku veličinu, koju u tom bloku nismo deklarisali, onda se pod njom podrazumijeva veličina deklarisana u okružujućem bloku. (Ako nije deklarisana ni u jednom okružujućem bloku, onda je program nepravilan.) Programer može u nekom unutrašnjem bloku ponovo deklarisati neko ime, iako je već deklarisano u spoljašnjem bloku. U tom slučaju imamo posla sa dvije veličine, koje međusobno nisu ni u kakvoj vezi. U unutrašnjem bloku važi unutrašnja deklaracija, a u spoljašnjem spoljašnja. Ovakav koncept nam omogućava da u svakom potprogramu možemo do mile volje upotrebljavati proizvoljna (lokalna) imena, a da se ne bojimo da će doći do bilo kakvog konflikta imena. S druge strane, imamo pravo da u potprogramu upotrebljavamo veličine koje su deklarisane van potprograma.

3. Razlika između PASCALA i drugih jezika

Ako upoređujemo PASCAL sa ALGOLOM 60, FORTRANOM ili PL/1 onda navodimo sljedeće značajne razlike.
– Deklaracije promjenljivih su obavezne.
– Neke ključne riječi, kao što su `begin`, `repeat` i slično, su rezervisane. To znači da smijemo da ih upotrebljavamo samo u onom značenju kako to PASCAL zahtijeva. U rukopisu te riječi podvlačimo, a na karticama te riječi pišemo potpuno normalno.
– Tačka–zarez se smatra za oznaku razdvajanja između dvije naredbe, a ne kao završetak naredbe (PL/1).
– Standardni tipovi podataka su cijeli brojevi, realni brojevi, logičke vrijednosti i znaci. Za strukturiranje podataka, PASCAL poznaje nizove, zapise (strukture u COBOLU i PL/1), skupove i datoteke. Te strukture lako kombinujemo u zapise sastavljene od nizova i skupova, u datoteke zapisa i slično. Strukture podataka možemo kreirati dinamički i pristupati im pomoću

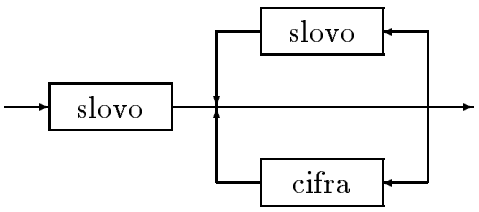
pokazivača. Dalje, dozvoljeno je da se definišu novi bazni tipovi podataka, čije vrijednosti su simboličke konstante.
– Struktura podataka "skup" (`set`) odgovara približno pojmu "bit string" u PL/1.
– Nizovi imaju proizvoljno mnogo indeksa, koji su proizvoljni, samo su granice konstantne. Nema dinamičkih nizova.
– Oznake naredbi su prirodni brojevi i treba da budu deklarisane.
– Sastavljena naredba je ista kao u ALGOLU 60.
– Naredba `case` nadoknađuje `switch` iz ALGOLA 60 i izračunatu GO TO naredbu iz FORTRANA.
– Naredba `for` odgovara naredbi DO u FORTRANU i može imati $+1$ ili -1 kao korak. Granice mogu biti i takve da se jezgro naredbe uopšte ne izvrši.
– Nema uslovnih izraza, ni višestrukih pridruživanja.
– Dozvoljeno je rekurzivno pozivanje potprograma i funkcija.
– Parametri potprograma prenose se po vrijednosti ili po referenci, ali ne mogu po imenu.
– Sve objekte moramo deklarisati prije upotrebe. Izuzeci su deklaracije tipova prilikom deklaracije pokazivača i poziva funkcije ili potprograma, ako se za to posebno pobrinemo.

4. Osnovni sintaksni elementi

Osnovni sintaksni elementi u PASCALU su imena, brojevi, specijalni simboli, rezervisane riječi i niske.

Imena koristimo za označavanje konstanti, tipova, promjenljivih, potprograma i funkcija. Ime je proizvoljne dužine, a obrazujemo ga po sljedećem pravilu

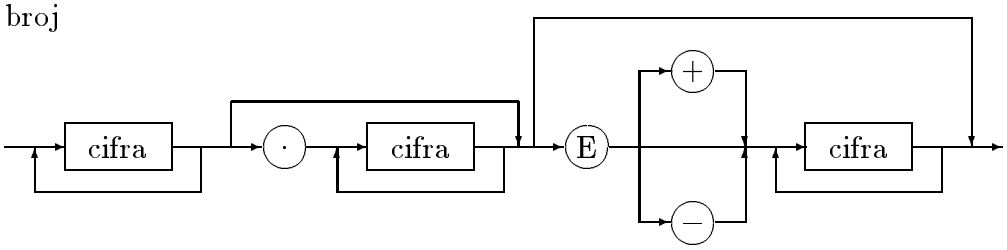
ime



Prevodilac se obavezuje da razlikuje imena koja se međusobno razlikuju u prvih osam znakova. Neka imena su unaprijed deklarisana, kao na primjer imena ugrađenih funkcija i slično. Za razliku od rezervisanih riječi, imamo pravo da ta imena ponovo deklarišemo sa različitim značenjem, u skladu sa lokalnošću imena. U sintaksnim pravilima navodimo kvalifikovana imena, na primjer "ime promjenljive", "ime konstante" i slično. To znači da ime o kome se radi mora da bude na odgovarajući način deklarisano.

Brojeve izražavamo u dekadnom sistemu, a mogu biti cijeli ili realni. Realni brojevi mogu imati i dekadni eksponent. Sljedeći sintaksni dijagram daje sintaksu brojeva

broj



Vrijednosti navedenih brojeva treba shvatiti onako isto kao u FORTRANU. Posebno valja podvući da iz dijagrama slijedi da ispred decimalne tačke, kao i iza nje, mora biti bar jedna cifra.

PASCAL koristi sljedeće specijalne simbole + − * / := . , : = ; < <= > >= <> [ ] ↑ .. Kada je specijalni simbol sastavljen od više od jednog znaka kao recimo := onda tako sastavljeni simbol treba da razumijemo kao nedjeljivu cjelinu.

Rezervisane riječi PASCALA su `and array begin case const div do downto else end file for function goto if in label mod nil not of or packed procedure program record repeat set then to type until var while with`

Niska je niz proizvoljnih znakova zaokružen između dva apostrofa, na primjer 'ovo je niska' Ako želimo da se u niski pojavi apostrof onda ga napišemo dvaput, na primjer
'i znak '' moze biti dio niske'

Komentar je proizvoljan niz znakova, sa izuzetkom znaka "}", koji se zaokružuje vitičastim zagradama, na primjer {ovo je komentar.} Umjesto vitičastih zagrada, može se koristiti kombinacija znakova (* i *)

Blanko, kraj reda i komentar valjaju kao oznake razdvajanja. Unutar osnovnog sintaksnog elementa ne smije se pojaviti oznaka razdvajanja. Izuzetak je niska, u kojoj se smiju pojavljivati blanko i vitičaste zagrade (koje tada ne znače komentar), s tim da čitava niska mora biti u jednom redu. Između osnovnih sintaksnih elemenata stoji po volji mnogo oznaka razdvajanja, a između dva uzastopna imena, broja ili rezervisane riječi mora stajati bar jedna oznaka razdvajanja.

5. Osnovne vrste struktura podataka

Pod strukturama podataka podrazumijevamo informacije koje program obrađuje. Svakako da su sve strukture podataka prikazane u memoriji računara kao kombinacije bita. Viši programski jezici pružaju veliku apstrakciju, tako da se može raditi sa raznim vrstama struktura podataka.

Tip strukture podataka definiše skup vrijednosti koje promjenljiva može uzimati. Iako strukture podataka u PASCALU mogu da budu veoma komplikovane, one su u svakom slučaju sastavljene od osnovnih struktura podataka. Te osnovne strukture podataka nazivamo skalarnim strukturama podataka, odnosno kažemo da su skalarnog tipa. PASCAL poznaje više skalarnih tipova: cjelobrojni (integer), realni (real), logički ili Booleov (Boolean), znakovni tip (char), kao još i cijelu skupinu skalarnih tipova koje sam programer može da deklariše.

5.1. Tip Boolean

Tip Boolean raspolaže sa samo dvije vrijednosti, true i false. Za Booleove veličine su definisane logičke operacije `and` logička konjunkcija, `or` logička disjunkcija, `not` logička negacija.

Imamo pravo da Booleove vrijednosti upoređujemo pomoću svih šest relacionih operacija = jednako, <> različito, <= manje ili jednako, >= veće ili jednako, < manje, > veće. Pri tome važi false < true.

I tri standardne funkcije daju Booleovu vrijednost:
odd(x) true, ako je cio broj x neparan, a inače false,
eoln(f) i eof(f) upoznaćemo kasnije.

5.2. Tip integer

Tip integer čine cijeli brojevi. Koji su to brojevi, zavisi od implementacije. Kao standardna konstanta postoji maxint čija vrijednost predstavlja najveći cio broj sa kojim se još da pravilno računati.

Za cijele brojeve PASCAL ima aritmetičke operacije:
$*$ množenje,
`div` cio dio količnika, u smislu 5 `div` 3 = 1, (−5) `div` 3 = −1,
`mod` ostatak pri dijeljenju, a `mod` b = a − ((a `div` b) $*$ b),
+ sabiranje,
− oduzimanje.

Relacijske operacije sa vrijednostima tipa integer daju nam Booleove vrijednosti.

Sljedeće standardne funkcije daju nam kao rezultat cijelu vrijednost: abs(x) apsolutna vrijednost cijelog broja x, sqr(x) kvadrat cijelog broja x, trunc(x) cio dio realnog broja x, trunc(3.7) = 3, trunc(−3.7) = −3, round(x) pravilno zaokružen realan broj x.

Formalno postoje još dvije funkcije: succ(i) nasljednik cijelog broja i, to je i+1, pred(i) prethodnik cijelog broja i, to je i−1, koje su u slučaju cijelih brojeva dosta suvišne.

5.3. Tip real

Tip real su realni brojevi. Do koje su veličine i koliki je stepen preciznosti prilikom reprezentacije, zavisi od implementacije.

Za realne brojeve PASCAL poznaje sljedeće aritmetičke operacije $*$ množenje, / dijeljenje, + sabiranje, − oduzimanje. Prilikom dijeljenja, bilo koji argument može biti cjelobrojan ili realan, rezultat je svakako realan. Kod ostale tri operacije, bar jedan argument mora biti realan, drugi smije da bude cijeli, da bi rezultat bio realan.

Relacijske operacije nam daju Booleovu vrijednost za proizvoljne kombinacije cjelobrojnih i realnih argumenata.

Funkcije abs(x) apsolutna vrijednost realnog broja i sqr(x) kvadrat realnog broja daju nam realnu vrijednost u slučaju realnih argumenata (a cjelobrojnu kada su argumenti cjelobrojni).

Sljedećih šest funkcija daće realan rezultat za cjelobrojne i za realne argumente sin(x) argument u radijanima, cos(x) argument u radijanima, arctan(x) glavna vrijednost u radijanima, ln(x) prirodni logaritam, exp(x) eksponencijalna funkcija, e^x, sqrt(x) kvadratni korijen, $\sqrt{x}$.

5.4. Tip char

Objekti tipa char su znaci koje je računar u stanju da napiše i učita. Koliko ima takvih znakova, koji su to i kako su uređeni, zavisno je od implementacije. U svakom slučaju, skup znakova obuhvata – sva slova, uređena su po abecedi, – sve cifre, uređene su po veličini i neposredno slijede jedna za drugom, – blanko.

Konstante tipa char zapisujemo kao niske sa jednim znakom.

Dvije ugrađene funkcije predstavljaju preslikavanja između znakova i prirodnih brojeva: ord(c) je prirodan broj koji odgovara znaku c, chr(i) je znak koji odgovara cijelom broju i.

Znaci su uređeni isto kao njihovi brojčani ekvivalenti ord(c). Imamo pravo da ih upoređujemo, kada se upoređuju njihovi brojčani ekvivalenti. U istom smislu su definisane funkcije pred(c) i succ(c), gdje je pred(c)= chr(ord(c)−1), succ(c) = chr(ord(c)+1).

5.5. Deklarisani skalarni tipovi

I sam programer može da deklariše nove skalarne tipove, čije su vrijednosti simbolička imena, na primjer:

tip operator sa vrijednostima plus, minus, puta, podijeljeno
tip pol sa vrijednostima masculinum, femininum

Za takva simbolička imena važi da su uređena onako kako su zapisana pri deklaraciji. Za njih su definisane funkcije pred(x) i succ(x) koje u ovom slučaju nisu suvišne, kao i funkcija ord(x) čija je vrijednost prirodan broj, tako da govori gdje je to ime, na primjer: succ(plus) = minus, pred(puta) = minus, ord(masculinum) = 1.

Tako deklarisani tipovi mogu se međusobno upoređivati. Svi izrazi plus < minus, masculinum <> femininum, minus = succ(plus) imaju vrijednost true.

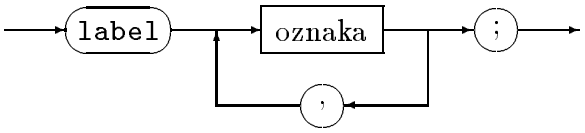
6. Deklaracije

U prvoj sekciji smo vidjeli da blok počinje sa nizom deklaracija, U ovoj sekciji ćemo razmotriti pojedine deklaracije.

6.1. Deklaracije oznaka

Svaka naredba u jezgru bloka može da ima oznaku. Oznake su cijeli brojevi bez predznaka i obrazovane su od najviše četiri cifre. Svaku oznaku koju koristimo moramo prethodno deklarisati pomoću sintaksnog dijagrama, po sintaksi

deklaracije oznaka

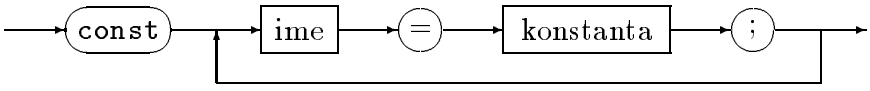


Na primjer `label` 23, 9999, 22;

6.2. Deklaracije konstanti

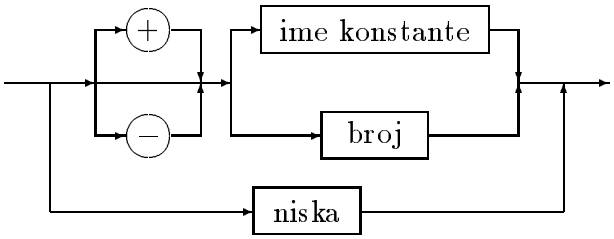
Pomoću deklaracija konstanti može se izvršiti korisna parametrizacija u programu. Na početku damo imena izabranim konstantama, pa ih zatim koristimo u programu preko imena (kao što važi i za unaprijed deklarisanu konstantu maxint). Prednost je u tome što konstantu možemo lako promijeniti. Treba je ispraviti samo jednom.

deklaracije konstanti



gdje je

konstanta



Primjer `const` a=273.18; b=−340;

cdcdcd=−a; xx= ' adresa ';

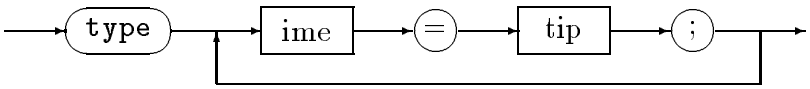
Primjer: Prilikom crtanja dijagrama na štampaču, zgodno je podesiti da bude odštampano po 6 redova i po 10 znakova na jedan palac, koji iznosi 25.4 mm. Početak tog programa može da glasi ovako

```
program crtanje(output);
label 2;
const nx = 6 {6 znakova po palcu u smjeru x};
ny = 10 {10 znakova po palcu u smjeru y};
palac = 25.4 {mm};
```

6.3. Deklaracije tipova

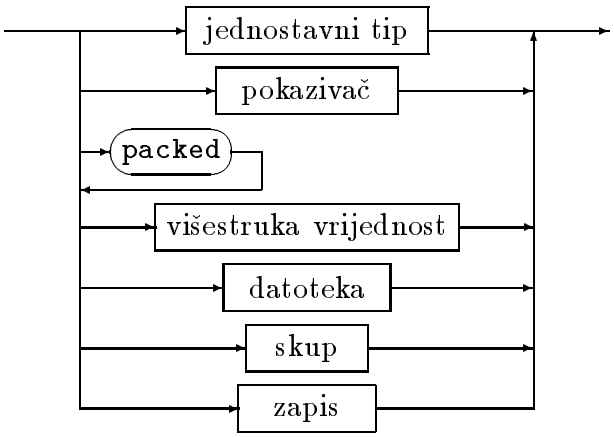
Pored ugrađenih tipova, i sam programer može definisati nove tipove i davati im imena. Za tu svrhu u PASCALU postoje deklaracije tipova

deklaracije tipova



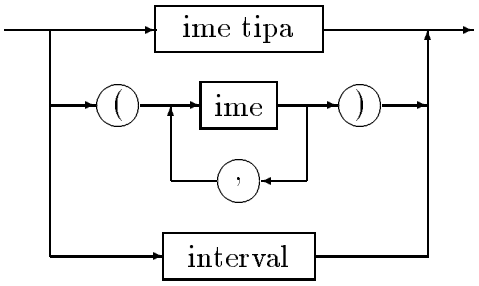
gdje je

tip



Sve tipove osim jednostavnih obradićemo kasnije, a za jednostavne važi

jednostavni tip



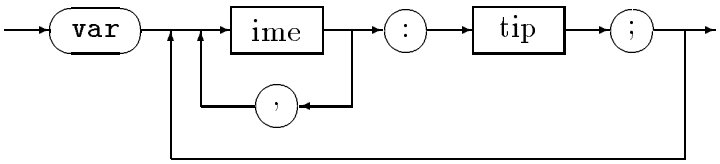
Opis intervala ostavljamo za kasnije, dok preostale dvije mogućnosti dozvoljavaju samo deklaraciju novog imena za već deklarisani tip i deklaraciju skalarnog tipa sa simboličkim imenima, na primjer

```
type operator = (plus, minus, puta, podijeljeno);
op = operator;
```

6.4. Deklaracije promjenljivih

Svaku promjenljivu koju koristimo u programu moramo prethodno deklarisati. Prilikom deklaracije kažemo njeno ime i njen tip. U istom bloku ne smijemo isto ime dvaput deklarisati, dok u drugom bloku isto ime može da bude još jednom deklarisano sa istim ili drukčijim smislom.

deklaracije promjenljivih



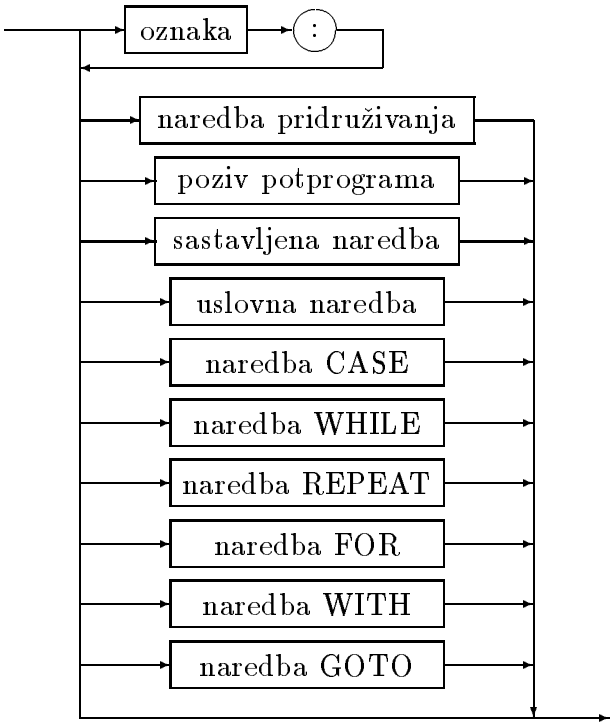
Primjeri: `var` i, j, p, q: integer;
x,y,z: real :
opr: operator ;

7. Naredbe

Pomoću naredbi opisujemo tok programa. Dijelimo ih na jednostavne naredbe (naredba pridruživanja, naredba goto i slično) i strukturirane naredbe (uslovna naredba, petlje i slično).

Iz sljedećeg dijagrama se vidi da ispred svake naredbe može da bude napisana oznaka, naravno samo one koje su deklarisane. Te oznake upotrebljavamo u naredbi GOTO, iako treba reći da u PASCALU postoji tako bogat skup strukturiranih naredbi da je naredba goto rijetko gdje potrebna.

naredba

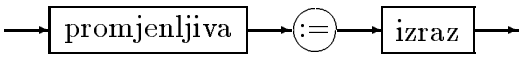


U nastavku ćemo detaljno razmotriti većinu tih naredbi.

