Elektrotehnički fakultet Univerziteta u Beogradu

Programiranje 1
Materijal za vežbe na tabli i pripremu ispita
verzija: 2009-12-30

Pregled iz teorije izložen u ovim materijalima treba koristiti PRVENSTVENO kao podsetnik, a ne kao jedini izvor znanja

Osim u beleškama sa predavanja, detaljna objašnjenja se mogu pronaći u sledećim knjigama:

- Jozo Dujmović, *Programski jezici i metode programiranja*
- Kathleen Jensen, Niklaus Wirth, *Pascal priručnik*

Više zadataka se može pronaći u zbirci prof. Krausa:

- Laslo Kraus, *Zbirka zadataka iz Programskih jezika*

Navedene knjige se mogu pronaći u skriptarnici ili u biblioteci fakulteta.

Za svaki zadatak koji se može pronaći u zbirci prof. Krausa, ovde je naveden samo tekst zadatka, bez rešenja. Svaki od zadataka je numerisan brojem koji označava stranu na kojoj se dati zadatak nalazi u zbirci. Priložena su i rešenja (kompletna ili odgovarajući deo) onih zadataka čiji je tekst blago izmenjen u odnosu na zbirku.

Zadaci koji su tipa ispitnih pitanja su priloženi u celosti, uključujući i obrazloženje rešenja tamo gde je to potrebno.

U materijal je uključeno i nekoliko ispitnih zadataka, koji su rešeni u potpunosti i detaljno komentarisani.

Materijal se stalno unapređuje, nove verzije će redovno biti dodavane na Internet stranicu predmeta:

- `http://rti.etf.rs/rti/ir1p1`

Molimo da sugestije, primedbe ili uočene greške pošaljete putem elektronske pošte na adresu **`zorz@etf.rs`**.

Sadržaj

Spisak izmena:

Verzija 2009-12-30: ispravljene su uočene greške u rešenju zadatka IZ 2005-06-22 (Jun 2005) i dopunjen je komentar zadatka.

Verzija 2009-12-16: postavka zadatka Z116 je formulisana preciznije. Identifikator datoteke *Tekst* je promenjen u *datoteka*, radi bolje čitljivosti programa.

Verzija 2009-12-15: ispravka stilske greške u postavci zadatka Z114 na strani 63.

Verzija 2009-11-26: ispravljene formule za zbir i razliku komplemenata na stranama 15 i 17.

Verzija 2009-10-27: dodat Z29 (pC)

BULOVA ALGEBRA

Bulova algebra je ovde predstavljena kratkim izvodom iz teorije, kao i nekim zadacima sa vežbi.

Izvod iz teorije

Neka su nad skupom B={a, b, c, …} definisane:
unarna operacija "¯" (komplementiranje),
binarna operacija "**+**" (sabiranje), i
binarna operacija "**·**" (množenje),
i to tako da:

$(\forall a \in B)(\exists b \in B)\ \bar{a} = b$,

$(\forall a, b \in B)(\exists c \in B)\ a + b = c$,

$(\forall a, b \in B)(\exists c \in B)\ a \cdot b = c$.

Drugim rečima, operacije su zatvorene nad skupom B.
Struktura koju čini skup B sa istaknutim elementima 0 i 1 i operacijama "¯" (komplementiranje), "**+**" (sabiranje) i "**·**" (množenje), predstavlja Bulovu algebru ako operacije zadovoljavaju sledeće aksiome:

1. zakon asocijativnosti: $a + (b + c) = (a + b) + c, a \cdot (b \cdot c) = (a \cdot b) \cdot c$
2. zakon komutativnosti: $a + b = b + a, a \cdot b = b \cdot a$
3. zakon o neutralnom elementu: $a + 0 = 0 + a = a, a \cdot 1 = 1 \cdot a = a$
4. zakon o komplementu: $a + \bar{a} = \bar{a} + a = 1, a \cdot \bar{a} = \bar{a} \cdot a = 0$
5. zakon distributivnosti: $a \cdot (b + c) = a \cdot b + a \cdot c, a + (b \cdot c) = (a + b) \cdot (a + c)$

Dualni Bulov izraz: ako je $Q(a, b, c, \ldots)$ Bulov izraz, dualni Bulov izraz $Q^d(a, b, c, \ldots)$ se dobija iz izraza Q tako što se svaka operacija sabiranja zameni operacijom množenja, svaka operacija množenja zameni operacijom sabiranja, i svaka konstanta svojim komplementom.

Teoreme:

1. zakon o jedinstvenom komplementu: $a + x = 1, a \cdot x = 0 \Rightarrow x = \bar{a}$
2. $\bar{0} = 1, \bar{1} = 0$
3. zakon involucije operacije komplementiranja: $\bar{\bar{a}} = a$
4. $a + 1 = 1, a \cdot 0 = 0$
5. zakon idempotentnosti: $a + a = a, a \cdot a = a$
6. zakon apsorpcije: $a + a \cdot b = a, a \cdot (a + b) = a$
7. zakon sažimanja: $a \cdot b + a \cdot \bar{b} = a, (a + b) \cdot (a + \bar{b}) = a$
8. zakon nepotpunog sažimanja:

   $a \cdot b + a \cdot \bar{b} = a + a \cdot b + a \cdot \bar{b}, (a + b) \cdot (a + \bar{b}) = a \cdot (a + b) \cdot (a + \bar{b})$
9. zakon generalisanog sažimanja:

   $a \cdot c + b \cdot \bar{c} + a \cdot b = a \cdot c + b \cdot \bar{c}, (a + c) \cdot (b + \bar{c}) \cdot (a + b) = (a + c) \cdot (b + \bar{c})$
10. De Morganova teorema: $\overline{a + b} = \bar{a} \cdot \bar{b}, \overline{a \cdot b} = \bar{a} + \bar{b}$
11. generalizovana De Morganova teorema: $\overline{Q(a, b, c, \ldots)} = Q^d(\bar{a}, \bar{b}, \bar{c}, \ldots)$
12. princip dualnosti: $Q(a, b, c, \ldots) = R(x, y, z, \ldots) \Rightarrow Q^d(a, b, c, \ldots) = R^d(x, y, z, \ldots)$

Primeri Bulove algebre:

1. u skupovima: $(+ \rightarrow \cup), (\cdot \rightarrow \cap), (\bar{a} \rightarrow A^c), (0 \rightarrow \{\}), (1 \rightarrow U)$
2. u matematičkoj logici: $(+ \rightarrow \vee), (\cdot \rightarrow \wedge), (\bar{a} \rightarrow \neg a), (0 \rightarrow F), (1 \rightarrow T)$

3. prekidačka logika:Bulova algebra na skupu `B={0, 1}`

Zadatak Z1BA

Uprostiti algebarske izraze:

A) $a \cdot b + a \cdot b \cdot c + b \cdot c$

B) $a \cdot b + a \cdot \bar{b} \cdot c + b \cdot c$

C) $(a \cdot \bar{b} + c) \cdot (\bar{a} + b) \cdot \bar{c}$

Rešenje:

A) $a \cdot b + a \cdot b \cdot c + b \cdot c =$ - apsorpcija: $ab + abc = ab \cdot (1 + c) = ab \cdot 1 = ab$

$a \cdot b + b \cdot c =$ - distibutivnost "$\cdot$" prema "**+**"

$b \cdot (a + c)$

B) $a \cdot b + a \cdot \bar{b} \cdot c + b \cdot c =$ - distibutivnost "$\cdot$" prema "**+**"

$a \cdot (b + \bar{b} \cdot c) + b \cdot c =$ - distibutivnost "**+**" prema "$\cdot$"

$a \cdot (b + \bar{b}) \cdot (b + c) + b \cdot c =$ - $b + \bar{b}$ je 1

$a \cdot (b + c) + b \cdot c =$

$a \cdot b + a \cdot c + b \cdot c$

C) $(a \cdot \bar{b} + c) \cdot (\bar{a} + b) \cdot \bar{c} =$ - distibutivnost "$\cdot$" prema "**+**"

$(a \cdot \bar{b} \cdot \bar{c} + c \cdot \bar{c}) \cdot (\bar{a} + b) =$ - $c \cdot \bar{c}$ je 0

$a \cdot \bar{b} \cdot \bar{c} \cdot (\bar{a} + b) =$ - oba sabirka će imati u sebi proizvod koji je 0

$a \cdot \bar{b} \cdot \bar{c} \cdot \bar{a} + a \cdot \bar{b} \cdot \bar{c} \cdot b = 0$ - prvi proizvod je nula zbog $a \cdot \bar{a}$, a drugi zbog $\bar{b} \cdot b$

Zadatak Z2BA

Dokazati jednakost: $bc + abd + a\bar{c} = bc + a\bar{c}$

Napomena: pri dokazivanju jednakosti, mogu se koristiti sledeća dva pristupa – jedan pristup je da se primenom teorema od jedne strane jednakosti dobije druga, dok se drugi pristup sastoji u tome da se obe strane jednačine uz pomoć teorema transformišu dok ne postanu identične. Iako ne postoji formalno pravilo, prvi pristup je bolji ako je jedan od izraza jednostavan, a drugi ako su oba izraza komplikovana.

Rešenje:

$bc + abd + a\bar{c} = bc + abd(c + \bar{c}) + a\bar{c} = bc + abcd + ab\bar{c}d + a\bar{c} = bc + a\bar{c}$

U poslednjem koraku upotrebljena je apsorpcija za prva dva, odnosno druga dva sabirka:

$bc + abcd = bc(1 + ad) = bc$, odnosno $ab\bar{c}d + a\bar{c} = a\bar{c}(bd + 1) = a\bar{c}$.

Zadatak Z3BA

Ako je $a\bar{b} + \bar{a}b = c$, dokazati da je $a\bar{c} + \bar{a}c = b$.

Rešenje:

$a\bar{c} + \bar{a}c = b$ - zameni se c

$a\overline{(a\bar{b} + \bar{a}b)} + \bar{a}(a\bar{b} + \bar{a}b) = b$ - primenimo De Morganovu teoremu, dva puta

$a(\overline{a\bar{b}} \cdot \overline{\bar{a}b}) + \bar{a}a\bar{b} + \bar{a}\bar{a}b = b$ - za pretposlednji sabirak važi $\bar{a}a\bar{b} = 0 \cdot \bar{b} = 0$

$a(\bar{a} + b) \cdot (a + \bar{b}) + 0 + \bar{a}b = b$

$a(\bar{a}a + ba + \bar{a}\bar{b} + b\bar{b}) + \bar{a}b = b$

$a(0 + ab + \bar{a}\bar{b} + 0) + \bar{a}b = b$

$aab + a\bar{a}\bar{b} + \bar{a}b = b$

$ab + 0 + \bar{a}b = b$ - ovde se mogao direktno primeniti zakon sažimanja

$(a + \bar{a})b = b$

$b = b$

Zadaci za samostalnu vežbu

Koji je od ponuđenih izraza Bulove algebre ekvivalentan izrazu $bc + abd + a\bar{c}$?
(A) $bc + a\bar{c}$ B) $\bar{c}\bar{d} + bc + a\bar{c}$ C) $bc + ab$

Dovoljan uslov da vrednost sledećeg izraza $(\bar{a} + b) \cdot \bar{c} + \bar{d}a + e\bar{b}$ Bulove algebre bude 1 je:
(A) $c = 0$ i $d = 0$ B) $d = 0$ i $a = 0$ **(C)** $e = 1$ i $c = 0$

Koji je od ponuđenih izraza Bulove algebre ekvivalentan izrazu $(a \cdot b) + (b \cdot c) + (c \cdot a)$?
(A) $(a + b) \cdot (b + c) \cdot (c + a)$ B) $a \cdot b \cdot c + \bar{a} \cdot \bar{b} \cdot \bar{c}$ C) $a + b + c$

Dovoljan uslov da vrednost sledećeg izraza $(\bar{a} + c) \cdot \bar{b} + \bar{d}a + e\bar{c}$ Bulove algebre bude 1 je:
(A) $b = 0$ i $d = 0$ B) $d = 0$ i $a = 0$ **(C)** $b = 0$ i $e = 1$

Koji od ponuđenih odgovora je ekvivalentan datom izrazu $(\bar{a} + b) \cdot (b + c) + a \cdot \bar{b}$ Bulove algebre:
(A) $a + b + c$ B) $(a + c) \cdot (\bar{b} + a)$ C) $(\bar{a} \cdot b + b \cdot c)(a + \bar{b})$

PREDSTAVLJANJE BROJEVA

Sistemi za predstavljanje nenegativnih celih brojeva

U *cifarskoj reprezentaciji* nenegativni ceo broj se predstavlja kao uređena *n-torka* elemenata koji se nazivaju *cifre*; *n*-torka se naziva *vektor cifara* i zapisuje se kao:

$$X = (X_{n-1}, X_{n-2}, \dots, X_i, \dots, X_1, X_0) = X_{n-1}X_{n-2} \dots X_i \dots X_1X_0$$

Brojni sistem određuju sledeći elementi:

1. *Skup* (numeričkih) *vrednosti cifara*. Sa D_i obeležavamo skup vrednosti koje uzima X_i, a sa $|D_i|$ broj elemenata skupa D_i. U decimalnom brojnom sistemu važi $D_i = [0,1,2,3,4,5,6,7,8,9]$, $|D_i| = 10, \forall i : 0 \le i \le n-1$.
2. *Pravilo interpretacije.* Reč je o pravilu preslikavanja skupa vrednosti vektora cifara u skup brojeva.

Skup brojeva predstavljenih vektorom cifara sa *n* elemenata je *konačan skup* sa najviše:

$$K = \prod_{i=0}^{n-1} |D_i|$$

elemenata, pošto je ovo najveći broj različitih vrednosti vektora cifara.
(Primer: za $D_i = D, \quad \forall i : 0 \le i \le n-1 \Rightarrow K = |D|^n$)

Neredundantan brojni sistem: ako svaki vektor cifara predstavlja različit broj (preslikavanje 1:1).
Redundantan brojni sistem: ako više različitih vektora cifara predstavlja isti broj.

Najčešće se koriste *težinski sistemi* (sistemi sa *težinskim pozicionim kodom*). Za njih važi pravilo preslikavanja:

$$X = \sum_{i=0}^{n-1} X_i W_i$$

gde je W_i "težina" *i*-tog razreda. Vektoru cifara X odgovara vektor težina: $W = (W_{n-1}, \dots, W_1, W_0)$.
U brojnim sistemima sa osnovom (eng. *radix*) vektor težina se nalazi u sledećem odnosu sa vektorom osnova $R = (R_{n-1}, \dots, R_1, R_0)$:

$W_0 = 1, W_i = W_{i-1}R_{i-1}$, što se svodi na:

$$W_0 = 1, W_i = \prod_{j=0}^{i-1} R_j, \forall i : 1 \le i \le n-1$$

Klasifikacija brojnih sistema sa osnovom

1. Prema *vektoru osnova* brojni sistemi se dele na one sa fiksnom i one sa mešovitom osnovom.

1.1. *Fiksna osnova:* svi elementi vektora *osnova* imaju istu vrednost *r*. Vektor težina je tada $W = (r^{n-1}, r^{n-2}, \dots, r^2, r^1, 1)$, a pravilo preslikavanja:

$$X = \sum_{i=0}^{n-1} X_i r^i$$

1.2. *Mešovita osnova:* elementi vektora osnova su različiti. Primer za ovakav brojni sistem može biti predstavljanje vremena vektorom cifara (<sat>, <minut>, <sekunda>). Vrednosti za vektor osnova i vektor težina su $R = (24,60,60); W = (3600,60,1)$.

2. Prema *skupu vrednost cifara* brojni sistemi sa osnovom se dele u kanoničke i nekanoničke sisteme.

2.1. *Kanonički sistemi:* skup vrednosti $D_i = [0,1,\ldots,R_i - 1]$.

2.2. *Nekanonički sistemi: skup* vrednosti D_i nije kanonički (primer: rimski brojevi).

Kanonički skupovi daju neredundantne sisteme.
Sistem sa fiksnom pozitivnom osnovom r i kanoničkim skupom vrednosti cifara naziva se *konvencionalnim brojnim sistemom sa osnovom r.*
Najčešće se koriste:

$r = 10,\ D = [1,2,3,4,5,6,7,8,9]$	decimalni (eng. *decimal*, skraćeno *dec.*)
$r = 2,\ D = [0,1]$	binarni (eng. *binary*, skraćeno *bin.*)
$r = 8,\ D = [0,1,2,3,4,5,6,7]$	oktalni (eng. *octal*, skraćeno *oct.*)
$r = 16,\ D = [0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F]$	heksadecimalni (eng. *hexadecimal*, skraćeno *hex.*)

Za predstavljanje brojeva u računaru koristi se binarni brojni sistem, a oktalni i heksadecimalni su pogodni za "skraćeno" zapisivanje binarno predstavljenih brojeva.

$X = 10110010_{(2)} = B2_{(16)} = 262_{(8)} = 178_{(10)}$

Konverzija iz sistema sa osnovom `r` u decimalni sistem

$$C_{n-1}C_{n-2}\ldots C_1C_{0(r)} \rightarrow X_{(10)}$$

$$X_{(10)} = C_{n-1}r^{n-1} + C_{n-2}r^{n-2} + \ldots + C_i r^i + C_0 r^0 = \left(\ldots\left(C_{n-1}r + C_{n-2}\right)r + \ldots + C_1\right)r + C_0$$

Primeri:

$X = 10011_{(2)} = ((((1 \cdot 2) + 0) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1$
$= (((2 + 0) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1$
$= ((4 \cdot 2) + 1) \cdot 2 + 1$
$= 9 \cdot 2 + 1 = 19_{(10)}$

$X = DA03_{(16)}$
$= ((13 \cdot 16 + 10) \cdot 16 + 0) \cdot 16 + 3$
$= ((208 + 10) \cdot 16 + 0) \cdot 16 + 3$
$= 3488 \cdot 16 + 3 = 55811_{(10)}$

Konverzija iz decimalnog u sistem sa osnovom `r`

$$X_{(10)} \rightarrow C_{n-1}C_{n-2}\ldots C_1C_{0(r)}$$

$$X / r = X_1[C_0];\ X_1 / r = X_2[C_1];\ldots;\ X_{n-1} / r = 0[C_{n-1}]$$

Primer:

$X = 19 / 2 = 9$ [1
$= 9 / 2 = 4$ [1
$= 4 / 2 = 2$ [0 ↑ $X = 10011_{(2)}$
$= 2 / 2 = 1$ [0
$= 1 / 2 = 0$ [1

$X = 55811 / 16 = 3488$ [3 = 3
$= 3488 / 16 = 218$ [0 = 0
$= 218 / 16 = 13$ [10 = A ↑ $X = DA03_{(16)}$
$= 13 / 16 = 0$ [13 = D

Sistemi za predstavljanje negativnih celih brojeva

Sistem znaka i magnitude

Ceo broj se predstavlja pomoću para (x_s, x_m), gde je x_s *znak*, a x_m *magnituda*. Dve vrednosti znaka (+,-) se predstavljaju pomoću binarnih vrednosti $(0,1)$, respektivno.
Magnituda se može predstaviti bilo kojim sistemom za predstavljanje nenegativnih celih brojeva. Ako se koristi konvencionalni sistem sa osnovom r opseg celih brojeva je $0 \le |x| \le r^n - 1$, gde je n broj cifara za predstavljanje magnitude.
Nula ima dve reprezentacije: $x_s = 0,\ x_m = 0$ (pozitivna nula), $x_s = 1,\ x_m = 0$ (negativna nula).

Sistem komplementa

U sistemu komplementa ne pravi se razdvajanje između predstavljanja znaka i predstavljanja magnitude, već se kompletan ceo broj sa predznakom predstavlja preko nenegativnog celog broja A_c na sledeći način:

$$A_c = A \bmod L,$$

gde je L pozitivan ceo broj, nazvan *komplementaciona konstanta*. Komplementaciona konstanta treba da zadovoljava uslov $|A|_{max} < \frac{L}{2}$ da bi se dobila jednoznačna predstava, tj. da se područja pozitivnih i negativnih brojeva ne bi preklapala. Po definiciji "mod" (moduo, ostatak pri celobrojnom deljenju) funkcije, imajući u vidu navedeni uslov, gornji izraz je ekvivalentan sa:

$$A_c = \begin{cases} A, & A \ge 0 \\ L - |A|, & A \le 0 \end{cases} \quad \text{obrnuto:} \quad A = \begin{cases} A_c, & 0 \le A_c < \frac{L}{2} \\ A_c - L, & \frac{L}{2} < A_c \le L \end{cases}$$

(Matematička definicija "mod" funkcije ima uslov: $A < 0$; ovde je za negativno A dato $A \le 0$, zbog uvođenja "negativne nule".)
Preslikavanje se može predstaviti tabelarno na sledeći način:

A	0 1 2 … ⌈L/2⌉-1	-(⌈L/2⌉-1) … −2 −1 −0
A_c	0 1 2 … ⌈L/2⌉-1	(⌈L/2⌉+1) … L−2 L−1 L

Napomena: A_c se posmatra kao pozitivan broj bez znaka (znak je implicitno zadat u A_c).
Primer: Predstaviti sve cele brojeve koji zadovoljavaju uslov: $|A|_{max} < L/2$, ako je

a) $L = 10$
Rešenje:
$L/2 = 5 \Rightarrow |A|_{max} = 4$

A	0	1	2	3	4	-4	-3	-2	-1	-0
A_c	0	1	2	3	4	6	7	8	9	10

b) $L = 9$
Rešenje
$L/2 = 4.5 \Rightarrow |A|_{max} = 4$

A	0	1	2	3	4	-4	-3	-2	-1	-0
A_c	0	1	2	3	4	5	6	7	8	9

Vrednost A_c se može predstaviti u bilo kom sistemu za pozitivne cele brojeve. Ako se koristi

konvencionalni brojni sistem sa osnovom r i vektorom od n cifara, opseg mogućih vrednosti je: $0 \le A_c \le r^n - 1$.

Uobičajeni izbor za komplementacionu konstantu L je $L = r^n$ ili $L = r^n - 1$.

Komplement osnove

Izbor $L = r^n$ definiše *sistem komplementa opsega* (eng. *Range Complement – RC*), takođe poznat kao *sistem komplementa osnove*, a za r=2, *komplement dvojke* (eng. *two's complement*). U ovom sistemu A_c=L je izvan opsega i stoga postoji samo jedna reprezentacija A=0. Opseg celih brojeva sa predznakom je: $0 \le |A| \le \left(\frac{1}{2} r^n - 1\right)$ podrazumevajući da je osnova parna. Vrednost $A_c = \frac{1}{2} r^n$ se ponekad koristi za predstavljanje $A = -\frac{1}{2} r^n$, što daje za rezultat nesimetričnu reprezentaciju pozitivnih i negativnih vrednosti.

A	0 1 2 … $\frac{1}{2} r^n - 1$	$-\frac{1}{2} r^n$ … -2 -1
A_c	0 1 2 … $\frac{1}{2} r^n - 1$	$\frac{1}{2} r^n$ … r^n-2 r^n-1

Komplement najveće cifre

Izbor $L = r^n - 1$ definiše sistem *komplementa najveće cifre* (eng. *Digit complement – DC*), takođe poznat kao *sistem umanjenog komplementa osnove*, a za r=2, *komplement jedinice* (eng. *one's complement*). U ovom sistemu A_c=L se može prikazati sa *n* cifara, tako da ovde postoje dve reprezentacije A=0: A_c=0 i A_c=r^n-1. Opseg celih brojeva sa predznakom je: $0 \le |A| \le \frac{1}{2} r^n - 1$ podrazumevajući da je osnova parna.

A	0 1 2 … $\frac{1}{2} r^n - 1$	$-(\frac{1}{2} r^n - 1)$ … -1 0
A_c	0 1 2 … $\frac{1}{2} r^n - 1$	$\frac{1}{2} r^n$ … r^n-2 r^n-1

Primer 1: Predstaviti $-4 \le A \le 3$ u sistemima komplementa dvojke i jedinice.

Rešenje:

U simetričnom opsegu: $|A|_{max} = 3 < 4 = L/2 \Rightarrow L = 8$, $L = r^n \Rightarrow 8 = 2^n \Rightarrow n = 3$

	A	3	2	1	0	-0	-1	-2	-3	-4
Komplement dvojke	A_c	3	2	1	0	0	7	6	5	4
	X	011	010	001	000	000	111	110	101	100
Komplement jedinice	A_c	3	2	1	0	7	6	5	4	3
	X	011	010	001	000	111	110	101	100	-

Osobine sistema komplementa dvojke i jedinice (iz primera):

1. Reprezentacija 0:
   1.1. u sistemu komplementa dvojke – jedinstvena
   1.2. u sistemu komplementa jedinice – dve reprezentacije 0.
2. Opseg brojeva:
   2.1. u sistemu komplementa dvojke nije simetričan: $\lfloor -2^{n-1}, 2^{n-1} - 1 \rfloor$. Sistem nije zatvoren u odnosu na operaciju promene znaka.
   2.2. u sistemu komplementa jedinice je simetričan: $\lfloor -(2^{n-1} - 1), 2^{n-1} - 1 \rfloor$.

Primer 2: Predstaviti broj $A=-236_{(8)}$ u komplementu osnove.

Rešenje:

Prvi način:

r=8, pretpostavimo da je n=3 dovoljno veliko.

$L = r^n = 8^3 = 512_{(10)}$

$|A| = 236_{(8)} = (2 \cdot 8 + 3)8 + 6 = 158_{(10)}$

$|A| = 158 < \frac{L}{2} = 256$, što znači da je n=3 dovoljno veliko.

$A_c = L - |A| = 512 - 158 = 354_{(10)}$

$354 / 8 = 44 \ [2$

$44 / 8 = 5 \ [4 \uparrow \quad A_c = 542_{(8)}$

$5 / 8 = 0 \ [5$

Drugi način:

$L=512_{(10)}=1000_{(8)}$

$|A| = 236_{(8)}$

$A_c = 1000_{(8)} - 236_{(8)} = 542_{(8)}$

Primer 3: Predstaviti broj $A = -8A2_{(16)}$ u komplementu cifre:

Rešenje:

Prvi način:

r=16, pretpostavimo $n = 3 \Rightarrow L = r^n - 1 = 16^3 - 1 = 4095_{(10)}$

$|A| = 8A2_{(16)} = (8 \cdot 16 + 10)16 + 2 = 2210_{(10)}$

$|A| = 2210 > \frac{L}{2} = \frac{4095}{2}, \Rightarrow n = 3$, pošto nije dovoljno veliko, uzimamo

$n = 4 \Rightarrow L = 16^4 - 1 = 65535$

$A_c = L - |A| = 65535 - 2210 = 63325$

$X = 63325 / 16 = 3957 \ [13 = D$

$= 3957 / 16 = 247 \ [5 = 5$

$= 247 / 16 = 15 \ [7 = 7 \uparrow A_c = F75D_{(16)}$

$= 15 / 16 = 0 \ [15 = F$

Drugi način:

$L = 16^4 - 1 = 65535_{(10)} = FFFF_{(16)}$

$L: \ FFFF_{(16)}$

$-|A|: \ -8A2_{(16)}$

$A_c: \ F75D_{(16)}$

Promena znaka

Ako se sa F označi broj (-A), tada važi da je $F_c = (-A)_c = L - A_c$, gde je L komplementaciona konstanta.