7.1. Naredba pridruživanja

Pomoću naredbe pridruživanja dodjeljuje se izabranoj promjenljivoj nova, izračunata vrijednost nekog izraza.

naredba pridruživanja



Pri tome promjenljiva predstavlja neko ime deklarisano u "deklaraciji promjenljivih", dok se izraz obrazuje po standardnim receptima. Sintaksu izraza pogledaćemo kasnije. Zasad kažimo samo da PASCAL poznaje operacije četiri prioriteta

najviši `not`
`*` / `div` `mod` `and`
+ − `or`
najniži = <> < <= > >= `in`

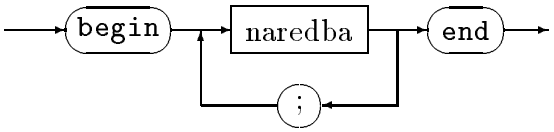
Ako se u izrazu pojavljuje uzastopno više operacija istog prioriteta onda važi da se izvode slijeva udesno. Tako je izraz a = b `or` c = d prije svega nepravilan, zato što se tumači kao a = (b `or` c) = d i treba ga napisati kao (a = b) `or` (c = d).

Pomoću naredbe pridruživanja mogu se dodjeljivati vrijednosti promjenljivima bilo kog tipa (osim tipa file). Izraz na desnoj strani mora da bude upravo istog tipa kao na lijevoj, sa popustom da izraz može biti cjelobrojan, ako je promjenljiva realna.

7.2. Sastavljena naredba

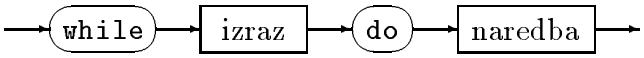
Često gramatika PASCALA dozvoljava da se na određenom mjestu napiše samo jedna naredba, a mi želimo da ih napišemo nekoliko. Potrebne naredbe u tom slučaju okružujemo simbolima `begin` i `end` čime dobijamo da to postane, sintaksno posmatrano, samo jedna naredba.

sastavljena naredba



7.3. Naredba WHILE

naredba WHILE



Naredba WHILE je najelegantnija naredba za obrazovanje petlji. Djeluje tako da se "naredba" ponavlja više puta, sve dok je Booleov izraz "izraz" jednak true. (Možda to ne bude nijednom.)

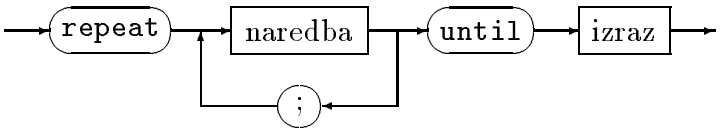
Primjeri:

```
while ch = blank do read(ch) ;

while abs(y − x) > eps do
      begin
          x := y;
          y := f(x)
      end ;
```

7.4. Naredba REPEAT

naredba REPEAT



Naredba REPEAT je veoma slična naredbi WHILE. Razlike su sljedeće: – u naredbi REPEAT jezgro smije da se sastoji iz nekoliko naredbi, čime uštedimo na pisanju simbola `begin`–`end`, – naredba će se izvršiti bar jednom. Detaljnije, niz naredbi između `repeat` i `until` ponavlja se toliko puta, sve dok logički izraz "izraz" ne postane true.

Pomoću naredbe REPEAT možemo prethodne primjere zapisati ovako:

`repeat` read(ch) `until` ch <> blank ;

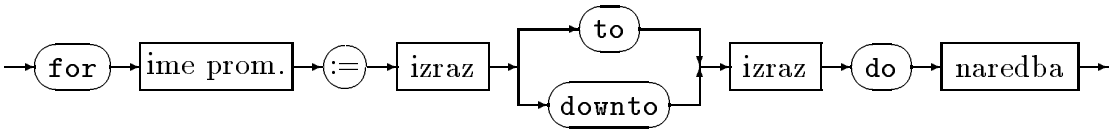
`repeat` x := y; y := f(x) `until` abs(x − y) <= eps ;

Dva primjera sa "read(ch)" mogu se riječima izraziti ovako: while: Pronađi prvi znak različit od blank, repeat: Pronađi prvi sljedeći znak različit od blank.

7.5. Naredba FOR

Naredba FOR odgovara naredbi DO u FORTRANU. Koristi se kada unaprijed znamo koliko puta treba izvršiti petlju.

naredba FOR



Izvođenje naredbe FOR možemo opisati na sljedeći način. Konstrukcija `for` v := e1 `to` e2 `do` S ekvivalentna je konstrukciji

`if` e1 <= e2 `then`
`begin` v := e1; S; v := succ(v); S; ... v := e2; S `end`;
v := nedefinisano

i analogno za drugu varijantu, kada stoji `for` v := e1 `downto` e2 `do` S što je ekvivalentno konstrukciji

`if` e1 >= e2 `then`
`begin` v := e1; S; v := pred(v); S; ... v := e2; S `end`;
v := nedefinisano

Kontrolna promjenljiva, početna vrijednost i krajnja vrijednost treba svi da budu istog skalarnog tipa (ne smiju biti realni) i te veličine ne smije da mijenja naredba u jezgru.

Primjeri:

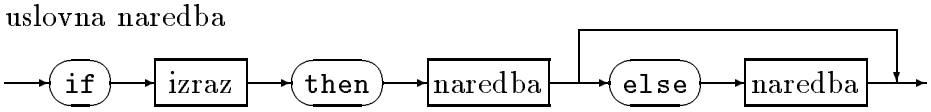
```
for i := 1 to n do s := s + 1/i

for j := m + n downto m − n do
    for k := j + 1 to m + n do
      begin s := s − j/k; t := t + k/j end

for spol := masculinum to femininum do test
```

7.6. Uslovna naredba

uslovna naredba



U kraćem obliku uslovna naredba znači da će se naredba, koja slijedi nakon simbola `then`, izvršiti samo pod uslovom da je vrijednost Booleovog izraza, koji dolazi nakon simbola `if`, jednaka true, na primjer `if` a < b `then begin` x := y; y := z `end`

U dužem obliku imamo dvije naredbe, dvije alternative. Izvršiće se prva ako je vrijednost Booleovog izraza jednaka true, a druga ako je vrijednost tog izraza jednaka false, na primjer

```
if x = 0 then error else
   begin
     y := a / x ;
     z := a + b
   end
```

Sljedeća naredba je dvosmislena: `if` p `then if` q `then` A `else` B jer je možemo shvatiti kao

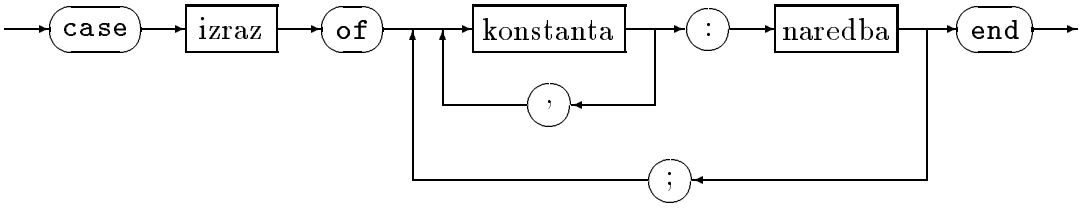
```
if p then
     begin if q then A else B end
```

ili pak kao `if` p `then begin if` q `then` A `end else` B Napravljeno je tako da, u takvim slučajevima, važi prva navedena varijanta.

7.7. Naredba CASE

Naredba CASE zamjenjuje izračunatu GO TO naredbu iz FORTRANA. U naredbi CASE možemo imati na raspolaganju više varijanti, od kojih želimo da izaberemo jednu. Naredba CASE je neuporedivo preglednija od izračunate GO TO naredbe, zato što su sve varijante napisane na jednom mjestu.

naredba CASE



Primjeri:

```
case i of
   0: x := 0 ;
   2: x := sin(x) ;
   1: begin t := 0; x := ln(x) end ;
   4: x := a + b
end
```

`case` ch `of` 'a','b','s': ch := succ(ch); 'p','q': ch := pred(ch); 'f': {prazna naredba} `end`

`case` spol `of` masculinum: `begin` ... `end`; femininum: `begin` ... `end end`

Izraz u naredbi CASE mora dati skalarnu vrijednost (ne realnu). Sve konstante, koje se pojavljuju kao oznake, moraju biti istog tipa kao izraz. Treba da budu međusobno različite.

Program teče ovako. Izračuna se vrijednost skalarnog izraza, zatim se odabere ona varijanta koja je označena sa konstantom koja ima tu vrijednost i uradi se odabrano, čime je naredba case okončana.

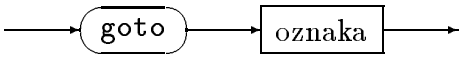
Treba napomenuti da konstante koje se pojavljuju kao oznake u naredbi CASE nisu oznake u sintaksnom smislu i ne treba ih deklarisati.

Ako se desi da izraz ima vrijednost koja nije navedena kao oznaka, onda je djelovanje programa nedefinisano.

7.8. Naredba GOTO

Uprkos veoma moćnim strukturiranim naredbama, u PASCALU se desi, mada samo ponekad, zaista samo ponekad, da je potrebno koristiti standardnu naredbu GOTO. Ona ima oblik

naredba GOTO



i znači da se program nastavlja sa označenom naredbom. Primjeri:

```
if x = 0 then goto 2;
goto 234;
```

8. Jednostavni primjeri

U ovoj sekciji sastavićemo nekoliko jednostavnih programa. Iako ćemo ponegdje za štampanje rezultata koristiti potprograme write i writeln, o njima ćemo govoriti u kasnijim sekcijama.

8.1. Pisanje rimskih brojeva

Sastavimo program koji će prikazati brojeve 1, 2, 4, 8, ..., samo da nisu veći od 5000, pomoću rimskih cifara.

```
program rimski(output);
const max = 5000;
var x, y: integer;
begin
y := 1;
repeat
      x := y; write(x, ' ');
      while x >= 1000 do
          begin write('m'); x := x - 1000 end;
      if x >= 500 then
          begin write('d'); x := x - 500 end;
      while x >= 100 do
          begin write('c'); x := x - 100 end;
      if x >= 50 then
          begin write('l'); x := x - 50 end;
      while x >= 10 do
          begin write('x'); x := x - 10 end;
      if x >= 5 then
```

```
            begin write('v'); x := x − 5 end;
        while x >= 1 do
            begin write('i'); x := x − 1 end;
        y := 2 * y; writeln
    until y > max
    end.
```

(Pregledno je ako posebni redovi i uvlačenje.)

8.2. Crtanje funkcije od dvije promjenljive

Pokušajmo da na štampaču nacrtamo sliku funkcije $f(x,y) = xy/(x^2+y^2+1)$ za $0 \le x \le 1$, $0 \le y \le 1$. To ćemo izvesti tako što ćemo tačke u kojima je $0 \le f < 0{,}05$, $0{,}1 \le f < 0{,}15$, $0{,}2 \le f < 0{,}25$ ili $0{,}3 \le f < 0{,}35$ otkucati sa zvjezdicama, dok ćemo tačke u kojima je $0{,}05 \le f < 0{,}1$, $0{,}15 \le f < 0{,}2$ ili $0{,}25 \le f < 0{,}3$ ostaviti neotkucanim.

```
program crtanje(output);
const nx = 10 {znakova po palcu};
ny = 6 {redova po palcu}; a = 10 {velicina slike u palcima};
var ix, iy, k: integer; x, y, f: real;
begin
page(output); {prelazak na novu stranicu}
for iy := 0 to a * ny do
begin
      write('         '); y := iy / (a * ny);
       for ix := 0 to a * nx do
           begin x := ix / (a * nx);
           f := x * y / (1 + sqr(x) + sqr(y)); k := trunc(20 * f);
           case k of 0, 2, 4, 6: write(' ');
           1, 3, 5: write('*') end end;
      writeln
end
end .
```

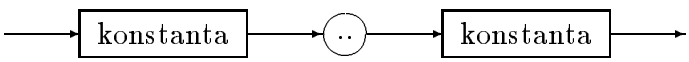
9. Intervali

Iz svakog skalarnog tipa, sa izuzetkom tipa real, možemo izdvajati podintervale, npr.

```
type
    dani = (po,ut,sr,ce,pe,su,ne);
    radni = po .. pe ; {podinterval dana }
    indeks = 0 .. 63 ; {podinterval cijelih brojeva }
    letter = 'a' .. 'z' ; {podinterval slova }
```

Sintaksa je veoma jednostavna

interval



Sa tipom interval dozvoljene su sve operacije koje su dozvoljene sa tipom iz koga je tip interval izdvojen, naravno uz ograničenje da intervalskoj promjenljivoj treba da bude pridružena vrijednost koja je legalna za interval.

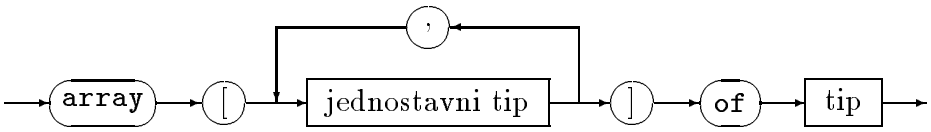
Kao što ćemo vidjeti, intervalske veličine koriste se kao indeksi u višestrukim vrijednostima, pa su zato neizbježne.

Pored toga, intervalske veličine su korisne zato što, uz svako pridruživanje, prevodilac doda kontrolu da li je vrijednost koja se pridružuje u tačnoj oblasti. O drugoj koristi saznaćemo kod upakovanih struktura.

10. Višestruke vrijednosti (tj. nizovi)

Više vrijednosti istog, a inače proizvoljnog, tipa možemo udružiti u niz. Pojedinačni elementi niza označavaju se pomoću indeksa, kojih može biti više. Kao indeks može poslužiti bilo koji skalarni tip, sa izuzetkom realnih brojeva.

višestruka vrijednost

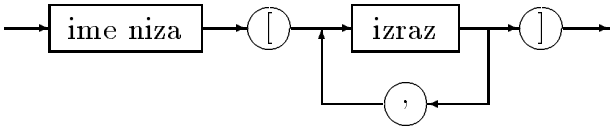


Primjeri:

```
type
    vektor = array [indeks] of real;
    matrica = array [indeks, indeks] of real;
    troskovi = array [dani] of integer;
var
    a : vektor;
    b : array [1 .. 25, 1 .. 30] of char;
```

Ako smo deklarisali promjenljivu tipa niz onda pojedinačnim elementima pristupamo tako što iza imena niza napišemo indekse u uglastim zagradama.

promjenljiva sa indeksom



Na primjer a[i+j] ili b[k+2, k+j−2]

Izrazi, koje upotrebljavamo kao indekse, moraju imati onakvu vrijednost kakvu indeks zahtijeva.

11. Zapisi

Zapisi predstavljaju drugo značajno sredstvo za grupisanje podataka. U zapis se može staviti proizvoljan broj objekata, koji mogu biti različitog tipa. Pojedinim poljima pristupamo pomoću selektora. Primjer:

```
var
datum : record
```

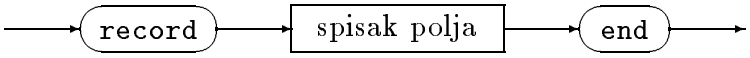
```
            mjesec : (jan,feb,mar,apr,maj,jun,jul,avg,sep,okt,nov,dec);
            dan : 1 .. 31;
            godina : integer
        end
```

Promjenljiva "datum" je zapis koji ima tri polja. Selektori polja su "mjesec", "dan" i "godina". Svako polje je drugačijeg tipa. Komponenti prilazimo tako što poslije imena zapisa stavimo tačku i odgovarajući selektor, na primjer

datum.mjesec := feb;
datum.dan := 23;
datum.godina := 2014

Polje u zapisu je proizvoljnog tipa, može biti i drugi zapis ili neki niz. Isto tako, možemo obrazovati i nizove čiji su elementi zapisi.

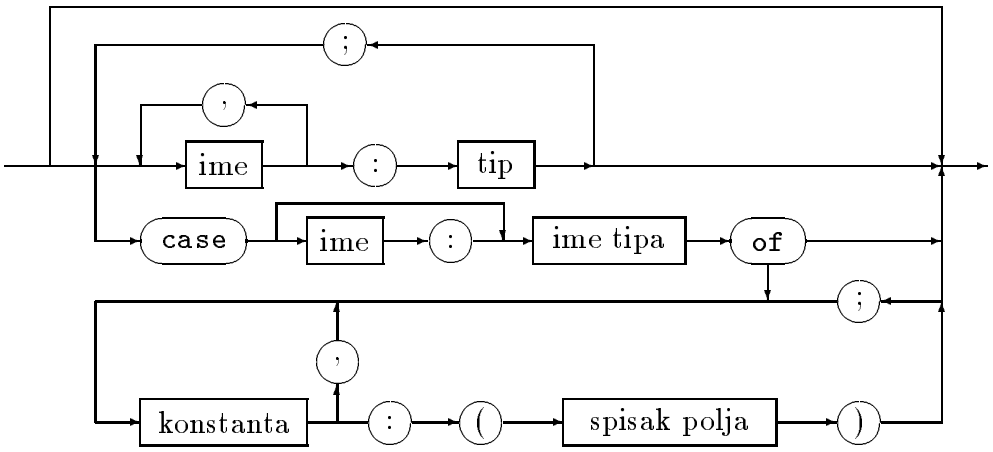
zapis



Sintaksa spiska polja je veoma komplikovana, budući da omogućava i obrazovanje zapisa koji imaju promjenljivih polja. To su tzv. zapisi sa varijantama. U takvom slučaju, jedno od fiksiranih polja koristimo za to da, u zavisnosti od vrijednosti tog polja, kažemo kakva su preostala polja, na primjer

```
type osoba =
    record pol : (muski,zenski);
        case pol of
            muski : (starost : integer);
            zenski : (mjere : array [ 1 .. 3 ] of integer)
    end
```

spisak polja



Ako u dijelu `case` navedemo i ime (selektora), što nije obavezno, onda to znači da smo u fiksirani dio zapisa dodali i to polje, po kome se biraju varijante, tako da ga ne treba posebno navoditi u fiksiranom dijelu.

Primjer. Uzmimo da želimo da sakupimo informacije o osobama u sljedećem obliku I. identifikacija (prezime i ime) II. broj lične karte III. pol (muški, zenski) IV. datum rođenja (dan, mjesec, godina) V. broj izdržavanih članova domaćinstva VI. bračno stanje. Dalji podaci zavise od stanja: ako je vjenčan ili udovac a. datum vjenčanja, ako je razveden a. datum razvoda,

b. prvi razvod (true, false), ako je samac a. stanuje odvojeno (true, false). Odgovarajući zapis može se ovako formulisati

```
type
    alfa = packed array [1 .. 10] of char {vidi kasnije} ;
    stanje = (vjencan,udovac,razveden,samac) ;
    datum =
        record
        dan : 1 .. 31 ;
        mjesec : (jan,feb,mar,apr,maj,jun,jul,avg,sep,okt,nov,dec) ;
        godina : integer
        end ;
    osoba =
        record
        ident : record prezime, ime : alfa end ;
        broj : integer ;
        pol : (muski, zenski) ;
       rodjen : datum ;
        izdr : integer ;
        case brak : stanje of
            vjencan, udovac : (vjencanje : datum) ;
            razveden : (draz : datum ; prvi : Boolean) ;
            samac : (stanuje : Boolean)
        end ;
```

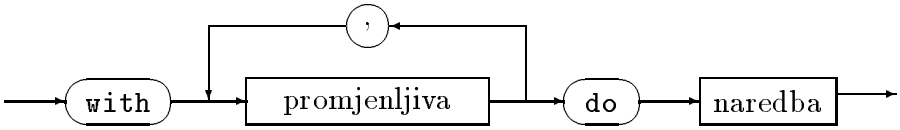
Ako sada još deklarišemo `var` p : osoba ; onda možemo pisati

```
p.broj := 2422 ; p.pol := muski ;
p.rodjen.dan := 23; p.rodjen.mjesec := avg ;
p.rodjen.godina := 1975 ; p.izdr := 2 ;
p.brak := samac ; p.stanuje := true ;
```

12. Naredba WITH

Ako zapis ima mnogo polja, naporno je stalno pisati ime zapisa ispred svakog selektora. Pomoću naredbe WITH možemo saopštiti potrebno ime jednom za uvijek.

naredba WITH



Umjesto p.a := ... ; p.b := p.c ... ; ..... možemo pisati

```
with p do
    begin a := ... ; b := c ... ; ..... end
```

Naredba `with` a, b `do` ... ekvivalentna je konstrukciji

```
with a do
    with b do ...
```

Tako da sada prethodna pridruživanja možemo zapisati kao

```
with p , rodjen do
```

```
begin broj := 2422 ; pol := muski ;
dan := 23; mjesec := avg ;
godina := 1975 ; izdr := 2 ;
brak := samac ; stanuje := true end ;
```

13. Upakovane strukture

Nizovi i zapisi mogu biti upakovani. To znači da će prevodilac pokušati da uštedi na prostoru, time što će se po nekoliko elemenata ili polja držati u istoj riječi. U slučaju upakovanog zapisivanja, dešava se da program sačuva nešto prostora za podatke, ali će zato svakako potrošiti više vremena, na razdvajanje pojedinačnih elemenata i polja. Zato treba sa razmišljanjem upotrebljavati upakovane strukture.

Upakovanu strukturu dobijamo kada ispred simbola `array` ili `record` napišemo simbol `packed`, kao na primjer `type` alfa = `packed array` [1 .. 10] `of` char ;

Niske dužine n znakova praktično predstavljaju konstante tipa `packed array` [1 .. n] `of` char

Kod upakovanih struktura, desiće se da program potroši puno vremena na izdvajanje komponenti. U slučaju nizova, pomažu nam standardni potprogrami "pack" i "unpack", sa kojima se niz može jednom za uvijek zapakovati ili raspakovati. Poziv pack(a, i, b) popuniće upakovani niz "b" sa elementima neupakovanog niza "a". Počeće da ih uzima od i–tog mjesta. Obrnuto, poziv unpac(b, a, i) raspakovaće čitavi niz "b" i staviće elemente u "a", počinjući od elementa "i".

14. Skupovi

Svaki skalarni tip sa konačno mnogo mogućih vrijednosti (to su deklarisani skalarni tipovi i intervali) možemo posmatrati kao (konačan) skup. PASCAL dozvoljava obrazovanje novih tipova, čije vrijednosti su podskupovi tog osnovnog skupa.

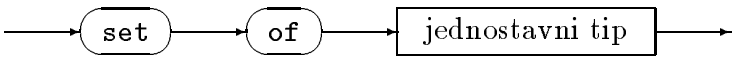
Primjer: Ako je osnovni tip deklarisan kao `type` operator = (plus, minus, puta, podijeljeno) onda se mogu razmatrati podskupovi

addop [plus, minus]
multop [puta, podijeljeno]
operat [plus .. podijeljeno]
prazan [ ]

Veličine koje smo nazvali "addop", "multop", "operat" i "prazan" su novog tipa. To su podskupovi tipa operator.

Za razmatrani tip imamo sljedeću sintaksu

skup

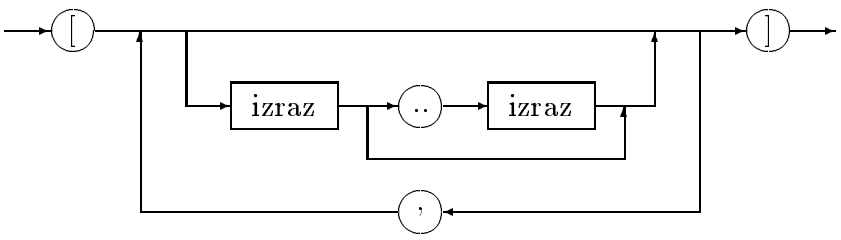


Na primjer `type` opers = `set of` operator; ili recimo

```
var
    x, y : set of 2 .. 8;
```

Ova nova vrsta ima podskupove kao vrijednosti, koji se ovako obrazuju

podskup



Na primjer x := [5, 2, 6 .. 8]

Za podskupove su definisane operacije + unija dva podskupa, * presjek dva podskupa, − razlika dva podskupa. U ova tri slučaja, rezultat je opet novi podskup. Na raspolaganju imamo i relacione operacije = jednakost dva podskupa. <> nejednakost dva podskupa, <= i >= inkluzija, `in` pripada. Kod `in`, lijevi argument mora biti onog jednostavnog tipa iz koga je podskup izdvojen, a vrijednost je true ako je lijevi argument element podskupa sa desne strane.

Primjeri:

```
type izbor = (bijela, crvena, plava) ;
     boja = set of izbor ;
var x, y : boja ;
. . . . . .
    x := [bijela] ; y := [ ] ;
    y := y + [succ(bijela)]
```

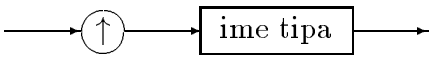
Upotrebom podskupova, program se lako pojednostavi. Umjesto `if` (ch = 'a') `or` (ch = 'b') `or` (ch = 'c') `or` (ch = 'd') `or` (ch = 'z') `then` ... možemo napisati `if` ch `in` ['a' .. 'd', 'z'] `then` ...

15. Pokazivači

Promjenljive koje su deklarisane na početku bloka nazivaju se statičkim, jer se o njima već prevodilac pobrine. PASCAL dopušta i upotrebu dinamičkih promjenljivih. Njih generiše program pomoću standardnog potprograma "new". Takve promjenljive nemaju imena, već su dostupne posredstvom pokazivača. Pokazivač je novi tip. Njegova vrijednost je upravo adresa promjenljive na koju ukazuje. Naravno da nisu svi pokazivači istog tipa. Pokazivač koji pokazuje na cjelobrojnu promjenljivu nije istog tipa kao pokazivač koji pokazuje na neki zapis.

"Pokazivački tip" ima veoma jednostavnu sintaksu

pokazivač



Na primjer

`type` pointer = ^osoba ;

`var` ptr : ^boja ;

Novi objekat nekog tipa stvara se tako što se pozove potprogram "new", da kao parametar ima pokazivačku promjenljivu koja može pokazivati na takav objekat. Tako poziv new(ptr) generiše novu promjenljivu tipa boja. Ta promjenljiva nema imena, ali "ptr" pokazuje na nju.

Za pokazivače je definisano vrlo malo operacija. Dva pokazivača mogu se upoređivati, da li su jednaki ili različiti. Pokazivač se može pridruživati drugim pokazivačima. Najvažnija

operacija je kada poslije pokazivača napišemo strelicu (tj. znak ^). Time dobijamo vrijednost na koju pokazivač pokazuje.

Pokazivači postaju značajni kada se u zapis stavi kao jedno polje pokazivač na neku drugu strukturu, recimo opet na zapis istog tipa. Tako se zapisi mogu povezati u liste, redove, stekove, drveta, ukratko u proizvoljne konačne grafove.

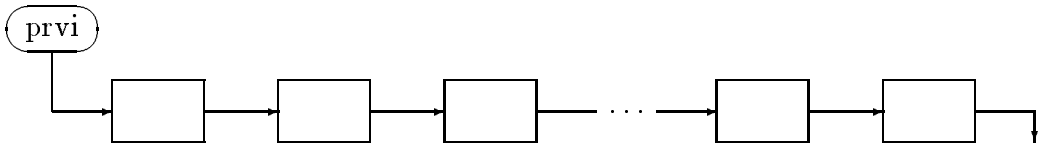
Primjer:

```
type
    pokazivac = ^osoba ;
    osoba =
        record ..........
        iduci : pokazivac
        .......... end ;
var
    pt : pokazivac ;
    prvi : pokazivac ;
```

Neka sada u programu uradimo sljedeće prvi := nil Time saopštavamo da sada pokazivač prvi nigdje ne pokazuje. U nastavku ćemo generisati niz osoba, poređanih u stek. To ćemo ostvariti ovako

```
while uslov do
    begin
        new(pt) ;
        pt^.podatak := ... {popunjava polja novog objekta}
        pt^.iduci := prvi ;
        prvi := pt
    end
```

Kao krajnji rezultat dobićemo stek, povezan onako kako to pokazuje slika



Prilikom obimnih manipulacija sa ovakvim strukturama može se desiti da poslije određenog vremena na neki objekat više ne pokazuje nijedan pokazivač. Takav objekat je za program izgubljen i samo bez potrebe zauzima memoriju (garbagge). To se može izbjeći tako što će se, kada i zadnji pokazivač na taj objekat preusmjerimo negdje drugo, kazati programu da nam taj objekat više nije potreban. Za tu svrhu služi standardni potprogram "dispose", kome se kao parametar navodi pokazivač na objekat koji je postao nepotreban. Od implementacije prevodioca zavisi šta će program uraditi sa takvim objektom. Može se desiti da program jednostavno ignoriše te pozive, ili će pak sprovesti temeljno čišćenje memorije, time što će sve suvišne objekte odnijeti u kantu, čime se stvori novi prostor u memoriji.

Posebno treba spomenuti pokazivače na zapise sa varijantama. Za svaki takav zapis u memoriji je predviđeno onoliko prostora koliko je dovoljno za čuvanje prostorno najvećeg zapisa. Prilikom dinamičkog generisanja, po pravilu znamo koja nam je varijanta potrebna. Zato to možemo i saopštiti, preko dodatnih parametara u potprogramu "new". Primjer:

```
type rec = record
a : integer ;
```

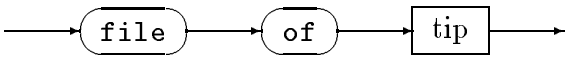
```
case ch : char of
    'a' : (b : array [1 .. 200] of real) ;
    'b' : (c : real) end ;
var ptr : ^rec ; ..... .
new(ptr, 'b')
```

Ovaj poziv će potrošiti samo malo prostora, zato što program predviđa da je potrebna memorija samo za drugu varijantu, varijantu 'b'. U ovakvom slučaju, svakako da ne smijemo mijenjati tip varijante, niti je dozvoljeno pridruživanje čitavog zapisa. Pored toga, i u potprogramu "dispose" moramo navesti iste parametre, da bi potprogram znao koliko prostora u memoriji više nije potrebno.

16. Datoteke

Niz objekata, čiji broj nije unaprijed ograničen i od kojih je u datom trenutku samo jedan dostupan, naziva se sekvencijalnom datotekom (objekata). Datoteke predstavljaju novi tip

datoteka



Datoteke se mogu obrazovati od znakova, zapisa, nizova i slično.

Sa deklaracijom datoteke, na primjer `var` out : `file of` osoba; povezana je jedna promjenljiva, koja se označava isto kao i ime datoteke sa dodatkom strelice, na primjer out^ Ta promjenljiva jeste onaj element datoteke koji je trenutno dostupan. Posebni potprogrami zaduženi su za pomijeranje te promjenljive tamo i ovamo po datoteci. Mogli bismo tu promjenljivu zvati "prozorom" kroz koji vidimo tekući element.

Ako prozor izađe preko kraja datoteke onda će standardna funkcija "eof(out)" dati vrijednost true, a inače false.

Uzmimo da je "f" deklarisano kao datoteka. Tada imamo na raspolaganju sljedeće pozive standardnih potprograma:

reset(f) Otvaranje datoteke za čitanje iz nje. Postavlja prozor na početak, sa ciljem da datoteku čitamo od početka. U slučaju da je datoteka prazna, eof(f) će postati jednak true, a f^ će postati nedefinisano. U suprotnom slučaju, postaće eof(f) jednako false, a f^ jednako prvom elementu datoteke.

rewrite(f) Otvara datoteku radi upisivanja. Ovim se datoteka izbriše. Eof(f) postaje jednako true i možemo iznova da popunjavamo datoteku, f^ je nedefinisano.

get(f) Preuzimanje jednog elementa, tokom čitanja. Ako postoji tekući element onda f^ postane jednak tekućem elementu, a prozor se pomjeri za jedno mjesto unaprijed. Ako tekuće komponente nema, postaće eof(f) jednako true, a f^ nedefinisano. Ako je još i prije poziva bilo eof(f) jednako true onda je čitav učinak poziva nedefinisan.

put(f) Upisivanje jednog elementa. Prije poziva mora eof(f) biti jednako true. Potprogram dodaje tekuću vrijednost promjenljive f^ u datoteku, uz pomijeranje prozora naprijed, f^ postaje nedefinisano, a eof(f) jednako true.

Primjer. Neka bude `var` a, b : `file of` ... Sastavimo dio programa koji će datoteku a prepisati na datoteku b.

```
reset(a); rewrite(b);
while not eof(a) do
    begin b^ := a^;
```

get(a); put(b) `end`

U slučaju softvera Turbo PASCAL firme Borland, u programu treba napisati (na početku niza naredbi) naredbu assign, čime se poziva standardni potprogram, koji će povezati unutrašnje i spoljašnje ime datoteke. Ta naredba ima opšti oblik assign(intname, 'extname'); Na primjer assign(f, 'fajl11.dat'); Unutrašnje ime pojavljuje se u programu. A pod spoljašnjim imenom ona se vodi na spoljašnjoj memoriji (na hard disku). Operativni sistem određuje format spoljašnjeg imena datoteke. Takođe, operativni sistem vrši i druge poslove koji se odnse na fajlove.

Dakle, za rad sa datotekama potrebno je da sarađuju operativni sistem i softver za PASCAL. Tako, na kraju niza naredbi treba napisati naredbu close za zatvaranje datoteke. Njen opšti oblik je close(intname); Recimo close(f);

Datoteke služe za trajno čuvanje podataka.

17. Tekstualne datoteke

Datoteke čiji su elemnti znaci nazivaju se tekstualnim datotekama. Za njih je definisan i standardni tip i to kao

`type` text = `file of` char;

tako da korisnik ima pravo da kaže recimo `var` printer : text; Tekstualne datoteke imaju još i dodatnu strukturu. One su podijeljene na posebne redove. Treba zamisliti da takva datoteka posjeduje, među dozvoljenim znacima, još i poseban znak – indikator kraja reda. Taj indikator ne možemo učitavati ili prikazivati kao znak, ali ga ipak možemo pronaći, preskočiti ili ostvariti.

Uz gornju deklaraciju promjenljive "printer" važi: writeln(printer) generiše indikator eol (end of line), readln(printer) preskoči sve znake u tekućem redu, uključujući i indikator eol, a printer^ postaje jednak prvom znaku u novom redu.

Na raspolaganju imamo i logičku funkciju eoln(printer). Njena vrijednost će postati true kada se nađemo na indikatoru eol (tada je printer^ jednak ' ').

Ako je f tekstualna datoteka onda imamo na raspolaganju sljedeće skraćenice. Imamo pravo da pišemo:

write(f, ch) umjesto f^ := ch; put(f)

read(f, ch) umjesto ch := f^; get(f)

Primjer. Prepišimo datoteku x (tekstualnu) na tekstualnu datoteku y tako da se struktura redova sačuva.

```
reset(x); rewrite(y);
while not eof(x) do
begin {prepiši red}
    while not eoln(x) do
        begin read(x, ch); write(y, ch) end;
    readln(x); writeln(y)
end
```

Dozvoljeno je da se u potprogramima write, read, writeln, readln, eof i eoln izostavi ime datoteke, ako se radi o imenu input ili output. Tako: write(ch) znači write(output, ch), read(ch) znači read(input, ch), writeln znači writeln(output), readln znači readln(input), eof znači eof(input), eoln znači eoln(input).

Potprogrami read i readln sposobni su da učitavaju i cijele i realne brojeve, a ne samo pojedinačne znake. Ti brojevi treba da budu međusobno razdvojeni sa bar jednim blankom ili novim redom. Potprogram će prvo preskočiti sve znake blanko, do prvog znaka koji nije blanko.

Tu broj počinje. Završava se sa prvim blankom ili novim redom. Ono što se unosi, mora da zadovoljava sintaksu brojeva u PASCALU.

Kada je read ili readln pročitao cijeli ili realan broj, onda je idući znak, koji je sada na redu, onaj znak blanko koji je okončao broj. A u slučaju potprograma readln važi da je potrebno, nakon učitavanja broja, preskočiti još i sve znake u tekućem redu, tako da je sada na redu prvi znak u idućem redu.

Pored toga, u oba potprograma imamo pravo da navedemo nekoliko parametara, na primjer read(filex, a, b, c, d) Prvi parametar je ime datoteke. Ako je to input, smijemo da ga izostavimo.

I potprogrami write i writeln u stanju su da štampaju ne samo znake, već i cijele i realne brojeve, Booleove veličine i čitave niske. Potprogrami mogu imati nekoliko parametara, na primjer write(datoteka, p1, p2, p3, ..., pn) Ako je ime datoteke output, možemo ga izostaviti. Svaki parametar smije da bude oblika

e ili e : e1 ili e : e1 : e2

U svakom slučaju, ovdje je "e" vrijedmost koja će biti ispisana, a smije biti tipa integer, real, Boolean, char, ili je pak niska. "e1" predstavlja neobaveznu širinu polja. Za ispis će se potrošiti e1 znakova. Ako e1 nije dato, važe lonvencije, zavisno od implementacije. "e2" dolazi u obzir samo u slučaju realnih brojeva i govori koliki je broj decimala.

Svaka datoteka je lokalna za blok u kome je deklarisana. Uobičajeno je da su datoteke lokalne za program. U zaglavlju programa trebalo bi navesti imena spoljašnjih datoteka koje će program koristiti. Pri tome nije potrebno deklarisati datoteke input i output, a ostale treba.

Da ponovimo. Privremene datoteke važe samo tokom izvršavanja programa i ne čuvaju se na spoljašnjoj memoriji. Dok trajne datoteke imaju suprotne karakteristike. Ako je ime datoteke navedeno u zaglavlju programa, onda se time proglašava da je to jedna trajna datoteka, a inače je privremena. Kao primjer zaglavlja programa navodimo `program` p(input, output, f); Navedena pravila važe za sve datoteke, s tim da egzistiraju samo dva izuzetka. To su datoteke input i output.

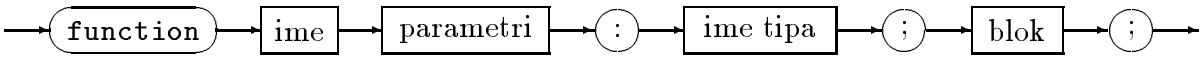
18. Potprogrami

Kada smo opisivali blok, kazali smo da sve veličine koje se upotrebljavaju u jezgru (sastavljena naredba) prethodno moraju da budu deklarisane. Isto važi i za potprograme (i funkcije). Tako potprogrami postaju lokalni tome bloku. Budući da i samo jezgro potprograma predstavlja jedan blok, može se govoriti i o potprogramima koji su lokalni potprogramu, i tako dalje.

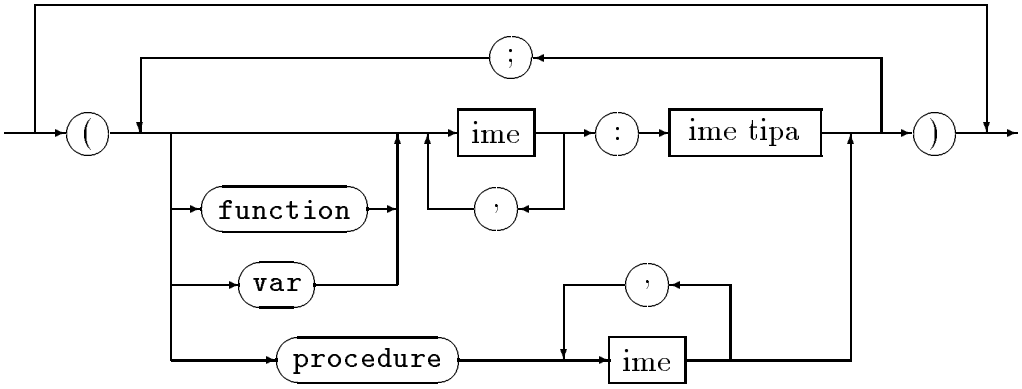
Razlika između potprograma i funkcija je ista kao i u drugim jezicima: potprograme pozivamo, dok funkcije upotrebljavamo u izrazima. Funkcije imaju vrijednost, a potprogrami nemaju. I jedni i drugi mogu imati formalne parametre.

deklaracija potprograma

deklaracija funkcije



parametri



Iz dijagrama se vidi da svaki parametar (ili grupa parametara) ima tip, definisan sa imenom, recimo

`procedure` xxx(a, b, c : real)

Ipak, to ne važi za parametre koji su potprogrami, jer potprogrami nemaju vrijednosti. Parametar potprograma ili funkcije može biti potprogram, kao recimo

`procedure` a(`procedure` b)

ili funkcija, s tim da u tom slučaju treba saopštiti kakav rezultat ona daje, kao recimo

`procedure` b(`function` f : real)

a obično su promjenljive. U tom slučaju raspolažemo sa dvije mogućnosti. Ako ispred imena napišemo `var` onda su to parametri koji se pozivaju po referenci, tj. potprogram koji poziva saopštava pozvanom potprogramu adrese stvarnih parametara. U tom slučaju, stvarni parametri moraju da budu promjenljive. Ako takvom parametru dodijelimo vrijednost u jezgru potprograma, onda tu vrijednost ustvari dobija stvarna promjenljiva. Formalni parametar je samo fornalan.