Dokaz:

$$(-A)_c = \begin{cases} -A, & -A \geq 0 \\ L - |-A|, & -A \leq 0 \end{cases} \Rightarrow \quad (-A)_c = \begin{cases} |A|, & A \leq 0 \\ L - A, & A \geq 0 \end{cases}$$

$$L - A_c = \begin{cases} L - A, & A \geq 0 \\ L - L + |A|, & A \leq 0 \end{cases} \Rightarrow \quad L - A_c = \begin{cases} L - A, & A \geq 0 \\ |A|, & A \leq 0 \end{cases}$$

$$(-A)_c = L - A_c$$

U *komplementu jedinice:* $(-A)_c = (2^n - 1) - A_c = 11\ldots11 - A_c = \overline{A}_c$ (bit po bit)

U *komplementu dvojke:* $(-A)_c = 2^n - A_c = (2^n - 1) - A_c + 1 = \overline{A}_c + 1$

Primer:

a) n=4, komplement jedinice

$A_c = 1011 = 11 \geq 8 \Rightarrow A = 11 - (2^4 - 1) = -4$

$\overline{A}_c = 0100 = 4 \Rightarrow (-A) = 4 \Rightarrow A = -4$

b) n=4, komplement dvojke

$A_c = 1011 = 11 \geq 8 \Rightarrow A_c = 11 - 16 = -5$

$\overline{A}_c = 0100$

$+1 = 0001$

$(-A_c) = 0101 = 5 \Rightarrow -A = 5 \Rightarrow A = -5$

Drugi, jednostavniji, način za određivanje drugog komplementa podrazumeva da se binarna reprezentacija broja deli na dva dela. Desni deo čine, čitajući sa desna na levo, sve nule i prva jedinica na koju se naiđe (u slučaju da je krajnje desna cifra jedinica, onda se desni deo sastoji samo od te jedinice). Taj deo se samo prepisuje u rezultat. Levi deo čine preostale cifre, i taj deo se bit po bit komplementira:

primer za n=4:

$A_c = 1011$

101 | 1

010 | 1

0101

primer za n=8:

$A_c = 10100100$

10100 | 100

01011 | 100

01011100

Sabiranje u sistemu komplementa dvojke ($L=2^n$)

Zbir brojeva A i B: F=A+B.

Prilikom sabiranja dva broja može doći do prekoračenja opsega (overflow: *V*), što daje pogrešan rezultat. Prekoračenje se dogodilo kada su *A i B negativni* (A_{n-1}=1, B_{n-1}=1), a zbir je *nenegativan* broj (F_{n-1}=0) ili kada su *A i B nenegativni* (A_{n-1}=0, B_{n-1}=0), a zbir *F negativan* broj (F_{n-1}=1).

$$V = \overline{A}_{n-1}\overline{B}_{n-1}F_{n-1} + A_{n-1}B_{n-1}\overline{F}_{n-1}$$

Prenos u i-ti razred obeležavamo sa C_i.

Ako su A_c i B_c komplementi dvojke brojeva *A* i *B*, a F_c komplement *F*, može se pokazati da važi

$$(A_c + B_c) \bmod (2^n) = F_c$$

Drugim rečima: sabirajući po modulu 2^n brojeve predstavljene u komplementu 2, dobijamo njihov zbir predstavljen u komplementu 2. Razmotrićemo posebno četiri moguća slučaja:

1. $A \geq B,\ B \geq 0 \Rightarrow A_{n-1} = 0,\ B_{n-1} = 0$

   Pošto postoji n binarnih razreda:

   ako je $A + B < 2^{n-1} \Rightarrow F_{n-1} = 0,\ V = 0$;

   ako je $A + B \geq 2^{n-1} \Rightarrow F_{n-1} = 1,\ V = 1$ (greška zbog prekoračenja).

$$(A_c + B_c) \bmod (2^n) = [A \geq 0, B \geq 0] = (A + B) \bmod (2^n) = (F) \bmod (2^n) = F_c$$

Primeri: za n=4 ($L=2^4=16$)

A=3, B=2
A_c: 0011 = 3
B_c: 0010 = 2
F_c: 0101 = 5
C_{i+1}: 0010

A=5, B=6
A_c: 0101 = 5
B_c: 0110 = 6
F_c: 1011 = $11 \geq L/2 = 8 \Rightarrow F = -5$
C_{i+1}: 0100, **prekoračenje!!!**

2. $A \geq B,\ B < 0 \Rightarrow A_{n-1} = 0,\ B_{n-1} = 1$
   $|A| \geq |B| : F_{n-1} = 0,\ V = 0$
   $|A| < |B| : F_{n-1} = 1,\ V = 0$

$$(A_c + B_c) \bmod (2^n) = (A + 2^n - |B|) \bmod (2^n) = (2^n + (A - |B|)) \bmod (2^n) =$$
$$= (2^n + (A + B)) \bmod (2^n)$$
$$= (A + B) \bmod (2^n)$$

pošto je $(x + 2^n) \bmod (2^n) = (x) \bmod (2^n)$, važi

Primeri: za n=4 ($L=2^4=16$)

a) *A*= 6, *B*= -3
A_c: 0110 = 6
B_c: 1101 = 13
F_c: 0011 = 3
C_{i+1}: 1100

b) *A*=3, *B*=-6
A_c: 0011 = 3
B_c: 1010 = 10
F_c: 1101 = $13 \geq L/2 \Rightarrow F = -3$
C_{i+1}: 0010

3. $A < B,\ B \geq 0 \Rightarrow A_{n-1} = 1,\ B_{n-1} = 0$
   $|A| > |B| : F_{n-1} = 1,\ V = 0$
   $|A| \leq |B| : F_{n-1} = 0,\ V = 0$

$$(A_c + B_c) \bmod (2^n) = (2^n - |A|) \bmod (2^n) = (2^n + (B - |A|)) \bmod (2^n)$$
$$= (2^n + (B + A)) \bmod (2^n) = (A + B) \bmod (2^n) = F_c$$

Primeri: za n=4 (L=2^4=16)

a) A= -7, B= 5
A_c: 1001 = 9
B_c: 0101 = 5
F_c: 1110 = 14 $\geq L/2 \Rightarrow F = -2$
C_{i+1}: 0001

b) A=-2, B=3
A_c: 1110 = 14
B_c: 0011 = 3
F_c: 1110 = 1
C_{i+1}: 1110

4. $A < B,\ B < 0 \Rightarrow A_{n-1} = 1,\ B_{n-1} = 1$

Pošto postoji n binarnih razreda:

ako je $|A + B| \leq 2^{n-1} \Rightarrow F_{n-1} = 1,\ V = 0$;
ako je $|A + B| > 2^{n-1} \Rightarrow F_{n-1} = 0,\ V = 1$, (greška usled prekoračenja)

$$(A_c + B_c) \bmod(2^n) = (2^n - |A| + 2^n - |B|) \bmod(2^n)$$
$$= (2^n + 2^n - (|A| + |B|)) \bmod(2^n) = (2^n + 2^n + (A + B)) \bmod(2^n)$$
$$= (A_c + B_c) \bmod(2^n) = F_c$$

Primeri: za n=4 (L=2^4=16)

A=-1, B=-3
A_c: 1111 = 15
B_c: 1101 = 13
F_c: 1100 = $12 \geq L/2 \Rightarrow F = -4$
C_{i+1}: 1111

A=-4, B=-7
A_c: 1100 = 12
B_c: 1001 = 9
F_c: 0101 = 5
C_{i+1}:1000, **prekoračenje!!!**

Ako se sa C_{n-1} označi prenos u poslednji (n-1) razred (razdred znaka), a sa C_n prenos iz poslednjeg razreda može se zaključiti da se prekoračenje može računati kao: $V = C_n \oplus C_{n-1}$ (ekskluzivno ili).

Oduzimanje u sistemu komplementa dvojke ($L=2^n$)

Razlika brojeva A i B: F=A-B.

Prilikom oduzimanja dva broja može doći do prekoračenja opsega (overflow: *V*), što daje pogrešan rezultat. Prekoračenje se dogodilo (*V*=1) kada je *A negativan* (A_{n-1}=1), *B nenegativan* (B_{n-1}=0), a razlika *F nenegativan* broj (F_{n-1}=0), ili kada je *A nenegativan* (A_{n-1}=0), *B negativan* broj (B_{n-1}=1), a razlika *F negativan* broj (F_{n-1}=1).

$$V = \bar{A}_{n-1}B_{n-1}F_{n-1} + A_{n-1}\bar{B}_{n-1}\bar{F}_{n-1}$$

Pozajmicu iz *i*-tog razreda obeležavamo sa C_i.

Ako su A_c i B_c komplementi dvojke brojeva *A* i *B*, a F_c komplement *F*, može se pokazati da važi $(A_c - B_c) \bmod (2^n) = F_c$.

Dokaz:

Kako iz: $(-B)_c = L - B_c = 2^n - B_c \Rightarrow -B_c = (-B)_c - 2^n$, dobija se:

$(A_c - B_c) \bmod (2^n) = (A_c + (-B) - 2^n) \bmod (2^n)$ pošto je $(x + 2^n) \bmod (2^n) = (x) \bmod (2^n)$, važi

$= (A_c + (-B)_c) \bmod (2^n)$ zbog $(A_c + X_c) \bmod (2^n) = (A + X) \bmod (2^n)$, važi

$= (A + (-B)_c) \bmod (2^n) = (A - B) \bmod (2^n) = (F) \bmod (2^n) = F_c$

Diskusija:

1. $A \geq B$, $B \geq 0 \Rightarrow A_{n-1} = 0$, $B_{n-1} = 0$

1.1. $|A| \geq |B|$: $F_{n-1} = 0$, $V = 0$

1.2. $|A| < |B|$: $F_{n-1} = 1$, $V = 0$

Primeri: za n=4 ($L=2^4=16$)

a) *A*= 3, *B*=2

A_c: 0011 = 3

B_c: 0010 = 2

F_c: 0001 = 1

C_{i+1}: 0000

b) *A*=5, *B*=6

A_c: 0101 = 5

B_c: 0110 = 6

F_c: 1111 = $15 \geq L/2 \Rightarrow F = -1$

C_{i+1}: 1110

2. $A \geq B$, $B < 0 \Rightarrow A_{n-1} = 0$, $B_{n-1} = 1$

Pošto postoji n binarnih razreda:

2.1. ako je $A - B \leq 2^{n-1} \Rightarrow F_{n-1} = 0$, $V = 0$;

2.2. ako je $A - B \geq 2^{n-1} \Rightarrow F_{n-1} = 1$, $V = 1$, (greška usled prekoračenja)

Primeri: za n=4 ($L=2^4=16$)

A=3, *B*=-2

A_c: 0011 = 3

B_c: 1110 = -2

F_c: 0101 = 5

C_{i+1}: 1100

A=5, *B*=-6

A_c: 0101 = 5

B_c: 1010 = 6

F_c: 1011 = $11 \geq L/2 = 8 \Rightarrow F = -5$

C_{i+1}: 1010 **prekoračenje!!!**

3. $A < B,\ B \geq 0 \Rightarrow A_{n-1} = 1,\ B_{n-1} = 0$

Pošto postoji n binarnih razreda:

3.1. ako je $|A - B| \leq 2^{n-1} \Rightarrow F_{n-1} = 1,\ V = 0$;

3.2. ako je $|A - B| > 2^{n-1} \Rightarrow F_{n-1} = 0,\ V = 1$, (greška usled prekoračenja)

Primeri: za n=4 (L=2^4=16)

A=-4, B=3	A=-8, B=2
A_c: 1100 = 12	A_c: 1000 = 8
B_c: 0011 = 3	B_c: 0010 = 2
F_c: 1001 = 9 $\geq L/2 = 8 \Rightarrow F = -7$	F_c: 0010 = 6
C_{i+1}: 0011	C_{i+1}: 0110 **prekoračenje!!!**

4. $A < B,\ B < 0 \Rightarrow A_{n-1} = 1,\ B_{n-1} = 1$

4.1. $|A| > |B|$: $F_{n-1} = 1,\ V = 0$

4.2. $|A| \leq |B|$: $F_{n-1} = 0,\ V = 0$

Primeri: za n=4 (L=2^4=16)

A=-7, B=-4	A=-1, B=-3
A_c: 1001 = 9	A_c: 1111 = 15
B_c: 1100 = 12	B_c: 1101 = 13
F_c: 1101 = 13 $\geq L/2 \Rightarrow F = -3$	F_c: 0010 = 2
C_{i+1}: 1100	C_{i+1}: 0000

Ako se sa C_{n-1} označi pozajmica iz poslednjeg (n-1) razreda (razreda znaka), a sa C_n pozajmica iz nepostojećeg (n-tog) razreda može se zaključiti da se prekoračenje može računati kao: $V = C_n \oplus C_{n-1}$.

Zadatak B1

Neka se u osmobitnim lokacijama *a* i *b* nalaze brojevi *A* i *B*. Odrediti broj *F* koji sadrži lokacija *f* nakon izvršene operacije oduzimanja *F*=*A*-*B* i vrednost indikatora u sledećim slučajevima:

a) A=+1011101, B=+0110011;
b) A=+1011101, B=-0110011;
c) A=-1011101, B=+0100010;
d) A=-1011101, B=+1010001;
e) A=-1011101, B=-0100010.

Brojevi se u računaru predstavljaju u komplementu dvojke, na širini od 8 bita.

Rešenje:

n=8, L=2^n=2^8=256, L/2=128

a) A_c: 01011101 = 93 A=93
B_c: 00110011 = 51 B=51
F_c: 00101010 = 42
C_{i+1}: 00101010 = 42
$F_c = 42 < 128 \Rightarrow F = 42$;
$V = 0 \oplus 0 = 0$

b) A_c: 01011101 = 93 A=93
B_c:11001101 = 205 A=-51
F_c: 10010000 = 144
C_{i+1}: 10000000
$F_c = 144 \geq 128 \Rightarrow F = 144 - 256 = -112$;
$V = 1 \oplus 0 = 1$

Na isti način se rešavaju problemi pod c), d) i e).

Zadaci sa vežbi na tabli

Zadatak IZ1

Sadržaj 10-bitne memorijske lokacije je $2A6_{16}$. Ako je na posmatranoj memorijskoj lokaciji smešten ceo broj (INTEGER) predstavljen u punom komplementu, kolika je decimalna vrednost posmatranog celog broja?

A) 678
B) 346
C) –346

Rešenje:

$x = 2A6 < 0$ (jer je bit najvećeg značaja = 1)

$\overline{x} = 159$

$+1$

$\overline{x} = 15A_{16} = (1 * 16 + 5) * 16 + 10 = 346_{10} \Rightarrow x = -346_{10}$

Odgovor: C

Zadatak IZ2

U računaru, koji ima memorijsku reč širine 14b, izvršena je operacija: Y:=MAXINT+X. Ako je pre operacije sadržaj memorijske lokacije X: $2A6C_{16}$, kolika je decimalna vrednost rezultata Y nakon izvršene operacije?

A) 2667
B) 19051
C) –1428

Rešenje:

MAXINT	$= {}_1$FFF
+ X	$= {}_2$A6C
Y	$= {}_0$A6B

$> 0 \Rightarrow Y = (10*16+6)*16+11 = 2667_{10}$

(vodeće cifre su namerno prikazane umanjene da bi se naglasilo da predstavljaju samo dva bita, a ne četiri kao ostale cifre u posmatranim brojevima)

Odgovor: A

Zadatak IZ (Oktobar 2007)

Posmatra se računar koji radi sa 9-bitnim brojevima predstavljenim u komplementu dvojke. Koja se **vrednost** dobije kada se na ovom računaru izračuna izraz (A-B)-(C-D)? **Vrednosti** operanada A i B su 217_{10} i $-F1_{16}$, **predstave** operanada C i D su 115_{16} i 011000101_2.

A) -122_{10}
(B) -86_{16}
C) 101111010_2

Rešenje:
Zadatak ilustruje razliku između **vrednosti** operanda i **predstave** operanda u memoriji računara. Vrednost operanda se dobija **interpretacijom** predstave operanda, primenom zadatih pravila. U ovom slučaju, pravila su određena sistemom *komplement dvojke*.

Zadatak je moguće rešiti na više načina. Ovde će biti prikazana dva pristupa.

1. Računanje na osnovu vrednosti operanada

Za operande A i B su date njihove vrednosti. Pri tome, operand A je dat u decimalnom brojnom sistemu, a operand B u heksadecimalnom. Radi jednostavnijeg računa, najpre treba odrediti vrednost operanda B u decimalnom brojnom sistemu: $B = -(15 * 16^1 + 1 * 16^0) = -241$.

Za operande C i D su date njihove predstave. Pri tome, predstava operanda C je data u heksadecimalnom brojnom sistemu, a predstava operanda D je data u binarnom brojnom sistemu. Slično prethodnom slučaju, da bi se jednostavnije odredile vrednosti ovih operanada, treba ih prikazati u binarnom brojnom sistemu: tako se direktno vidi vrednost svakog bita broja, čime se smanjuje mogućnost greške prilikom interpretacije. Reprezentacija C: 100010101_2. Očigledno, broj u lokaciji C je negativan. Da bi se odredila njegova vrednost, najpre ga treba pretvoriti u pozitivan broj. Reprezentacija –C : 011101011_2, odnosno C = -235. Određivanje vrednosti operanda C je moglo i jednostavnije da se obavi: negativan broj se interpretira kao ceo broj bez znaka i od njega se oduzme 2^n, gde je n u ovom slučaju 9, dakle oduzme se 512. Interpretacijom reprezentacije operanda C kao celog broja bez znaka dobija se vrednost 256+16+4+1 = 277. Prema tome, 277-512 = -235, što je tražena vrednost. Slično se dobija D = 128+64+4+1 = 197.

Sada treba razmatrati parcijalne sume A-B i C-D zato što računar rezultat svake od njih mora da smesti u memoriju pre nego što obavi sledeću operaciju. Prilikom svake od operacija može doći do prekoračenja, pa je zato preporučljivo da se najpre odrede vrednosti **minINT** i **maxINT** za ovaj računar. Za 9-bitni računar, minINT = -256, maxINT = 255. A-B = 217-(-241) = 458. Očigledno **dolazi do prekoračenja** (rezultat je veći od maxINT). Prekoračenje označava da je rezultat operacije netačan. Ipak, rezultat te operacije (iako netačan) će biti upisan u memoriju računara. Vrednost rezultata je 458-512 = -54. Slično tome, C-D = -235-197 = -432. I u ovom slučaju dolazi do prekoračenja a rezultat je -432+512 = 80. Konačno, vrednost celog izraza je -54-80 = -134. Očigledno, odgovor pod A) je netačan. Takođe, odgovor pod C) je netačan jer je u pitanju pozitivan broj, a vrednost ovog izraza je negativan broj. Odgovor pod B) je tačan: $8*16^1 + 6 * 16^0 = 128+6 = 132$.

NAPOMENA: u zadatku se tražila **VREDNOST** rezultata. Da se tražila **predstava** rezultata u memoriji, onda bi rešenje pod C) bilo tačno. Takođe treba primetiti da je prekoračenje kod aritmetičkih operacija sasvim normalna pojava i da rezultat, iako netačan, postoji.

2. Računanje na osnovu binarne reprezentacije operanada

Binarna reprezentacija operanda A je 011011001_2, a operanda B je 100001111_2. S obzirom na to da je operand B negativan i da se traži razlika A-B, jednostavnije je da se B pretvori u pozitivan broj i da se onda odredi zbir.

A	0	1	1	0	1	1	0	0	1
-B	0	1	1	1	1	0	0	0	1
+	1	1	1	0	0	1	0	1	0

Na sličan način se određuje predstava razlike C-D:

C	1	0	0	0	1	0	1	0	1
D	0	1	1	0	0	0	1	0	1
-	0	0	1	0	1	0	0	0	0

Konačno:

A-B	1	1	1	0	0	1	0	1	0
C-D	0	0	1	0	1	0	0	0	0
-	1	0	1	1	1	1	0	1	0

Na kraju ostaje da se odredi **VREDNOST** rezultata čija je **PREDSTAVA** na datom računaru 101111010_2. Očigledno, u pitanju je negativan broj, a njegova *apsolutna* vrednost je $10000110_2 = 86_{16} = 132_{10}$. Prema tome, tačan rezultat je B) pošto je u pitanju negativan broj.

Diskusija

Ovaj zadatak pokazuje da je veoma bitno praviti razliku između **VREDNOSTI** i **PREDSTAVE** broja. Osim toga, naročito ako se zadatak rešava na 2. način (od načina ovde prikazanih), ne treba prebrzo donositi zaključak da je C) tačan odgovor samo zato što je ponuđen isti niz bita.

Zadaci za samostalnu vežbu (sa ranijih ispita)

U nekom računaru celi brojevi se predstavljaju u 10-bitnim lokacijama u drugom (potpunom) komplementu. U kojem od navedenih slučajeva će prilikom sabiranja celih brojeva I i J doći do prekoračenja?

A) Sadržaj lokacije u koju je smešteno I je $3FA_{(16)}$, a lokacije u koju je smešteno J je $1FF_{(16)}$
B) Vrednost broja I je -202, a vrednost broja J je -310 u dekadnom brojnom sistemu.
(C) Vrednost broja I je 110, a vrednost broja J je 423 u dekadnom brojnom sistemu.

Na računaru koji ima memorijsku reč širine 14 bita izvrši se operacija: Y:=minint-X. Ako je pre operacije sadržaj memorijske lokacije X, koja sadrži 14-bitni ceo broj, jednak $2A6C_{16}$, kolika je decimalna vrednost celobrojnog rezultata Y nakon izvršene operacije?

(A) -2668 B) 2668 C) 2667

Celi brojevi A, B i C prikazani su u drugom komplementu na širini od 8 bita. Neka je vrednost broja $A=105_{10}$, a binarna predstava broja B u memoriji se može zapisati kao 75_{16}. Ako je potrebno izračunati izraz A:=A-B+C, koji je uslov neophodan i dovoljan da bi se obezbedilo da pri računanju ne dođe do prekoračenja? Napomena: Prvo se vrši oduzimanje, pa sabiranje.

A) C>0 **(B)** $C \geq -116_{(10)}$ C) $C \leq 104_{(10)}$

Ukoliko je sadržaj lokacije u koju je smešten najveći ceo broj MAXINT prikazan u drugom komplementu na nekom računaru $7FFF_{(16)}$, kako onda na tom računaru izgleda prikaz broja koji se dobija kao zbir MININT i broja čiji je prikaz $03F0_{(16)}$?

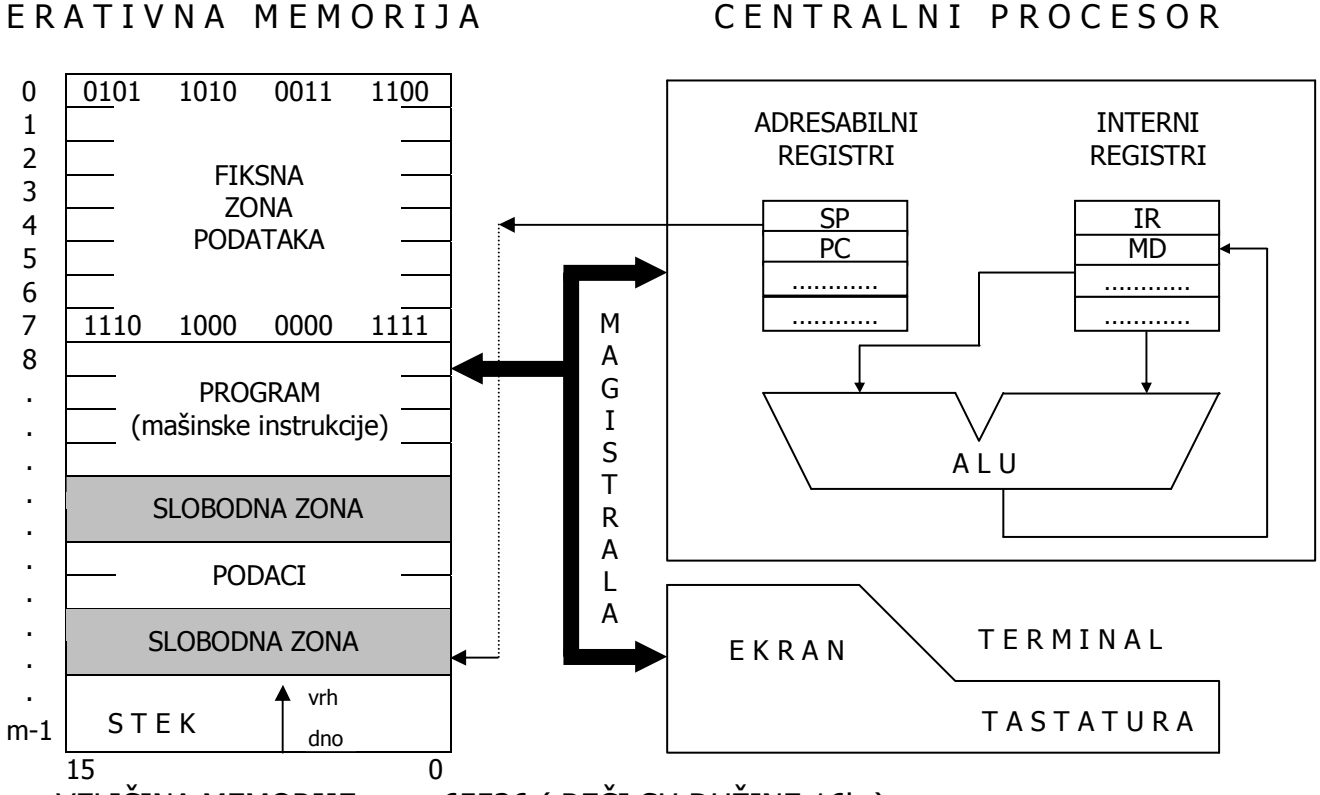
A) $1000\ 0011\ 1111\ 0001_{(2)}$ B) $101\ 740_{(8)}$ C) $43F0_{(16)}$

Dva broja prikazana su u drugom komplementu na dužini od 8 bita. Vrednost broja A iznosi -99, a binarni sadržaj lokacije u kojoj se nalazi drugi broj B je 11001011. Kolika je vrednost razlike A-B?