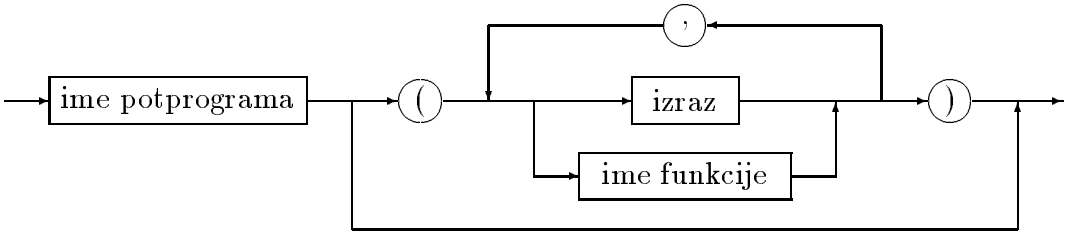
Ako se ispred imena parametra ništa ne napiše, kao na primjer

`function` f(x : real) : real

tada se parametar poziva po vrijednosti. U ovom slučaju, formalni parametar je lokalna promjenljiva. Stvarni parametar u tom slučaju smije da bude proizvoljan izraz, samo da je odgovarajućeg tipa. Odgovarajući formalni parametar dobiće vrijednost izračunatog izraza i sa njim se dalje računa. U PASCALU nema pozivanja parametara po imenu, kakvo dozvoljava ALGOL 60.

Potprograme pozivamo tako što napišemo ime potprograma i u zagradi po potrebi napišemo stvarne parametre.

poziv potprograma



Poziv potprograma predstavlja jednu naredbu. Funkcije pozivamo tako što napišemo ime funkcije i u zagradi navedemo po potrebi stvarne parametre i sve to zajedno upotrebimo u izrazu.

Posebno treba spomenuti da u jezgru funkcije treba jednom definisati njenu vrijednost, obično u zadnjoj naredbi. To se ostvaruje tako što se imenu funkcije pridruži odgovarajuća vrijednost.

Primjer. Izračunajmo nulu funkcije $f(x) = 2\sin x - x$.

Objašnjenje. Nula funkcije pripada intervalu $\pi/2 < x < \pi$. Koristi se metoda polovljenja intervala. Ponavlja se sve dok dužina intervala sasvim ne iščezne.

```
program nula(output);
{program nula će izračunati pozitivnu nulu
funkcije f(x) = 2 * sin(x) - x}
const pi = 3.14159;
var a, b, c : real ;

function f(x : real) : real;
    begin f := 2.0 * sin(x) - x end ;

begin
a := pi / 2; b := pi; c := (a + b) / 2;
while (a < c) and (c < b) do
    begin
    if f(c) > 0 then a := c else b := c;
    c := (a + b) / 2
    end ;
writeln(c)
end .
```

PASCAL dopušta rekurzivnu upotrebu funkcija, tj. potprogram ili funkcija smije pozivati samu sebe, direktno ili indirektno. Primjer:

```
function djelilac(a, b : integer) : integer ;
{funkcija djelilac daje najveći zajednički djelilac prirodnih
brojeva a i b, izračunat po Euklidovom algoritmu}
begin
    if b = 0 then djelilac := a else djelilac := djelilac(b, a mod b)
end ;
```

Ako do rekurzije dođe tako što, na primjer, potprogram a poziva potprogram b, a potprogram b poziva potprogram a, onda ne možemo deklarisati potprograme u takvom redosljedu da bismo svaki potprogram prvo deklarisali, pa tek nakon toga pozivali. Navedeni problem prevazilazi se tako što se potprogram prvo samo najavi, a biće deklarisan kasnije.

Potprogram (ili funkcija) najavljuje se tako što se napiše zaglavlje potprograma, a umjesto bloka napiše se samo simbol forward. Takav potprogram sada možemo upotrebljavati. Deklarisaćemo ga kasnije, time što ćemo još jednom napisati zaglavlje, ovog puta bez parametara, a zatim blok, neposredno u nastavku. Primjer:

```
procedure a(x : tip); forward ;

procedure b(x : tap);
begin ... a(s) ... end ;
```

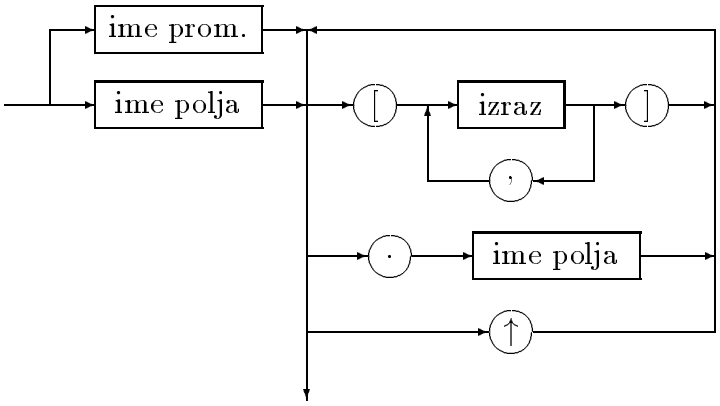
```
procedure a;
begin ... b(s) ... end ;
```

Prilikom upotrebe potprograma i funkcija ima još nekih ograničenja, koja omogućavaju efikasno prevođenje i jednostavan prevedeni program. – Vrijednost funkcije mora biti skalarnog tipa, interval ili pokazivač. Nisu dozvoljeni nizovi i zapisi. – Potprogrami i funkcije, koje dajemo kao paranetre drugim potprogramima, smiju imati samo parametre koji se pozivaju po vrijednosti. – Komponente upakovanih struktura ne smijemo pozivati po referenci.

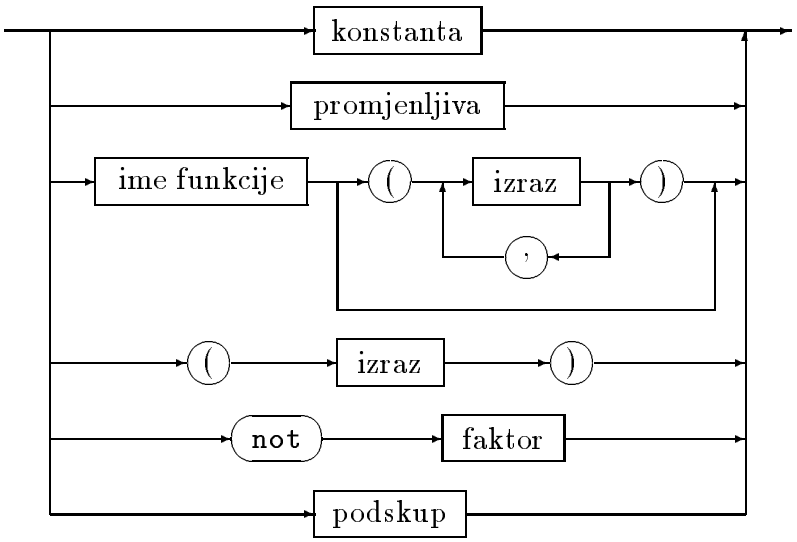
19. Sintaksa izraza

U ovoj sekciji detaljno ćemo obraditi sintaksu izraza, i to bilo kakvih izraza, koji mogu da se pojave u programu.

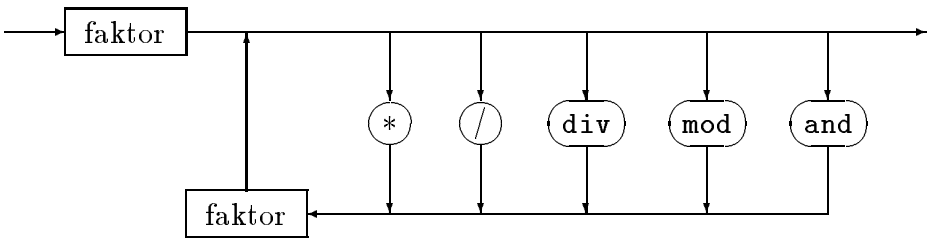
promjenljiva



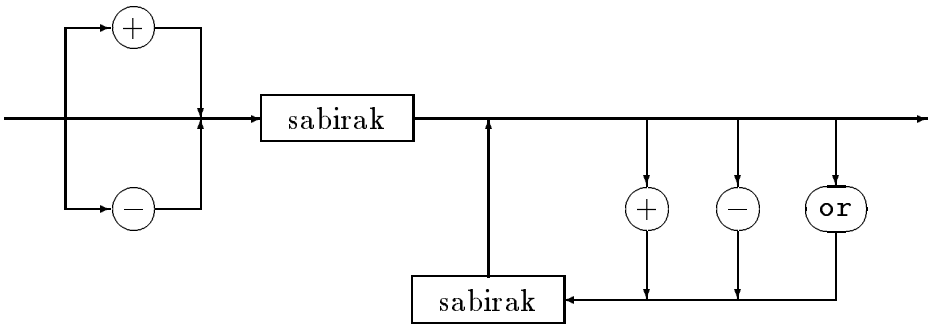
faktor



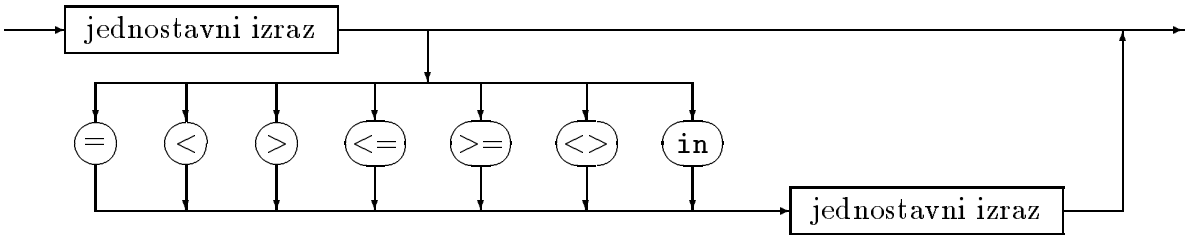
sabirak



jednostavni izraz



izraz



SADRŽAJ

Literatura

Egon Zakrajšek: Programski jezik pascal, Društvo matematikov, fizikov in astronomov SR Slovenije, Ljubljana, 1979.

PASCAL POSTAVKE ZADATAKA

1) Uvod u programiranje

1. Sljedeće matematičke formule zapiši kao aritmetičke izraze jezika Pascal: a) $3a + \frac{b}{2c+d}$, b) $\frac{0{,}3x}{(x+y)^2}$, c) $\frac{10^{-15}+x^{-2}}{\sqrt{1+x^{-1}}}$, d) $\sqrt{1+\left|\sin\frac{|x|}{2}\right|}$, e) $\frac{x}{1+\frac{x}{1+\frac{x}{1+x}}}$, f) $\frac{\sqrt{x}+\sin x^2}{x+e^{-2x}} + \ln\left|\operatorname{tg}\frac{x}{2}\right| + \frac{A+Bx}{a+cx}$, g) $\frac{1}{1+\left(\frac{RG}{K}\right)^2}$.

2. Naredne matematičke obrasce zapiši kao naredbe pridruživanja jezika Pascal: a) $R = \frac{a+bx}{c+dx}$, b) $P = \frac{ab^2}{cd} - 3a + \frac{bc}{2c+d}$, c) $L' = \sqrt{x^2 + |y - A|}$, d) $h_{max} = x \operatorname{tg}\frac{x}{2} + 2\ln\left|\cos\frac{x}{2}\right|$, e) $K = A\left(1 + \frac{1}{P}\right)^n$, f) $H_3 = \log_2 h$.

3. Napiši naredbe pridruživanja za: a) veći korijen kvadratne jednačine, b) površinu trougla čije su dužine stranica a, b i c, c) dužinu stranice c trougla sa datim stranicama a i b i uglom γ koga one zaklapaju, d) uglove trougla, ako su poznate dužine sve tri stranice, e) površinu trougla, ako su poznati poluprečnik upisanog kruga i sva tri unutrašnja ugla, f) stranicu pravilnog 12–ugla upisanog u krug poluprečnika r, g) zapreminu i površinu tetraedra čija je stranica a, h) površinu valjkaste cijevi čiji su poluprečnici R i r, a visina h, i) oštri ugao između pravih $a_1x + b_1y = c_1$ i $a_2x + b_2y = c_2$, j) m–ti korijen pozitivnog broja a, gdje je m prirodan broj. Dozvoljeno je da se u izrazima upotrebljavaju standardne funkcije jezika Pascal.

4. Odrediti vrijednosti sljedećih izraza: a) a + b − c div 3 + c / 3, b) 4 ∗ a + 1 / 5, c) 17 + c div d ∗ d, d) round (sqrt(c) + trunc(b) mod c − 1) ako promjenljive u izrazima imaju vrijednosti: a = −12.3 , b = 3.04 , c = 41 i d = 13 . Napomena: tip promjenljive vidi se po njenoj vrijednosti (realni brojevi sadrže decimalnu tačku). Obratiti pažnju na redosljed operacija u izrazima.

5. Kakvu vrijednost će imati promjenljiva x, kada se izvede sljedeći dio programa:

x := 2 ; x := x + sqr(x) ; x := x + sqr(x) ; x := x + sqr(x) ;

6. Naredbe a := 2 ∗ a ; b := a − b ; u nekom dijelu programa izvedu se triput uzastopno. Kakve vrijednosti će imati promjenljive a i b, ako je na početku bilo a = 1 i b = 12.

7. Sastaviti dio programa u kome će promjenljive a i b međusobno zamijeniti vrijednosti. Pritom: a) dozvoljeno je da se upotrebi pomoćna promjenljiva c, b) ne smiju se upotrebljavati pomoćne promjenljive.

8. Šta će stvoriti niz od sljedeće dvije naredbe: x := x + y ; y := x − 2 ∗ y ;

9. Zapisati sljedeće uslove u obliku Booleovih izraza jezika Pascal: a) $x \neq y$, b) $-1 \leq z < 1/2$, c) $x \in [0, 1]$, d) $|x| < 1$, e) tačka $T(a, b)$ leži u krugu poluprečnika 2 sa centrom u koordinatnom početku, f) $a \mid b$ (a dijeli b), g) broj x je paran, h) broj a je djeljiv sa 5 i 7, i) iz $a < b$ slijedi $c < d$, j) broj x leži u intervalu $[a, b)$, k) $|x - y| < 0{,}25$, l) količnik između $|x \cdot y|$ i kvadratnog korijena iz $x + y$ jednak je 1, m) (realan) broj x je cio broj, n) prirodan broj n je potpun kvadrat, o) tačka $T(p, q)$ ne leži u prvom kvadrantu, p) tačka $A(a, b)$ leži u presjeku kruga čiji je centar $R(0, 1)$ i čiji je poluprečnik $r = 1$ i kruga sa centrom $S(-1, 2)$ poluprečnika isto $r = 1$, q) tačka $A(a, b)$ leži u simetričnoj razlici krugova poluprečnika $r = 2$ sa centrima $R(1, -3)$ i $S(-1/2, -1)$. Napomena: simetrična razlika skupova X i Y je skup $(X - Y) \cup (Y - X)$.

10. Kakve će nam vrijednosti dati sljedeći Booleovi izrazi: a) true < false, b) (true < false) = false, c) (2 >= 5) or (1 <= 1), d) (3 <> 3) and (3 <> 4) or (a = a).

11. Za pojedine vrijednosti promjenljivih A) x=−1 y=0 z=0 B) x=10000 y=0.01 z=2 C) x=5 y=1 z=−4 odrediti vrijednosti sljedećih Booleovih izraza: a) x <= 2.0E5 b) (x > y) or (y >= z) c) not (x <> x) and (y + 1 < z − 2) d) ((x ∗ y ∗ z) / 2 >= x + 1) or not (x <= 0)

12. Opiši postupak za računanje e^x pomoću reda: $e^x = 1 + x/1! + x^2/2! + x^3/3! + \ldots$ Vrijednost e^x želimo da izračunamo precizno, tako da vrijednost zadnjeg uračunatog člana treba da bude manja od 1.0E−8 .

13. Šta je pogrešno u algoritmu koji bi se mogao opisati ovako:

SVE DOK JE $|x| \leq 1$ PONAVLJAJ $x \leftarrow 2/x$

14. Opiši postupak koji će ciklično pomjeriti elemente u nizu dužine n za k mjesta u desno ($k \leq n$). Na raspolaganju imamo operaciju ZAMIJENI(i, j) koja međusobno zamjenjuje elemente na i–tom i j–tom mjestu. Mimo toga ne smiju se upotrebljavati bilo kakve promjenljive, osim brojača. Primjer: ako niz 1, 2, 3, 4, 5, 6, 7, 8 ciklično pomjerimo za tri mjesta onda dobijamo niz 6, 7, 8, 1, 2, 3, 4, 5. Ocijeniti broj operacija (tj. poziva operacije ZAMIJENI) u zavisnosti od n i k. Pokušaj da nađeš algoritam koji će tokom svog izvođenja što manje upotrebljavati operaciju ZAMIJENI.

2) Osnovni programi

15. Sastaviti program koji će ugao, dat u radijanima, pretvoriti u ugao izražen u stepenima, minutima i sekundama.

16. Sastaviti program koji će ugao, dat u stepenima, minutima i sekundama, pretvoriti u ugao izražen u radijanima. Pri tome redukovati ugao na vrijednost između $-\pi$ i π.

17. Sastaviti program koji učitava težinu nekog predmeta izraženu u funtama, a prikazuje je u funtama i kilogramima (1 funta = 0,45359 kg).

18. Avion se (u ovom trenutku) nalazi na visini h pod uglom α (u odnosu na horizontalu) i sa brzinom v preleće položaj protivavionske rakete. Brzina rakete je V i ima goriva za 10 s letenja. Avion i raketa lete pravolinijski. Sastaviti program koji, prilikom preleta aviona, na osnovu (radarskih) podataka, odlučuje da li će raketa biti ispaljena (ispaljuje se ako će raketa za manje od 10 s pogoditi avion). Ako je odgovor potvrdan, neka program još napiše i ugao δ (u stepenima) pod kojim treba ispaliti raketu, kao i vrijeme T za koje će raketa dostići avion. Veličine δ i T računaju se ovako: $\delta = \arccos(v\cos\alpha/V)$, $T = h/(V\sin\delta - v\sin\alpha)$.

19. Sastaviti program koji učitava brojeve a i b, koji predstavljaju realni i imaginarni dio kompleksnog broja $z = a+bi$, pa računa i štampa odgovarajuće parametre r i φ polarnog zapisa $z = re^{i\varphi} = r(\cos\varphi + i\sin\varphi)$ tog broja. Kao što znamo, važi: $\varphi = \mathrm{arctg}(b/a)$ i $r = \sqrt{a^2+b^2}$. Neka program odštampa i – u kom kvadrantu leži broj z. Neka se ponavlja za četiri broja z.

20. Sastaviti program koji učitava koordinate tačke (x, y) i saopštava odgovor na pitanje: pripada li ta tačka oblasti definisanoj nejednakostima: $y < 3$, $x + y > 1$, $y < 2x + 1$ i $y > x^2$.

21. Sastaviti program koji će za date vrijednosti a, b, c, d i x (koje se učitavaju preko tastature) izračunati i napisati vrijednost polinoma $f(x) = ax^3 + bx^2 + cx + d$. Uputstvo: prilikom računanja vrijednosti polinoma, upotrebi izraz koji daje istu vrijednost (Horner): $f(x) = ((ax + b)x + c)x + d$.

22. Kubna jednačina oblika $x^3 + bx^2 + cx + d = 0$ može se, pomoću smjene $x = y - b/3$, pretvoriti u kubnu jednačinu oblika $y^3 + py + q = 0$, pri čemu važi: $p = (3c - b^2)/3$ i $q = (27d-9bc+2b^3)/27$. Sastaviti program koji učitava koeficijente b, c i d i ispisuje kubnu jednačinu u oba izdanja. Dodatak: neka program još nađe i ispiše i sve realne korijene razmatrane kubne jednačine.

23. Napisati program koji učitava cijele brojeve x i y, prikazuje te brojeve, kao i njihov količnik i ostatak pri dijeljenju. U slučaju da je y = 0, neka program saopšti poruku 'GRESKA'.

24. Napisati program koji učitava neku cijenu i prikazuje je preko jedinica plaćanja. Na primjer: 138.40 eura = 1 * 100 eura + 3 * 10 eura + 1 * 5 eura + 1 * 2 eura + 1 * 1 euro + 2 * 20 centi.

25. U prvom redu date su, kao ulazni podaci, koordinate centra kruga i njegov poluprečnik. U svakom idućem redu date su koordinate jedne tačke. Napisati program koji za svaku tačku saopštava da li se ona nalazi unutar, na granici ili van kruga. Broj tačaka može da bude proizvoljno velik.

26. Za mjerenje temperature postoji nekoliko skala: Celsius, Fahrenheit, Kelvin, Rankine, među kojima postoje sljedeće veze: F = 32 + 9 * C / 5, K = C + 273, R = F + 460. Učitava se niz podataka o temperaturama. Pojedini podatak sastoji se od broja stepeni i oznake skale (C, F, K, R). Napisati program koji će dati niz temperatura prikazati po sve četiri skale.

27. Napisati program koji učitava realne brojeve (svaki je u posebnom redu) i određuje koliko imaju cijelih mjesta. Na primjer: 0.0023 0 cijelih mjesta, −34.2735 2 cijela mjesta, 15000.01 5 cijelih mjesta.

28. Učitavaju se podaci o trouglovima. U pojedinom redu nalaze se sljedeća tri podatka: dužine stranica a i b i ugao γ među njima (u stepenima). Ako je $a < 0$ onda to znači kraj podataka. Za svaku trojku podataka treba izračunati treću stranicu c po kosinusnoj teoremi, a zatim prikazati na ekranu podatke kao i rezultat.

29. Napisati program koji učitava tri broja x, y i z i onda ih štampa tako da prvo odštampa najveći, a zadnje najmanji broj.

30. Napisati program koji preko tastature učitava rimski broj, pretvara ga u dekadni i saopštava ga u oba oblika.

31. Napisati program za rješavanje kvadratne jednačine $ax^2 + bx + c = 0$, gdje su a, b i c ulazni podaci. Obuhvati i slučaj kada jednačina ima kompleksne korijene, kao i slučaj kada je linearna ($a = 0$).

32. Sastaviti program koji će n puta prikazati tvoje ime i prezime, gdje je broj n ulazni podatak.

33. Sastaviti program koji će napisati jednom 1, dvaput 2, triput 3, ... i deset puta 10, tj. 122333444455555...1010.

34. Sastaviti program koji će u obliku tabele prikazati brojeve x (ulazni podatak), $\sin x$, $\operatorname{tg} x$, $\arcsin x$ i $\operatorname{arctg} x$.

35. Program za rješavanje sistema jednačina oblika 2×2: $a_{11}x_1 + a_{12}x_2 = b_1$, $a_{21}x_1 + a_{22}x_2 = b_2$, gdje se pretpostavlja da je determinanta sistema $D \neq 0$. Učitavaju se a_{ij} i b_i, a treba izračunati i odštampati rješenje sistema x_1, x_2. U slučaju $D = 0$, dovoljno je da se odštampa kratka poruka.

36. Program koji računa i štampa prvih 30 članova niza $p_n = \frac{\lambda^n}{n!}e^{-\lambda}$, gdje je broj λ ulazni podatak. Neka se još odštampa i $\sum_{n=1}^{30} p_n$.

3) Kontrolne strukture

37. Napisati program koji štampa tabelu recipročnih vrijednosti brojeva od 1 do 20 sa n decimala, gdje se vrijednost n učitava.

38. Napisati program koji će na ekranu prikazati prvih 10 članova niza $\{x_n\}$, gdje je $f_{n+1} = f_n + f_{n-1}$, $x_n = f_n/f_{n-1}$, $f_0 = a$, $f_1 = b$, gdje su a i b ulazni podaci.

39. Napisati program koji učitava brojeve n, m i k, a onda štampa n članova niza m, $-m$, $m + k$, $-(m + k)$, $m + 2k$, $-(m + 2k)$, ...

40. Neka je $\binom{n}{k}$ binomni koeficijent – broj kombinacija od n elemenata k–te klase. Za binomne koeficijente važi $\binom{n}{k} = (n + 1 - k)\binom{n}{k-1}/k$ i $\binom{n}{0} = 1$. Napisati program koji će na ekranu prikazati binomne koeficijente za $n = 0, 1, \ldots, 10$ u obliku Pascalovog trougla.

41. Napisati program koji učitava prirodan broj n i štampa sve prirodne brojeve između 1 i 1000 čiji je zbir cifara jednak n. Na primjer, ako je $n = 4$ onda program treba da odštampa

brojeve 4, 13, 22, 31, 40, 103, ..., 400.

42. Napisati program koji će odštampati one prirodne brojeve između 1 i 1000 koji su djeljivi sa zbirom svojih cifara. Za svaki takav broj odštampati i zbir cifara, kao i količnik. Na primjer, broj 330 ima zbir cifara $3 + 3 + 0 = 6$. Kako je 330 djeljivo sa 6, to taj broj treba prikazati, a pored toga i zbir 6 i količnik 55.

43. Za broj se kaže da je savršen ako je jednak zbiru svojih činilaca, izuzev njega samog. Na primjer $28 = 1 + 2 + 4 + 7 + 14$. Odrediti sve savršene brojeve manje od 1000.

44. Napisati program koji će pronaći sva rješenja jednačine $x^2 + y^2 = z^2$ u prirodnim brojevima (x, y i z su prirodni brojevi). Pritom, neka je $z \leq 100$ i $x < y$.

45. Pronađi sve trojke cifara a, b i c koje zadovoljavaju jednakost $abc = a^3 + b^3 + c^3$. Napomena: abc označava kada se tri cifre napišu redom (a ne proizvod).

46. Sastaviti program za nalaženje svih cjelobrojnih rješenja jednačine $i^3 + j^3 + k^3 = 3$, pri čemu važe ograničenja: $|i| \leq 10$, $|j| \leq 8$, $|k| \leq 6$.

47. Napisati program koji učitava prirodne brojeve a i b ($a < b$) i određuje razlomak oblika p/q ($a \leq q \leq b$) koji se najmanje razlikuje od broja $\pi = 3{,}14159$.

48. Napisati program za približno računanje vrijednosti određenog integrala po Simpsonovoj formuli $\int_a^b f(x)dx = (y_0 + 4y_1 + 2y_2 + 4y_3 + \ldots + 4y_{2n-1} + y_n) \cdot h/3$, gdje je $h = (b-a)/(2n)$ i $y_i = f(a + ih)$. Program testiraj za $n = 5$ i $n = 10$ u slučaju: $a = 0$, $b = 1$, $f(x) = 1/(1 + x^2)$.

49. Napisati program koji učitava niz pozitivnih realnih brojeva i među njima određuje najvećeg i najmanjeg. Prestaje se sa učitavanjem kada se naiđe na prvi negativan broj.

50. Sastaviti program koji učitava i prikazuje brojeve i sabira posebno negativne, a posebno pozitivne. Učitavanje se prekida kada se prvi put učita broj 0. Na kraju prikaže i oba zbira. Dodatak: ukoliko je ukupan zbir negativan, neka saopšti i upozorenje *** MANJAK.

51. Sastaviti program koji učitava i prikazuje brojeve. Prekida učitavanje kada prvi put pročita neki negativan broj. Tokom učitavanja neka pronađe najveći među učitanim brojevima i neka ga na kraju saopšti. Dodatak: još treba odrediti i saopštiti i – koliko je brojeva, među datim brojevima, bilo jednako najvećem.

52. Fibonaccijevi brojevi mogu se računati na dva načina: a) $f_{i+1} = f_i + f_{i-1}$, $f_0 = 0$, $f_1 = 1$, b) $f_i = \text{round}(c^i/\sqrt{5})$, $c = (1 + \sqrt{5})/2$. Pronaći najmanje i za koje će tako dobijeni brojevi početi da se razlikuju, ako si uzeo da je $\sqrt{5} = 2{,}23607$.

53. Napisati program koji učitava prirodan broj i rastavlja ga na proste činioce. Na primjer: $204 = 2 * 2 * 3 * 17$, $83 = 83$, $81 = 3 * 3 * 3 * 3$. Napisati program tako da učitava i rastavlja nekoliko brojeva.

54. Napisati sva moguća razbijanja broja n (ulazni podatak) na tri različita sabirka, $n = a + b + c$. Uputstvo: pretpostaviti da je $a < b < c$. Sa izborom a i b, tačno je određen c.

55. Napisati program za tabeliranje funkcije $y = a \sin x + b \cos x$ za vrijednosti $x = 0$ $x = 0{,}1$ $x = 0{,}2$ ... sve dok ne budu dvije uzastopne vrijednosti funkcije različitog predznaka. Ovdje su a i b podaci koje računar učitava.

56. Metodom polovljenja intervala odrediti nulu funkcije $f(x) = x - \text{tg}\, x$ koja leži u intervalu $(\pi/2, 3\pi/2)$. Polovljenje ponavljati sve dok ne postane $|f(x)| < 0{,}00001$.

57. Izračunati vrijednost zbira $1 - 1/2 + 1/3 - 1/4 + \ldots + 1/9999 - 1/10000$ na četiri načina: a) slijeva u desno, b) zdesna u lijevo, c) slijeva u desno, pozitivne članove odvojeno od negativnih, d) zdesna u lijevo, pozitivne članove odvojeno od negativnih.

58. Napisati program koji će kompleksan broj z pomnožiti 500 puta sa brojem $c = 0{,}6 + 0{,}8i$. Budući da je $|c| = 1$, mora ostati $|z|$ nepromijenjen. Provjeriti.

59. Odrediti koliko uzastopnih članova niza $a_n = n^2 - n + 41$ ($n = 1, 2, \ldots$) su prosti brojevi i odštampati ih.

60. Profesor A. stanuje u srazmjerno dugačkoj ulici. Kućni broj kuće gdje on stanuje je trocifren i ima jedno zanimljivo svojstvo, da je zbir svih kućnih brojeva u ulici koji su manji od njega jednak zbiru svih većih. Odrediti kućni broj i broj kuća u ulici.

61. Napisati program koji će na ekranu prikazati sliku kruga, što je bolje moguće, u tzv. ASCII grafici. Uzeti u obzir da jedno slovo po širini zauzima 1/10 inča, a po visini 1/6 inča. Znamo da je 1 inč = 2,54 cm.

62. Pravilna raspodjela. Neka je $n \geq 2$. Neka je $f(x) = (ax + b) \bmod n$, gdje je $a = 21$, $b = 33$, $n = 1000$. Neka je $0 \leq x_1 \leq n - 1$ (npr. $x_1 = 500$) i $x_{k+1} = f(x_k)$ $(k \geq 1)$. Navedene formule definišu jedan generator slučajnih brojeva $x_1, x_2, \ldots$ i želimo da ispitamo njegov kvalitet. Napisati program u kome se računa prvih 100 slučajnih brojeva $x_1, \ldots, x_{100}$ i saopštava njihov raspored npr. po četvrtinama (0 – 250 – 500 – 750 – 1000), da bi se vidjelo da li je raspodjela približno ravnomjerna.

63. Dužina periode. Prilikom izbora parametara a i b za generator slučajnih brojeva $\{x_k\}$ koji se bazira na formuli oblika $y = (ax + b) \bmod n$ (linearna kongruencija), poželjno je da budu ispunjeni sljedeći uslovi: a) broj $a - 1$ je djeljiv sa svim prostim činiocima broja n, b) ako je n djeljiv sa 4 onda je i $a - 1$ djeljiv sa 4, c) $b \neq 0$, d) b i n su uzajamno prosti, jer bi tada i samo tada trebalo da perioda u nizu $\{x_k\}$ bude maksimalna moguća (to znači n), tj. do zatvaranja ciklusa dolazi tek kada to postane zaista obavezno. Napisati program u kome se ispituje istinitost navedenog tvrđenja, u slučaju nekoliko generatora, od kojih neki zadovoljavaju navedene uslove, a neki ih ne zadovoljavaju.

64. Približna vrijednost za π. Razmotrimo dvije geometrijske figure u ravni: jedinični kvadrat $[0, 1] \times [0, 1]$ i u njega upisani krug, čija površina očito iznosi $\pi/4$. Napisati program u kome se generiše veći broj parova slučajnih brojeva (x, y) $(0 \leq x, y \leq 1)$ i za svaku tačku (x, y) se određuje da li pripada krugu. Izračunati količnik K broja tačaka koje leže u krugu i ukupnog broja tačaka i štampati veličinu $4K$, koja služi kao procjena za π.

4) Sastavljene strukture podataka

65. Napisati program koji će tabelirati vrijednosti polinoma $p(x) = a_0x^n + a_1x^{n-1} + \ldots + a_n$ za $x = 0(0,1)1$. Učitati stepen i koeficijente. Uputstvo: upotrebi Hornerovu šemu.

66. Napisati program koji učitava koordinate (x, y, z) pet tačaka i računa tačku u kojoj prava kroz prve dvije tačke prodire ravan kroz preostale tri tačke. Prikazati podatke i rezultat.

67. Učitati podatke o 20 materijalnih tačaka, gdje o svakoj tački imamo sljedeće podatke: njene koordinate (x, y, z) i njenu masu m. Napisati program u kome se podaci učitavaji i prikazuju, pa zatim računa masa ukupnog sistema, njegovo težište, kao i koordinate i redni broj najteže tačke. Ispisati rezultate.

68. Napisati program na Pascalu za računanje skalarnog proizvoda dva vektora $(x_1, \ldots, x_n)$ i $(y_1, \ldots, y_n)$. Vektori se učitavaju i prikazuju, a prikazati i izračunati skalarni proizvod.

69. Sastaviti program koji će izbrojati koliko se različitih slova pojavljuje u jednom redu, koji se učitava preko datoteke input.

70. Sastaviti program u kome se učitava kvadratna matrica oblika 4×4 i određuje njen najveći element. Neka program odštampa: matricu, vrijednost najvećeg elementa i brojeve retka i stupca u kome se on nalazi.

71. Napisati program koji učitava matricu veličine 3×3 i računa njenu determinantu. Odštampati matricu i determinantu.

72. Napisati program koji učitava 20 cijelih brojeva i uređuje ih u rastući oblik. Treba odštampati brojeve u originalnom i uređenom redosljedu. Uputstvo: nađi najmanji broj i zamijeni ga sa prvim. Nađi najmanji broj od drugog pa dalje i zamijeni ga sa drugim. Nađi najmanji broj od trećeg pa dalje i zamijeni ga sa ....

73. Napisati program koji učitava dva 50–cifarska broja i sabira ih. Saopštiti rezultat.

74. Napisati program koji štampa prvih 20 faktorijela $n!$ Moraju biti odštampane sve cifre, ne važe realni brojevi.

75. Prvih 50 stepena broja 2.

76. Napisati program koji će pomoću Eratostenovog sita pronaći sve proste brojeve od 1 do 1000 i odštampati ih.

77. Po Goldbachovoj hipotezi, svaki paran broj $n \geq 4$ može se prikazati kao zbir dva prosta broja. Provjeriti hipotezu za parne brojeve $n \leq 200$.

78. Napisati program koji učitava prirodan broj, pretvara ga u sistem sa osnovom 3 i štampa original i rezultat.

79. Napisati program za štampanje magičnog kvadrata neparnog reda. Magični kvadrat reda n je kvadratna matrica oblika $n \times n$ čiji su elementi $1, 2, \ldots, n^2$ tako raspoređeni da je zbir u svakom retku, u svakom stupcu i po svakoj dijagonali jedan te isti.

80. Napisati program za mijenjanje po padežima u jednini ženske imenice prve vrste, kao npr. pjesma, pjesme, pjesmi, pjesmu, oj pjesmo, sa pjesmom, u pjesmi. Može se pretpostaviti da imenica ima najviše 10 slova. Neka program prvo provjeri da li se zaista završava sa slovom "a".

81. Sastaviti tip podataka za opis studenta, gdje su potrebne sljedeće informacije: prezime i ime, datum i mjesto rođenja, pol, stalna adresa, godina i ocjene 30 ispita.

82. Binarnu relaciju R na skupu $X = \{1, \ldots, n\}$ možemo predstaviti u Pascalu na nekoliko načina. Obično se koriste sljedeće strukture podataka: type X = 1 .. n ; relacija = array [X] of set of X ; ili type relacija = array [X, X] of Boolean ; Ako je R tipa relacija onda važi $(x, y) \in R$ u prvom slučaju ako i samo ako je x u skupu R[y], dok u drugom slučaju $(x, y) \in R$ ako i samo ako važi R[x,y]. Napisati program koji utvrđuje da li je data relacija: a) simetrična, b) antisimetrična, c) tranzitivna, d) refleksivna, e) relacija ekvivalencije. Uputstvo: za relaciju ekvivalencije, upotrebi potprograme koji provjeravaju svojstva (a), (c) i (d). Sam odluči o obliku ulaznih podataka.

83. Data je binarna relacija R na skupu $\{1, \ldots, n\}$ ($n \leq 50$). Sastaviti program koji utvrđuje da li je to: a) relacija linearnog uređenja, b) relacija parcijalnog uređenja.

84. Data je binarna relacija R. Sastaviti program koji će dati: a) tranzitivno zatvorenje relacije R (najmanju tranzitivnu relaciju koja sadrži R), b) najmanju relaciju ekvivalencije koja sadrži R i c) klase ekvivalencije relacije pod (b). Uputstvo: relaciju pod (b) dobićeš tako što ćeš prvo proširiti R do refleksivne i simetrične relacije, pa zatim uzeti njeno tranzitivno zatvorenje.

85. Napisati program za računanje kvadrata date binarne relacije R.

86. Date su dvije binarne relacije R i S na skupu brojeva $\{1, \ldots, n\}$, gdje se broj $n \leq 50$ učitava. Sastaviti program za dobijanje proizvoda (kompozicije) $R \circ S$ dvije relacije.

87. Strukturu grupoida (skupa X sa binarnom operacijom "$*$") možemo u Pascalu opisati sljedećim tipom: type X = 1 .. n ; grupoid = array [X, X] of X ; Ako je G tipa grupoid onda to znači da je kompozicija elemenata x i y, $x * y$, jednaka G[x, y]. Sastaviti program koji učitava i prikazuje tablicu kompozicije datog grupoida.

88. Dat je grupoid G. Sastaviti program koji utvrđuje da li u G postoji neutralni element (lijevi, desni neutralni element) i, u slučaju da postoji, prikazuje sve invertibilne (slijeva, zdesna invertibilne) elemente, kao i njihove inverze (lijeve, desne).

89. Sastaviti program koji ispituje asocijativnost operacije u datom grupoidu.

90. Linearnu listu možemo u Pascalu predstaviti preko pokazivača. Deklaracija tipa izgleda ovako:

```
type pointer = ^ karika ;
     karika = record kljuc : integer ;
```

novi : pointer end ;

Sastaviti sljedeće potprograme za rad sa linearnim listama:

a) Ulazne podatke (cijeli brojevi) povezati u linearni spisak. Odluči se za jednu od sljedećih mogućnosti: – da zadnji učitani broj bude na početku liste, – da zadnji učitani broj bude na kraju liste, ili – da lista bude uređena (direktno umetanje).

b) Pronaći element liste čiji je ključ k. Ako takvog elementa nema, onda se zaustavi na kraju liste (varijanta: ako je lista uređena, zaustavi se ispred prvog elementa čiji je ključ veći).

c) Ispred (iza) elementa liste čiji je ključ k umetni element čiji je ključ h (ulazni podatak). Ako u listi nema elementa sa ključem k, onda dodaj h na kraj liste (ako je lista uređena, na odgovarajuće mjesto u listi). Neka ti potprogram (b) pomogne.

d) Element čiji je ključ k odstraniti iz liste.

5) Potprogrami

91. Sastaviti program za nalaženje nule polinoma $p(x)$ po Newtonovoj metodi: $x_{r+1} = x_r - p(x_r)/p'(x_r)$, gdje je x_0 pogodno izabrana početna aproksimacija. Za računanje vrijednosti polinoma i njegovog izvoda upotrebi funkcijski potprogram, koji se temelji na Hornerovoj šemi.

92. Sastaviti potprogram za množenje dva polinoma.

93. Sastaviti sljedeće potprograme za računanje sa kompleksnim brojevima: saberi — saberi dva kompleksna broja, oduzmi — oduzmi dva kompleksna broja, mnozi — pomnoži dva kompleksna broja, dijeli — podijeli dva kompleksna broja, korijen — kvadratni korijen (sa nenegativnim realnim dijelom). Na primjer procedure saberi (x, y, u, v: real; var a, b: real) ; treba da sabere $x + iy$ i $u + iv$ i da upiše rezultat u $a + ib$.

94. Napisati program za tabeliranje niza kompleksnih brojeva: $z_{n+1} = \sqrt{z_n z_{n-1} + 1}$, gdje su z_0 i z_1 podaci. Za računanje upotrebi potprograme iz prethodnog zadatka.

95. Dat je polinom sa realnim koeficijentima. Istabelirati ga u cjelobrojnim kompleksnim tačkama $z = x+iy$, $-2 \le x, y \le 2$. Za računanje upotrebi potprograme iz prethodnih zadataka.