A) 104 B) -47 **(C)** -46

PICOCOMPUTER

Referentni priručnik za picoComputer



SLIKA 1 SASTAVNI DELOVI I ORGANIZACIJA pC-a

Za komuniciranje sa spoljnim svetom (ulaz podataka i prikaz rezultata), pC koristi terminal, koji predstavlja dva uređaja – tastaturu i ekran u istom kućištu. Unos podataka u memoriju pC-a je posredstvom tastature, a prikaz podataka iz memorije je na ekranu. Sva tri sastavna dela pC-a (memorija, procesor i terminal), razmenjuju podatke preko odgovarajuće bidirekcione magistrale.

Struktura mašinskih instrukcija, koje izvršava pC, je:

kôd operacije				i1	a1			i2	a2			i3	a3		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Kôd operacije (bitovi 12–15) je četvorobitan, što znači da je moguće direktno definisati najviše 16 različitih kodova operacije, a potrebno je obezbediti instrukcije bar za sledeća četiri osnovna tipa:

1. instrukcije prenosa podataka
2. aritmetičke instrukcije
3. kontrolne instrukcije
4. ulazno–izlazne instrukcije

Tabela 1.1: Osnovne mašinske instrukcije koje izvršava pC

NAZIV INSTRUKCIJE	BINARNI I SIMBOLIČKI KÔD OPERACIJE		OPIS AKTIVNOSTI, KOJU REALIZUJE INSTRUKCIJA
POMERANjE	0000	MOV	i3=0, a3=0 ⇒ A1:=A2 i3=1, a3=0 ⇒ A1:=val(val(PC)) i3=0, a3>0 ⇒ A1[j]:=A2[j], j=0,1,…,val(A3)–1 i3=1, a3=7 ⇒ A1[j]:=A2[j], j=0,1,…,val(val(PC))–1
SABIRANjE	0001 1001	ADD ADD	A1:=A2+A3 (svi argumenti su promenljive) A1:=A2+val(val(PC)), ili A1:=val(val(PC))+A3
ODUZIMANjE	0010 1010	SUB SUB	A1:=A2–A3 (svi argumenti su promenljive) A1:=A2–val(val(PC)), ili A1:=val(val(PC))–A3
MNOŽENjE	0011 1011	MUL MUL	A1:=A2·A3 (svi argumenti su promenljive) A1:=A2·val(val(PC)), ili A1:=val(val(PC))·A3
DELjENjE	0100 1100	DIV DIV	A1:=⌊A2/A3⌋ (svi argumenti su promenljive) A1:=⌊A2/val(val(PC))⌋, ili A1:=⌊val(val(PC))/A3⌋ Za kodove > 8 ⇒ mora biti adr(A2)>0 ∧ adr(A3)>0
USLOVNI SKOK, AKO JE PRVI ARGUMENT JEDNAK DRUGOM	0101	BEQ	Uslovni skokovi realizuju operaciju: ako je Uslov_Ispunjen ⇒ PC:=Adresa_Skoka. Moguće je koristiti tri vrste uslova: (1) A1=A2, ili A1>A2 (kodirano: a1>0 ∧ a2>0) (2) A1=0, ili A1>0 (kodirano: a1>0 ∧ i2=a2=0) (3) 0=A2, ili 0>A2 (kodirano: i1=a1=0 ∧ a2>0) Uslovi: adr(A1)>0 ∧ adr(A2)>0 Dozvoljene su dve adrese skoka: (1) i3=0 ∧ 0≤a3≤7 ⇒ Adresa_Skoka=val(a3) (2) i3=1 ∧ a3=0 ⇒ Adresa_Skoka=val(val(PC))
USLOVNI SKOK, AKO JE PRVI ARGUMENT VEĆI OD DRUGOG	0110	BGT	
ULAZ PODATAKA	0111	IN	Prenos n celobrojnih vrednosti, posredstvom tastature, na lokacije A1[j], J=0,……,n-1. Ako je i2=0 ⇒ bitovi 0–6 desne polureči sadrže n, pa je n≤127. Ako je i2=1 ∧ a2=0 ⇒ n=A3. Uvek mora biti n>0.
IZLAZ PODATAKA	1000	OUT	Prenos n celobrojnih veličina, sa lokacija A1[j], j=0,…,n–1 na ekran terminala. Interpretacija n je ista, kao u instrukciji IN.
SKOK U POTPROGRAM	1101	JSR	stek:=PC+1; PC:=val(val(PC)). Početna vrednost PC–a nakon skoka je adresa druge reči instrukcije.
POVRATNI SKOK IZ POTPROGRAMA	1110	RTS	PC:=stek (programski brojač se puni vrednošću povratne adrese uzete sa steka)
KRAJ RADA	1111	STOP	Prestanak inkrementiranja PC-a. Ako je sadržaj adresnih polja različit od 0000, onda će, na ekranu terminala, pre završetka rada, biti prikazane krajnje vrednosti odgovarajućih argumenata (ovako nije moguće prikazati krajnji sadržaj lokacije 0).

OBJAŠNjENjE: A1[j] je oznaka lokacije, čija je adresa adr[A1]+j, j≥0. A1[0] je isto što i A1.

Tabela 1.2: Pregled strukture mašinskih instrukcija za pC

KÔD OPERACIJE	MAŠINSKA INSTRUKCIJA	OPIS AKTIVNOSTI, KOJU INSTRUKCIJA REALIZUJE
MOV 0000	0000 xxxx yyyy 0000	X:=Y
	0000 xxxx ???? 1000 cccc cccc cccc cccc	X:=C
	0000 xxxx yyyy 0nnn	X[j]:=Y[j], j=0,.....,N–1
	0000 xxxx yyyy 1111 cccc cccc cccc cccc	X[j]:=Y[j], j=0,.....,C–1
ADD b001 SUB b010 MUL b011 DIV b100	0kkk xxxx yyyy zzzz	X : = Y ⊕ Z, ⊕ ∈ { +, –, *, / }
	1kkk xxxx yyyy 0000 cccc cccc cccc cccc	X : = Y ⊕ C (yyyy>0)
	1kkk xxxx 0000 zzzz cccc cccc cccc cccc	X : = C ⊕ Z (zzzz>0)
BEQ 0101 BGT 0110	kkkk xxxx yyyy 1000 cccc cccc cccc cccc	if X ® Y then PC:=ccccccccccccccc (xxxx>0, yyyy>0)
	kkkk xxxx 0000 0nnn	if X ® Y then PC:=val(nnn) (xxxx>0)
	0101 xxxx xxxx 1000 cccc cccc cccc cccc	PC:=ccccccccccccccc (xxxx>0) (GOTO C)
IN 0111 OUT 1000	kkkk xxxx 1000 nnnn	ULAZ/IZLAZ za X[j], j=0,.....,N–1
	kkkk xxxx 0ccc cccc	ULAZ/IZLAZ za X[j], j=0,.....,C–1
JSR 1101	1101 ???? ???? ???? cccc cccc cccc cccc	Stek:=PC+1, PC:=ccccccccccccccc
RTS 1110	1110 ???? ???? ????	PC:=stek
STOP 1111	1111 0000 0000 0000	Kraj rada
	1111 xxxx yyyy zzzz	Prikaz X, Y i kraj rada (xxxx>0, yyyy>0)

Tabela 1.3: Sintaksa simboličkog mašinskog jezika pC

Sintaksa SMJ za pC je ovde priložena u notaciji sa zagradama.

Specifikacija simbola: simbol = konstanta [; komentar]

Specifikacija početka programa: ORG konstanta [; komentar]

Sintaksa izvršnih instrukcija:

$$[simbol:]\ \ MOV\ \ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\},\ \left\{\begin{matrix} \left\{\begin{matrix} \text{konstanta} \\ \#\,simbol \\ simbol \\ (simbol) \end{matrix}\right\} \\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\},\ \left\{\begin{matrix} \text{konstanta} \\ \#\,simbol \\ simbol \end{matrix}\right\} \end{matrix}\right\}\ [;\ komentar]$$

$$[simbol:]\ \left\{\begin{matrix} ADD \\ SUB \\ MUL \\ DIV \end{matrix}\right\}\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\},\ \left\{\begin{matrix} \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\},\ \left\{\begin{matrix} \text{konstanta} \\ \#\,simbol \\ simbol \\ (simbol) \end{matrix}\right\} \\ \left\{\begin{matrix} \text{konstanta} \\ \#\,simbol \end{matrix}\right\},\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\} \end{matrix}\right\}\ [;\ komentar]$$

$$[simbol:]\ \left\{\begin{matrix} BEQ \\ BGT \end{matrix}\right\}\ \left\{\begin{matrix} \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\},\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\} \\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\},\ 0 \\ 0,\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\} \end{matrix}\right\},\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\ [;\ komentar]$$

$$[simbol:]\ \left\{\begin{matrix} IN \\ OUT \end{matrix}\right\}\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\ \left[\ ,\ \left\{\begin{matrix} \text{konstanta} \\ \#\,simbol \\ simbol \\ (simbol) \end{matrix}\right\}\ \right]\ [;\ komentar]$$

$$[simbol:]\ \ JSR\ \ simbol\ \ [;\ komentar]$$

$$[simbol:]\ \ RTS\ \ [;\ komentar]$$

$$[simbol:]\ \ STOP\ \ \left[\left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\left[\ ,\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\left[\ ,\ \left\{\begin{matrix} simbol \\ (simbol) \end{matrix}\right\}\right]\right]\right]\ [;\ komentar]$$

Uputstvo za korišćenje programa PCAS

Početak rada: treba pokrenuti PCAS:EXE
PCAS je prevodilac za simbolički mašinski jezik pC-a i simulator pC-a. Princip rada: unošenje zahteva je omogućeno izborom jedne od ponuđenih mogućnosti, ili unošenjem traženog podatka – princip MENU-a.

Raspoložive operacije (glavni MENU):

A – Učitavanje programa za pC (simbolički mašinski jezik). PCA je standardni tip za datoteke sa izvornim tekstom (ime_prog.PCA). Datoteke kreirati EDITOR-om teksta.

M – Učitavanje programa za pC (mašinski jezik). HEX je standardni tip za datoteke sa mašinskim Programom (ime_prog.HEX). Ove datoteke nastaju posle prevođenja izvornog programa, ali mogu biti kreirane i EDITOR-om teksta. Format datoteke je:

aaaa	Početna adresa programa (hex)
iiii	Sukcesivne mašinske reči programa (hex)
…	
iiii	

S – Ispisivanje teksta izvornog programa, koji je u radnoj memoriji

D – Prikazivanje mašinskog programa, koji je u radnoj memoriji

C – Prevođenje simboličkog mašinskog programa, koji je u radnoj memoriji, formiranje datoteke ime_prog.HEX i ubacivanje mašinskog programa u radnu memoriju nakon zahteva

R – Izvršavanje mašinskog programa, koji je u radnoj memoriji

T – Naizmenično uključivanje/isključivanje trasiranja rada programa. Ako je uključeno trasiranje pre izvršavanja svake mašinske naredbe, biće prikazan trenutni sadržaj brojača naredbi, biće prikazani sadržaji memorijskih lokacija, čije su adrese 0–7 i biće prikazan kôd te naredbe.

L – Naizmenično uključivanje/isključivanje zapisivanja podataka o radu sistema PCAS u datoteci ime_prog.LOG (dnevnik-datoteka; izveštaj-datoteka). Ako u radnoj memoriji ne postoji program u trenutku uključivanja zapisivanja ⇒ PCAS.LOG je naziv datoteke, pa je omogućeno beleženje rezultata rada korisnikovog programa zbog kasnijeg lakšeg štampanja.

H – Traženje pomoći o načinu korišćenja PCASA-a i o sintaksi simboličkog mašinskog jezika za pC

Q – Kraj rada
Uobičajeni redosled:

$ EDIT ime_prog.PCA	PRIPREMA IZVORNOG PROGRAMA
$ PCAS	STARTOVANjE PCAS-A
A	UBACIVANjE IZVORNOG PROGRAMA
L	POČETAK FORMIRANjA ime_prog.LOG
C	PREVOĐENjE IZVORNOG PROGRAMA
R	IZVRŠAVANjE PREVEDENOG PROGRAMA
Q	KRAJ RADA (KRAJ KORIŠĆENjA PCAS-A)
$ TYPE ime_prog.LOG	BRZI PREGLED DOBIJENIH REZULTATA

Zadatak Z22

Sastaviti program na simboličkom mašinskom jeziku računara pC za izračunavanje i ispisivanje vrednosti izraza $c = a^2 + a \cdot b + b^2$, gde su a i b celi brojevi koji se učitavaju sa tastature.

```
A=1         IN A,2
B=2         MUL C,A,A
C=3         MUL T,A,B
T=4         ADD C,C,T
ORG 8       MUL T,B,B
            ADD C,C,T
            STOP C
```

Zadatak Z23s (skraćena varijanta zadatka sa strane 23)

Sastaviti program na mašinskom jeziku pC koji učitava 2 cela broja sa tastature i ispisuje ih po nerastućem redosledu.

Rešenje:
Osnovni algoritam:

a) Učitaj a i b
b) Ako je a<b zameni mesta
c) Ispiši a i b

Opis promenljive ili radnje	Adr	Sadržaj lokacije operativne memorije							Hex
P	0								
A	1								
B	2								
	. . .								
1. Učitaj A i B	8	IN	0	#A	0			2	7102
2. Ako je A>B, skočimo na (7)	9	BGT	0	#A	0	#B	1	0	6128
	10	adr(naredba 7)							0010
3. Ako je A=B, skočimo na (7)	11	BEQ	0	#A	0	#B	1	0	5128
	12	adr(naredba 7)							0010
4. P ← A	13	MOV	0	#P	0	#A	0	0	0010
5. A ← B	14	MOV	0	#A	0	#B	0	0	0120
6. B ← P	15	MOV	0	#B	0	#P	0	0	0200
7. Ispiši A i B i završi	16	STOP	0	#A	0	#B	0		F120

Jednostavnije rešenje:

a) Učitaj a i b
b) Ako je a>b ispiši a i b i zaustavi program
c) U suprotnom ispiši b i a i zaustavi program

```
A=1             IN A,2
B=2             BGT A,B,ISPIS1
ORG 8           STOP A,B
ISPIS1: STOP B,A
```

Napomena: rešenje jeste jednostavnije ali je u praksi primenljivo samo u situaciji kada se obrađuju dva broja (porede po vrednosti). Takođe, dobra praksa je da se izbegava višestruka pojava instrukcije za prekid programa (STOP) jer to program generalno čini manje preglednim i težim za razumevanje.

Zadatak Z23

Sastaviti program na mašinskom jeziku računara pC koji učitava 3 cela broja sa tastature i ispisuje ih po nerastućem redosledu.

Rešenje:
Osnovni algoritam:

a) Učitaj A, B, C
b) Ako je A < B zameni im mesta
c) Ako je A < C zameni im mesta
d) Ako je B < C zameni im mesta
e) Ispiši A, B, C i završi

Napomena: savetuje se studentima da za ovaj zadatak sami pokušaju da napišu program na simboličkom mašinskom jeziku korišćenjem rešenja iz prethodnog zadatka.

Zadatak Z24

Sastaviti program na mašinskom jeziku računara pC koji učitava n celih brojeva sa tastature, a zatim izračunava i ispisuje celobrojni deo aritmetičke sredine tih brojeva.

Rešenje:
Osnovni algoritam:

a) Učitaj n
b) Učitaj n celih brojeva
c) Izračunaj sumu učitanih brojeva
d) Srednja vrednost je suma podeljena sa n
e) Ispiši rezultat i završi

```
N=1             IN N                    CIKLUS: ADD SUMA,SUMA,(NIZ)
SUMA=2          SUB SUMA,SUMA,SUMA              SUB I,I,1
NIZ=3           MOV NIZ,#ADRNIZ                 ADD NIZ,NIZ,1
I=4             IN (NIZ),N                      BGT I,0,CIKLUS
ADRNIZ=100      MOV I,N                         DIV SUMA,SUMA,N
ORG 8                                           STOP SUMA
```

Zadatak IZ6

Koji od sledećih programa za pC ispisuje vrednost 1?

A)
```
A=1
ORG 8
MOV (A), #A
STOP A
```

B)
```
A=1
ORG 8
MOV A, #A
STOP A
```

C)
```
A=1
ORG 8
OUT A
STOP
```

Obrazloženje:
Program pod A) upisuje 1 na lokaciju, čija je adresa slučajna, jer sadržaj lokacije A nije definisan ⇒ rezultat ispisivanja sadržaja lokacije A je slučajna vrednost.
Program pod B) upisuje 1 na lokaciju A ⇒ rezultat ispisivanja sadržaja lokacije A je 1.
Program pod C) ispisuje slučajnu vrednost (sadržaj lokacije A), koja nije definisana.

Odgovor: B

Zadatak Z25

Sastaviti program na simboličkom mašinskom jeziku računara pC za izračunavanje zbira prvih n prirodnih brojeva i zbira kvadrata prvih n prirodnih brojeva.

Rešenje:

$$S_1 = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \qquad S_2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Zadatak Z26

Sastaviti program na simboličkom mašinskom jeziku računara pC kojim se na osnovu dva data niza brojeva a[i] i b[i] formira novi niz c[i], tako da važi c[i] = a[i] + b[i] (i = 0, 1, 2, ..., n – 1).

Zadatak IZ7

Koje vrednosti ispisuje priloženi program za pC?

```
X=1
Y=2
Z=3
ORG 8
MOV X, #Y
ADD Y, X, #X
MOV (Y), #Y
STOP X, Y, Z
```

A) 1 2 3
B) 2 3 3
C) 2 3 2

Obrazloženje:

```
X=1              ; promenljiva X je na lokaciji, čija je adresa 1
Y=2              ; promenljiva Y je na lokaciji, čija je adresa 2
Z=3              ; promenljiva Z je na lokaciji, čija je adresa 3
ORG 8            ; program će biti smešten od adrese 8 u memoriji pC
MOV X, #Y        ; X := adr ( Y ) = 2
ADD Y, X, #X     ; Y := 2 + adr ( X ) = 2 + 1 = 3
MOV (Y), #Y      ; Z := adr ( Y ) = 2
STOP X, Y, Z     ; biće ispisane vrednosti: 2  3  2
```

Odgovor: C

Zadatak Z27

Sastaviti program na simboličkom mašinskom jeziku računara pC kojim se iz datog niza celih brojeva izostavljaju svi elementi čije su vrednosti parne.

Zadatak Z28

Sastaviti potprogram na simboličkom mašinskom jeziku računara pC za izračunavanje n!.

Zadatak Z29 (dodat u verziji 2009-10-27)

Šta ispisuje sledeći program na SMJ za pC, ako se unesu vrednosti 10 i 12?

```
AdrA=100
A=1
V=2
T=3
R=4
ORG 8
IN V
JSR B
OUT V
IN V
JSR B
STOP V
```

```
B: MOV A, #AdrA
C: DIV T, V, 2
   MUL T, T, 2
   BEQ V, T, D
   MOV (A), 1
   BEQ V, V, E
D: MOV (A), 0
E: ADD A, A, 1
   DIV V, V, 2
   BGT V, 0, C
```

```
   MOV R, #AdrA
   MOV V, (R)
F: ADD R, R, 1
   BEQ A, R, KRAJ
   MUL V, V, 2
   ADD V, V, (R)
   BEQ V, V, F
KRAJ: RTS
```

A) 14 16 (**B**) 5 3 C) 13 7

Zadatak IZ24

U memoriji picoComputera nalazi se lista celih brojeva predstavljena na sledeći način: ako se na lokaciji sa adresom A nalazi ceo broj, na lokaciji sa adresom A+1 se nalazi adresa sledećeg celog broja u listi. Adresa 0 označava kraj liste. Šta označava potprogram PP?

```
U=1
S=2
ORG=8
```

```
PP:     MOV S,0
PETLJA: ADD S,S,(U)
        ADD U,U,1
        BEQ (U),0,KRAJ
        MOV U,(U)
        BEQ U,U,PETLJA
KRAJ:   RTS
```

A) Broj elemenata u listi na koju ukazuje U.
(**B**) Zbir elemenata u listi na koju ukazuje U.
C) Zbir elemenata uvećanih za jedan u listi na koju ukazuje U.

Zadatak IZ 2006-11-30 (Kolokvijum 2006)

Sledeći program na simboličkom mašinskom jeziku za picoComputer, za zadate ulazne vrednosti 3 i 4 respektivno, ispisuje:

```
M=0
N=1
Q=2
X=3
Y=4
Z=5
ORG 8
```

```
MOV M, #Q
IN Q, M
SUB (X), X, #X
SUB (Y), (Q), (X)
DIV Z, (N), (X)
OUT M, X
STOP
```

A) 3 4 1 4
B) 3 4 -1 4
(**C**) 2 3 3 4

Zadatak IZ 2006-11-30 (Kolokvijum 2006)

Ako su ispravno deklarisani svi simboli, a u lokaciji A se nalazi adresa prvog elementa niza celih brojeva čija se dužina nalazi u lokaciji N, sledeći potprogram PP:

```
PP:  MOV I,0
     SUB S,S,S
L0:  BGT 0,(A),L1
     ADD S,(A),S
     BEQ S,S,L2
L1:  SUB S,S,(A)
L2:  ADD A,A,1
     ADD I,I,1
     BGT N,I,L0
     RTS
```

A) u lokaciju S smešta sumu apsolutnih vrednosti elemenata niza
B) u lokaciju S smešta sumu pozitivnih vrednosti elemenata niza
(**C**) u lokaciju S smešta sumu pozitivnih umanjenu za sumu negativnih vrednosti elemenata niza

Zadatak IZ 2006-11-30 (Kolokvijum 2006)

Čime treba zameniti #### u sledećem programu za picoComputer da bi on za učitanu pozitivnu vrednost N ispisivao sve cele brojeve veće od 1 sa kojima je uneti broj deljiv?

```
N=1                    |      ####
I=2                    |      OUT I
L=3                    |L1:   ADD I,I,1
T=4                    |      BEQ I,I,L0
ORG 8                  |L2:   STOP
   IN N                |
   DIV L,N,2           |
   MOV I,2             |
```

A)
```
L0:  BGT I,L,L2
     DIV T,N,I
     MUL T,T,I
     BGT N,T,L0
```

(**B**)
```
L0:  BGT I,L,L2
     DIV T,N,I
     MUL T,T,I
     BGT N,T,L1
```

C)
```
L0:  BGT I,L,L2
     DIV T,N,I
     MUL N,T,I
     BGT N,T,L1
```

SINTAKSNE NOTACIJE

- sintaksa i semantika programskih jezika: sintaksa se bavi formalnom ispravnošću nekog iskaza na objektnom jeziku bez ulaženja u logičku ispravnost.

primer: `a+b*c` - ispravno napisano
`a+c*` - neispravno napisano

- metajezik je niz pravila za opis sintakse objektnog jezika, tzv. gramatika objektnog jezika.

Elementi notacija

Svaka notacija ima svoje elemente kojima se opisuje objektni jezik. Ovde su navedeni elementi četiri notacije, iako jednu od njih, i to **notaciju sa zagradama, nije potrebno spremati za ispit.**

BNF notacija

1. neterminalni simboli	`<x>`
2. terminalni simboli	`slovo`
3. metajezički operator dodele	`::=`
4. metajezički operator nadovezivanja	`<x>slovo`
5. metajezički operator izbora	`<x> ::= <y>|<z>` isto sto i $\begin{matrix} <x> ::= <y> \\ <x> ::= <z> \end{matrix}$
6. ponavljanje prethodnog elementa	$\{x\}_{i}^{j}$ j ←podrazumeva se ∞, i ←podrazumeva se 1

Primer:

```
<niz cifara> ::= <cifra> | <niz cifara> <cifra>
<cifra> ::= 0|1|2|3|4|5|6|7|8|9
```

primer za 12: **<niz cifara> → <niz cifara> <cifra> → <cifra> <cifra> → 12**

- rekurzivna definicija ostvaruje opis beskonačnog broja nizova cifara
- objektni jezik je skup nizova terminalnih znakova, odnosno nizova cifara {0,1,...,9,00,01,...,123}

EBNF notacija

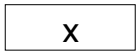
1. neterminalni simboli	x
2. terminalni simboli	`"slovo"`
3. metajezički operator dodele	=
4. metajezički operator nadovezivanja	x"slovo"
5. metajezički operator izbora	`x = (y|z)` isto sto i $\begin{matrix} x = y \\ x = z \end{matrix}$ (zagrade preciziraju operande operatora izbora)
6. ponavljanje prethodnog elementa	$\{x\}_{i}^{j}$ j ←podrazumeva se ∞, i ←podrazumeva se 0
7. opcija	[x]
8. na kraju pravila stoji tačka	.