96. Sastaviti (nerekurzivan) potprogram function gcd (u, v: integer): integer ; za računanje najvećeg zajedničkog djelioca prirodnih brojeva u i v. Potprogram smije da upotrebljava samo sljedeća svojstva najvećeg zajedničkog djelioca: a) iz $u > v$ slijedi $\gcd(u, v) = \gcd(u - v, v)$, b) $\gcd(u, v) = \gcd(v, u)$, c) $\gcd(u, u) = u$.

97. Neka je x dati broj, $1 < x < 2$. Definišimo nizove: $a_0 = 1$, $c_0 = 1 - x$, $a_{i+1} = a_i(1 + c_i)$, $c_{i+1} = c_i^2$ $(i = 0, 1, 2, \ldots)$. Dokazati da važi $\lim_{n\to\infty} a_n = 1/x$, $\lim_{n\to\infty} c_n = 0$ i na osnovu toga sastaviti funkcijski potprogram za računanje recipročne vrijednosti broja x (naravno bez dijeljenja).

98. Neka je x dati broj, $1 < x < 2$. Definišimo nizove: $a_0 = x$, $c_0 = 1-x$, $a_{i+1} = a_i(1+c_i/2)$, $c_{i+1} = (c_i(3 + c_i)/4)^2$ $(i = 0, 1, 2, \ldots)$. Dokazati da važi $\lim_{n\to\infty} c_n = 0$, $\lim_{n\to\infty} a_n = \sqrt{x}$. Na osnovu toga, sastaviti funkcijski potprogram za računanje kvadratnog korijena realnog broja x $(1 < x < 2)$.

99. Sastaviti potprogram function exp (x : real) : real ; na sljedećim osobinama: a) $\exp(-x) = 1/\exp(x)$, b) $\exp(x) = \exp(\mathrm{trunc}(x))\exp(x - \mathrm{trunc}(x))$, $x \ge 0$, c) $\exp(x) = 1 + x + x^2/2! + x^3/3! + \ldots$, $0 \le x < 1$. U slučaju prirodnog broja n, $\exp(n)$ izračunati pomoću potprograma za stepenovanje realnog broja na prirodni eksponent.

100. Napisati generator slučajnih brojeva po Fibonaccijevoj metodi $x_{n+1} = (x_n + x_{n-1}) \bmod 2^k$. Izraditi potprogram RANSET koji će definisati početne vrijednosti generatora (x_0, x_1 i k) i funkcijski potprogram RANDOM : real za generisanje slučajnih brojeva između 0 i 1.

101. Sastaviti funkcijski potprogram UCITAJ koji će pročitati prvi znak različit od blanko. Ako do kraja datoteke nema nijednog odgovarajućeg znaka, neka vrati blanko kao pročitani znak.

102. Sastaviti funkcijski potprogram CARD koji vraća kardinalni broj datog skupa tipa set of 1 .. max , gdje je max konstanta, deklarisana u glavnom programu.

103. Sastaviti funkcijski potprogram za računanje prve norme $\|A\|_1$ date kvadratne matrice. Prva norma matrice A je broj $\max_{1\le j\le n}\sum_{i=1}^{n}|a_{ij}|$.

104. Sastaviti program za računanje svojstvenih vrijednosti date realne matrice A veličine 2×2. Uzeti u obzir da svojstvene vrijednosti mogu biti realni ili konjugovano kompleksni brojevi. Uputstvo: znamo da su svojstvene vrijednosti λ rješenja jednačine $\det(A-\lambda I)=0$.

6) Rekurzija

105. Napiši funkcijski potprogram PROIZVOD sa cjelobrojnim parametrima a i b za računanje vrijednosti sljedeće funkcije:

$$f(a,b)=\begin{cases} 0\text{ , ako je } b=0\\ f(a,b-1)+a\text{ , ako je } b \text{ neparan}\\ 2f(a,b/2)\text{ , ako je } b \text{ paran } (b\neq 0)\end{cases}$$

Pomoću matematičke indukcije, dokaži da je $f(a,b)=ab$, samo da je $b\ge 0$. Šta se dešava u slučaju kada je $b<0$?

106. Napisati efikasan (rekurzivan) funkcijski potprogram za računanje n–tog stepena realnog broja, gdje je n prirodan broj. Uputstvo: iskoristi prethodni zadatak.

107. Funkciju $f(n)$ (n prirodan broj), definisanu rekurzivnom relacijom $f(n)=f(n-1)+f(n-2)$, $f(0)=a$, $f(1)=b$ (a i b su podaci), možemo računati na dva načina: a) rekurzivno po gornjoj definiciji ili b) iterativno, tako što redom računamo $f(2)$, $f(3)$, ..., $f(n)$. Napisati program za računanje $f(25)$ na oba načina. Izmjeriti vrijeme koje je računar potrošio prilikom rekurzivnog, odnosno iterativnog računanja. Šta primjećeš? Da li je rekurzija u ovom slučaju efikasna?

108. Šta će uraditi sljedeći potprogram:

```
procedure palindrom ; var ch : char ;
begin read ( ch ) ;
if ch <> ' '
        then begin write ( ch ) ; palindrom end
        else writeln ;
write ( ch ) end ;
```

109. Sastaviti program koji će ispisati sve moguće permutacije pet elemenata.

110. Razmotrimo aritmetički izraz E sastavljen od znakova binarnih operacija + i − i imena promjenljivih a, b, c i d. Pretpostavlja se da je izraz gramatički ispravan. Učitava se izraz, a vrijednosti promjenljivih su date: $a=20$, $b=22$, $c=24$, $d=30$. Program treba da izračuna vrijednost izraza i da saopšti dobijeni rezultat. Npr. ako je $E=a-d+a$ onda imamo 10 kao rezultat.

111. Isto kao prethodni zadatak, samo što je sada dopušteno da u izrazu E učestvuju još dva simbola: lijeva i desna zagrada (i).

112. Kakav je učinak sljedećeg potprograma:

```
procedure napisati ( n : integer ) ;
begin if n >= 10 then napisati ( n div 10 ) ;
write ( n mod 10 : 1 ) end ;
```

113. Napisati kratak program (bez upotrebe nizova) za prikazivanje broja n (podatak) u binarnom sistemu. Uputstvo: iskoristi prethodni zadatak. Šta se dešava kada je $n\le 0$?

114. Napisati program koji će prikazati sve moguće rasporede osam kraljica na šahovskoj ploči, tako da se nikoje dvije među postavljenim kraljicama ne mogu ponijeti.

115. Binarno drvo T je sljedećeg tipa:

```
type drvo = ^ korijen ;
     korijen = record elt : element ; kljuc : integer ;
                      lijevi : drvo ; desni : drvo end ;
var T : drvo ;
```

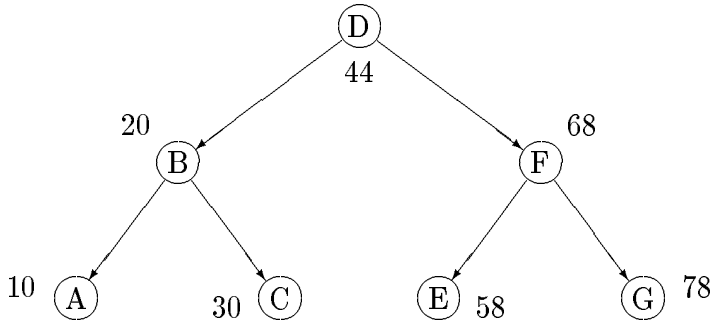
Ako posmatramo neki element u drvetu, onda bi trebalo da su u lijevom poddrvetu samo elementi sa manjim, a u desnom samo elementi sa većim ili jednakim ključem.

a) Napisati potprogram UMETNI sa parametrima x (ključ) i p (tipa drvo) koji će u binarno drvo p umetnuti element E (globalna promjenljiva) čiji je kliuč x.

b) Napisati (rekurzivan) potprogram za ispisivanje elemenata binarnog drveta u uređenom redosljedu.

116. Visina $h(T)$ i težina $w(T)$ binarnog drveta T definišu se rekurzivno ovako: $h(T) = -1$, ako je drvo prazno ($T = \text{nil}$), $h(T) = 1 + \max(h(T.\text{lijevi}), h(T.\text{desni}))$, inače i $w(T) = 0$, ako je drvo prazno ($T = \text{nil}$), $w(T) = 1 + w(T.\text{lijevi}) + w(T.\text{desni})$, inače. Napisati potprogram za računanje visine i težine binarnog drveta.

117. Neka su deklarisane vrste podataka type drvo = ^ vrh ; vrh = record e : char ; k : integer ; l , r : drvo end ; i promjenljiva var T : drvo ; tako da T predstavlja jedno binarno drvo. Napisati program za generisanje potpunog binarnog drveta visine $h = 2$ sa elementima i ključevima kako pokazuje slika:



Napomena: u nastavku bi trebalo izvršiti neku obradu koja se tiče drveta, npr. odrediti najmanji ključ u drvetu. Drvetu se pristupa posredstvom promjenljive T koja pokazuje na korijen drveta. Drukčije rečeno, preko T^ počinje pregledanje drveta.

118. Date su deklaracije:

```
type prizma = record a , h : real ; n : integer end ;
var p : prizma ;
```

tako da promjenljiva p sadrži podatke o jednoj pravilnoj prizmi: dužina stranice osnovice, visina prizme i koliko strana ima osnovica. Napisati program u kome se učitavaju podaci o jednoj pravilnoj prizmi, a zatim računaju i saopštavaju njena površina i zapremina. Još, da li je p kocka.

7) Datoteke i sortiranje

119. Napisati potprogram za učitavanje cijelog broja (bez potprograma read).

120. Napisati potprogram za pisanje cijelog broja (bez potprograma write).

121. Napisati program koji preko tastature učitava niz brojeva i prepisuje ih u datoteku F tipa file of integer. Uzeti da su brojevi koji se učitavaju pozitivni, da se svaki nalazi u posebnom redu datoteke input i da pojava negativnog broja označava kraj niza.

122. Napisati potprogram MERGE za spajanje dvije uređene datoteke sa cijelim brojevima.

123. Napisati program u kome se na osnovu postojeće datoteke T1 tipa text formira nova datoteka T2 istog tipa, tako da se više uzastopnih blankova zamjenjuje sa samo jednim blankom. Struktura redova treba da bude sačuvana.

124. Napisati program za prepisivanje sadržaja iz jedne tekstualne datoteke u drugu datoteku istog tipa, s tim da u drugoj datoteci smije da bude najviše 80 karaktera u jednom redu.

125. Napisati program koji učitava proizvoljni program na Pascalu i saopštava koliko puta se pojavljuju rezervisane riječi BEGIN i END. Pretpostaviti da nema niski i komentara.

126. Napisati program koji će izbrojati koliko ima samoglasnika u tekstu koji se preuzima preko standardne ulazne datoteke input.

127. Data je datoteka f tipa file of integer. Napisati program koji će kazati kolika je dužina najduže čete u njoj (uređenog rastućeg podniza).

128. Data je datoteka f tipa file of integer. Napisati program koji će saopštiti koliko u njoj ima negativnih brojeva, koliko nula i koliko pozitivnih.

129. Napisati program koji prati sportsko takmičenje, recimo trku na 100 m ili skok u dalj. Učitavaju se imena takmičara i njihovi rezultati, što treba da čini jednu datoteku. Druga datoteka neka sadrži redosljed prvih 10 učesnika. Detalje sam odredi.

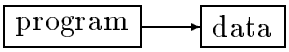
130. Napisati program za uređivanje niza cijelih brojeva $x_1, \ldots, x_n$ $(n \geq 1)$ po metodi quicksort.

131. Uporediti vrijeme koje se potroši prilikom sortiranja slučajnih brojeva po sljedećim metodama: a) obični algoritam: nađi najmanji broj, zatim nađi najmanji među preostalim brojevima, itd. b) quicksort: razvrstaj sve elemente na manje i veće od izabranog elementa, uredi manje (veće) brojeve i c) mergesort: vrši se spajanje dvije uređene polovine, do kojih se dolazi istim postupkom. Za svaku od mogućnosti (a) – (c), izmjeriti vrijeme u slučaju $n = 100$, $n = 500$, $n = 1000$ i $n = 2000$ podataka. Pojedini podatak je cio broj.

132. Neka je deklarisana promjenljiva

var data : file of integer ;

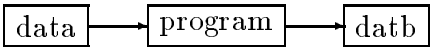
Napisati program koji će u datoteku data upisati sve proste brojeve manje od 1000. Zatim prikazati na ekranu sve proste brojeve između $a = 30$ i $b = 200$, putem čitanja iz data. Ilustracija:



133. Neka su deklarisane promjenljive

var data , datb : file of integer ;

Napisati program koji će u datoteku datb upisati sve proste brojeve do 200. Popunjavanje treba izvršiti preuzimanjem elemenata iz datoteke data iz prethodnog zadatka. Dalje, prikazati na ekranu sve proste brojeve manje od 200, putem čitanja iz datoteke. Ilustracija:



134. Neka su deklarisane promjenljive

var data : file of integer ; datc : file of char ;

Napisati program u kome se obrazuje nova datoteka datc, da sadrži sve proste brojeve do 1000. Popunjavanje datoteke datc vrši se preuzimanjem podataka iz već postojeće datoteke data od maloprije. Neka bude po 10 brojeva u jednom redu nove datoteke.

Objašnjenje: tekstualne datoteke su promjenljive tipa file of char ili svejedno rečeno tipa text. Znamo da se tekstualne datoteke mogu slobodno editovati, iz operativnog sistema.

135. Neka su deklarisane promjenljive

var datc , datd : file of char ;

Napisati program koji će u datoteku datd upisati sve proste brojeve do 200. Popunjavanje treba izvršiti preuzimanjem elemenata iz datoteke datc iz prethodnog zadatka.

136. Razmotrimo deklaracije var datc , date : file of char ; gdje smo maloprije govorili o tome šta datc sadrži. Napisati program za obrazovanje datoteke date, treba da sadrži sve proste brojeve p kao i datoteka datc (to znači $p \leq 1000$), s tim da ispred svakog prostog broja treba da piše njegov redni broj i, uzmimo u zagradi. Tako da će nova datoteka izgledati ovako: (1) 2 (2) 3 (3) 5 (4) 7 (5) 11 .... Naravno da se tokom rada programa vrši prepisivanje brojeva iz prve datoteke u drugu.

Trebalo bi da u novoj datoteci bude najviše 80 karaktera u jednom redu.

8) Dodatni zadaci

137. Napisati program na Pascalu za tabeliranje vrijednosti funkcije $y = \sin x$, gdje u jednom redu treba prkazati po pet parova (x, y), argument x kreće se od 0 do 90 stepeni sa korakom po 1 stepen, a vrijednosti funkcije y neka budu na pet decimala.

138. Učitavaju se koordinate tačaka tj. parovi cijelih brojeva (x_i, y_i) za $1 \leq i \leq 100$, gdje je $0 \leq x_i, y_i \leq 20$. Želimo da na ekranu prikažemo raspored tačaka, u obliku matrice. Na mjestu (j, k) treba odštampati znak + ako postoji samo jedna tačka sa takvim koordinatama, a znak $*$ ako takvih tačaka ima više. Ostaviti prazno mjesto ako nema takve tačke, gdje je $0 \leq j, k \leq 20$. Obrati pažnju na pravi smjer ose y.

139. Napisati program za određivanje najmanjeg prirodnog broja koji se može izraziti bar na dva načina kao zbir dva kvadrata.

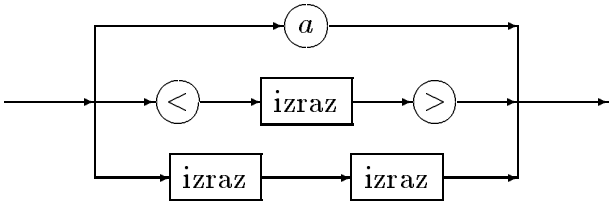
140. Napisati program koji će ispitati da li su u aritmetičkom izrazu, koji se učitava, zagrade pravilno postavljene. Izraz je sastavljen od +. −, a, b, c, d, (i).

141. Učitavaju se brojevi i i j, gdje je $1 \leq i, j \leq 8$, čime je određeno jedno polje na šahovskoj tabli. Još se učitava i jedno slovo, govori koja se figura na tom polju nalazi: Kralj, Dama, Top, Skakač, Lovac ili Pion. Uzmimo da nema drugih figura na tabli. Napisati program u kome se računa koliko polja napada postavljena figura i saopštava izračunata vrijednost.

142. Na šahovskoj tabli nalaze se samo dvije figure istog tipa. Koliko polja tuče makar jedna figura.

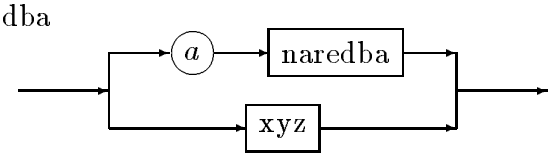
143. Učitava se niz karaktera w u kome učestvuju a, < i > (na primjer do znaka x), npr. w = <<aaa>> ili $w = aa$<<. Napisati program u kome se utvrđuje da li izraz w pripada jeziku definisanom sljedećim sintaksnim dijagramom:

izraz

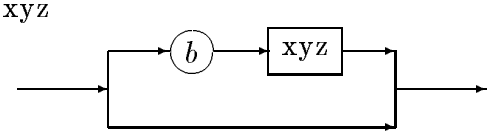


144. Riječ w predstavlja ulazni podatak programa. Treba odrediti da li w pripada jeziku L, tj. predstavlja li ona naredbu u smislu definicije date sljedećim sintaksnim dijagramima:

naredba



xyz



145. Tekstualna datoteka T (to znači var T : text ;) sastoji se od riječi razdvojenih znacima blanko i (uopšte uzev) ima više redova. Napisati program koji će izbrojati koliko se puta u datoteci T pojavljuje riječ "tekst", odnosno riječ "fajl".

146. Neka je data deklaracija var T : text ; Da li su riječi u razmatranoj datoteci T uređene leksikografski (kao u rječniku).

147. Podaci o neusmjerenom grafu G sastoje se od prirodnog broja n, čime se definiše da je skup tačaka $\{1,\ldots,n\}$, i matrice koincidencije A oblika $n \times n$, čiji su elementi 0 i 1, koja definiše veze između tačaka: $a_{ij} = 1$ ako i samo ako (i,j) je ivica u grafu. Znamo da je matrica A simetrična i da ima nule po glavnoj dijagonali.

Napisati program u kome se učitavaju podaci o jednom neusmjerenom grafu G i ispituje da li je graf povezan. Saopštiti odgovor. Saopštiti još i spisak povezanih komponenti.

148. Napisati program u kome se određuje da li je dati neusmjereni graf G bipartitan. Drugim riječima, da li se tačke grafa mogu podijeliti u dvije grupe, tako da nema ivica između tačaka iz svake od dvije grupe. Uputstvo: ako su dvije tačke povezane ivicom onda su one sigurno u različitim grupama. Zato su svi susjedi neke tačke u onoj drugoj grupi (nisu u njenoj), svi njihovi susjedi su u istoj grupi kao i izabrana tačka, itd.

149. Za graf se kaže da je Eulerov ako može da bude nacrtan u jednom potezu, bez podizanja olovke ili duplog povlačenja neke linije, s tim da crtanje počinje i završava se u istoj tački. Dokazano je da je graf Eulerov ako i samo ako je stepen svake tačke (broj njenih susjeda) paran, samo još da je povezan. Napisati program u kome se učitavaju podaci o grafu G i određuje da li je Eulerov. Ako jeste, odštampati još i niz vrhova i grana koje ćemo povući, da bismo nacrtali graf u jednom potezu.

150. Kažemo da je graf pravilno obojan ako su svake dvije susjedne tačke različite boje. Napisati program koji će pravilno obojati tačke datog grafa po sljedećem postupku: – uredi tačke po opadajućim stepenima, – u tom redosljedu, ofarbaj svaku tačku prvom slobodnom bojom. Za boje uzeti prirodne brojeve redom.

151. Za $n \geq 1$, Eulerova funkcija $\varphi = \varphi(n)$ definisana je sa: $\varphi(n) = \#\{k \mid 1 \leq k \leq n, \mathrm{NZD}(k,n) = 1\}$ (koliko ima uzajamno prostih), npr. $\varphi(12) = \#\{1,5,7,11\} = 4$. Napisati program u kome se tabelira funkcija $\varphi = \varphi(n)$ za $1 \leq n \leq 100$.