Notacija sa zagradama

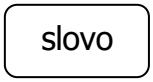
1. neterminalni simboli	x
2. terminalni simboli	<u>slovo</u>
3. metajezički operator dodele	=
4. metajezički operator nadovezivanja	x <u>slovo</u>
5. metajezički operator izbora	$\left\{\begin{matrix} x \\ y \end{matrix}\right\}$
6. ponavljanje prethodnog elementa	x...
7. opcija	[x]

Sintaksni dijagrami

1. neterminalni simboli

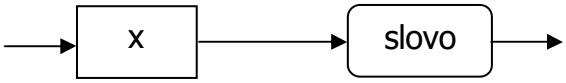


2. terminalni simboli

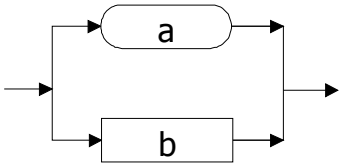


3. dodela

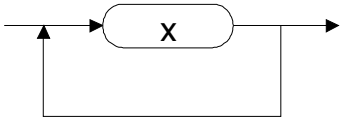
4. nadovezivanje



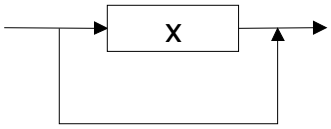
5. izbor



6. ponavljanje



7. opcija



Zadatak IZ32A (integralni ispit, 14.03.2002. godine) *

U nekom jeziku celobrojne konstante se mogu pisati u heksadekadnom ili binarnom brojnom sistemu. Ako se pišu u heksadekadnom brojnom sistemu, moraju počinjati cifrom 0-9 i moraju se završiti sufiksom H. Ako se pišu u binarnom brojnom sistemu, ne smeju počinjati nulom i moraju se završiti sufiksom B. Koju sintaksnu definiciju treba dodati datim definicijama da bi se dobila ispravna sintaksna definicija konstante u ovom jeziku?

```
<cons>::=<bcon>B|<hcon>H
<bcon>::=1|<bcon><bc>
<bc>::=0|1
<dc>::=<bc>|2|3|4|5|6|7|8|9
<hc>::=A|B|C|D|E|F
```

A) `<hcon>::=<hc>|<hcon><dc>`

B) `<hcon>::=<dc>|<hcon><hc>|<dc><hcon>`

C) `<hcon>::=<dc>|<hcon><hc>|<hcon><dc>`

Obrazloženje:

Interesantna stvar u ovom zadatku je način na koji je definisana dekadna cifra. Umesto prostog navođenja svih mogućnosti, autor zadatka se odlučio na korišćenje već postojeće definicije binarne cifre. Binarne konstante su ispravno definisane. Ostaje da se vidi koje od ponuđenih rešenja zadovoljava uslov da heksadekadne konstante moraju počinjati dekadnom cifrom.

Odgovor pod A) ne zadovoljava pomenuti uslov zadatka, zato što dozvoljava da konstanta počne heksadekadnom cifrom. Odgovor pod B) ne pokriva bilo koji slučaj kada u heksadekadnom broju postoji dekadna cifra posle heksadekadne, primer: $A1_{16}$, $B12C_{16}$... Odgovor pod C) je ispravan jer pokriva sve slučajeve i zadovoljava uslov zadatka vezan za heksadecimalne konstante.

Zadatak IZ8

U nekom jeziku sintaksa izraza se definiše u BNF notaciji na sledeći način:

```
<i>::= <a> | <i>, <i><o>
<a>::= A|B|C|D|E
<O>::= +|-|*|/
```

Koji od sledećih izraza sintaksno odgovara datoj definiciji?

A) A, B+, C+, D, E-/
B) A+B, C/, D/
C) A, A, A*+, B, C-/

Rešenje:

A) A , B + , C + , D , E - /
<a>, <a><o>, <a><o>, <a>, <a><o><o>
<i>, <i><o>, <a><o>, <i>, <i><o><o>
<i> , <i><o>, <i> <o>
<i> , <i> <o>
<i>

B) A + B , C / , D /
<a><o><a>, <a><o>, <a><o>
<i><o><i>, <i><o>, <i.<o>
???

C) A , A , A * + , B , C - /
<a>, <a>, <a><o><o>, <a>, <a><o><o>
<i>, <i>, <i><o><o>, <i>, <i><o><o>
<i>, <i> <o>, <i> <o>
<i> , <i> <o>
<i>

Komentar: gornja definicija izraza odgovara zapisu u postifiksnoj notaciji – izrazi pod A) i C)
Odgovor: V (A i C)

Zadatak IZ9

Koja od ponuđenih definicija u EBNF notaciji pravilno dopunjuje sledeću definiciju apsolutne vrednosti normalizovane mantise realnog broja u decimalnom brojnom sistemu?

```
Mantisa = "0." Dec.
NnCifra = ("1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9").
```

A) `Dec = (NnCifra| Dec "0" | Dec NnCifra).`
B) `Dec = (NnCifra| Dec NnCifra).`
C) `Dec = (NnCifra| ("0" | NnCifra) Dec).`

Rešenje:
Normalizovana mantisa realnog broja u decimalnom sistemu ima oblik: 0.XY... ,
pri čemu je X obavezna cifra različita od 0, a Y... proizvoljan broj proizvoljnih decimalnih cifara.

A) dozvoljava da Dec bude jedna nenulta cifra ili proizvoljan niz cifara koji počinje nenultom cifrom a završava 0, ili proizvoljan niz cifara koji počinje nenultom cifrom, a završava takođe nenultom cifrom, što je sve ispravno.
B) omogućava samo niz nenultih cifara, što nije ispravno.
C) omogućava da prva cifra bude 0 što nije ispravno, a ne omogućava da poslednja cifra bude 0, što takođe nije ispravno.

Odgovor: A

Zadatak IZ 2006-11-31 (Kolokvijum 2006)

Data je sintaksna definicija u BNF notaciji: <a>::= 0 | 010<a>1 | <a>1<a>
Koji od sledećih nizova odgovara datoj sintaksnoj definiciji?

A) 1001001 B) 0100001 (**C**) 0101001

Zadatak IZ10

Koja od navedenih sintaksnih definicija ispravno definiše skup malih slova (na primer: ['a','c','s'..'w']) pod pretpostavkom da ne treba da se definiše malo slovo?

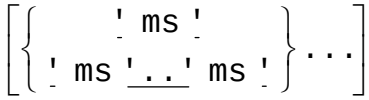
A) u BNF notaciji

```
<skup_ms> ::= [<niz>]
<niz> ::= <elem>|<elem>,<niz>
<elem> :: '<ms>'|'<ms>'..'<ms>'
```

B) u EBNF notaciji

```
Skup_MS = "[" Niz "]".
Niz = [ Elem { "," Elem }].
Elem = ( "'" MS "'" | "'" MS "'..'" MS "'").
```

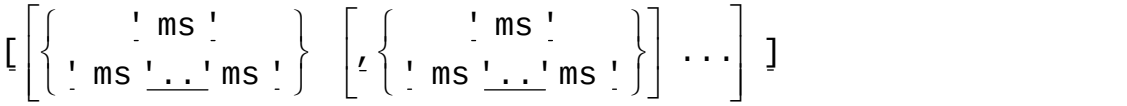
C) U notaciji sa zagradama:



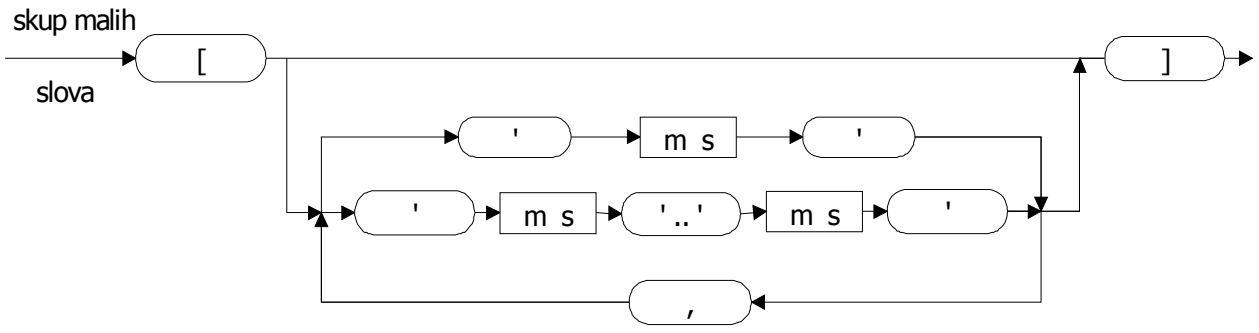
Rešenje:
Odgovor pod A) je neispravan, jer ne omogućava definisanje praznog skupa, jer lista navedena između srednjih zagrada mora da sadrži bar jedan element. Trebalo bi:

```
<skup ms> ::= [<niz>]|[]
<niz> ::= <elem>|<elem>,<niz>
<elem> :: '<ms>'|'<ms>'..'<ms>'
```

Odgovor pod B) je ispravan, jer po EBNF notaciji ono što je između zagrada [ ] može da se izostavi, ili pojavi jednom, pa je moguće definisati i prazan i neprazan skup. Neprazan skup ima samo jedan element ili više elemenata razdvojenih zarezom, što je omogućeno korišćenjem zagrada { }, koje znače da se ono što je između njih navedeno može pojaviti 0, 1 ili više puta. Element može biti slovo ili opseg slova. Apostrof se pojavljuje kao terminalni simbol pa se navodi pod navodnicima, po pravilima EBNF notacije.
Odgovor pod C) je neispravan, jer se nigde srednje zagrade ne pojavljuju kao terminalni simboli. Takođe se nigde ne pojavljuje zarez kojim se razdvajaju elementi skupa. Trebalo bi:



Skup možemo definisati i korišćenjem sintaksnih dijagrama:



Odgovor:B

Zadatak Z72 (modifikovan)

Napomena: ovo je stariji zadatak, preuzet iz materijala za EF2PJ, i odnosi se na pseudojezik, a ne na programski jezik Pascal. U svakom slučaju, zadatak predstavlja dobar primer za ilustraciju razlika između pojedinih notacija.

Sastaviti BNF notaciju, EBNF notaciju, sintaksni dijagram i notaciju sa zagradama za opis sintakse podataka nizovnog tipa sa skalarnim elementima.

(Primer: `Matrica: ARRAY [ 10..100, LOGICAL ] OF INTEGER`)

Rešenje:

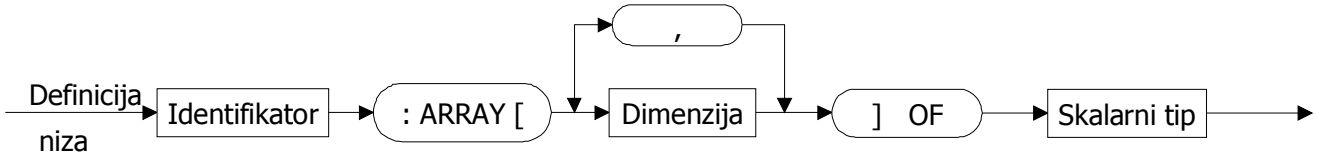
A) BNF notacija:

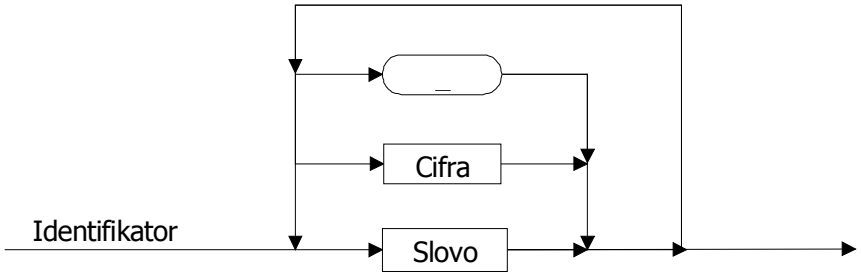
```
<definicija niza> ::= <identifikator> : ARRAY [ <niz dimenzija> ] OF <skalarni tip>
<identifikator> ::= <slovo> | <identifikator><slovo> | <identifikator><cifra> |
                    <idenitifikator>_
<niz dimenzija> ::= <dimenzija> | <dimenzija> , <niz dimenzija>
<skalarni tip> ::= LOGICAL | CARDINAL | INTEGER | CHARACTER | REAL |
                   (<niz identifikatora>) | <opseg> | <identifikator>
<slovo> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q| R |
            S | T | U | V | W | X | Y | Z
<cifra> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<dimenzija> ::= LOGICAL | CHARACTER | <opseg> | <identifikator>
<niz identifikatora> ::= <identifikator>,<niz identifikatora>
<opseg> ::= <konstanta> .. <konstanta>
<konstanta> ::= <ceo broj> | '<znak>' | <identifikator>
<ceo broj> ::= <predznak><kardinalan broj>
<znak> ::= <slovo | <cifa> | <specijalan znak>
<predznak> ::= + | - | <prazno>
<kardinalan broj> ::= <cifra> | <cifra><kardinalan broj>
<specijalan znak> ::= + | - | * | / | = | . | , | : | ; | ( | ) | ' | " | ! | # |
                      $ | % | & | _ | ? | @ | ^ | \ | ~ | ` | [ | ]
<prazno> ::=
```

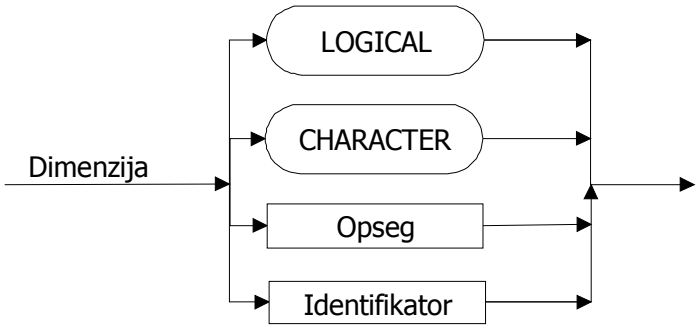
B) EBNF notacija

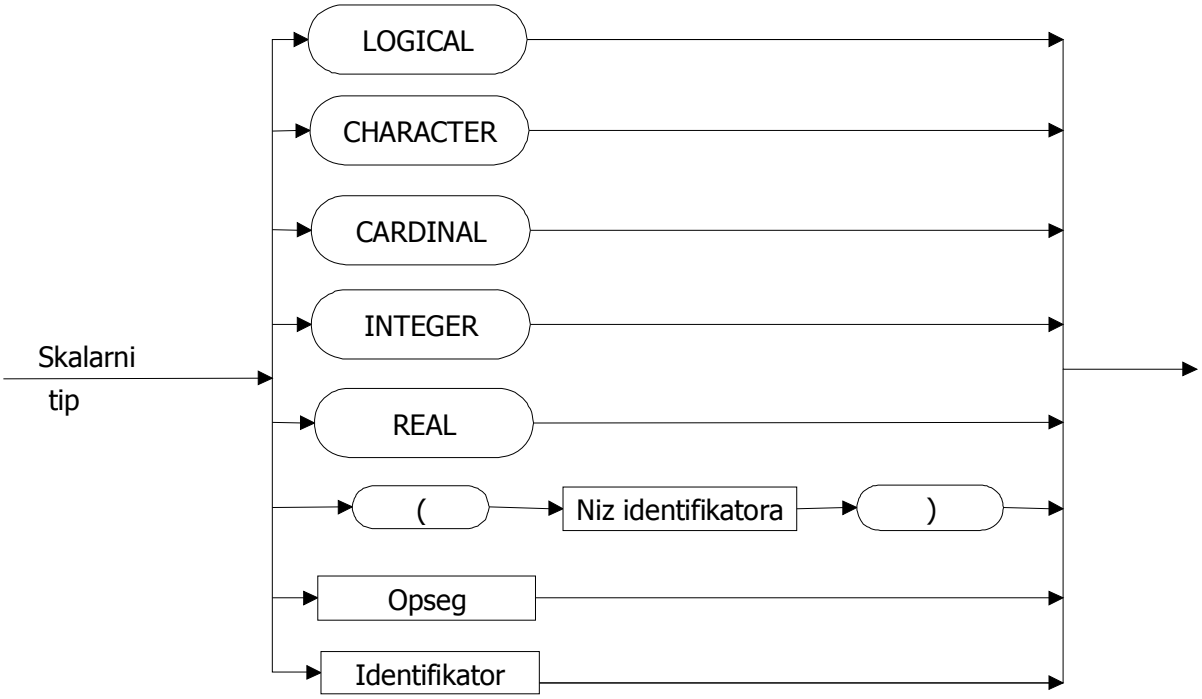
```
DefinicijaNiza = Identifikator ": ARRAY [" Dimenzija { "," Dimenzija } "] OF"
                 SkalarniTip.
Identifikator = Slovo { ( Slovo | Cifra | "_" ) }.
Dimenzija = ( "LOGICAL" | "CHARACTER" | Opseg | Identifikator ).
SkalarniTip = ( "LOGICAL" | "CHARACTER" | "CARDINAL" | "INTEGER" | "REAL" |
              "(" NizIdentifikatora ")" | Opseg | Identifikator ).
Slovo = ( "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" |
          "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" |
          "Y" | "Z" ).
Cifra = ( "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ).
Opseg = Konstanta ".." Konstanta.
NizIdentifikatora = Identifikator { "," Identifikator }.
Konstanta = ( CeoBroj | "'" Znak "'" | Identifikator ).
CeoBroj = [ ( "+" | "-" ) ] KardinalanBroj.
Znak = ( Slovo | Cifra | SpecijalanZnak ).
KardinalanBroj = Cifra { Cifra }.
SpecijalanZnak = ("+" | "-" | "*" | "/" | "=" | "." | "," | ":" | ";" | "(" | ")" |
                 "'" | """ | "!" | "#" | "$" | "%" | "&" | """" | "?" | "@" | "^" |
                 "\" | "~" | "`" | "[" | "]"| "{" | "}" | "<" | ">" | "¦" ).
```

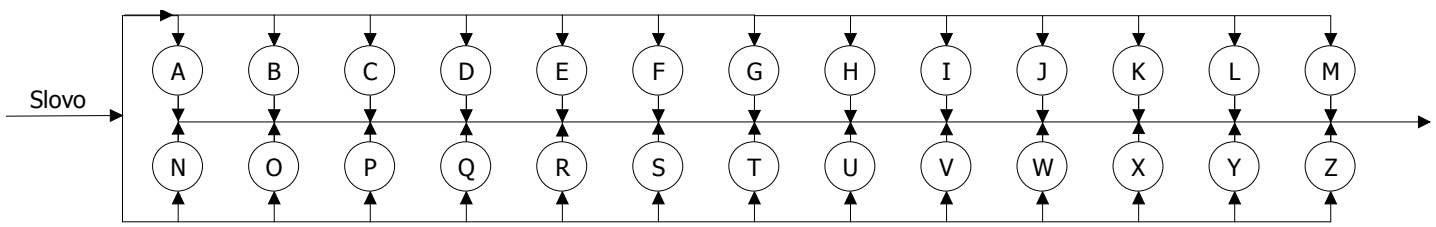
C) Sintaksni dijagram

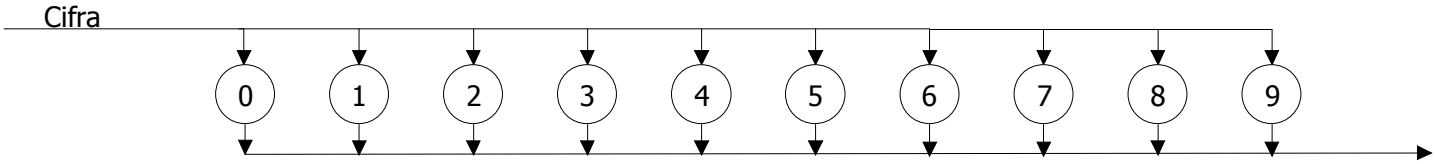


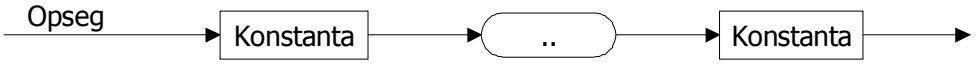


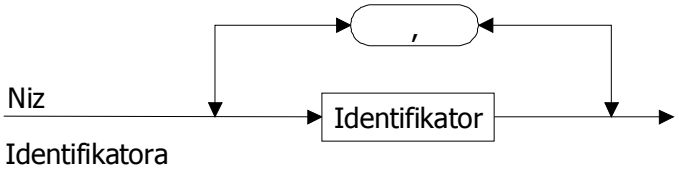


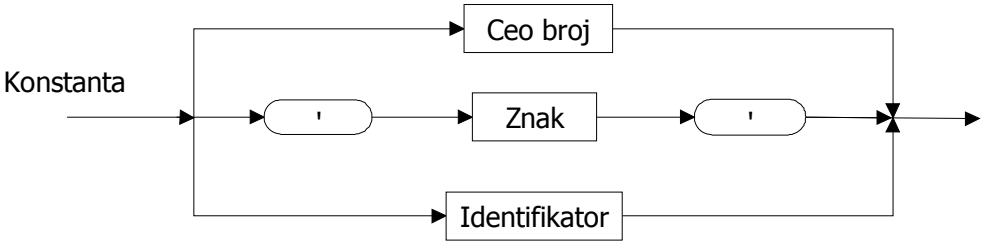


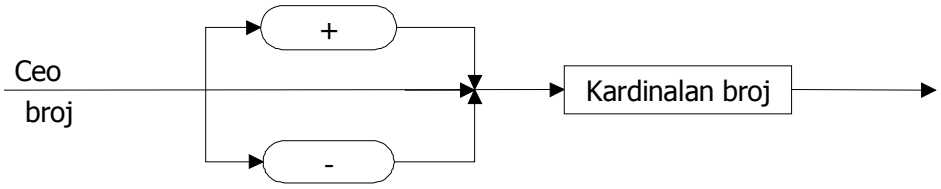


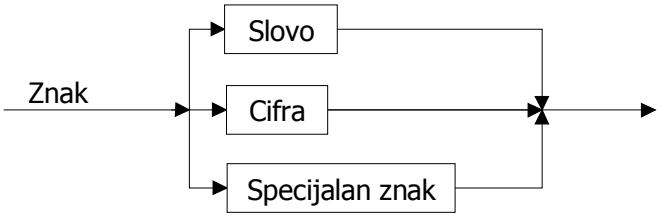

Cifra
0
1
2
3
4
5
6
7
8
9

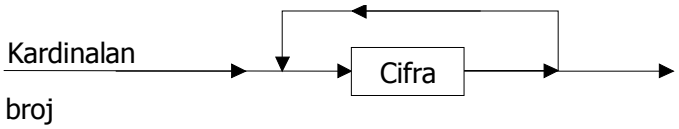

Opseg
Konstanta
..
Konstanta

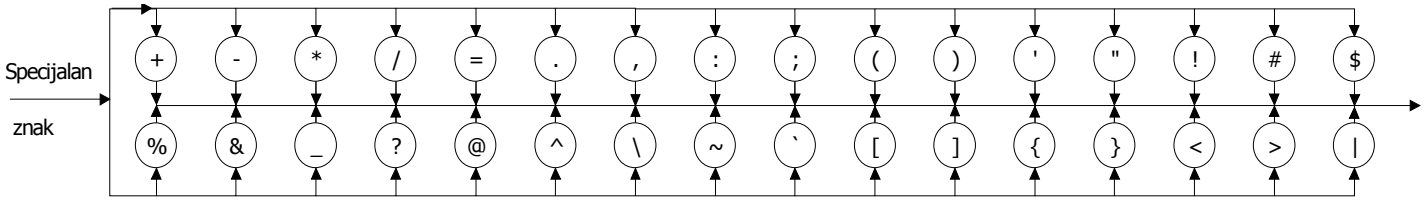

,
Niz
Identifikator
Identifikatora


Ceo broj
Konstanta
'
Znak
'
Identifikator


+
Ceo
broj
Kardinalan broj
-


Slovo
Znak
Cifra
Specijalan znak


Kardinalan
Cifra
broj


Specijalan
znak
+
-
*
/
=
.
,
:
;
(
)
'
"
!
#
$
%
&
_
?
@
^
\
~
`
[
]
{
}
<
>
|

D) Notacija sa zagradama

definicija_niza = identifikator : ARRAY [niz_dimenzija] OF skalarni_tip

identifikator = slovo [{ slovo | cifra | _ } K]

niz_dimenzija = dimenzija [{ , dimenzija } ...]

skalarni_tip = { LOGICAL | CARDINAL | INTEGER | CHARACTER | REAL | (niz_identifikatora) | opseg | identifikator }

slovo = A | B | C | D | E | F | G | D | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
cifra = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

dimenzija = { LOGICAL | CHARACTER | opseg | identifikator }

niz_identifikatora = identifikator [{ , identifikator } ...]
opseg = konstanta .. konstanta

konstanta = { ceo_broj | 'znak' | identifikator }

ceo_broj = [{ + | - }] {*cifra*} ...

znak = { slovo | cifra | specijalan_znak }

specijalan_znak = + | - | * | / | = | . | , | : | ; | (|) | ' | " | ! | # | $ | % | & | _ | ? | @ | ^ | \ | ~ | ` | [|] |

Zadatak IZ11 (Januar 2007)

Koji od ponuđenih izraza odgovaraju sledećoj sintaksnoj definiciji <start>:

```
<start>::= b<x>
<x>::= a<y>|<x><y>
<y>::= <y>a|bb<z>
<z>::= aa
```

A) baabaabaabaa
B) baaabbaabbaab
(C) babbaaabbaa

Rešenje

Posmatranjem datih pravila, može se uočiti da ne postoji smena koja omogućava da se neterminal **b** pojavi na kraju niza. Prema tome, odmah se može zaključiti da je odgovor pod B) netačan. Slično tome, pravila dozvoljavaju da se pojedinačni neterminal **b** pojavi samo na početku izraza. U ostatku izraza, neterminal **b** mora da se pojavljuje u parovima. Prema tome, odgovor A) je netačan. Ostaje da se analizira odgovor C) za koji se posmatranjem ne može odmah dati odgovor.

C)

```
babbaaabbaa
babbaaabb<z>
babbaaa<y>
babb<z>a<y>
ba<y>a<y>          ba<y>a<y>
ba<y><y>           ba<y><x>
b<x><y>            b<x><x>
b<x>               ?
<start>
```

Ako je ovaj odgovor tačan, par terminala **bb** se pojavljuje samo ako za njim sledi par neterminala **aa**. Zato se zaključuje da se kraj izraza dobija iz neterminala **y**. Slično se postupa sa drugim parom neterminala **bb**.
Od tada, kao što je prikazano, moguća su dva puta u smenjivanju, od kojih jedan ne može da se svede na neterminal **start**. U takvoj situaciji treba probati sve mogućnosti jer u suprotnom može pogrešno da se zaključi da ni ovaj izraz ne odgovara datoj sintaksnoj definiciji.

Zadatak IZ 2005-12-02 (Kolokvijum 2005)

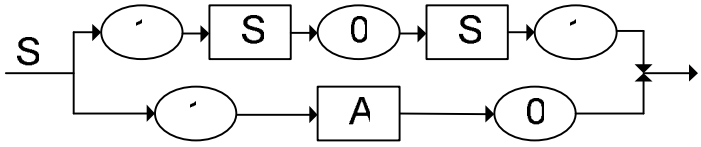
Koji od prikazanih realnih brojeva odgovara ponuđenoj sintaksnoj definiciji u EBNF notaciji?