152. Neka su p i q prosti brojevi $(p \neq q)$ i $k,l \geq 1$ i stavimo $n = p^k q^l$. Tada važi: $\varphi(n) = p^{k-1}(p-1)q^{l-1}(q-1)$. Napisati program u kone se provjerava navedena teorema za brojeve n razmatranog oblika, pri čemu $p,q \in \{2,3,5,7,11,13\}$, $k,l \in \{1,2,3\}$.

Sa adrese www.freepascal.org može se besplatno preuzeti softver (misli se prevodilac), pa smo onda u prilici da vježbamo.

Ili preuzeti fajl tp55.zip za Turbo Pascal.

PASCAL RJEŠENJA ZADATAKA

1) Uvod u programiranje

12. Opiši postupak za računanje e^x pomoću reda: $e^x = 1 + x/1! + x^2/2! + x^3/3! + \ldots$ Vrijednost e^x želimo da izračunamo precizno, tako da vrijednost zadnjeg uračunatog člana treba da bude manja od 1.0E−8 .

Rješenje.

Neka je $x \geq 0$. U cilju efikasnog računanja $y = e^x$, rastavimo ga kao $x = a + b$, gdje $a \in N_0$, $0 \leq b < 1$. Imamo da je $e^x = e^{a+b} = e^a \cdot e^b$. Za računanje e^b treba iskoristiti razvoj u stepeni red:

$$e^b = 1 + b + \frac{b^2}{2!} + \frac{b^3}{3!} + \ldots$$

budući da tada opšti član reda (to je $b^n/n!$) brzo teži ka nuli kad $n \to \infty$, pa je dovoljno da se zadrži samo mali broj sabiraka.

Što se tiče e^a, vrši se uzastopno množenje. Naime, konstanta $e = 2{,}7182818284$ bila je ranije izračunata i zapamćena.

Dalje, u slučajevima kada je a veliko, treba i stepenovanje izvoditi po nekom efikasnom postupku.

13. Šta je pogrešno u algoritmu koji bi se mogao opisati ovako:

SVE DOK JE $|x| \leq 1$ PONAVLJAJ $x \leftarrow 2/x$

Rješenje.

Neka je $a \geq 1$ (npr. $a = 2$). Za računanje $\sqrt{a}$ može da posluži Heronov obrazac: $x_1 = 1$, $x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$ $(n \geq 1)$. Naime, tada važi $\lim_{n\to\infty} x_n = \sqrt{a}$. Zapaziti da je $x = \sqrt{a}$ rješenje jednačine $x = \frac{1}{2}\left(x + \frac{a}{x}\right)$.

2) Osnovni programi

19. Sastaviti program koji učitava brojeve a i b, koji predstavljaju realni i imaginarni dio kompleksnog broja $z = a + bi$, pa računa i štampa odgovarajuće parametre r i φ polarnog zapisa $z = re^{i\varphi} = r(\cos\varphi + i\sin\varphi)$ tog broja. Kao što znamo, važi: $\varphi = \mathrm{arctg}(b/a)$ i $r = \sqrt{a^2 + b^2}$. Neka program odštampa i – u kom kvadrantu leži broj z. Neka se ponavlja za četiri broja z.

Rješenje. Smatramo da su kvadranti otvorene oblasti. Program bi trebalo dopuniti, da radi i kada je $(a, b) = (0, 0)$.

```
program kompleksni ( input , output ) ;
var pi, a, b, r, fi: real ; brojac, kvad: integer;
begin  pi:=3.1415926535;
for brojac := 1 to 4 do
    begin  read( a, b ); write( 'a b r fi ' );
    write( a:12, b:12 ); r := sqrt( a*a + b*b );
    if a>0 then fi := arctan( b / a );
    if a<0 then fi := arctan( b / a ) + pi;
    if a=0 then if b>0 then fi := pi / 2 else fi := 3 * pi / 2;
    write( r:12, fi:12 ); kvad := 0;
    if (a>0) and (b>0) then kvad := 1;
```

```
    if (a<0) and (b>0) then kvad := 2;
    if (a<0) and (b<0) then kvad := 3;
    if (a>0) and (b<0) then kvad := 4;
        Case kvad of 1: writeln(' I kvadrant'); 2: writeln(' II kvadrant');
        3: writeln(' III kvadrant'); 4: writeln(' IV kvadrant');
        0: writeln End  end;
writeln( 'kraj programa' )  end.
```

Test primjer: `2 2 3 -1 1 0 1.73 1` Izlazi:

```
a b r fi  2.00000E+00 2.00000E+00 2.82843E+00 7.85398E-01 I kvadrant
a b r fi  3.00000E+00-1.00000E+00 3.16228E+00-3.21751E-01 IV kvadrant
a b r fi  1.00000E+00 0.00000E+00 1.00000E+00 0.00000E+00
a b r fi  1.73000E+00 1.00000E+00 1.99822E+00 5.24112E-01 I kvadrant
kraj programa
```

23. Napisati program koji učitava cijele brojeve x i y, prikazuje te brojeve, kao i njihov količnik i ostatak pri dijeljenju. U slučaju da je y = 0, neka program saopšti poruku 'GRESKA'.

Rješenje. Ugrađena funkcija div, npr. $a \operatorname{div} b$, ima cjelobrojne argumente a i b i daje cjelobrojni rezultat – količnik a i b (cio dio njihovog količnika). Slično, i standardna funkcija mod, npr. $a \bmod b$ ima cijele argumente i vraća cijeli rezultat – ostatak pri dijeljenju $a : b$.

```
program divmod ( input , output ) ;
var x , y : integer ;
begin
writeln( 'unesite x i y' );
read( x, y );
writeln( 'x=', x, ' y=', y );
if y<>0 then
    writeln( 'x div y=', x div y, ' x mod y=', x mod y )
    else writeln( 'greska y=0' ) ;
end .
```

Samo se napominje da, u programu, zadnji znak tačka–zarez može da se izostavi. Dali smo ulazne podatke: `555 12` Rezultat:

```
unesite x i y
x=555 y=12
x div y=46 x mod y=3
```

3) Kontrolne strukture

38. Napisati program koji će na ekranu prikazati prvih 10 članova niza $\{x_n\}$, gdje je $f_{n+1} = f_n + f_{n-1}$, $x_n = f_n/f_{n-1}$, $f_0 = a$, $f_1 = b$, gdje su a i b ulazni podaci.

Rješenje. Za niz Fibonaccijevih brojeva $\{f_n\}$ važi relacija: $f_n = c_1((1-\sqrt{5})/2)^n + c_2((1+\sqrt{5})/2)^n$, gdje $c_1, c_2 \in R$ zavise od a i b. Zato, lako se vidi da je $\lim_{n\to\infty} f_n/f_{n-1} = (1+\sqrt{5})/2$, osim kada je $c_2 = 0$. Imamo da je $\sqrt{5} = 2{,}23607$, $(1-\sqrt{5})/2 = -0{,}61803$, $(1+\sqrt{5})/2 = 1{,}61803$.

```
program Fibonacci( input , output ) ;
var f : array [ 0 .. 10 ] of integer ;
x : array [ 1 .. 10 ] of real ;
a, b, i : integer ;
begin
read( a, b ); f[ 0 ] := a; f[ 1 ] := b;
for i := 2 to 10 do f[ i ] := f[ i-1 ] + f[ i-2 ];
for i := 1 to 10 do x[ i ] := f[ i ] / f[ i-1 ];
writeln( 'i,  f(i),  f(i)/f(i-1)' );
i := 0; writeln( i:3, f[ i ]:5 );
for i := 1 to 10 do
      writeln( i:3, f[ i ]:5, x[ i ]:14 );
end.
```

Ulaz: 1 1 Izlaz:

```
i,  f(i),  f(i)/f(i-1)
  0    1
  1    1 1.0000000E+00
  2    2 2.0000000E+00
  3    3 1.5000000E+00
  4    5 1.6666667E+00
  5    8 1.6000000E+00
  6   13 1.6250000E+00
  7   21 1.6153846E+00
  8   34 1.6190476E+00
  9   55 1.6176471E+00
 10   89 1.6181818E+00
```

53. Napisati program koji učitava prirodan broj i rastavlja ga na proste činioce. Na primjer: $204 = 2 * 2 * 3 * 17$, $83 = 83$, $81 = 3 * 3 * 3 * 3$. Napisati program tako da učitava i rastavlja nekoliko brojeva.

Rješenje. Označimo sa n prirodan broj koji se učitava. U programu, djeljivost broja n brojem k ispituje se pomoću izraza n mod k (da li je ostatak prilikom dijeljenja jednak nuli). Ako se pronađe neki činilac k onda se razmatrani broj n sa njim podijeli, pa se nastavlja dalje. Treba uzeti u obzir da k može da bude i višestruki činilac. U programu se ispituje da li je n djeljivo sa $k = 2, 3, 4, \ldots$ redom, gdje ima i brojeva k koji nisu prosti, što ne dovodi do greške.

```
program faktor ( input , output ) ;
label 20, 40, 99 ;
const blanko = ' ' ;
var k, n: integer; ch: char;
    begin
20: read( n ); if n <= 1 then goto 99;
    writeln; write( n, blanko );
    ch := '='; k := 2;
40: if n mod k = 0 then
        begin write( ch, blanko, k, blanko );
        n := n div k; ch := '*';
```

```
        if n = 1 then goto 20;
        if n mod k = 0 then goto 40 end;
    k := k + 1; goto 40;
99: writeln; writeln( 'kraj programa' )
    end.
```

Program treba otkucati, prevesti i izvršiti. Ako ulazni podaci glase:
`2 3 4 5 45 111 1111 2222 32767 44 1`
onda ćemo imati kao rezultat:

```
2 = 2
3 = 3
4 = 2 * 2
5 = 5
45 = 3 * 3 * 5
111 = 3 * 37
1111 = 11 * 101
2222 = 2 * 11 * 101
32767 = 7 * 31 * 151
44 = 2 * 2 * 11
kraj programa
```

4) Sastavljene strukture podataka

65. Napisati program koji će tabelirati vrijednosti polinoma $p(x) = a_0x^n + a_1x^{n-1} + \ldots + a_n$ za $x = 0(0{,}1)1$. Učitati stepen i koeficijente. Uputstvo: upotrebi Hornerovu šemu.

Rješenje. Da bi se izračunala vrijednost polinoma $p = p(x)$ za jedno fiksirano x po Hornerovoj šemi, treba primijeniti niz formula:

$$t \leftarrow a_0,\ t \leftarrow xt + a_1,\ t \leftarrow xt + a_2,\ \ldots,\ t \leftarrow xt + a_n$$

gdje je zadnje $t = p(x)$. Mi treba da izračunamo $p(x)$ za $x = kh$, gdje je $h = 0{,}1$ i $0 \le k \le 10$.

```
program Horner ( input , output ) ;
var a : array [ 0 .. 10 ] of real ;
n, i : integer ; x, h, p, t : real ;
begin
read( n );
for i := 0 to n do read( a[i] );
x := 0.0; h := 0.1;
writeln( '        X                  Y' );
while x < 1.05 do
    begin  t := a[0];
    for i := 1 to n do t := x * t + a[i];
    p := t; writeln ( x, '  ', p );
    x := x + h  end;
writeln( '------ Horner ------' )
end.
```

Uzmimo da ulazni podaci glase: `4 1 1 -2 0 -2`. To znači da je $p(x) = x^4 + x^3 - 2x^2 - 2$. Tada, izvršavanjem programa, dobijamo ovakav rezultat:

```
        X                   Y
 0.0000000000E+00  -2.0000000000E+00
 1.0000000000E-01  -2.0189000000E+00
 2.0000000000E-01  -2.0704000000E+00
 3.0000000000E-01  -2.1449000000E+00
 4.0000000000E-01  -2.2304000000E+00
 5.0000000000E-01  -2.3125000000E+00
 6.0000000000E-01  -2.3744000000E+00
 7.0000000000E-01  -2.3969000000E+00
 8.0000000000E-01  -2.3584000000E+00
 9.0000000000E-01  -2.2349000000E+00
 1.0000000000E+00  -2.0000000000E+00
------ Horner ------
```

77. Po Goldbachovoj hipotezi, svaki paran broj $n \geq 4$ može se prikazati kao zbir dva prosta broja. Provjeriti hipotezu za parne brojeve $n \leq 200$.

Rješenje. U prvoj fazi rada programa, formira se spisak svih prostih brojeva $n \leq 200$. Da bi $n \geq 13$ bio prost, treba da nije djeljiv sa $k = 3$, niti sa većim neparnim k ($k^2 \leq n$). Članove spiska stavljamo u niz "prost".

U drugoj fazi rada programa, svi prosti brojevi $n \leq 200$ stavljaju se u skup čije je ime "primes" (prepisuju se iz niza).

U trećoj fazi, za parne brojeve $n \geq 8$, ispituje se mogućnost njihovog predstavljanja u obliku zbira dva člana skupa primes. Gleda se da li $k = 3$ može da posluži kao prvi sabirak (samim tim određen je drugi sabirak $n - k$), zatim eventualno veće k ($k \leftarrow k + 2$), itd.

```
program Goldbach ( output ) ;
label 15, 25, 35, 45, 55, 99 ;
var prost : array [ 1 .. 80 ] of integer ;
pr, k, n, i, broj, razl : integer ;
primes : set of 1 .. 200 ;
Boole : Boolean ;
    begin  prost[1] := 2; prost[2] := 3;
    prost[3] := 5; prost[4] := 7;
    prost[5] := 11; pr := 5;
    n := 13;
15: k := 3;
25: if ( n mod k ) = 0 then  {n je slozen}
        goto 35;
    if k * k < n then  {jos se ne zna}
        begin k := k + 2; goto 25 end
    else  {n je prost}
        begin pr := pr + 1; prost[pr] := n end;
35: if n < 199 then begin n := n + 2; goto 15 end;
    writeln( 'Goldbachova hipoteza' );
    writeln( 'prosti brojevi' );
    for i := 1 to pr do
        begin write( prost[i]:5 );
        if i mod 10 = 0 then writeln end;
```

```
    writeln; writeln( 'paran = prost + prost' );
    primes:=[ ];
    for i := 1 to pr do
        begin broj := prost[i];
        primes := primes + [ broj ] end;
    write( '              ' );
    n := 4; k := 2; razl := n - k;
    write( n:6, '=', k:3, '+', razl:3 );
    n := 6; k := 3; razl := n - k;
    write( n:6, '=', k:3, '+', razl:3 );
    n := 8;
45: k := 3;
55: razl := n - k;
    Boole := ( k in primes ) and ( razl in primes );
    if not Boole and ( k + k < n ) then  {jos se ne zna}
        begin k := k + 2; goto 55 end;
    if not Boole and not ( k + k < n ) then  {ne postoji p+p}
        begin writeln( 'propada za n = ', n:3 ); goto 99 end;
    if Boole then  {n=p+p}
        begin write( n:6, '=', k:3, '+', razl:3 );
        if ( n mod 10 ) = 0 then writeln end;
    n := n + 2;
    if n < 201 then goto 45;
    writeln( 'vazi u svim slucajevima' );
99: writeln( 'kraj programa' )  end .
```

Za niz prostih brojeva – prilikom njegove deklaracije – izdvojili smo (u memoriji) 80 mjesta, jer ne znamo unaprijed koliko tačno ima prostih brojeva $p \leq 200$. Ispostaviće se da ih ima $\mathrm{pr} = 46$. Treba napomenuti da je dio programa gdje piše "propada" u suštini suvišan, zato što se sigurno neće ostvariti, budući da je Goldbachova hipoteza provjerena do jedne jako velike granice N. Rezultat:

```
Goldbachova hipoteza
prosti brojevi
    2    3    5    7   11   13   17   19   23   29
   31   37   41   43   47   53   59   61   67   71
   73   79   83   89   97  101  103  107  109  113
  127  131  137  139  149  151  157  163  167  173
  179  181  191  193  197  199
paran = prost + prost
                   4=  2+  2     6=  3+  3     8=  3+  5    10=  3+  7
    12=  5+  7    14=  3+ 11    16=  3+ 13    18=  5+ 13    20=  3+ 17
    22=  3+ 19    24=  5+ 19    26=  3+ 23    28=  5+ 23    30=  7+ 23
    32=  3+ 29    34=  3+ 31    36=  5+ 31    38=  7+ 31    40=  3+ 37
    42=  5+ 37    44=  3+ 41    46=  3+ 43    48=  5+ 43    50=  3+ 47
    52=  5+ 47    54=  7+ 47    56=  3+ 53    58=  5+ 53    60=  7+ 53
    62=  3+ 59    64=  3+ 61    66=  5+ 61    68=  7+ 61    70=  3+ 67
    72=  5+ 67    74=  3+ 71    76=  3+ 73    78=  5+ 73    80=  7+ 73
    82=  3+ 79    84=  5+ 79    86=  3+ 83    88=  5+ 83    90=  7+ 83
```

```
    92=  3+ 89    94=  5+ 89    96=  7+ 89    98= 19+ 79   100=  3+ 97
   ( ... ... ... )
   172=  5+167   174=  7+167   176=  3+173   178=  5+173   180=  7+173
   182=  3+179   184=  3+181   186=  5+181   188=  7+181   190= 11+179
   192= 11+181   194=  3+191   196=  3+193   198=  5+193   200=  3+197
vazi u svim slucajevima
kraj programa
```

5) Potprogrami

103. Sastaviti funkcijski potprogram za računanje prve norme $\|A\|_1$ date kvadratne matrice. Prva norma matrice A je broj $\max_{1\le j\le n}\sum_{i=1}^{n}|a_{ij}|$.

Rješenje. Napisali smo ukupni program. Matrica A je oblika $n\times n$, gdje je $n\ge 1$. Potprogram norma ima dva parametra, prvi "a" je tipa matrice, a drugi "n" je tipa cijelog broja. Budući da je function, ima i rezultat, tipa real.

```
program normamatrice ( input , output ) ;
type matrica = array [ 1 .. 10 , 1 .. 10 ] of real ;
var a , b : matrica ; i , j , n : integer ; nor : real ;

function norma(a: matrica; n: integer): real;
var suma, max: real; i, j: integer;
begin max := 0;
    for j := 1 to n do
    begin suma := 0;
    for i := 1 to n do suma := suma + abs( a[i,j] );
    if suma > max then max := suma end;
norma := max end;

begin write('pocetak ');
read( n );
for i := 1 to n do
    for j := 1 to n do read( a[i,j] );
nor := norma( a, n ); writeln('norma=', nor);
for i := 1 to n do
    for j := 1 to n do b[i,j] := a[i,j] + 1.0;
nor := norma( b, n ); write('norma=', nor);
writeln(' kraj') end.
```

Da bi se program testirao, dajemo kao ulaz: `3 1 2 10 3 4 20 5 6 10` a kao odgovor dobijamo:

```
pocetak norma= 4.0000000000E+01
norma= 4.3000000000E+01 kraj
```

što znači da je $\|A\|_1 = 40$, ako je

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}=\begin{bmatrix} 1 & 2 & 10\\ 3 & 4 & 20\\ 5 & 6 & 10\end{bmatrix}$$

Vidimo da smo program ustvari malo dopunili, radi provjere njegove korektnosti, tako da je uvedena i druga matrica B istog oblika kao i A, njeni elementi su veći, tj. važi $b_{ij}=a_{ij}+1$. Kao što je već dato, program saopštava $\|B\|_1 = 43$.

104. Sastaviti program za računanje svojstvenih vrijednosti date realne matrice A veličine 2×2. Uzeti u obzir da svojstvene vrijednosti mogu biti realni ili konjugovano kompleksni brojevi. Uputstvo: znamo da su svojstvene vrijednosti λ rješenja jednačine $\det(A - \lambda I) = 0$.