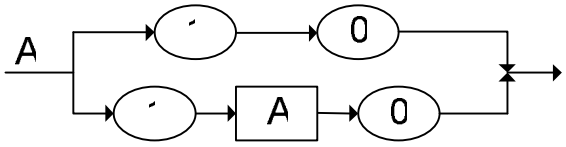
```
RealanBroj = Znak Cifra "." NizCifara "E" Znak Eksp.
Znak = [("+" | "-")].
Cifra = ("1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9").
NizCifara = {(Nula | Cifra)}.
Nula = "0".
Eksp = (Cifra [Cifra] | Nula [Cifra] | [Cifra] Nula).
```

A) +3.14159265E00
(**B**) -2.718281E-10
C) 128.E+3

Zadatak IZ 2005-12-02 (Kolokvijum 2005)

Za koja od ponuđenih sintaksnih pravila S generiše istu sekvencu kao dat sintaksni dijagram?





A) BNF:

```
<S> ::= 1<S>0<S>1|1<A>0
<A> ::= 1<p>0
<p> ::= {1<A>0}
```

B) EBNF:

```
S = "1" A {A} ("1" | "0").
A = "1" [A] "0".
```

C) EBNF:

```
S = (B|"1"{A}"0").
B = "1" S "0" S "1".
A = "1" {A} "0".
```

PASCAL

Zadatak UVODNI_01

Sastaviti program na programskom jeziku Pascal koji na standardnom izlazu ispisuje poruku "Zdravo svete!".

```
PROGRAM ZDRAVO_SVETE(output);
BEGIN
  WRITELN(output, 'Zdravo svete!')
END.
```

Tradicionalno, prvi program koji se piše na višem programskom jeziku (poput jezika Pascal) treba da bude "Zdravo svete!" ("Hello world!" na engleskom jeziku).

Prema standardu jezika Pascal, svaki program počinje rezervisanom reči `PROGRAM` nakon čega sledi naziv programa koji progamer sam bira, u ovom slučaju `ZDRAVO_SVETE`. Generalni savet u vezi imenovanja programa jeste da ime treba da jezgrovito opiše namenu programa, ako je to moguće. Treba izbegavati generička imena poput `MOJ_PROGRAM`. Potom se između zagrada navode logički nazivi ulaznih i izlaznih uređaja koje će program koristiti, u ovom slučaju oznaka za standardni izlaz – `output` – koji je najčešće ekran računara. Rezervisana reč `PROGRAM` i ostatak reda predstavljaju deklarativni deo programa koji se ne pretvara u mašinski kod, već služi kao uputstvo prevodiocu. Neki prevodioci za Pascal, poput Turbo Pascal-a, ne insistiraju na tome da program mora da počne na opisan način. Ipak studentima se savetuje da se drže propisanog standarda prilikom pisanja programa.

Program zapravo počinje sa rezervisanom reči `BEGIN` a završava se rezervisanom reči `END` iza koje, u prikazanom slučaju (kraj programa) sledi tačka ("."). Ove dve rezervisane reči označavaju granice u okviru kojih se nalazi program koji se izvršava. Sam program ima tačno jednu instrukciju: `WRITELN`. Njen naziv potiče od "write line", što u prevodu sa engleskog jezika znači "piši liniju". Ova instrukcija ispisuje navedeni znakovni niz na navedenom izlaznom uređaju, nakon čega se izlaznom uređaju signalizira da tekst koji bude sledio (ako ga bude) treba ispisati u narednoj liniji. Jednostavno rečeno, nakon ispisa zadatog teksta prelazi se u sledeći red. Treba primetiti da instrukcija `WRITELN` u ovom programu ima dva argumenta, od kojih je prvi logički naziv uređaja na kojem treba ispisati poruku, a drugi sama poruka. Generalno, instrukcija `WRITELN` može imati više od dva argumenta, ali prvi mora biti logički naziv izlaznog uređaja. Još jednom, neki prevodioci za Pascal dozvoljavaju da se ne navede naziv izlaznog uređaja i tada se podrazumeva standardni izlaz. Kod takvih prevodilaca, instrukcija `WRITELN('Zdravo svete!')` bi imala isti efekat kao i gore navedena. Ponovo, studentima se preporučuje da eksplicitno navode logičke nazive ulazno-izlaznih uređaja.

Primetiti da se u programskom jeziku Pascal znakovni nizovi navode pod apostrofima a ne pod znacima navoda.

Zadatak UVODNI_02

Registarski broj se predstavlja kao petocifren ceo broj u opsegu od 10000 do 30000, od čega prve četiri cifre čine registarski broj u užem smislu a peta cifra čini kontrolnu cifru. Uloga kontrolne cifre jeste da spreči pogrešan unos registarskog broja. Registarski broj je ispravan ako je suma svih cifara deljiva sa 7. Napisati program na programskom jeziku Pascal koji sa standardnog ulaza čita registarski broj firme i proverava da li je broj ispravan. U slučaju ispravnog broja na standardnom izlazu ispiše "Ispravan" a u suprotnom "Neispravan".

```
PROGRAM REG_BR(input, output);
VAR
        i, broj, suma : INTEGER;
BEGIN
        READ(input, broj);
        IF (broj < 10000) OR (broj > 30000) THEN
                WRITELN(output, 'Neispravan')
        ELSE
        BEGIN
                suma := 0;
                FOR i := 1 TO 5 DO
                BEGIN
                        suma := suma + broj MOD 10;
                        broj := broj DIV 10
                END;

                IF suma MOD 7 = 0 THEN
                        WRITELN(output, 'Ispravan')
                ELSE
                        WRITELN(output, 'Neispravan');
        END
END.
```

Program ilustruje korišćenje kontrolne strukture **IF-THEN-ELSE** kao i korišćenje **FOR** ciklusa.

Popularni prevodioci za Pascal poput Turbo Pascal-a ili Free Pascal-a predstavljaju cele brojeve na dužini od 16 bita, što znači da je vrednost najvećeg celog broja 32767. Da bi registarski broj mogao da se učita sa standardnog ulaza kao ceo broj, uvedeno je veštačko ograničenje da je registarski broj petocifren, u opsegu od 10000 do 30000. Studentima se preporučuje da razmisle i da napišu program koji bi istu proveru vršio nad registarskim brojevima sa većim brojem cifara (na primer dvanaest).

Pre rezervisane reči **BEGIN** kojom počinje program, nalazi se deklarativna sekcija programa koja počinje rezervisanom reči **VAR** i u kojoj se prevodiocu navode nazivi i tipovi promenljivih koje će biti korišćene u programu. U ovom programu se koriste tri celobrojne promenljive. Praktičan savet za imenovanje promenljivih jeste da njihovo ime odražava njihovu namenu. Na primer, u promenljivoj **broj** će biti smešten registarski broj koji se unosi sa standardnog ulaza a u promenljivoj **suma** će biti smešten zbir vrednosti svih cifara registarskog broja. Promenljiva **i** u ovom programu ima sporednu ulogu jer služi kao brojač ciklusa, pa ne mora da ima smisleno ime (kao brojači ciklusa tradicionalno se koriste jednoslovni identifikatori poput i, j, k, ...). Treba izbegavati davanje jednoslovnih naziva bez posebnog značenja većem broju promenljivih jer utiču na smanjenu preglednost i razumljivost programa.

U programu se, najpre, instrukcijom **READ** sa standardnog ulaza (**input**), koji najčešće predstavlja tastaturu računara, čita ceo broj i smešta se u promenljivu **broj**. Nakon toga se ispituje vrednost učitanog broja. Ako je vrednost promenljive **broj** manja od 10000 ili veća od 30000, onda ona prema postavci zadatka ne predstavlja registarski broj firme. Zbog toga se, ako je taj uslov ispunjen,

ispisuje poruka "Neispravan". U suprotnom, izvršenje programa preskače instrukciju **WRITELN** i nastavlja se unutar **ELSE** grane.

ELSE grana počinje rezervisanom reči **BEGIN** i završava se sa **END**. U ovom slučaju **BEGIN** i **END** ne označavaju početak i kraj programa već označavaju blok instrukcija koje logički predstavljaju jednu celinu u programu. Naime, ako bi u **ELSE** grani postojala samo jedna instrukcija, onda par **BEGIN** i **END** ne bi bio neophodan jer se tada nedvosmisleno zna koja instrukcija potpada pod **ELSE** granu. U suprotnom slučaju, par **BEGIN** i **END** je neophodan jer drugačije prevodilac ne bi mogao da zna gde se završava niz instrukcija koje logički spadaju pod **ELSE** granu. Ovaj princip (par **BEGIN** i **END**) se koristi kod svih kontrolnih struktura koje obuhvataju više instrukcija.

U **ELSE** grani se najpre vrednost promenljive **suma** postavlja na 0. **Veoma je bitno** navesti početnu vrednost svih promenljivih (osim, naravno, onih čija se vrednost čita sa nekog ulaznog uređaja) pre njihovog korišćenja u nekom izrazu. U suprotnom ponašanje programa može biti nepredvidivo, osim ako prevodilac ne pravi takav kod da po startovanju programa automatski postavi vrednosti svih promenljivih na podrazumevanu vrednost. Dobra programerska praksa jeste da se ne treba oslanjati na mogućnosti prevodioca jer one ne moraju biti standardne, pa će se isti program preveden na drugom prevodiocu drugačije ponašati.

Nakon toga se otpočinje ciklus sa 5 iteracija i u svakoj iteraciji će se najpre promenljivoj **suma** dodati ostatak deljenja promenljive **broj** sa 10 a zatim će se izvršiti celobrojno deljenje promenljive broj sa 10. Na primer ako promenljiva **broj** ima vrednost 12203, nakon prve iteracije promenljiva **suma** će imati vrednost 3 a promenljiva **broj** 1220, nakon druge 3 i 122, nakon treće 5 i 12, nakon četvrte 7 i 1 i na kraju 8 i 0. **FOR** ciklus će se ponoviti tačno 5 puta (tokom kojih će promenljiva i menjati vrednost od 1 do 5, što je u ovom slučaju nebitno).

Na kraju se ispituje da li je vrednost promenljive **suma** deljiva sa 7, odnosno da li je ostatak prilikom deljenja sa 7 jednak 0. U slučaju da jeste, ispisaće se poruka "Ispravan" a u suprotnom slučaju "Neispravan".

Napomena u vezi formatiranja izvornog teksta programa: preglednost programa je daleko veća ako se instrukcije između jednog para **BEGIN** i **END** pomere za nekoliko mesta udesno jer je tada vizuelno jasno koji skup instrukcija pripada datom bloku. Pravilno formatiranje ne predstavlja preduslov uspešnog prevođenja programa, jer bi prevodilac uspešno preveo i program koji se u potpunosti nalazi u jednom redu teksta. Pravilno formatiranje pomaže programeru u procesu programiranja i – što je još važnije – kasnijeg prepravljanja i unapređivanja programa.

Zadatak Z89

Sastaviti program na programskom jeziku Pascal za nalaženje zbira elemenata zadatog niza brojeva. Program treba da učita preko glavne ulazne jedinice broj elemenata niza a zatim i elemente niza, da ispiše na glavnoj izlaznoj jedinici učitani niz brojeva i rezultat. Prethodne operacije treba ponavljati sve dok se za dužinu niza ne učita nula.

```
PROGRAM z89(input,output);
CONST MaxDuz = 100;
VAR
  niz:ARRAY[1..MaxDuz] OF integer;
  n,i,zbir:integer;
BEGIN
  write(output,'Unesite broj elemenata niza: ');
  readln(input,n);

  WHILE(n>0) AND (n<=MaxDuz) DO BEGIN
    zbir:=0;

    FOR i:=1 TO n DO BEGIN
      write(output, 'Unesite ', i,'. element niza');
      readln(input,niz[i])
    END;

    FOR i:=1 TO n DO zbir:=zbir+niz[i];

    write(output,'Uneti niz glasi:');
    FOR i:=1 TO n-1 DO write(output,' ',niz[i],',');
    writeln(output,' ',niz[n],'.');
    writeln(output,'Rezultat je ',zbir,'.');

    write(output,'Unesite broj elemenata niza: ');
    readln(input,n)
  END
END.
```

Zadatak Z91

Sastaviti program na programskom jeziku Pascal za tabeliranje polinoma zadatog pomoću koeficijenata.

```
PROGRAM TabeliranjePolinoma (input,output);
VAR i, n: integer;
    Polinom: ARRAY [0..100] OF real;
    xmin, xmax, dx, x, p: real;
BEGIN
  writeln;
  write('Red polinoma: '); readln (n);
  write ('Koeficijenti polinoma: ');
  FOR i := n DOWNTO 0 DO read (Polinom[i]); readln;
  write('xmin,xmax,dx: '); readln (xmin, xmax, dx);
  writeln ('======================');
  writeln ('     x          p(x)');
  writeln ('======================');
  x := xmin;
  WHILE x <= xmax DO
    BEGIN
      p := Polinom[n];
      FOR i := n - 1 DOWNTO 0 DO p := p * x + Polinom[i];
      writeln (x:10:3, p:12:3);
      x := x + dx
    END;
  writeln ('======================');
END.
```

Zadatak Z41.PAS

Sastaviti program na programskom jeziku Pascal kojim se u tekstu, koji se u proizvoljnom broju redova učitava preko standardnog ulaza (**input**), odredi broj cifara, velikih slova, malih slova, i ostalih znakova. Tekst koji treba obraditi se završava praznim redom. Rezultate ispisati na standardnom izlazu (**output**).

```
PROGRAM znakovi(input, output);

VAR
  n_cif, n_vel, n_mal, n_ost, i: integer;
  linija: STRING[255]; { max. duzina stringa za Turbo Pascal }
  znak: char;

BEGIN
  n_cif := 0; n_vel:= 0; n_mal := 0; n_ost := 0;
  writeln(output, 'Unesite linije koje treba obraditi (prazan red za kraj unosa)');
  readln(input, linija);
  WHILE (Length(linija) > 0) DO
  BEGIN
    FOR i := 1 TO Length(linija) DO
      BEGIN
        znak := linija[i];
        IF (znak >= '0') AND (znak <= '9') THEN
          n_cif := n_cif + 1
        ELSE IF(znak >= 'A') AND (znak <= 'Z') THEN
          n_vel := n_vel + 1
        ELSE IF(znak >= 'a') AND (znak <= 'z') THEN
          n_mal := n_mal + 1
        ELSE
          Inc(n_ost)
      END; { FOR i: 1 TO Length(linija) }
    readln(input, linija);
  END; { WHILE (Length(linija) > 0) }
  writeln(output, 'Cifara ima ukupno: ', n_cif);
  writeln(output, 'Velikih slova ima ukupno: ', n_vel);
  writeln(output, 'Malih slova ima ukupno: ', n_mal);
  writeln(output, 'Ostalih znakova ima ukupno: ', n_ost);

  writeln(output, 'Pritisnite [ENTER] za kraj programa');
  readln(input)
END.
```

Zadatak ilustruje korišćenje **STRING** tipa podataka, korišćenje **IF-ELSE IF-ELSE** kontrolne strukture, kao i korišćenje petlje sa izlazom na dnu. Poslednja dva reda pokazuju kako se programira čekanje da korisnik pročita ispisane poruke, pa tek onda kada sam odluči, završava izvršavanje programa.

Skupovi, nabrojani tip, potprogrami, zapisi

Zadatak Z38.PAS

Sastaviti program na programskom jeziku Pascal za određivanje broja različitih elemenata u zadatom nizu celih brojeva koji spadaju u opseg od 0 do 255.

```
PROGRAM Skupovi(input, output);

CONST
  MAX_DUZ = 300;

VAR
  niz: ARRAY[1..MAX_DUZ] OF 0..255;
  broj: integer;
  n, k, i: 0..MAX_DUZ;
  skup: SET OF 0..255;

BEGIN
  REPEAT
    { unos elemenata niza }
    n := 1;
    REPEAT
      writeln('Unesite ', n, '. element niza (0..255, van opsega za kraj unosa): ');
      readln(input, broj);
      IF (broj >= 0) AND (broj <= 255) THEN
        BEGIN
          niz[n] := broj;
          n := n + 1;
        END
    UNTIL (broj < 0) OR (broj > 255) OR (n>MAX_DUZ);

    { ukoliko niz nije prazan, odredjivanje broja razlicitih elemenata }
    IF (n > 1) THEN
      BEGIN
        skup := [];
        k := 0;
        FOR i:= 1 TO n-1 DO
          IF NOT (niz[i] IN skup) THEN
            BEGIN
              { povecavamo brojac za broj razlicitih elemenata }
              k := k + 1;
              { ovde + NIJE operacija sabiranja, vec operacija unije skupova }
              { element pretvaramo u skup koji sadrzi samo taj element }
              { to se obavlja tako sto element stavimo u [] }
              skup := skup + [niz[i]]
            END;

        { ispisivanje rezultata }
        writeln(output, 'Broj razlicitih elemenata je: ', k)
      END

  { ponavljamo ceo proces sve dok se ne unese prazan niz }
  UNTIL n = 1
END.
```

Zadatak ilustruje korišćenje skupovskog tipa, petlje sa izlazom na dnu i brojačke petlje. Pokazano je i korišćenje komentara.

Zadatak IZ53

Pod pretpostavkom da je sledeći progamski segment na programskom jeziku Pascal semantički ispravan, šta ispisuje?

```
a := [3];  b := [2];  c := (a+b) * (a-b);
FOR i := 0 TO 5 DO IF i IN c THEN write (i: 2);
```

A) 3 2 B) 5 C) 3

Obrazloženje:

Kontekst određuje da su **a**, **b** i **c** skupovne promenljive, pa je prema tome **c** presek unije skupova **a** i **b** { [3, 2] } i razlike skupova **a** i **b** { [3] }, tj. skup { [3] }.

Dakle , ispisivanje će se izvršiti samo za **i** = 3.

Zadatak IZ50

Ako je deklarativni deo programa na programskom jeziku Pascal:

```
TYPE
  ptice = ( slavuj, roda, vrabac );
  avioni = ( mig, boing, mirage );

VAR
  p: ptice;
  a: avioni;
  aa: ARRAY [ptice] OF avioni ;
```

koja će onda sekvenca naredbi tog programa biti formalno ispravna?

A)

```
  a := mig;
  FOR p := slavuj TO vrabac DO
    aa[p] := succ( succ( a ) );
```

B)

```
  p := pred( vrabac );
  a := succ( boing );
  IF p = a THEN writeln;
```

C)

```
  p := vrabac;
  a := mirage;
  IF ord( pred(p) ) = ord( a )
    THEN writeln(a);
```

Odgovor: A

Obrazloženje:

A) ispravno popunjava vektor aviona podatkom čija je vrednost **mirage**;

B) poredi promenljive različitog tipa **p = a**;

C) sadrži ispis (**writeln**) nabrojivog tipa, što nije defiisano ISO standardom za Pascal.

Zadatak IZ 2006-09-20 (MODIFIKACIJA)

Napisati program na jeziku Pascal kojim se iz jednodimenzionog niza brojeva tipa REAL izdvaja najduži podniz koji je uređen strogo rastuće. Pretpostaviti da niz sadrži bar jedan element. Program treba da učita niz, izdvoji najduži podniz u drugi niz i ispiše drugi niz na standardnom izlazu. Program treba da ponavlja ove korake sve dok se za dužinu niza unosi prirodan broj.

```
Program Zakdatak1 (input, output);
Const MAX=100;
Type niz=array[1..MAX] of real;
Var
   nizA, nizB: niz;
   duzinaA, duzinaB, i: integer;
   trduz, maxduz: integer;
   trind, maxind: integer;

Begin
   Repeat
     writeln(output);
     write(output,'Unesite duzinu niza(0 za kraj rada) ');          | Unos dužine niza i
     readln(input, duzinaA);                                        | provera validnosti
     If duzinaA > 0 Then
     Begin
       writeln(output, 'Unesite elemente niza ');                   | Unos elemenata niza
       for i:= 1 To duzinaA Do
          read(input, nizA[i]);

       maxduz:= 1; maxind:= 1; trduz:= 1; trind:= 1;                | Inicijalizacija
                                                                    | promenljivih

       For i:=2 To duzinaA Do                                       | Glavna obrada:
       Begin                                                        | pronalaženje najdužeg
         If nizA[i] > nizA[i-1] Then                                | podniza koji
           trduz:= trduz +1                                         | zadovoljava zadati
         Else                                                       | kriterijum.
         Begin
           If trduz > maxduz Then                                   | U ciklusu se prolazi
           Begin                                                    | kroz sve elemente
             maxduz:= trduz;                                        | niza, počevši od
             maxind:= trind                                         | drugog, i utvrđuje se
           End;                                                     | da li je između dva
           trind:= i;                                               | uzastopna elementa
           trduz:= 1                                                | postoji traženi
         End                                                        | poredak.
       End;
       If trduz > maxduz Then
       Begin
         maxduz:= trduz;
         maxind:= trind
       End;

       For i:=maxind To (maxind+maxduz-1) Do                        | Prepisivanje uočenog
         nizB[i-maxind+1]:= nizA[i];                                | podniza niza A u
       duzinaB:= maxduz;                                            | niz B

       writeln(output,'Najduzi strogo rastuci podniz je');          | Ispis niza B
       For i:= 1 To duzinaB Do
         write(output, nizB[i]:2:2, ' ')

     End

   Until duzinaA <= 0

End.
```

U postavci zadatka nije naznačeno, pa se – kao razumna pretpostavka – uzima da je najveća dužina niza koji se unosi 100 elemenata. Definisan je poseban tip podatka **niz** za potrebe deklarisanja promenljivih tipa niza (nizA i nizB).

Ideja za rešenje ovog programskog problema je sledeća: treba analizirati sve uzastopne elemente niza i brojati koliko uzastopnih elemenata zadovoljava postavljeni kriterijum: traži se najduži strogo rastuće uređen podniz. Drugim rečima, traži se podniz kod kojeg je svaki element veći od prethodnog. Kad god se bude ustanovilo da dati kriterijum nije zadovoljen, treba prekinuti brojanje i utvrditi da li je novootkriveni (tj. upravo analizirani podniz) duži od prethodno poznatog najdužeg podniza. Ako jeste, onda novootkriveni podniz treba proglasiti za najduži. U suprotnom treba ga odbaciti. U postavci zadatka nije precizirano šta treba raditi ako postoji više podnizova iste (maksimalne) dužine. Zbog toga će u rešenju biti usvojena pretpostavka da prvi takav podniz na koji se naiđe prilikom obrade treba proglasiti za "pobednika" i njega izdvojiti.

Za potrebe uočavanja traženog podniza u nizu A koristiće se dve promenljive: **trduz** i **trind**. Njhova namena je sledeća: **trduz** označava **TR**enutnu **DUŽ**inu podniza koji se analizira, a **trind** (**TR**enutni **IND**eks) označava indeks u nizu A od kojeg počinje podniz koji se analizira. Te dve informacije (gde počinje podniz i koja je njegova dužina) su dovoljne da bi moglo da se izvrši njegovo izdvajanje u drugi niz. S obzirom na to da traženi podniz može početi od prvog elementa niza A, početna vrednost za **trind** treba da bude 1. Takođe, prilikom planiranja treba pretpostaviti da niz A može biti uređen nerastuće (odnosno naredni element u nizu je jednak ili manji od prethodnog). U tom slučaju ne postoji podniz dužine dva ili duži koji zadovoljava traženi uslov, pa je zbog toga početna vrednost za **trduz** postavljena na 1.

Nakon ovog objašnjenja, smisao promenljivih **maxduz** i **maxind** je jasna: u njima će se pamtiti dužina najdužeg podniza i indeks u nizu A od kojeg počinje najduži podniz. Svaki put kada se uoči da traženi poredak nije zadovoljen između dva susedna elementa u nizu A, program proverava da li je dužina novootkrivenog podniza veća od dužine prethodno poznatog najdužeg podniza. Ako jeste, program prepisuje vrednosti **trduz** i **trind** u **maxduz** i **maxind**, respektivno, i time novootkriveni podniz "proglašava" za najduži podniz. U nastavku, u svakom slučaju (bilo da je novootkriveni podniz duži od prethodnog najdužeg ili ne) se vrednost promenljive **trduz** postavlja na 1 a vrednost promenljive **trind** na indeks elementa niza A koji je "prekinuo" traženu sekvencu. Time se praktično inicira traženje novog podniza počevši od "prekidnog" elementa, a kao i ranije, pretpostavlja se da je njegova nova dužina 1.

Kada se završi FOR ciklus, potrebno je još jednom proveriti da li je novootkriveni podniz duži nego prethodno otkriven najduži podniz. To je potrebno uraditi u posebnom slučaju kada se najduži podniz nalazi na kraju niza A, odnosno kada je poslednji element podniza ujedno i poslednji element niza A. Ako je to slučaj, FOR ciklus će doći do poslednjeg elementa niza A, a kako je za njega ispoštovan traženi poredak, program neće izvršiti granu koja postavlja nove vrednosti za **maxind** i **maxduz**. U slučaju da se nakon FOR ciklusa još jednom ne izvrši potrebna provera, stvarni najduži podniz, koji se nalazi na kraju niza A, bi ostao "neprimećen".

Nakon toga se vrši prepisivanje uočenog podniza iz niza A u niz B. Uočeni podniz počinje od onog elementa niza A koji odgovara indeksu zapamćenom u **maxind**, a dužina tog niza je zapamćena u **maxduz**. Zbog toga poslednji element podniza ima indeks maxind+maxduz-1 u nizu A. Slično tome, elementi koji se prepisuju u niz B se smeštaju počevši od indeksa 1. Kako je ciklus formiran od brojačke promenljive **i** koja služi za indeksiranje niza A, odgovarajući indeks niza B se dobija oduzimanjem **maxind** od tekuće vrednosti promenljive **i** i dodavanja 1 (da bi prvi element bio pod indeksom 1).

Zadatak Z97

Sastaviti program na programskom jeziku Pascal za izračunavanje aritmetičke i geometrijske sredine zadatog niza pozitivnih brojeva.

Prvo rešenje (loše)

```
PROGRAM z97(input,output);
VAR  n,i:integer;
     niz:ARRAY[1..100] OF real;
     a,g,tmp:real;

BEGIN
  write(output,'Unesite duzinu niza: ');
  read(input,n);
  writeln(output,'Unesite clanove niza:');
  FOR i:=1 TO n DO niz[i]:=tmp;
  a:=0;  g:=1;
  FOR i:=1 TO n DO BEGIN
    a:=a+niz[i]/n;
    g:=g*exp(ln(niz[i])/n)
  END;
  writeln(output,'Aritmeticka sredina je ',a:2:2,'.');
  writeln(output,'Geometrijska sredina je ',g:2:2,'.')
END.
```

Iako tačno, prikazano rešenje je loše iz više razloga. Najpre, ne proverava se da li su unete vrednosti korektne. Na primer, da li su za dužinu niza i za elemente niza unete pozitivne vrednosti. Zatim računanje aritmetičke i geometrijske sredine, iako korektno napisano, nije spremno za višestruku upotrebu: naime, ako bi se tražila modifikacija programa takva da se ove vrednosti određuju za više nizova da bi se u zavisnosti od rezultata vršila obrada tih nizova, u ovakvom rešenju bi bilo neophodno da se napišu novi FOR ciklusi koji vrše računanje ovih vrednosti, ali za druge dužine i za druge podatke. To je izrazito nepraktično i podložno je greškama. Osim toga nije pregledno.