Rješenje. Navedimo pregled potrebnih formula:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \lambda I = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix},\ A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},\ \lambda I - A = \begin{bmatrix} \lambda - a_{11} & -a_{12} \\ -a_{21} & \lambda - a_{22} \end{bmatrix}$$

$\det(\lambda I - A) = 0$, $(\lambda - a_{11})(\lambda - a_{22}) - a_{12}a_{21} = 0$, $\lambda^2 - (a_{11} + a_{22})\lambda + a_{11}a_{22} - a_{12}a_{21} = 0$, $\lambda^2 + p\lambda + q = 0$, $\lambda_{1,2} = -p/2 \pm \sqrt{p^2/4 - q}$ ako je $D = p^2/4 - q \geq 0$, a ako $D < 0$ onda $\lambda_{1,2} = -p/2 \pm i\sqrt{-D}$

```
program lambda(input, output);
var a: array [1 .. 2, 1 .. 2] of real;
p, q, d: real; lam1, lam2, x, y: real;
begin
read(a[1,1], a[1,2], a[2,1], a[2,2]);
p := -(a[1,1] + a[2,2]);
q := a[1,1] * a[2,2] - a[1,2] * a[2,1];
d := sqr(p/2)-q;
if d>=0 then
begin lam1 := -p/2 + sqrt(d); lam2 :=- p/2 - sqrt(d);
writeln('lam1=', lam1); writeln('lam2=', lam2) end
else
begin x := -p/2; y := sqrt(-d);
writeln('lam1=', x, '+i*', y); writeln('lam2=', x, '-i*', y) end
end.
```

Ako unesemo `2 2 1 3` onda ćemo dobiti

```
lam1= 4.0000000000E+00
lam2= 1.0000000000E+00
```

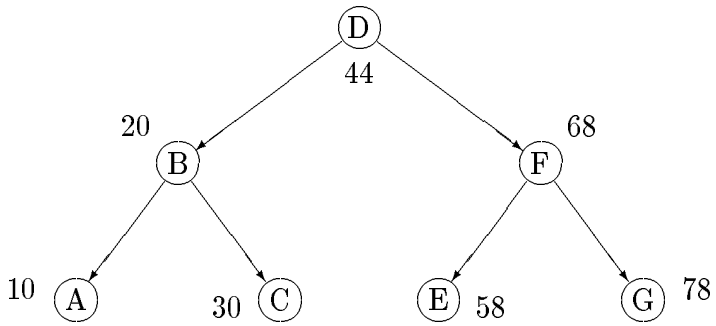
što znači da svojstvene vrijednosti matrice

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}$$

iznose $\lambda_1 = 4$, $\lambda_2 = 1$.

6) Rekurzija

117. Neka su deklarisane vrste podataka type drvo = ^ vrh ; vrh = record e : char ; k : integer ; l , r : drvo end ; i promjenljiva var T : drvo ; tako da T predstavlja jedno binarno drvo. Napisati program za generisanje potpunog binarnog drveta visine $h = 2$ sa elementima i ključevima kako pokazuje slika:



Napomena: u nastavku bi trebalo izvršiti neku obradu koja se tiče drveta, npr. odrediti najmanji ključ u drvetu. Drvetu se pristupa posredstvom promjenljive T koja pokazuje na korijen drveta. Drukčije rečeno, preko T^ počinje pregledanje drveta.

Rješenje. Efekat poziva npr. new(p4) je sljedeći: u memoriji se izdvoji prostor čija je veličina dovoljna za čuvanje jednog objekta odgovarajućeg tipa (u našem slučaju, objekat ima četiri komponente). Za razmatrani memorijski objekat koristi se oznaka p4^ odnosno njemu se pristupa posredstvom te oznake. Kod nas, veličina p4^ je, dimenziono posmatrano, jedan zapis (record), pa se komponentama zapisa pristupa preko tačke. Postoje četiri komponente, prva p4^.e je tipa karakter (element vrha), druga p4^.k je cio broj (ključ jednog vrha u drvetu), dok su treća i četvrta komponenta, u oznaci p4^.l onosno p4^.r istog tipa kao i p4, lijevi i desni pokazivač.

```
program binarnodrvo ( output ) ;
type drvo = ^vrh ;
vrh = record e: char; k: integer; l, r: drvo end ;
var p1, p2, p3, p4, p5, p6, p7, t, u, v: drvo;
ch: char; i: integer;
begin  new(p1); p1^.e:='A'; p1^.k:=10; p1^.l:=nil; p1^.r:=nil;
new(p3); p3^.e:='C'; p3^.k:=30; p3^.l:=nil; p3^.r:=nil;
new(p5); p5^.e:='E'; p5^.k:=58; p5^.l:=nil; p5^.r:=nil;
new(p7); p7^.e:='G'; p7^.k:=78; p7^.l:=nil; p7^.r:=nil;
new(p2); p2^.e:='B'; p2^.k:=20; p2^.l:=p1; p2^.r:=p3;
new(p6); p6^.e:='F'; p6^.k:=68; p6^.l:=p5; p6^.r:=p7;
New(p4); p4^.e:='D'; p4^.k:=44; p4^.l:=p2; p4^.r:=p6;
t := p4;
ch := t^.e; write( ch ); write(' ');
i := t^.k; write( i ); write(' ');
u := t^.l; ch := u^.e; write( ch ); write(' ');
v := t^.r; ch := v^.e; write( ch )  end.
```

Vidimo da je pri kraju program malo dopunjen, da bi nešto i odštampao, npr. o osnovnom vrhu drveta. Neka odštampa element u korijenu, ključ korijena i elemente u vrhovima neposredno lijevo, odnosno desno od korijena razmatranog binarnog drveta. Dobićemo: `D 44 B F`

118. Date su deklaracije:

type prizma = record a , h : real ; n : integer end ;

var p : prizma ;

tako da promjenljiva p sadrži podatke o jednoj pravilnoj prizmi: dužina stranice osnovice, visina prizme i koliko strana ima osnovica. Napisati program u kome se učitavaju podaci o

jednoj pravilnoj prizmi, a zatim računaju i saopštavaju njena površina i zapremina. Još, da li je p kocka.

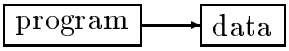
Rješenje. Predviđene su samo mogućnosti $n = 3$, $n = 4$ i $n = 6$ za broj strana osnovice. Uvedimo oznake B za površinu osnovice i O za površinu omotača. Poznate su formule za površinu i zapreminu: $P = 2B + O$, $V = Bh$. Pored toga, $B = a^2\sqrt{3}/4$ ako je $n = 3$, $B = a^2$ ako je $n = 4$ i $B = 3\sqrt{3}a^2/2$ u slučaju $n = 6$. Dalje, očito je da pravilna prizma predstavlja kocku ako je $n = 4$ i $a = h$.

```
program pravilnaprizma(input, output);
type prizma = record a, h : real; n : integer end;
var p: prizma;
o, b, povr, zapr: real; Boole: Boolean;
begin read( p.a, p.h, p.n );
o := p.n * p.a * p.h;
    case p.n of 3 : b := sqrt(3) * sqr(p.a) / 4;
    4 : b := sqr(p.a);
    6 : b := 3 * sqrt(3) * sqr(p.a) / 2 end;
povr := 2 * b + o; zapr := b * p.h;
write('p=', povr, ' v=', zapr);
Boole := ( p.n = 4 ) and ( p.a = p.h );
if Boole then write(' jeste kocka') else write(' nije kocka') end.
```

Rad programa isprobali smo na sljedećim ulaznim podacima: `20 2 6` Treba pritisnuti "enter" da bi podaci bili prihvaćeni. Računar je dao:
`p= 2.3184609691E+03 v= 2.0784609691E+03 nije kocka`

7) Datoteke i sortiranje

132. Neka je deklarisana promjenljiva
var data : file of integer ;
Napisati program koji će u datoteku data upisati sve proste brojeve manje od 1000. Zatim prikazati na ekranu sve proste brojeve između $a = 30$ i $b = 200$, putem čitanja iz data. Ilustracija:



Rješenje. Pomoću naredbe reset, npr. reset(d), otvara se već postojeća datoteka. Ona se otvara sa ciljem da se iz nje preuzimaju podaci. Preuzimanje (čitanje) sprovodi se redom: od početka datoteke, stalno naprijed, obično do njenog kraja. Sa get, npr. get(d), preuzima se jedan podatak u d^ (to znači: d^ ← podatak). Slično, poziv potprograma rewrite, npr. rewrite(d), služi za otvaranje nove datoteke d, koja je u tom trenutku prazna. Ona se otvara radi upisivanja elemenata u nju. Upisivanje se sprovodi redom: od početka datoteke i stalno naprijed. Za te svrhe upotrebljava se standardni potprogram put. Upravo, tekuća vrijednost d^ šalje se (upisuje se) u datoteku putem poziva put(d). Uzeli smo da je d ime datoteke. Tada se tekuća vrijednost označava sa d^ (kao što je već). Generalno, datoteke se drže na disku i služe za trajno čuvanje podataka.

Primjedba. Naglasimo da je naredba assign, recimo assign(data, 'mydata.dat') potrebna ako se koristi softverski paket Turbo Pascal čiji je proizvođač Borland i da predstavlja specifiku tog softvera. Ta naredba služi da se izjednači (da se poveže) ime datoteke u programu (znači

data) sa njenim imenom na disku (znači mydata.dat). Slično važi i za potprogram close, npr. close(data). Njegov efekat je da se datoteka zatvori.

Takođe, u slučaju Turbo Pascal, ne smije se pisati get(d) ; x := d^ ; već treba read(d, x) ; Isto tako, umjesto d^ := x ; put(d) ; treba pisati write(d, x) ;

Ukratko o algoritmu. Vidimo da je datoteka nazvana data. U prvoj fazi rada programa, ona se popunjava. Za $n = 2, 3, \ldots$ redom ispituje se da li je n prost. Ako jeste, onda se upisuje u datoteku. Tako se nastavlja sve do $n = 1000$. U drugoj fazi rada programa, vrši se čitanje podataka iz datoteke. Preuzima se jedan po jedan broj x. Prvih nekoliko brojeva se propuštaju, jer su manji od donje granice a. Zatim se na "output" šalju prosti brojevi x, takvi da je $a < x < b$. Brojevi veći od gornje granice b neće se štampati. Nastavlja se sa pregledanjem, sve dok se ne dođe do kraja datoteke (end of file). Promjenljiva m služi za formatiranje izlaza.

Ispitivanje primalnosti jednog broja izdvojeno je u poseban potprogram tipa "procedure". Prvi parametar n je ulaznog karaktera, a drugi izlaznog: B = true znači da je broj n prost, a B = false znači da je složen. Kako se odvija potprogram? Za jednocifrene brojeve n, automatski se daje odgovor, a slično i za ostale parne brojeve, dok se za ostale neparne n sprovodi ispitivanje. Da bi takav n bio prost, treba da nije djeljiv sa $k = 3$, niti sa većim neparnim k ($k^2 \leq n$). Zapaziti da u zaglavlju potprograma (u spisku parametara) ispred B: Boolean stoji var. To znači da se taj parametar prenosi, između potprograma i programa, po referenci ili svejedno rečeno po adresi (ako ne piše var, onda se prenosi po vrijednosti).

```
program prog ( output , data ) ;
var data: file of integer ;
a, b, n, x, m, min, max, uku: integer ; Boole: Boolean ;

procedure prost( n: integer; var B: Boolean );
var k: integer;  begin {potp}  if n in [ 2, 3, 5, 7 ] then B := true;
if n in [ 4, 6, 8, 9 ] then B := false;
if (n > 10) and (n mod 2 = 0) then B := false;
if (n > 10) and (n mod 2 <> 0 ) then
    Begin B := true; k := 3;
    while (B = true) and (k * k <= n) do
        begin if n mod k = 0 then B := false;
        k := k + 2 end End  end {potp} ;

begin  a := 30; b := 200; assign( data, 'mydata.dat' );
rewrite( data ); for n := 2 to 1000 do
    begin prost( n, Boole );
    if Boole = true then write( data, n ) end;
close( data ); reset( data ); read( data, x ); min := x;
uku := 1; writeln( ' min = ', min ); while x < a do
    begin read( data, x ); uku := uku + 1 end;
m := 0;  while x < b do
    begin write( x:4 ); m := m + 1;
    if m = 10 then writeln; if m = 10 then m := 0;
    read( data, x ); uku := uku + 1 end;
max := x;  while not eof( data ) do
    begin read( data, x ); uku := uku + 1; max := x end;
```

```
close( data ); writeln; write( 'max = ', max );
writeln( '  ukupno = ', uku ) end.
```

Izlaz:

```
min = 2
  31  37  41  43  47  53  59  61  67  71
  73  79  83  89  97 101 103 107 109 113
 127 131 137 139 149 151 157 163 167 173
 179 181 191 193 197 199
max = 997  ukupno = 168
```

Kada se rad programa završi i kada izađemo iz softvera onda (iz operativnog sistema) vidimo da je sada u direktorijumu prisutan novi fajl mydata.dat i da zauzima 336 bajta.

134. Neka su deklarisane promjenljive
var data : file of integer ; datc : file of char ;

Napisati program u kome se obrazuje nova datoteka datc, da sadrži sve proste brojeve do 1000. Popunjavanje datoteke datc vrši se preuzimanjem podataka iz već postojeće datoteke data od maloprije. Neka bude po 10 brojeva u jednom redu nove datoteke.

Objašnjenje: tekstualne datoteke su promjenljive tipa file of char ili svejedno rečeno tipa text. Znamo da se tekstualne datoteke mogu slobodno editovati, iz operativnog sistema.

Rješenje. U trenutku kada počinje rad programa, u direktorijumu je već odranije prisutna datoteka mydata.dat. Tokom rada programa, iz nje se čita jedan po jedan element i šalje se u datoteku datc tipa text, čije je formiranje u toku. Tako se nastavlja sve dok se ne stigne do kraja prve datoteke. Vidimo da je rad sa tekstualnim datotekama vrlo sličan radu sa standardnim datotekama input i output. To je istina zato što se radi o objektima istog tipa (tipa text). Takvi objekti imaju i dodatnu strukturu, podijeljeni su na redove (na linije).

```
program progra ( output , data , datc ) ;
var data: file of integer; datc: text; x, m: integer;
begin  writeln( '  pocetak rada programa' );
assign( data, 'mydata.dat' ); assign( datc, 'mydatc.dat' );
reset( data ); rewrite( datc ); m := 0;
repeat read( data, x );
    write( datc, x:6 ); m := m + 1;
    if m = 10 then writeln( datc );
    if m = 10 then m := 0
until eof( data );
close( data ); close( datc );
writeln( '  zavrsetak rada programa' )  end.
```

Izlaz:

```
pocetak rada programa
zavrsetak rada programa
```

Nakon svega, vidimo da je u direktorijumu prisutan novi fajl mydatc.dat i da zauzima 1040 bajta. Taj fajl može se jednostavno editovati. Za te svrhe služi recimo norton editor, notepad, wordpad ili crimson editor. Vidimo da novi fajl sadrži po deset brojeva u jednom redu i da pojedini broj zauzima šest mjesta (pozicioniran je desno).

8) Dodatni zadaci

137. Napisati program na Pascalu za tabeliranje vrijednosti funkcije $y = \sin x$, gdje u jednom redu treba prkazati po pet parova (x, y), argument x kreće se od 0 do 90 stepeni sa korakom po 1 stepen, a vrijednosti funkcije y neka budu na pet decimala.

Rješenje. Što se tiče ugrađene funkcije "sin" (sinus), njen argument izražava se u radijanima. U programu, štampanje je oblikovano tako da, kada vrijednost brojača "m" postane djeljiva sa pet, onda se prelazi na štampanje od početka novog reda. Sa "m:4" definisan je format prikazivanja: broj zauzima 4 mjesta. Slično, kada piše "y:8:5" onda to znači: prilikom štampanja navedene veličine potrošiće se 8 pozicija, od čega 5 za decimale.

```
program sinus( output );
var m: integer; pi, koef, x, y: real;
begin pi := 4.0 * arctan( 1.0 ); koef := pi / 180.0;
m := 0; x := m * koef;
    repeat y := sin( x );
    write( m:4, y:8:5 );
    m := m + 1; x := m * koef;
    if m mod 5 = 0 then writeln
    until m = 91;
write( '  x stepeni,  y = sin(x)' ) end.
```

Kao rezultat izvršavanja programa, imaćemo na ekranu prikazano ovako:

```
   0 0.00000   1 0.01745   2 0.03490   3 0.05234   4 0.06976
   5 0.08716   6 0.10453   7 0.12187   8 0.13917   9 0.15643
  10 0.17365  11 0.19081  12 0.20791  13 0.22495  14 0.24192
  15 0.25882  16 0.27564  17 0.29237  18 0.30902  19 0.32557
  ( ... ... ... )
  80 0.98481  81 0.98769  82 0.99027  83 0.99255  84 0.99452
  85 0.99619  86 0.99756  87 0.99863  88 0.99939  89 0.99985
  90 1.00000  x stepeni,  y = sin(x)
```

151. Za $n \geq 1$, Eulerova funkcija $\varphi = \varphi(n)$ definisana je sa: $\varphi(n) = \#\{k \mid 1 \leq k \leq n, \mathrm{NZD}(k, n) = 1\}$ (koliko ima uzajamno prostih), npr. $\varphi(12) = \#\{1, 5, 7, 11\} = 4$. Napisati program u kome se tabelira funkcija $\varphi = \varphi(n)$ za $1 \leq n \leq 100$.

Rješenje. Za dato $n \geq 1$, da bi se odredila vrijednost $\varphi(n)$, treba ispitati da li je broj k uzajamno prost sa n. Drugim riječima, za svako k, $1 \leq k \leq n$, treba ispitati da li ispunjava $\mathrm{NZD}(k, n) = 1$. Ukupan broj vrijednosti k koje ispunjavaju uslov, upravo i predstavlja $\varphi(n)$.

Računanje najvećeg zajedničkog djelioca dva broja $a \geq 1$ i $b \geq 1$ izdvojeno je u poseban potprogram. Za određivanje NZD koristi se Euklidov algoritam. Drugim riječima, rekurzivno se primjenjuje formula: $\mathrm{NZD}(a, b) = \mathrm{NZD}(b, a \bmod b)$. Isto tako, važi i formula $\mathrm{NZD}(a, a) = a$. Pored toga, primjenjuje se i $\mathrm{NZD}(a, 0) = a$, samo da je $a \neq 0$.

```
program Euler ;
var fi : array [ 1 .. 100 ] of integer ;
n, k, koliko, i : integer ;

function nzd(a, b: integer): integer;
begin if a = 0 then nzd := b;
```

```
if b = 0 then nzd := a;
if (a > 0) and (b > 0) then nzd := nzd ( b, a mod b ) end;

BEGIN  writeln( 'Eulerova funkcija' );
writeln( '(i, fi(i)), i=1,...,100' );
for n := 1 to 100 do
    Begin koliko := 0;
    for k := 1 to n do
        if nzd( k, n ) = 1 then koliko := koliko + 1;
    fi[ n ] := koliko End;
for i := 1 to 100 do
    Begin write( i:4 , fi[ i ]:3 );
    if ( i mod 10 ) = 0 then writeln End;
writeln( '------ nema vise ------' )  END .
```

Predviđeno je da program tabelira vrijednosti $1, \varphi(1), 2, \varphi(2), 3, \varphi(3), \ldots, 100, \varphi(100)$. Zato, u rezultatu rada programa, imamo na ekranu:

```
Eulerova funkcija
(i, fi(i)), i=1,...,100
   1  1   2  1   3  2   4  2   5  4   6  2   7  6   8  4   9  6  10  4
  11 10  12  4  13 12  14  6  15  8  16  8  17 16  18  6  19 18  20  8
  21 12  22 10  23 22  24  8  25 20  26 12  27 18  28 12  29 28  30  8
  31 30  32 16  33 20  34 16  35 24  36 12  37 36  38 18  39 24  40 16
  41 40  42 12  43 42  44 20  45 24  46 22  47 46  48 16  49 42  50 20
  51 32  52 24  53 52  54 18  55 40  56 24  57 36  58 28  59 58  60 16
  61 60  62 30  63 36  64 32  65 48  66 20  67 66  68 32  69 44  70 24
  71 70  72 24  73 72  74 36  75 40  76 36  77 60  78 24  79 78  80 32
  81 54  82 40  83 82  84 24  85 64  86 42  87 56  88 40  89 88  90 24
  91 72  92 44  93 60  94 46  95 72  96 32  97 96  98 42  99 60 100 40
------ nema vise ------
```

Urađeni su sljedeći primjeri:

Iz oblasti (1) Uvod u programiranje: 12. Eksponencijalna funkcija $y = e^x$ 13. Heronov obrazac (za računanje $\sqrt{a}$) Iz oblasti (2) Osnovni programi: 19. Kompleksan broj z u polarnom obliku 23. Primjer sa funkcijama div i mod Iz oblasti (3) Kontrolne strukture: 38. Niz Fibonaccijevih brojeva $\{f_n\}$ 53. Rastavljanje broja na proste činioce Iz oblasti (4) Sastavljene strukture podataka: 65. Računanje polinoma po Hornerovoj šemi 77. Goldbachova hipoteza Iz oblasti (5) Potprogrami: 103. Norma matrice A oblika $n \times n$ 104. Svojstvene vrijednosti matrice oblika 2×2 Iz oblasti (6) Rekurzija: 117. Generisanje binarnog drveta (rad sa pokazivačima) 118. Površina i zapremina pravilne prizme Iz oblasti (7) Datoteke i sortiranje: 132. Formiranje nove datoteke 134. Prepisivanje iz jedne datoteke u drugu datoteku tipa text Iz oblasti (8) Dodatni zadaci: 137. Tabeliranje funkcije $y = \sin x$ 151. Eulerova funkcija $\varphi = \varphi(n)$