Mnogo bolje rešenje, sa programerske tačke gledišta, jeste da se napišu posebni potprogrami koji bi vršili određenu aktivnost: učitavanje broja, učitavanje niza, računanje neke vrednosti, itd. Takav program je pregledan, značajno lakši za analizu, jednostavniji za održavanje (modifikovanje) i unapređivanje.

Kako odrediti kada treba neki deo programa smestiti u potprogram? Potprogram treba da bude dobro izolovana celina koja obavlja neku aktivnost. Generalno govoreći, kada god se uoči sekvenca od nekoliko instrukcija (dve ili više) kojima se može pripisati jasno značenje, naročito ako je potrebno da ta sekvenca postoji na više mesta u programu. U gornjem primeru, računanje aritmetičke sredine se svodi na jedan ciklus u kojem se obavlja jedna jednostavna radnja, pa se na prvi pogled možda može reći da nije pravi kandidat za potprogram. Međutim, smisao te kratke sekvence je sasvim jasan i njena upotreba je precizno ograničena čineći je zapravo dobrim kandidatom za potprogram.

Bitno je da potprogrami ne koriste globalne promenljive (tj. promenljive glavnog programa). Jedna od bitnih osobina potprograma je njihova višestruka upotrebljivost. Ta osobina se ostvaruje parametrizacijom potprograma njihovim argumentima: potprogramima se dostavljaju podaci nad kojima oni obavljaju neku obradu. Kako konkretne vrednosti tih podataka (argumenata) nisu unapred poznate, potprogrami se mogu pozivati iz različitih konteksta čime je ostvarena višestruka upotrebljivost.

Drugo rešenje (**dobro**)

```
PROGRAM z97m(input,output);
TYPE NizBrojeva = ARRAY[1..100] OF real;
VAR  n,i:integer;
     niz:NizBrojeva;
     a,g,tmp:real;
PROCEDURE citaj_duzinu(var n : integer);
BEGIN
  REPEAT
    read(input, n);
    IF n < 1 THEN writeln(output,'Unesite pozitivnu vrednost!')
  UNTIL n > 0
END;

PROCEDURE citaj_poz_vrednost(var r : real);
BEGIN
  REPEAT
    read(input, r);
    IF r < 1 THEN writeln(output,'Unesite pozitivnu vrednost!')
  UNTIL r > 0
END;

PROCEDURE citaj_niz(var niz : NizBrojeva; var duz : integer );
VAR i : integer;
BEGIN
  writeln(output,'Unesite duzinu niza');
  citaj_duzinu(duz);
  writeln(output,'Unesite elemente niza');
  FOR i := 1 TO duz DO citaj_poz_vrednost(niz[i])
END;

FUNCTION ArSr(var niz : NizBrojeva; duz : integer) : real;
VAR i : integer; as:real;
BEGIN
  as := 0;
  FOR i := 1 TO duz DO as := as + niz[i];
  ArSr:=as
{Iako standard ne specificira nista osim da se promenljivoj koja je implicitno definisana
nazivom funkcije dodeli vrednost bar jednom, vecina prevodilaca nece dozvoliti citanje te
promenljive, bez obzira da li je inicijalizovana ili ne. Turbo Pascal ce, na primer,
prijaviti sintaksnu gresku ukoliko pokusamo da procitamo vrednost promenljive ArSr}
END;

FUNCTION GeomSr(var niz : NizBrojeva; duz : integer) : real;
VAR i : integer;
    gs : real;
BEGIN
  gs := 1;
  FOR i := 1 TO duz DO gs := gs * exp(ln(niz[i])/duz);
  GeomSr := gs
END;

BEGIN
  citaj_niz(niz, n);
  writeln(output,'Aritmeticka sredina je ', ArSr(niz, n):2:2, '.');
  writeln(output,'Geometrijska sredina je ', GeomSr(niz, n):2:2,'.')
END.
```

Nakon ove izmene, glavni program je značajno kraći i – očigledno – mnogo pregledniji.
Pretpostavka zadatka jeste da korisnik neće uneti dužinu niza veću od 100. Studentima se savetuje da za vežbu preprave program tako da obavlja dodatnu proveru na maksimalnu dužinu niza. Savet: prepraviti potprogram za čitanje dužine tako da dobije još jedan argument, a to je maksimalna dozvoljena dužina niza.

Zadatak IZ 2007-09-18 (Oktobar 2007-1)

Napisati program na programskom jeziku Pascal koji vrši obradu nad matricom celih brojeva. Nakon učitavanja dimenzija i elemenata matrice, treba urediti vrste prema rastućem zbiru njihovih elemenata. Ispisati uređenu matricu. Broj vrsta i broj kolona ne prelaze 40.

```
program ZAD1_OKT07(input, output);
type MATRICA = ARRAY[1..40, 1..40] of integer;
var m : MATRICA;
    i, j, k, v, zbir1, zbir2 : integer;

function zbir(var m:MATRICA; v,k:integer) : integer;
var i, z : integer;
begin
    z := 0;
    for i := 1 to k do z := z + m[v,i];
    zbir := z
end;
```

Određivanje zbira vrednosti elemenata zadate vrste matrice

```
procedure zameni(var m:MATRICA; v1, v2, k : integer);
var i,t : integer;
begin
    for i := 1 to k do
    begin
        t := m[v1, i];  m[v1, i] := m[v2, i]; m[v2, i] := t
    end
end;
```

Zamena elemenata dve vrste matrice

Glavni program

```
begin
        write(output, 'Broj vrsta?'); readln(input, v);
        write(output, 'Broj kolona?'); readln(input, k);

        for i := 1 to v do
                for j := 1 to k do
                begin
                        write(output, 'Element[',i,',',j,']?');
                        readln(input, m[i,j])
                end;
```

Učitavanje dimenzija i elemenata matrice

```
        for i := 1 to v do
        begin
                zbir1 := zbir(m, i, k);
                for j := i+1 to v do
                begin
                        zbir2 := zbir(m, j, k);
                        if zbir1 > zbir2  then
                        begin
                                zameni(m, i, j, k);
                                zbir1 := zbir2;
                        end
                end
        end;
```

Glavna obrada: uređivanje vrsta matrice prema rastućoj vrednosti sume elemenata jedne vrste

```
        writeln(output, 'Uredjena matrica:');
        for i := 1 to v do
        begin
                for j := 1 to k do
                        write(output, m[i,j], ' ');
                writeln(output);
        end
end.
```

Ispis matrice nakon obrade

Ovaj zadatak predstavlja ispitnu modifikaciju jednostavnog problema uređivanja (sortiranja) niza celih brojeva: umesto obrade niza, ovde treba da se obradi matrica koja je zapravo niz nizova.

U postavci zadatka se sugeriše redosled aktivnosti programa: najpre se učitavaju podaci (dimenzije matrice i njeni elementi), zatim se vrši tražena obrada koja predstavlja jezgro programa i na kraju se ispisuje matrica nakon obrade. U prikazanom rešenju glavni program je saglasno tome podeljen na tri zasebne celine. S obzirom na to da je u postavci zadat maksimalan broj vrsta i kolona matrice, nije neophodno da se ista dinamički kreira. Istom prilikom se ističe da se ne očekuje da korisnik koji koristi program unese vrednost veću od zadatog maksimalnog broja, pa se u rešenju ne proverava korektnost unetih podataka, što bi u praksi trebalo raditi.

Ideja za glavnu obradu, gde se vrši uređivanje matrice po vrstama, je sledeća: za svaku vrstu matrice (indeksiranu promenljivom `i`) traži se neka druga vrsta (indeksirana promenljivom `j`) tako da je zbir elemenata `i`-te vrste veći od zbira elemenata `j`-te vrste. Kada se nađe takav par vrsta, vrši se njihova zamena. Prolaz po vrstama matrice je realizovan u vidu dva ugneždena `FOR` ciklusa, spoljni po promenljivoj `i`, unutrašnji po promenljivoj `j`. Nakon zamene vrsta matrice još je potrebno postaviti promenljivu `zbir1`, koja predstavlja sumu elemenata `i`-te vrste, na novu vrednost.

Izračunavanje zbira elemenata jedne vrste matrice i zamena elemenata dve vrste matrice su realizovani kao potprogrami. U postavci zadatka nije traženo da se pišu potprogrami, ali se takvo rešenje prirodno nameće: radi se o netrivijalnim aktivnostima od kojih se jedna koristi dva puta. Na ovaj način je sasvim jednostavno praćenje rada programa i kontrola mogućih grešaka zato što suština obrade nije opterećena načinom računanja sume elemenata jedne vrste odnosno načinom zamene elemenata dve vrste matrice.

Funkcija `zbir` računa sumu elemenata jedne vrste matrice. Ova funkcija prima 3 argumenta: matricu, redni broj vrste čija se suma traži i broj kolona. Procedura `zameni` vrši zamenu elemenata dve vrste matrice (premešta sve elemente jedne vrste na mesto druge i obrnuto). Ova procedura prima četiri argumenta: matricu, redni broj prve i druge vrste i broj kolona. Aktivnosti koje obavljaju ovi potprogrami su krajnje jednostavni i nema potrebe komentarisati ih.

U postavci nije rečeno, ali **pravila dobrog stila u programiranju zahtevaju da potprogrami ne koriste promenljive glavnog programa** već da isključivo koriste svoje argumente kao sredstvo za prenos podataka. Zbog toga se matrica i broj kolona, iako vidljivi u potprogramima, prosleđuju kao argumenti.

Zadatak IZ Februar 2007-2 (Modifikovan)

Napisati funkciju **zbir** koja određuje decimalnu vrednost zbira brojeva **a** i **b**. Argumenti **a** i **b** su znakovni nizovi od tačno 3 elementa i predstavljaju pozitivne cele brojeve u heksadecimalnom brojnom sistemu. Pri tome, znak koji predstavlja cifru najveće težine je prvi u nizu (tj. element sa indeksom 1). Napisati glavni program koji sa standardnog ulaza učitava dva niza od najviše 3 znaka koji treba da predstavljaju heksadecimalne cifre, a potom pozivanjem funkcije **zbir** računa i na standardnom izlazu ispisuje zbir ta dva broja u decimalnom brojnom sistemu. Nije potrebno raditi proveru ulaznih podataka.

```
PROGRAM hex(Input, Output);
TYPE HexDigits = ARRAY[1..3] OF char;
VAR prvi, drugi: HexDigits;
    brPrvi, brDrugi, i: integer;

FUNCTION HEX_U_DEC(cif : integer; hc : HexDigits) : integer;
VAR i, dec : integer;
BEGIN
   dec := 0;
   FOR i := 1 TO cif DO
        IF hc[i] IN ['a'..'f'] THEN
                dec := dec * 16 + ord(hc[i]) - ord('a') + 10
        ELSE IF hc[i] IN ['A'..'F'] THEN
                dec := dec * 16 + ord(hc[i]) - ord('A') + 10
        ELSE
                dec := dec * 16 + ord(hc[i]) - ord('0');
        HEX_U_DEC := dec
END;

FUNCTION saberi(cif1, cif2: integer; hc1, hc2: HexDigits):integer;
BEGIN
        saberi := HEX_U_DEC(cif1, hc1) + HEX_U_DEC(cif2, hc2)
END;

BEGIN
  writeln('Unesite broj cifara, a zatim cifre prvog broja');
  readln(brPrvi);
  FOR i := 1 TO brPrvi DO read(prvi[i]);
  writeln('Unesite broj cifara, a zatim cifre drugog broja');
  readln(brDrugi);
  FOR i := 1 TO brDrugi DO read(drugi[i]);
  writeln('Zbir brojeva je: ', saberi(brPrvi, brDrugi, prvi, drugi))
END.
```

U postavci se zahteva da nizovi koji se učitavaju sa standardnog ulaza mogu imati manje od 3 znaka. Zbog toga je neophodno da se od korisnika najpre traži da unese broj cifara za svaki od brojeva. U ispitnom zadatku su nizovi tačno dužine 3 znaka, pa je prikazano rešenje složenije ali i fleksibilnije.

Sama funkcija koja određuje zbir dva broja u heksadecimalnom zapisu je krajnje jednostavna i koristi pomoćnu funkciju **HEX_U_DEC** (koja nije tražena u postavci ali značajno pojednostavljuje rešenje) koja određuje vrednost jednog heksadecimalnog broja.

S obzirom na to da se kao heksadecimalne cifre vrednosti veće od 9 koriste prvih 6 slova abecede, ostavljena je mogućnost da se te cifre pišu malim ili velikim slovima. Zbog toga funkcija koja vrši konverziju mora posebno da ispituje vrednost znaka koji se obrađuje – da li pripada skupu malih ili velikih slova. Ako se utvrdi da je u pitanju slovo, na redni broj tog slova umanjen za redni broj slova **a** se dodaje 10 (jer heksadecimalna cifra **a** ima vrednost 10). U suprotnom, u pitanju je cifra (od 0 do 9) pa osim vrednosti te cifre nije potrebno ništa dodavati.

Zadatak IZ51

Koje vrednosti ispisuje sledeći program na programskom jeziku Pascal?

```
PROGRAM xy (output);
VAR a, b, c, d, e, f: integer ;
PROCEDURE  P (VAR c: integer; d, e: integer; VAR  f, g: integer);
BEGIN
  a := a + 1;
  b := b + 1;
  c := c + 1;
  d := d + 1;
  e := e + 1;
  f := f + 1;
  g := g + 1
END;

BEGIN
  a:= 1; b:= 1; c:= 1; d:= 1; e:= 1; f:= 1;
  P(b, f, b, d, e);
  writeln (a:3, b:3, c:3, d:3, e:3, f:3)
END.
```

Formalni argumenti potprograma kojima je dodat modifikator VAR uzrokuju drugačije ponašanje prenosa stvarnih argumenata u potprogram. Bez modifikatora VAR, za svaki stvarni argument se prilikom poziva potprograma rezerviše prostor u memoriji u koji se smesti **kopija** vrednosti stvarnog argumenta (prenos po **vrednosti**). Kod takvog načina prenosa, početna vrednost formalnog argumenta je ista kao i vrednost stvarnog argumenta u trenutku poziva potprograma. Međutim, stvarni i formalni argument zauzimaju dve **različite** memorijske lokacije i zbog toga svaka izmena vrednosti formalnog argumenta ostaje **nevidljiva** za ostatak programa (tj. za mesto poziva potprograma).

Sa modifikatorom VAR, **ne pravi se kopija** stvarnog argumenta, već se formalni argument tretira kao alternativno ime za stvarni argument – odnosno radi se o istoj memorijskoj lokaciji (prenos po **referenci**). Svaka izmena vrednosti formalnog argumenta je zapravo izmena vrednosti stvarnog argumenta i samim tim je **vidljiva** za ostatak programa (tj. za mesto poziva potprograma).

U programu koji se razmatra u ovom zadatku postoji dodatna komplikacija: neki od formalnih argumenata potprograma koriste isti identifikator kao i promenljive glavnog programa (tj. globalne promenljive). Takav način imenovanja formalnih argumenata je dozvoljen. Ne postoji mogućnost da Pascal prevodilac pogreši prilikom pristupa promenljivim jer uvek prvo pristupa onim promenljivim koje su deklarisane u najužem kontekstu u odnosu na mesto korišćenja. Na primer, promenljiva **a** nije deklarisana u potprogramu, ali jeste kao promenljiva glavnog programa, pa se nedvosmisleno zna da se u dodeli **a:=a+1** misli na promenljivu **a** iz glavnog programa. Međutim, promenljiva **c** je deklarisana u okviru potprograma i u okviru glavnog programa. Zato se kod dodele **c:=c+1** koristi lokalno **c** (jer je mesto njegove deklaracije bliže mestu korišćenja od deklaracije u glavnom programu).

Kod rešavanja ovakvih zadataka, verovatno je najjednostavnije nacrtati tablicu preslikavanja stvarnih argumenata u formalne: za prenos po vrednosti treba prepisati vrednost stvarnog argumenta; za prenos po referenci treba nacrtati strelicu od formalnog argumenta ka stvarnom. To je prikazano na sledećoj slici.

glavni program	a	b	c	d	e	f
	1	1	1	1	1	1

potprogram	c	d	e	f	g
		1	1		

Prvi formalni argument potprograma je **c** i ima modifikator VAR. Prvi stvarni argument na mestu poziva potprograma je **b**. Prema tome, u okviru potprograma, svako referisanje ka formalnom argumentu **c** je zapravo referisanje stvarnog argumenta **b**. Naredna dva formalna argumenta su **d** i **e** i oni se prenose po vrednosti. Za njih se rezerviše memorijski prostor u koji se upisuju kopije vrednosti stvarnih argumenata. U ovom slučaju to su **f** i **b**: vrednost **f** se upisuje u lokaciju koja odgovara promenljivoj **d** iz potprograma a vrednost **b** u lokaciju **e**. Konačno, poslednja dva formalna argumenta imaju modifikator VAR, pa se oni zapravo odnose na stvarne argumente **d** i **e** iz glavnog programa, respektivno. Da ne bi došlo do konfuzije, polja u kojima bi se normalno upisala vrednost odgovarajućeg argumenta, za VAR argumente je zatamljena. Time se simbolički ukazuje na to da date memorijske lokacije ne postoje, već da se dati argumenti odnose na stvarne argumente na koje ukazuju odgovarajuće strelice.

Kada je jednom formirana ovakva tablica, potrebno je pratiti izvršenje programa i menjati vrednosti u odgovarajućim lokacijama saglasno instrukcijama. Svaki upis u formalni argument prenesen po referenci zapravo treba izvršiti kao upis u stvarni argument.

Sadržaj globalnih promenljivih će se menjati na sledeći način:

```
a  b  c  d  e  f
1  1  1  1  1  1
2                    a := a + 1;
   2                 b := b + 1;
   3                 c := c + 1; - jer je c alternativno ime za b iz glavnog programa
                     d := d + 1; - jer je d lokalna promenljiva
                     e := e + 1; - jer je e lokalna promenljiva
       2             f := f + 1; - jer je f alternativno ime za d iz glavnog programa
         2           g := g + 1  - jer je g algernativno ime za e iz glavnog programa
```

Konačno:

```
a  b  c  d  e  f
2  3  1  2  2  1
```

prema tome, program će ispisati sledeće vrednosti: 2 3 1 2 2 1

Zadatak IZ Februar 2007 – P4

Šta ispisuje sledeći program na programskom jeziku Pascal?

```
PROGRAM Zad4 (output);
VAR a, b: integer;
PROCEDURE p(VAR a: integer);
  PROCEDURE q(VAR x,y:integer);
  VAR a:integer;
  BEGIN
    a:=x; x:=y; y:=a+x
  END;
BEGIN
 a := a + 1; b := b * 2;
 q(b,a);
 write(a, b);
 b := b div 2; a := a - 1
END;
BEGIN
 a := 5;
 b := 1;
 p(b);
 write(a, b)
END.
```

A) 7743
B) 1044
(C) 8853

Zadatak Z105

Sastaviti program na programskom jeziku Pascal za izračunavanje zbira dva vremena zadatih u obliku sata, minuta i sekundi.

```
PROGRAM SabiranjeVremena (input, output);

TYPE Vreme = RECORD
       Sati: integer;
       Minute, Sekunde: 0..59
     END;

VAR
  T1, T2, T: Vreme;

PROCEDURE CitajVreme (VAR T: Vreme); (* Ucitavanje Podataka tipa Vreme *)
BEGIN
  read (T.Sati, T.Minute, T.sekunde)
END;

PROCEDURE PisiVreme (T:Vreme); {Ispisivanje podataka tipa Vreme }
BEGIN
  WITH T DO write (Sati:3, ':',
        (Minute div 10):1, (Minute mod 10):1, ':',
        (Sekunde div 10):1, (Sekunde mod 10):1)
END;

PROCEDURE SaberiVreme(T1, T2:Vreme; var T3:Vreme);
VAR  k: 0.. 119;
BEGIN
  WITH T3 DO
   BEGIN
      k := T1.Sekunde + T2.Sekunde;  Sekunde := (k mod 60);
      k := T1.Minute + T2.Minute + (k div 60); Minute := (k mod 60);
      Sati := T1.Sati + T2.Sati + (k div 60)
   END
END;

BEGIN
  REPEAT
  (* Ucitavanje vremena *)
  CitajVreme (T1); CitajVreme (T2); readln;
  IF T1.Sati >= 0 THEN
    BEGIN
       (* Ispisivanje vremena *)
       writeln;
       write ('t1, t2= '); PisiVreme (T1); write (', ');
       PisiVreme (T2); writeln;
       (* Izracunavanje zbira vremena *)
       SaberiVreme(T1, T2, T);
       (* Ispisivanje rezultata *)
       write ('t1 + t2 ='); PisiVreme (T); writeln
    END
  UNTIL T1.Sati < 0
END.
```

Program ilustruje korišćenje podatka tipa **RECORD** i ključne reči **WITH**, kao i upotrebu operatora celobrojnog deljenja **div** i operatora ostatka **mod**. Prikazano je i formatiranje podatka o vremenu pri ispisu (primer: 1 sat, 2 minuta, 3 sekunde, umesto da bude prikazano kao `1:2:3`, biće prikazano kao `01:02:03`).

Rekurzija

Zadatak Z103

Sastaviti potprogram na programskom jeziku Pascal za izračunavanje binomnog koeficijenta B(n, k) i glavni program za ispisivanje Pascal-ovog trougla za $0 \leq n \leq 12$.

Iterativno rešenje:

```
PROGRAM z103(input,output);
VAR
  i,j,n: integer;
FUNCTION BinKo(n,k: integer): integer;
VAR
  i,b: integer;
BEGIN
  b:=1;
  FOR i:=1 TO k DO BEGIN
    b:=b*n;
    dec(n)
  END;
  FOR i:=k DOWNTO 1 DO b:=b div i;
  BinKo:=b
END;
BEGIN
  write(output,'Unesite n: ');
  read(input,n);
  FOR i:=0 TO n DO BEGIN
    FOR j:=i+1 TO n DO write(output,'   '); (* tri prazna mesta*)
    FOR j:=0 TO i DO write(output,BinKo(i,j):6);
    writeln(output)
  END
END.
```

Rekurzivno rešenje:

```
PROGRAM z103r(input,output);
VAR n, k: integer;
FUNCTION BinKo(n, k: integer): integer;
BEGIN
  IF (k > 0) AND (k < n) THEN
    BinKo := BinKo(n-1, k) + BinKo(n-1, k-1)
  ELSE
    BinKo := 1
END;
```

Glavni program je isti kod oba rešenja.

Zadatak IZ52

Šta izračunava funkcija `calc` za y >= 1?

```
FUNCTION calc(x, y: integer): integer;
BEGIN
  IF (y=1) THEN calc:=x
  ELSE calc:=calc(x,y-1) + x
END;
```

A) $x+y$ B) $x \cdot y$ C) x^y

Odgovor: B

Zadatak IZ Januar 2007 – P4

Šta ispisuje sledeći program na programskom jeziku Pascal?

```
PROGRAM Zad4 (output);
VAR  a, b: integer;
PROCEDURE p(VAR a: integer);
BEGIN
 a := a + 1; b := b * 2;
 if b < 11 then p(a);
 write(a, b);
 b := b div 2; a := a - 1
END;
BEGIN
 a := 5;
 b := 1;
 p(b)
END.
```

A) 515473321
(B) 2222101044
C) 816788

Zadatak IZ 2006-02-01 (Januar 2006 – P5)

Šta ispisuje dati program na programskom jeziku Pascal?

```
PROGRAM P(Output);
VAR a, b : INTEGER;
FUNCTION z (x:INTEGER):INTEGER;
BEGIN
   IF x>=3 THEN z:=x-2 ELSE z:=z(x+3)
END;
BEGIN
   a:=5;
   FOR b:=1 TO a DO BEGIN WRITE(Output, z(b)) END
END.
```

(A) 23123
B) 21212
C) 12012

Zadatak IZ 2006-06-21 (Jun 2006 – P6)

Šta ispisuje sledeći program na programskom jeziku Pascal za 10 učitanih celih brojeva?

```
PROGRAM Rekurzija (Input, Output);
CONST MAX=10;
VAR niz: ARRAY [1..MAX] OF INTEGER; i:INTEGER;
FUNCTION Test (i:INTEGER):INTEGER;
BEGIN
   IF (i = MAX) OR (niz[i] > Test(i+1)) THEN Test:=niz[i]
   ELSE Test:=Test(i+1)
END;
BEGIN
   FOR i:=1 TO MAX DO READ(Input, niz[i]); READLN(Input);
   WRITELN(Output, Test(1))
END.
```

A) poslednji učitani ceo broj

(B) maksimalani ceo broj među učitanim brojevima

C) ceo broj koji se najčešće pojavljuje među učitanim brojevima

Datoteke

UVODNA NAPOMENA

U računarskoj tehnici se generalno pravi razlika između dve vrste datoteka: binarnih i tekst. Između binarnih i tekst datoteka postoji razlika u nameni:

- Tekst datoteka se sastoji od ograničenog skupa simbola kojima se predstavlja tekst proizvoljnog sadržaja. Simboli su: slova abecede (velika i mala), razmak, znakovi interpunkcije, i znaci za formatiranje teksta (tabulacija i prelazak u novi red). Podaci su zapisani tako da su jednostavnom interpretacijom (najčešće ASCII kod) direktno čitljivi od strane čoveka.

- Binarna datoteka se sastoji od podataka koji nisu namenjeni za neposredno čitanje od strane čoveka i mogu biti raznoliki. Generalno sadrži podatke formi u kojoj su oni smešteni u memoriji računara. Tipični primeri binarnih datoteka su izvršni programi (rezultati prevođenja programa u mašinski jezik), digitalne slike, zvuk, itd.

  U odnosu na opšti slučaj, programski jezik Pascal je nešto restriktivniji prema binarnim datotekama jer zahteva da se datoteka sastoji od slogova fiksne dužine. Format svih slogova u jednoj datoteci mora biti istovetan, odnosno binarna datoteka ne trpi raznorodne slogove.

Iako se formalno pravi razlika između binarnih i tekst datoteka, strogo govoreći tekst datoteke su specijalizovana vrsta binarnih datoteka za skladištenje teksta – pomenuti slog iz binarne datoteke je jedan simbol u tekst datoteci. Dakle, tekst datoteka je zapravo binarna datoteka čiji je slog jedan karakter. Specijalizacija tekst datoteka se između ostalog ogleda u tome što kod njih postoji pojam "novog reda", odnosno specijalnog simbola kojim se označava da tekst koji sledi treba ispisati u narednom redu. Sa druge strane binarna datoteka "ne poznaje" pojmove poput oznake za kraj linije ili razmaka jer oni u tom kontekstu nemaju smisla. Zbog toga se nad binarnim datotekama ne mogu primeniti procedure **`readln`** i **`writeln`**, kao ni ispitivanje da li se pri čitanju stiglo do kraja linije uz pomoć **`eoln(ime_dat_promenljive)`**. Imajući ovo u vidu, jasno je da postoji razlika između datoteka deklarisanih kao **`text`** i **`file of char`** – prve su tekstualne, a potonje su binarne tipa **`char`**.

Pošto može biti prazna (veličine 0 bajtova), datoteka uvek čita sa:
```
while NOT eof(ime_dat) do ...
```
Kod tekstualne datoteke, ako se radi obrada znak po znak, uvek se čita sa:
```
while NOT eoln(ime_dat) do ...
```
Datoteka može biti (Turbo Pascal terminologija):

- **`(Closed)`** – *zatvorena:* na samom početku, pre nego se izvrši **`assign`**,
- **`(Closed, 'pera.txt')`** – *zatvorena:* nakon **`assign(tekstdat, 'pera.txt')`**,
- **`(INPUT, 'pera.txt')`** – *otvorena za čitanje*: nakon **`reset(tekstdat)`**,
- **`(OUTPUT, 'pera.txt')`** – *otvorena za pisanje*: nakon **`rewrite(tekstdat)`**,
- **`(OPEN, 'pera.dat')`** – *otvorena za čitanje i pisanje*: nakon **`reset(bin_dat)`** ili **`rewrite(bin_dat)`**,
- **ponovo** **`(CLOSED, 'pera.txt')`** – nakon **`close(tekstdat)`**, što je obavezno da bi ono što je upisano u datoteku moglo da se iskoristi za čitanje, bilo iz istog ili nekog drugog programa. Iako Turbo Pascal dozvoljava da se istoj binarnoj datoteci može pristupati i čitanjem i pisanjem, treba podrazumevati da se datoteci, od otvaranja (**`reset/rewrite`**) do zatvaranja (**`close`**) pristupa samo na jedan način – i to: samo čitanje posle **`reset`** ili samo pisanje posle **`rewrite`**.

Zadatak ZD1

Napisati program koji prepisuje brojeve, prvo iz ulazne tekstualne u binarnu, a potom iz te novoformirane datoteke u izlaznu tekstualnu datoteku.

```
PROGRAM datoteke(ulbin, izbin, ultekst, iztekst);
VAR ulbin, izbin: file OF integer;
    ultekst, iztekst: text;
    broj: integer;
BEGIN
  assign(ultekst, 'ulaz.txt'); { navodimo ime datoteke na disku }
  assign(izbin, 'brojevi.dat'); { podrazumeva se tekuci direktorijum }
  reset(ultekst); { otvaramo tekstualnu datoteku za citanje }
  rewrite(izbin); { pripremamo binarnu datoteku za upis - STARI SADRZAJ SE BRISE!!! }
  WHILE NOT eof(ultekst) DO begin { citamo do kraja datoteke }
    WHILE NOT eoln(ultekst) DO BEGIN {citamo red po red }
      read(ultekst, broj); { citamo broj iz tekstualne datoteke }
      write(izbin, broj) { upisujemo broj u binarnu datoteku }
    END;
    readln(ultekst)
  END;
  close(ultekst); close(izbin); { zatvaramo datoteke }
  assign(ulbin, 'brojevi.dat'); { datoteka koju smo formirali u prvom delu programa }
  assign(iztekst, 'izlaz.txt'); { tekst datoteka koju cemo napraviti u ovom delu }
  reset(ulbin); rewrite(iztekst); { pripremamo za citanje, odnosno za pisanje }
  WHILE NOT eof(ulbin) DO BEGIN
    read(ulbin, broj);
    writeln(iztekst, broj) { ispisujemo jedan broj po jednom redu }
  END;
  close(ulbin); close(iztekst) { zatvaramo datoteke }
END.
```

Ulazni tekst ne mora biti uređen po redovima; izlazni će biti uređen, po jedan broj u jednom redu.

Zadatak Z115

Datoteka MAGACIN o zalihama robe u magacinu se sastoji od zapisa sa sledećim sadržajem:

- šifra robe (ceo broj),
- naziv robe (niz od najviše 30 znakova),
- količina robe (realan broj), i
- jedinična cena.

U datoteci može da postoji više zapisa sa istom šifrom robe koji se nalaze jedan za drugim. Sastaviti program na programskom jeziku Pascal kojim se svi zapisi sa istom šifrom sažimaju u jedan zapis u kojem je količina robe jednaka zbiru količina u svim zapisima sa tom šifrom.

```
PROGRAM z115(input,output,f);
TYPE
  slog=RECORD
    sifra:integer;
    naziv:string[30];
    kolicina,cena:real
  END;
VAR
  f:file OF slog;
  niz:ARRAY[1..100]OF slog;
  a:slog;
  i,j:integer;
BEGIN
  assign(f,'magacin.dat');
  reset(f);
  i := 0;
  WHILE NOT(eof(f)) DO BEGIN
    read(f,a);
    IF ((i=0)OR(a.sifra<>niz[i].sifra)) THEN
    BEGIN
      i := i + 1;
      niz[i]:=a
    END
    ELSE
      niz[i].kolicina:=niz[i].kolicina+a.kolicina
  END;
  close(f);
  assign(f,'magacin.dat');
  rewrite(f);
  FOR j:=1 TO i DO write(f,niz[j]);
  close(f)
END.
```

Zadatak Z114

Binarna datoteka studenata za svakog studenta sadrži ime studenta, broj indeksa, profil, semestar, broj polaganih ispita i podatke o ispitima. Broj indeksa se sastoji od registarskog broja i godine upisa. Podaci o ispitima su šifra predmeta, datum polaganja i ocena. Datum se sastoji od dana, meseca i godine. Sastaviti program na programskom jeziku Pascal za formiranje nove datoteke od podataka onih studenata čija je srednja ocena položenih ispita najmanje 8.

```
PROGRAM ocena(DatStudent,Prosek_8);
TYPE
  ZapStudent = RECORD
      Ime:ARRAY [1..30] OF char;
      BrIndeksa:
          RECORD
            RegBroj:1..9999;
            GodUpisa:1975..2010
          END;
      Profil:ARRAY [1..3] OF char;
      Semestar:1..10;
      BrOcena:0..50;
      Ocene:ARRAY[1..50] OF
            RECORD
              Predmet:ARRAY [1..5] OF char;
              Datum:RECORD
                  Dan:1..31;
                  Mesec:1..12;
                  Godina:1975..2010
                END; (*Datum*)
              Ocena:5..10
            END (* Ocene *)
    END; (* ZapStudent *)
VAR
  student: ZapStudent;
  dat_student, prosek_8: file OF ZapStudent;
  i, n: integer;
  prosek: real;
BEGIN
  assign(dat_student,'student.dat'); { ovo je potrebno kad se koristi Turbo Pascal }
  assign(prosek_8,'student8.dat');   (* povezuje promenljivu sa datotekom na disku *)
  reset(dat_student);
  rewrite(prosek_8);
  WHILE NOT eof(dat_student) DO BEGIN
      read(dat_student,student);
      (*umesto ove naredbe u zbirci stoji naredba koja ne radi ako se koristi
      Turbo Pascal: student:=dat_student^;*)
      prosek:=0;
      n:=0;
      FOR i:=1 TO student.BrOcena DO
        IF student.Ocene[i].Ocena>5 THEN BEGIN
            prosek:=prosek+student.Ocene[i].Ocena;
            n:=n+1
          END;
      IF n>0 THEN BEGIN
          prosek:=prosek/n;
          IF prosek>=8 THEN
            (*umesto ove naredbe u zbirci stoji prosek_8:=student; put (prosek_8) *)
            write(prosek_8,student)
        END;
  END
  close(dat_student);
  close(prosek_8);
END.
```

U rešenju ovog zadatka su prikazana oba načina za pisanje komentara u programskom jeziku Pascal.

Zadatak IZ 2006-02-01 (Januar 2006)

Data je ulazna tekstualna datoteka *Brojevi.txt*, u kojoj se u proizvoljnom broju redova nalaze pozitivni celi brojevi (u svakom redu ima bar jedan broj). Napisati program na programskom jeziku Pascal koji formira novu datoteku *Izlaz.txt* prepisujući brojeve iz ulazne datoteke uz očuvanje redosleda (poretka pojavljivanja), tako da se svaki broj nalazi u zasebnom redu. U izlaznu datoteku se prepisuju samo oni brojevi koji zadovoljavaju uslov da njihova vrednost nije veća od 10% sume vrednosti svih do tada pročitanih brojeva.

```
PROGRAM P1_Jan06_2(ulaz, izlaz);
VAR
  ulaz, izlaz: text;
  suma, broj: integer;
BEGIN
  assign(ulaz, 'Brojevi.txt');
  reset(ulaz);

  assign(izlaz, 'Izlaz.txt');
  rewrite(izlaz);

  suma := 0;
  WHILE NOT eof(ulaz) DO BEGIN
    WHILE NOT eoln(ulaz) DO BEGIN
      read(ulaz, broj);
      suma := suma + broj;
      IF broj <= suma / 10
        THEN writeln(izlaz, broj)
    END;
    readln(ulaz)
  END;
  close(ulaz);
  close(izlaz)
END.
```

Zadatak Z116

Sastaviti program na programskom jeziku Pascal za određivanje srednje dužine reči u tekst datoteci "roman.txt". Smatrati da se u datoteci nalaze isključivo slova i blanko znaci (razmak, tabulacija, novi red).

```
PROGRAM DuzinaReci(output,datoteka);
FUNCTION SrDuzRec(ime: String):real;
VAR
  datoteka:text;
  BrRec, ZbDuz:integer;
  prvi:boolean;
  znak:char;
BEGIN
  assign(datoteka, ime); (* ovo je potrebno kad se koristi Turbo Pascal *)
  reset(datoteka);
  BrRec:=0;
  ZbDuz:=0;
  WHILE NOT eof(datoteka) DO
    BEGIN
      prvi:=true;
      WHILE NOT eoln(datoteka) DO
        BEGIN
          read(datoteka,znak);
          IF znak=' '
          THEN prvi:=true
          ELSE
            BEGIN
              ZbDuz:=ZbDuz+1;
              IF prvi THEN BrRec:=BrRec+1;
              prvi:=false
            END
        END;
      readln(datoteka)
    END;
  close(datoteka);
  SrDuzRec:=ZbDuz/BrRec
END;
BEGIN
  writeln('Srednja duizina reci u datoteci roman.txt je',SrDuzRec('roman.txt'):5:3)
END.
```

Zadatak Z117

Sataviti program na programskom jeziku Pascal kojim se sadržaj datoteke, u kojoj se nalazi tekst na govornom jeziku, prepiše u drugu datoteku uz pretvaranje početnih slova rečenica u velika slova, a svih ostalih slova u mala slova. Uređenost teksta u redove treba da se očuva.

```
PROGRAM recenice(ulaz,izlaz);
VAR
  ulaz,izlaz:text;
  znak:char;
  prvi:boolean;

BEGIN
  assign(ulaz,'ulaz.txt');
  reset(ulaz);
  assign(izlaz,'izlaz.txt');
  rewrite(izlaz);
  prvi:=true;
  WHILE NOT eof(ulaz) DO
    BEGIN
      WHILE NOT eoln(ulaz)DO
        BEGIN
          read(ulaz,znak);
          IF znak IN ['A'..'Z'] THEN
            IF NOT prvi THEN
              znak:=chr(ord(znak)+ord('a')-ord('A'))
            ELSE
            prvi:=false
          ELSE
            IF znak IN['a'..'z']
            THEN
            BEGIN
              IF prvi THEN
              BEGIN
                znak:=chr(ord(znak)+ord('A')-ord('a'));
                prvi:=false
              END
            END
            ELSE
              IF znak IN['.','!','?'] THEN prvi:=true;
          write(izlaz,znak)
        END;
      readln(ulaz);
      writeln(izlaz)
    END;
  close(ulaz);
  close(izlaz)
END.
```

Zadatak IZ 2006-09-20 (Septembar 2006) – Modifikacija

Napisati program na jeziku Pascal za određivanje namirnica koje se mogu kupiti u prodavnicama makrobiotičke ishrane, a čiji je procentualni udeo kalcijuma veći od zadate vrednosti. Podaci o namirnicama se nalaze u tekst datoteci **namirnice.txt**. U svakom redu datoteke se navode podaci za jednu vrstu namirnica prema sledećem formatu: naziv namirnice (jedna reč, niz od najviše 30 karaktera), masa pakovanja izražena u gramima (realan broj), masa kalcijuma u datom pakovanju izražena u miligramima (realan broj). Podaci su međusobno razdvojeni blanko znacima. Program treba da sa standardnog ulaza pročita željeni procentualni udeo kalcijuma, a zatim na standardnom izlazu ispiše podatke (naziv i procentualni udeo kalcijuma) za one namirnice koje zadovoljavaju zadati uslov, u zasebnom redu za svaku namirnicu.

```
PROGRAM Z2_0610(INPUT,OUTPUT,NAMIRNICE);
TYPE
      NAMIRNICA = RECORD
            NAZIV:STRING[30];
            MASA:REAL;
            MASA_CA:REAL;
      END;
VAR
      NAMIRNICE:TEXT;
      NAM:NAMIRNICA;
      PROCENAT_CA,PROC:REAL;

BEGIN
      ASSIGN(NAMIRNICE,'NAMIRNICE.TXT');
      RESET(NAMIRNICE);

      WRITELN(OUTPUT, 'UNESITE % CA:');
      READLN(INPUT,PROCENAT_CA);

      WHILE NOT EOF(NAMINIRCE) DO
      BEGIN
            READLN(NAMIRNICE,NAM.NAZIV,NAM.MASA,NAM.MASA_CA);
            PROC:=(NAM.MASA_CA*0.1)/NAM.MASA;
            IF PROC > PROCENAT_CA THEN
                  WRITELN(OUTPUT, NAM.NAZIV,PROC);
      END;

      CLOSE(NAMIRNICE);
END.
```

Napomena: u originalnom zadatku se tražilo da se podaci najpre učitaju u dinamički alociranu memoriju (oblast koja će kasnije biti obrađena). U ovom rešenju se podaci ne smeštaju najpre u memoriju već se odmah nakon čitanja ispisuju traženi podaci ako je uslov zadovoljen. Studentima se preporučuje da nakon proučavanja oblasti dinamičke alokacije memorije sami urade ovaj zadatak tako što će najpre podatke smestiti u ulančanu listu a zatim ispisati tražene podatke i osloboditi listu iz memorije.

Pravljenje posebnog tipa podataka **NAMIRNICA** nije neophodno ali je iskorišćeno zbog elegancije rešenja (videti napomenu narednog zadatka).

Zadatak IZ 2006-06-21 (Jun 2006)

Na programskom jeziku Pascal napisati program koji vrši analizu podataka prikupljenih od automatske meteorološke stanice. Podaci se prikupljaju svakog punog sata i smeštaju se u tekst datoteku *vracar.txt*. Svaki red u datoteci sadrži sledeće informacije (odvojene jednim ili više blanko znakova): vreme merenja (ceo broj između 0 i 23), temperatura (realan broj), pritisak (realan broj), vlažnost vazduha (realan broj). Poznato je da temperaturni senzor ima fabrički defekt pa povremeno vrši loše očitavanje i vraća vrednost -50.0. Takođe je poznato da postoje problemi u komunikaciji sa stanicom, tako da je moguće da u trenutku obrade nisu prikupljene sve informacije (za svih 24 časa) te datoteka ne mora da sadrži tačno 24 linije sa podacima. Podaci koji su prikupljeni su smešteni u datoteku po rastućim vrednostima za vreme. Program treba da pročita datoteku sa prikupljenim podacima i odredi minimalnu, maksimalnu i prosečnu dnevnu temperaturu i ispiše ih na standardnom izlazu. Loša očitavanja ignorisati (ne uzimati u obzir temperaturu -50.0). Na kraju treba izvestiti da li su detektovani problemi u komunikaciji sa automatskom meteorološkom stanicom.

```
PROGRAM P1_Jun06_1(Ulaz, Output);
CONST
  DefektTemp = -50.0;
  BrojPod = 24;

TYPE     Info = RECORD
           vreme: 0..23;
           temp, pv, vv: real
         END;
VAR
  br_pod: integer;
  pod: Info;
  Ulaz: text;
  mintemp, maxtemp, sr_temp: real;
BEGIN
  br_pod := 0;
  assign(Ulaz, 'vracar.txt');
  reset(Ulaz);

  WHILE NOT eof(Ulaz) DO BEGIN
    readln(Ulaz, pod.vreme, pod.temp, pod.pv, pod.vv);
    IF pod.temp <> DefektTemp THEN BEGIN
      IF br_pod = 0 THEN
      BEGIN
          mintemp := pod.temp; maxtemp := pod.temp; sr_temp := 0
      END
      ELSE
      BEGIN
          IF pod.temp > maxtemp THEN maxtemp := pod.temp;
          IF pod.temp < mintemp THEN mintemp := pod.temp
      END;
      sr_temp := sr_temp + pod.temp;
      br_pod := br_pod + 1;
    END {IF}
  END; {WHILE}

  IF br_pod <> 0 THEN
  BEGIN
    sr_temp := sr_temp / br_pod;
    writeln ('min: ',mintemp:7:5,', max: ', maxtemp:7:5,', sr: ', sr_temp:7:5);
  END;

  IF br_pod <> BrojPod
    THEN writeln('Detektovana greška u komunikaciji sa meteoroloskom stanicom.');

  close(Ulaz)
END.
```

Prikazano rešenje čita podatke prema zadatom formatu iz tekst datoteke. Studentima se preporučuje da za vežbu da izmene program tako da čita iz binarne datoteke. **Obavezno** proveriti da li se program ispravno izvršava (biće potrebno napisati poseban program koji formira binarnu datoteku).

Nasuprot onome što bi se očekivalo na osnovu postavke zadatka, informacije iz datoteke se ne smeštaju u niz da bi posle bile analizirane već se odmah obrađuju nakon čitanja. Naravno da nije pogrešan pristup kojim se najpre čitaju svi podaci u niz a zatim se obrađuju, ali je to u ovom slučaju očigledno suvišno.

U rešenju je takođe suvišno čitanje vazušnog pritiska i vlage (jer se ti podaci ne koriste). Suvišna je i upotreba zapisa za izveštaj, kada se zapravo koriste samo dva podatka (vreme i temperatura). To je ipak obavljeno zbog kompletnosti i elegancije rešenja. U slučaju da se koristi binarna datoteka, ovaj zapis bi bio neophodan.

Treba obratiti pažnju na način kako je izvršeno određivanje minimalne i maksimalne temperature: ispituje se broj učitanih podataka. Ako je taj broj 0, to znači da je upravo pročitan prvi validan podatak pa on ujedno predstavlja minimalnu i maksimalnu temperaturu. Za sve ostale validne podatke mora se izvršiti provera da li je u pitanju ekstremna vrednost (minimum ili maksimum).

Zadatak IZ 2006-06-21 (Jun 2006)

Napisati program na programskom jeziku Pascal koji iz tekst datoteke *sfere.txt* čita podatke o sferama i određuje da li se dve uzastopno zapisane sfere presecaju, dodiruju ili ne dodiruju. U svakom redu datoteke su smešteni podaci za jednu sferu, razdvojeni blanko znacima. Podaci za sferu su četiri realna broja. Prva tri su koordinate centra sfere (X Y Z), a četvrti je poluprečnik sfere. Program treba da čita podatke iz datoteke i ispiše odgovarajuću poruku (presecaju se, dodiruju se ili ne dodiruju se) u vezi međusobnog odnosa svake dve uzastopne sfere (prve i druge, druge i treće, treće i četvrte, itd). U datoteci postoje podaci za najmanje dve sfere. Nije potrebno proveravati ispravnost sadržaja datoteke.

```
PROGRAM SFERE(input, output);
TYPE sfera = RECORD
       R,X,Y,Z:REAL;
       END;
VAR
  f:text;
  s1,s2:sfera;
  d,r:REAL;
  b:INTEGER;

BEGIN
  assign(f, 'sfere.txt');
  reset(f);
  b:=1;
  readln(f,s1.X,s1.Y,s1.Z,s1.R);
  WHILE NOT EOF(f) DO
  BEGIN
    readln(f,s2.X,s2.Y,s2.Z,s2.R);
    d:=sqr(s1.X-s2.X)+sqr(s1.Y-s2.Y)+sqr(s1.Z-s2.Z);
    r:=sqr(s1.R)+sqr(s2.R);
    write('Sfere ', b, ' i ', b+1, ' se ');
    if d>r then writeln(output, 'ne dodiruju')
    else if d=r then writeln(output, 'dodiruju')
    else writeln(output, 'seku');
    s1:=s2;
    b:=b+1
  END
  close(f);
END.
```

Dinamička alokacija memorije

Zadatak Z123

Sastaviti program na programskom jeziku Pascal koji učitava niz celih brojeva iz proizvoljnog broja redova i posle ih ispisuje po suprotnom redosledu. Kraj niza se obeležava praznim redom. Koristiti dinamičke strukture podataka.

```
PROGRAM NizBrojeva(input,output);
const
  DuzinaReda=10; (*broj podataka koji se ispisuju u jednom redu*)
  BrojKolona=7;  (*broj kolona za ispisivanje jednog podatka*)
TYPE
  pokazivac=^ElementListe;
  ElementListe=RECORD
        broj:integer;
        sledeci:pokazivac
      END;
VAR
  poslednji: pokazivac;      (*poslednji ucitani podatak*)
  novi: pokazivac;           (*novi element u dinamickoj memoriji*)
  prvi: pokazivac;           (*prvi podatak za ispisivanje*)
  prazan_red: boolean;       (*indikator praznog reda*)
  broj_podataka: integer;    (*brojac podataka radi prelaska u novi red*)

BEGIN
  poslednji:=NIL;
  REPEAT
    prazan_red:=true;
    write(output, 'Unesite sledeci broj:');
    WHILE NOT eoln DO
      BEGIN
        new(novi);                  (*dodela prostora novom elementu*)
        read(novi^.broj);           (*ucitavanje podatka u novi element*)
        novi^.sledeci:=poslednji;   (*ukazivanje na prethodni element*)
        poslednji:=novi;            (*ukazivanje na novi kraj niza*)
        prazan_red:=false           (*red nije prazan - nije kraj niza podataka*)
      END;
    readln
  UNTIL prazan_red;
  broj_podataka:=0;
  prvi:=poslednji;    (*pocetak ispisivanja, ukazivanje na poslednji ucitani podatak*)
  WHILE prvi<>NIL DO
    BEGIN
      write(output, prvi^.broj:BrojKolona);        (*ispisivanje podatka*)
      prvi:=prvi^.sledeci;                  (*ukazivanje na sledeci element*)
      broj_podataka := broj_podataka+1;        (*brojanje podatka i prelaz u *)
      IF (broj_podataka mod DuzinaReda)=0 THEN writeln(output) (*sledeci red po potrebi*)
    END;
  (*prelazak u novi red ako poslednji nije ceo*)
  IF (broj_podataka mod DuzinaReda)<>0 THEN writeln(output);

  (*dealociranje memorije*)
  (*OBAVEZNO URADITI NA ISPITU !!! *)
  prvi:=poslednji;
  WHILE prvi<>NIL DO
  BEGIN
    poslednji:=prvi^.sledeci;
    dispose(prvi);
    prvi:=poslednji
  END
END.
```

Zadatak IZ54

Kvadratna matrica se predstavlja pomoću lančane strukture u kojoj za svaki element matrice postoji jedan čvor koji sadrži vrednost elemenata (**broj**), pokazivač na naredni element u istoj koloni (**dole**) i pokazivač na naredni element u istoj vrsti (**desno**). Pod pretpostavkom da je argument pokazivač na gornji levi element matrice, priložena funkcija na programskom jeziku Pascal određuje zbir:

A) svih elemenata matrice
B) elemenata prve vrste matrice
C) elemenata na glavnoj dijagonali matrice

```
PROGRAM matrica(input,output);
TYPE
  ukaz=^matr;
  matr=RECORD
         broj:real;
         dole,desno:ukaz
  END;

FUNCTION XYZ(prvi:ukaz):real;
BEGIN
  IF prvi<>NIL THEN
    IF prvi^.desno<>NIL THEN
      XYZ:=prvi^.broj+XYZ(prvi^.desno^.dole)
    ELSE
      XYZ:=prvi^.broj
  ELSE
    XYZ:=0.0
END;

BEGIN
END.
```

Odgovor: C)

Zadatak IZ Februar 2007 – P5

Koja od priloženih funkcija ispravno ispisuje jednostruko ulančanu listu i kao rezultat vraća broj elemenata liste kada se pozove sa stvarnim argumentom koji pokazuje na prvi element liste celih brojeva? U programu postoje i sledeće definicije:

```
TYPE pok = ^elem;
elem = RECORD broj: integer; sled: pok END.
```

A)
```
function ispis(prvi: pok): integer;
  begin
    if prvi^.sled=nil
    then begin writeln(output, prvi^.broj); ispis := 1 end
    else ispis := ispis(prvi^.sled)
  end;
```

B)
```
function ispis(prvi: pok): integer;
  begin
    if prvi=nil then ispis := 0 else ispis := 1+ispis(prvi^.sled);
    writeln(output prvi^.broj)
  end;
```

(C)
```
function ispis(prvi: pok): integer;
   begin
     if prvi<>nil
        then begin writeln(output, prvi^.broj); ispis := 1 + ispis(prvi^.sled) end
     else ispis := 0
   end;
```

Zadatak IZ Januar 2007 – P5

Koji od navedenih programskih segmenata na programskom jeziku Pascal na ispravan način u dvostruko ulančanu listu iza elementa na koji pokazuje pokazivač `pok`, a ispred narednog elementa, ubacuje element na koji ukazuje pokazivač `pom`? Pretpostaviti da su `pok`, `pom` i `pok^.sled` različiti od `nil`. Polje `pred` ukazuje na prethodni, a polje `sled` na sledeći element liste.

(A)
```
pom^.pred := pok;
pom^.sled := pok^.sled;
pok^.sled := pom;
pom^.sled^.pred := pom;
```

B)
```
pok^.sled := pom;
pom^.pred := pok;
pom^.sled := pok^.sled;
pom^.sled^.pred := pom;
```

C)
```
pom^.sled := pok;
pom^.pred := pok^.pred;
pok^.pred^.sled := pom;
pok^.pred := pom;
```

Zadatak IZ 2006-06-21 (Jun 2006 – P5)

Šta ispisuje sledeći program na programskom jeziku Pascal?

```
PROGRAM Prog(Output);
TYPE num = ^integer;
VAR p, q, r, s, t: num;
PROCEDURE rotate(x, y:num; VAR z:num);
BEGIN
  t:=x; x:=y; y:=z; z:=t
END;
```

```
BEGIN
  new(p); p^:= 1;
  new(q); q^:= 2;
  new(r); r^:= 3;
  new(s); s^:= 4;
  new(t); t^:= 5;
  rotate(r, p, q); rotate(r, s, t);
  writeln(p^:1, q^:1, r^:1, s^:1, t^:1);
END.
```

(A) 13343 B) 12141 C) 13333

Zadatak Z125

Sastaviti potprogram na programskom jeziku Pascal kojim se iz niza celih brojeva, koji je predstavljen u obliku linearne jednostruko ulančane liste, izostave svi elementi koji imaju neku zadatu vrednost.
Napomena: ovde su navedeni samo deklarativni deo programa i glavni program. Ostatak se ne razlikuje u odnosu na zbirku.

```
PROGRAM Izostavljanje(input,output);
TYPE
  ukaz = ^elem;
  elem =
    RECORD
      broj:integer;
      sledeci:ukaz
    END;
VAR
  lista,prvi: ukaz;
  k: integer;

PROCEDURE Pisi(niz:ukaz);(*ispisuje listu u jednom redu*)
VAR prvi:ukaz;
BEGIN
  prvi:=niz;
  WHILE prvi<>NIL DO
    BEGIN
      write(prvi^.broj:1,' ');
      prvi:=prvi^.sledeci
    END;
  writeln
END;
```

```
PROCEDURE Izost(VAR niz:ukaz; k:integer); (*Izostavlja zadate elemente*)
VAR preth,tekuci,stari:ukaz;
BEGIN
  tekuci:=niz;
  preth:=NIL;
  WHILE tekuci<>NIL DO
    IF tekuci^.broj=k THEN
      BEGIN
        stari:=tekuci;
        tekuci:=tekuci^.sledeci;
        IF preth=NIL
          THEN niz:=tekuci
          ELSE preth^.sledeci:=tekuci;
        dispose(stari)
      END
    ELSE
      BEGIN
        preth:=tekuci;
        tekuci:=tekuci^.sledeci
      END
END;

PROCEDURE Citaj(VAR niz:ukaz);(*ucitava niz iz jednog reda i pravi listu*)
VAR posl,novi:ukaz;
BEGIN
  niz:=NIL;
  posl:=NIL;
  WHILE NOT eoln DO
  BEGIN
    new(novi);
    read(input, novi^.broj);
    novi^.sledeci:=NIL;
    IF posl=NIL
      THEN niz:=novi
      ELSE posl^.sledeci:=novi;
    posl:=novi
  END;
  readln
END;

BEGIN
  WHILE NOT eoln DO
    BEGIN
      Citaj(lista);
      write(output, 'ucitani niz je: ');
      Pisi(lista);
      readln(k);
      writeln(output, 'Izostavlja se = ',k:1);
      Izost(lista,k);
      write(output, 'novi niz je: ');
      Pisi(lista);
      (* dealociranje memorije *)
      (* OBAVEZNO URADITI NA ISPITU!!! *)
      prvi:=lista;
      WHILE prvi<>NIL DO
        BEGIN
          lista:=prvi^.sledeci;
          dispose(prvi);
          prvi:=lista
        END
    END
END.
```

Zadatak IZ 2005-06-22 (Jun 2005)

Data je binarna datoteka *vlasnici.dat* koja je formirana na osnovu informacija o vlasnicima kućnih ljubimaca. Svaki zapis u datoteci sadrži ime i prezime vlasnika (maksimalno 30 karaktera), ime kućnog ljubimca (maksimalno 20 karaktera) i identifikacioni zapis. Identifikacioni zapis sadrži šifru ljubimca (ceo broj) i jedan karakter koji označava vrstu ljubimca ('P' za pse, 'M' za mačke i 'O' za ostale vrste). Napisati program na programskom jeziku Pascal koji na osnovu sadržaja navedene datoteke formira listu samo onih zapisa koji se odnose na pse. Zatim je potrebno sa standardnog ulaza čitati šifre izgubljenih pasa sve dok se ne unese negativna vrednost. Za svaku učitanu šifru izgubljenog psa, potrebno je ispisati informacije o vlasniku na standardnom izlazu. Podrazumeva se da datoteka *vlasnici.dat* već postoji.

```
PROGRAM JUN05(Input, Output, ulaz);
TYPE ZAPIS = RECORD
     IMEVL:ARRAY[1..30] OF CHAR;
     IME:ARRAY[1..20] OF CHAR;
     ID: RECORD
       SIFRA:INTEGER;
       VRSTA:CHAR;
     END;
  END;

  LISTPOK = ^LISTZAPIS;
  LISTZAPIS = RECORD
     zap:ZAPIS;
     sled:LISTPOK
  END;

VAR  ulaz:FILE OF ZAPIS;
     glava, pok:LISTPOK;
     sifra:INTEGER;

PROCEDURE FORMIRAJ_LISTU(var glava: LISTPOK);
var  novi, posl : LISTPOK;
     podat : ZAPIS;
begin
  assign(ulaz, 'vlasnici.dat');
  reset(ulaz);
  posl:=nil; glava:=nil;
  while not eof(ulaz) do
  begin
     read(ulaz, podat);
     if podat.ID.vrsta = 'p' then
     begin
        new(novi);
        novi^.sled = nil;
        novi^.zap = podat;
        if glava = nil then
                  glava := novi
        else
          posl^.sled := novi;
        posl := novi

     end
  end;
  close(ulaz)
end;
```

```
PROCEDURE OBRISI_LISTU(var glava:LISTPOK);
var tek:LISTPOK;
begin
  while glava <> nil do
  begin
     tek := glava;
     glava := glava^.sled;
     dispose(tek)
  end
end;

FUNCTION NADJI_SIFRU(glava:LISTPOK, sifra:INTEGER):LISTPOK;
var nasao:boolean;
begin
  nasao:=false; NADJI_SIFRU:=nil;
  while not nasao and glava <> nil do
    if glava^.zap.ID.sifra = sifra then nasao:=true
    else glava:=glava^.sled
  end;
  NADJI_SIFRU := glava
end;

begin
  FORMIRAJ_LISTU(glava);
  if glava <> nil then
  begin
    sifra := 1;
    while sifra > 0 do
    begin
      read(sifra);
      if sifra > 0 then
      begin
        pok:=NADJI_SIFRU(glava, sifra);
        if pok <> nil then
          writeln(pok^.zap.IMEVL)
      end
    end;
    OBRISI_LISTU(glava)
  end
end.
```

Zadatak kombinuje oblasti dinamičke alokacije memorije i datoteka. S obzirom na to da je glavna obrada smeštena u potprograme, glavni program je jednostavan za praćenje: najpre se formira lista zapisa i odmah proveri da li je lista prazna. Ako nije, sve dok se unosi šifra vrednosti veće od 0 vrši se pretraga liste zapisa na datu šifru i ako je zapis sa datom šifrom nađen, ispisuju se informacije iz zapisa. Studentima se preporučuje da napišu poseban potprogram **OBRADA** koji vrši glavnu obradu, tako da se glavni program sastoji od tri uzastopna poziva potprograma: **FORMIRAJ_LISTU**, **OBRADA**, **OBRISI_LISTU**.

Zadatak Z127

Sastaviti potprograme na programskom jeziku Pascal za izmenu redosleda elemenata (tj. za zamenu prvog sa poslednjim, drugog sa pretposlednjim, itd.) zadatog niza realnih brojeva koji je predstavljen u obliku jednostruko i u obliku dvostruko ulančane liste. Napisati glavni program čiji iskazni deo ne obavlja nikakve operacije.

Napomena: priloženo rešenje se razlikuje od onog u zbirci. Oba su ispravna, a studentima se preporučuje da prouče oba rešenja i porede razlike.

```
PROGRAM z127(input,output);
TYPE
  pok1=^slog1;
  slog1=RECORD
    broj:real;
    sledeci:pok1
  END;
  pok2=^slog2;
  slog2=RECORD
    broj:real;
    prethodni,sledeci:pok2
  END;
VAR
  p1:pok1;
  p2:pok2;
PROCEDURE form(VAR p1:pok1;VAR p2:pok2);
VAR
  n,i:integer;
  t11,t12:pok1;
  t21,t22:pok2;
BEGIN
  write(output,'Unesite broj clanova niza: ');
  read(input,n);
  IF (n>0) THEN BEGIN
    new(t11);
    new(t21);
    p1:=t11;
    p2:=t21;
    writeln(output,'Unesite clanove niza:');
    read(input,t11^.broj);
    t21^.broj:=t11^.broj;
    t21^.prethodni:=NIL;
    FOR i:= 2 TO n DO BEGIN
      new(t12);
      new(t22);
      t11^.sledeci:=t12;
      t21^.sledeci:=t22;
      t22^.prethodni:=t21;
      read(input,t12^.broj);
      t22^.broj:=t12^.broj;
      t11:=t12;
      t21:=t22
    END;
    t11^.sledeci:=NIL;
    t21^.sledeci:=NIL
  END
  ELSE BEGIN
    p1:=NIL;
    p2:=NIL
  END
END;
...
```

```
...
PROCEDURE izmeni1(VAR p1:pok1);
VAR
  t1,t2,t3:pok1;
BEGIN
  IF (p1<>NIL) THEN BEGIN
    t1:=p1;
    t2:=p1;
    t3:=p1^.sledeci;
    WHILE (t2<>NIL) DO BEGIN
      t2^.sledeci:=t1;
      t1:=t2;
      t2:=t3;
      IF (t3<>NIL) THEN t3:=t3^.sledeci
    END;
    p1^.sledeci:=NIL;
    p1:=t1
  END
END;
PROCEDURE izmeni2(VAR p2:pok2);
VAR
  t1,t2:pok2;
BEGIN
  IF(p2<>NIL)THEN BEGIN
    t1:=p2;
    t2:=p2^.sledeci;
    WHILE(t2<>NIL)DO BEGIN
      t2^.prethodni:=t2^.sledeci;
      t2^.sledeci:=t1;
      t1:=t2;
      t2:=t2^.prethodni
    END;
    p2^.prethodni:=p2^.sledeci;
    p2^.sledeci:=NIL;
    p2:=t1
  END
END;
PROCEDURE ispisiizbrisi(VAR p1:pok1;VAR p2:pok2);
VAR
  t1:pok1;
  t2:pok2;
BEGIN
 WHILE(p1<>NIL) DO BEGIN
   write(output,p1^.broj,' ');
   t1:=p1^.sledeci;
   dispose(p1);
   p1:=t1
 END;
 writeln(output);
 WHILE(p2<>NIL) DO BEGIN
   write(output,p2^.broj,' ');
   t2:=p2^.sledeci;
   dispose(p2);
   p2:=t2
 END
END;
BEGIN
  (* iako se u tekstu zadatka kaze da treba ostaviti prazan glavni program,
     ovde je zbog kompletnosti resenja pokazano kako treba pozivati gornje potprograme *)
  form(p1,p2);
  izmeni1(p1);
  izmeni2(p2);
  ispisiizbrisi(p1,p2)
END.
```

SLOŽENOST ALGORITMA

Često je potrebno, makar okvirno, proceniti vremensku i prostornu složenost algoritma kojim se neki problem rešava. Kako se svaki programerski problem može rešiti primenom više različitih algoritama, korisno je prilikom izbora znati koliko je svaki od odgovarajućih algoritama vremenski, odnosno prostorno, složen. Ovo poglavlje se bavi vremenskom složenošću algoritma.

Pravila za prebrojavanje instrukcija radi utvrđivanja I(n)

- Svaka od instrukcija u sekvenci se broji kao 1, bez obzira na to koliko složen izraz može predstavljati i koliko je stvarno mašinskih instrukcija iskorišćeno za ostvarivanje konkretnog dela sekvence.
- **FOR** se broji kao N+1, ako se telo petlje izvrši N puta,
- **BEGIN** i **END** se ne broje,
- **IF** se broji kao 1, svaki put kad se "postavi pitanje", a ono što **IF** uslovljava se broji onoliko puta koliko se proceni prema konkretnoj situaciji,
- **WHILE** se broji kao N+1, ako se telo petlje izvrši N puta,
- **REPEAT** se ne broji (0), a **UNTIL** se broji onoliko puta koliko puta se izvrši telo petlje, samim tim i sam test izlaska iz petlje.
- sama **PROCEDURE** ili **FUNCTION** instrukcija se ne broji, pošto se u sekvenci iz koje se potprogram poziva poziv datog potprograma već broji kao jedna instrukcija.

Linearno traženje

(**q** je verovatnoća pronalaženja)

`NASAO:=false;`	`1`
`FOR i:=1 TO N DO`	`N+1`
`  IF (NIZ[i]=K) THEN`	`N`
`    NASAO:=true;`	`q (1 ili 0)`

Bubble sort

(**p** je verovatnoća zamene)

`FOR I:=1 TO N-1 DO`	`N`
`  FOR J:=N-1 DOWNTO I DO`	`N + N-1 +…+2`
`    IF A[J]>A[J+1] THEN`	`(N-1) + (N-2) +…+1`
`      BEGIN`	
`        B=A[J]; A[J]=A[J+1]; A[J+1]=B;`	`p x ((N-1) + (N-2) +…+1)`
`      END;`	

Binary search

(brojanje je za najgori slučaj; **q** verovatnoća nalaženja, **p** verovatnoća kretanja "na gore", a **1-p** "na dole" prilikom traženja)

`POCETAK:=1; KRAJ:=N; NASAO:=false;`	`3`
`WHILE NOT NASAO AND (POCETAK <= KRAJ) DO`	$\log_2 N+1$
`  BEGIN`	
`  TEKUCI:=(POCETAK+KRAJ) div 2;`	$\log_2 N$
`  IF NIZ[TEKUCI]>BROJ THEN`	$\log_2 N$
`    KRAJ=TEKUCI-1`	$p \times \log_2 N$
`  ELSE  IF NIZ[TEKUCI]<BROJ THEN`	$(1-p) \times \log_2 N$
`      POCETAK:=TEKUCI+1`	$(1-p) \times \log_2 N - q$
`    ELSE NASAO:=true`	`q (1 ili 0)`
`  END;`	

Bubble sort (kada se koristi flag)
(teško je prebrojati, zato što zavisi od prethodne uređenosti niza)

	min	max
I:=1; GOTOVO:=false;	2	2
WHILE NOT GOTOVO DO	2	N
BEGIN		
GOTOVO:=true;	1	N-1
J:=N-1;	1	N-1
WHILE (J>=I) DO	2	N + N-1 +…+2
BEGIN		
IF A[J]>A[J+1] THEN	N-1	N-1 +…+1
BEGIN		
B=A[J]; A[J]=A[J+1]; A[J+1]=B;	0	3 x (N-1 +…+1)
GOTOVO=false	0	N-1 +…+1
END;		
J=J-1	N-1	N-1 +…+1
END;		
I=I+1	1	N-1
END;		

Zadatak NZ1

Izračunati funkciju složenosti za dati program na programskom jeziku Pascal koji učitava, sortira u neopadajućem poretku i ispisuje niz od najviše 100 realnih brojeva.

```
PROGRAM nz1(input,output);
VAR
  niz:ARRAY[1..100] OF real;
  tmp:real;
  n,i,j:integer;
begin
  read(input,n);
  FOR i:=1 TO n DO read(input,niz[i]);
  FOR i:=1 TO n-1 DO
    FOR j:=i+1 TO n DO
      IF niz[i]>niz[j] THEN BEGIN
        tmp:=niz[i];
        niz[i]:=niz[j];
        niz[j]:=tmp
      END;
  FOR i:=1 TO n DO writeln(output,niz[i]:2:2)
END.
```

read(n);	1
FOR i:=1 TO n DO read(niz[i]);	(n+1)+n=2n+1
FOR i:=1 TO n-1 DO	n
FOR j:=i+1 TO n DO	[(n-1)+1]+[(n-2)+1]+...+2=(n+2)(n-1)/2
IF niz...	(n-1)+(n-2)+...+1=n(n-1)/2
tmp:=niz[i];	n(n-1)/2
niz[i]:=niz[j];	n(n-1)/2
niz[j]:=tmp	n(n-1)/2
FOR i:=1 TO n DO writeln(niz[i]:2:2)	2n+1

$I(n)=1+(2n+1)+n+(n^2+n-2)/2+4(n^2-n)/2+(2n+1)=(5n^2+7n)/2+2$

Napomena: Ovde se usvaja da je **if** uslov uvek ispunjen, što predstavlja najgori slučaj. U prosečnom slučaju, složenost ima manju vrednost konstante (ali red složenosti ostaje isti).

Zadatak NZ2

Izračunati funkciju složenosti za dati program na programskom jeziku pascal koji učitava niz realnih brojeva sortiran po neopadajućem redosledu i u tom nizu pronalazi zadati realni broj metodom binarnog traženja.

```
PROGRAM nz2(input,output);
VAR
  niz:ARRAY[1..100] OF real;
  x:real;
  nasao:boolean;
  prvi,poslednji,tekuci,n,i:integer;
BEGIN
  read(input,n,x);
  FOR i:=1 TO n DO read(input,niz[i]);
  prvi:=1;
  poslednji:=n;
  nasao:=false;
  WHILE ((NOT(nasao)) AND (prvi<=poslednji)) DO BEGIN
    tekuci:=(prvi+poslednji) div 2;
    IF (niz[tekuci]>x) THEN poslednji:=tekuci-1
    ELSE IF (niz[tekuci]<x) THEN prvi:=tekuci+1
    ELSE nasao:=true
  END;
  IF nasao THEN write(output,'Element pronadjen na poziciji ',tekuci,'.')
  ELSE write(output,'Element nije pronadjen.')
END.
```

`read(input,n,x);`	`1`
`FOR i:=1 TO n DO read(input,niz[i]);`	`(n+1)+n=2n+1`
`prvi:=1;`	`1`
`poslednji:=n;`	`1`
`nasao:=false;`	`1`
`WHILE ((NOT(nasao)) AND (prvi<=poslednji)) DO BEGIN`	$\log_2 n+1$
`tekuci:=(prvi+poslednji) div 2;`	$\log_2 n$
`IF (niz[tekuci]>x) THEN poslednji:=tekuci-1`	$\log_2 n+p*\log_2 n$ (p je verovatnoća da je uslov ispunjen)
`ELSE IF (niz[tekuci]<x) THEN prvi:=tekuci+1`	$(1-p)*\log_2 n+[(1-p)*\log_2 n-q]$ (q je verovatnoća nalaženja elementa)
`ELSE nasao:=true`	`q`
`IF nasao THEN write(output,'Element pronadjen na poziciji ',tekuci,'.')`	`1+q`
`ELSE write(output,'Element nije pronadjen.')`	`1-q`

$I(n)=1+(2n+1)+1+1+1+(\log_2 n+1)+\log_2 n+(\log_2 n+p*\log_2 n)+\{(1-p)*\log_2 n+[(1-p)*\log_2 n-q]\}$
$+q+(1+q)+(1-q)=2n+(5-p)\log_2 n+8$

U ovom primeru, postoje dve komponente funkcije složenosti; jedna je linearno, a druga logaritamski zavisna od dimenzije problema. Kako se određivanje reda funkcije složenosti radi za dimenziju problema takvu da teži beskonačnosti, očigledno je da ovaj program ima linearnu vremensku složenost. Sa druge strane, ako zanemarimo unos podataka i posmatramo samo deo koji se bavi binarnim pretraživanjem, onda je jasno da je vremenska složenost logaritamski zavisna od dimenzije problema.

Zadatak Z64.PAS

Odrediti funkciju složenosti i red funkcije složenosti datog programa na programskom jeziku Pascal:

```
PROGRAM zadatak64(input, output);

PROCEDURE strel(n: integer);
VAR
  korak, nzvez, i, j: integer;
  znak: char;
BEGIN
  korak := 1;
  znak := ' ';
  nzvez := 0;
  i := 1;
  WHILE (i<=3*n) DO BEGIN
    nzvez := nzvez+korak;
    FOR j := 1 TO 3*n DO write(output, znak);
    FOR j := 1 TO nzvez DO write(output, '*');
    writeln(output);
    IF i = n
      THEN znak := '*'
    ELSE IF i = 3*n div 2 + 1
      THEN korak := -1
    ELSE IF i = 2*n
      THEN znak := ' ';
    i := i + 1
  END
END;

BEGIN
  strel(7)
END.
```

Ovaj program "crta" trougao na standardnom izlazu, takav da je jedna od ivica trougla vertikalna i pomerena za `3*n` mesta od leve ivice.

`strel(7)`	1
`korak:=1; znak:=' '; nzvez:=0; i:=1;`	4
`WHILE (i<=3*n) DO BEGIN`	3n+1
`Nzvez := nzvez+korak;`	3n
`FOR j := 1 TO 3*n DO write(output, znak);`	$3n(3n+1+3n)=18n^2+3n$
`FOR j := 1 TO nzvez DO write(output, '*');`	$3n+2(3n \text{ div } 2+1)^2$ za neparno n, $(9/2)n^2+9n$ za parno
`writeln(output);`	3n
`IF i = n`	3n
`THEN znak := '*'`	1
`ELSE IF i = 3*n div 2 + 1`	3n-1
`THEN korak := -1`	1
`ELSE IF i = 2*n`	3n-2
`THEN znak := ' ';`	1
`i := i + 1`	3n
`END;`	0

dakle, približno je $I(n)=(45/2)n^2+33n+6$, što znači da je $I(n)=O(n^2)$

Zadatak Z65.PAS

Odrediti funkciju složenosti i red funkcije složenosti datog programa:

```
PROGRAM zadatak65(input, output);

VAR
  n_cif, n_vel, n_mal, n_ost, i: integer;
  linija: STRING[255]; { max. duzina stringa za Turbo Pascal }
  znak: char;

BEGIN
  n_cif := 0; n_vel:= 0; n_mal := 0; n_ost := 0;
  writeln(output, 'Unesite liniju koje treba obraditi: ');
  readln(input, linija);
  FOR i := 1 TO Length(linija) DO
    BEGIN
      znak := linija[i];
      IF (znak >= '0') AND (znak <= '9') THEN
        n_cif := n_cif + 1
      ELSE IF(znak >= 'A') AND (znak <= 'Z') THEN
        n_vel := n_vel + 1
      ELSE IF(znak >= 'a') AND (znak <= 'z') THEN
        n_mal := n_mal + 1
      ELSE
        n_ost := n_ost + 1
    END; { FOR i: 1 TO Length(linija) }

  writeln(output, 'Cifara ima ukupno: ', n_cif);
  writeln(output, 'Velikih slova ima ukupno: ', n_vel);
  writeln(output, 'Malih slova ima ukupno: ', n_mal);
  writeln(output, 'Ostalih znakova ima ukupno: ', n_ost);
  writeln(output, 'Pritisnite [ENTER] za kraj programa');
  readln(input)
END.
```

Program je izveden iz programa koji je prikazan u zadatku Z41.PAS, radi pojednostavljenja prvobitnog zadatka.

`n_cif:=0; n_vel:=0; n_mal:=0; n_ost:=0;`	4
`writeln(output, 'Unesite liniju koje treba obraditi: ');`	1
`readln(input, linija);`	1
`FOR i := 1 TO Length(linija) DO BEGIN`	d+1, d je length(linija)
`  znak := linija[i];`	D
`  IF (znak >= '0') AND (znak <= '9') THEN`	D
`    n_cif := n_cif + 1`	d*p1, p1 je verovatnoća prethodnog uslova
`  ELSE IF(znak >= 'A') AND (znak <= 'Z') THEN`	d(1-p1)
`    n_vel := n_vel + 1`	d(1-p1)p2, p2 je verovatnoća drugog uslova, ako prvi nije ispunjen
`  ELSE IF(znak >= 'a') AND (znak <= 'z') THEN`	d(1-p1)(1-p2)
`    n_mal := n_mal + 1`	d(1-p1)(1-p2)p3, p3 je verovatnoća trećeg uslova, ako prvi i drugi nisu ispunjeni
`  n_ost := n_ost + 1`	d(1-p1)(1-p2)(1-p3)
`END;`	0
`writeln(output, 'Cifara ima ukupno: ', n_cif);`	1
`writeln(output, 'Velikih slova ima ukupno: ', n_vel);`	1
`writeln(output, 'Malih slova ima ukupno: ', n_mal);`	1
`writeln(output, 'Ostalih znakova ima ukupno: ', n_ost);`	1
`writeln(output, 'Pritisnite [ENTER] za kraj programa');`	1
`readln(input)`	1

I(n)=d(6-2p1-p2+p1p2)+13

I(n)=O(d)

Zadatak Z68a.PAS

Odrediti funkciju složenosti i red funkcije složenosti datog programa na programskom jeziku Pascal:

```
PROGRAM zadatak68a(input, output);

VAR
  k, s, i, j, n: integer;

BEGIN
  write(output, 'Unesite broj iteracija: ');
  readln(input, n);
  k := 1;
  s := 0;
  FOR i := 1 TO n DO BEGIN
    FOR j := 1 TO k DO
      s := s + i*j;
    k := k + k;
  END
END.
```

Vrlo često najupadljivija radnja (sabiranje u ovom slučaju) ne doprinosi suštinski redu funkcije složenosti.

`write(output, 'Unesite broj iteracija: ');`	1
`readln(input, n);`	1
`k := 1;`	1
`s := 0;`	1
`FOR i := 1 TO n DO BEGIN`	n+1
`  FOR j := 1 TO k DO`	$1+2+4+8+...+2^{n-1}+n=2^n-1+n$
`    s := s + i*j;`	2^n-1
`  k := k + k;`	n

$I(n)=2^{n+1}+3n+3$

$I(n)=O(2^n)$

Komentar: red funkcije složenosti nije $O(2^{n+1})$ zato što je 2^{n+1} zapravo $2 * 2^n$