1. Napisati Pascal program (NPP) koji će na standardni izlaz ispisati poruku *Zdravo svete.*

Rešenje 1:

```
Program Prvi;
Begin
    writeln('Zdravo svete');
END.
```

Rešenje 2:

```
Program Prvi;
BEGIN
    writeln('Zdravo svete');
END.
```

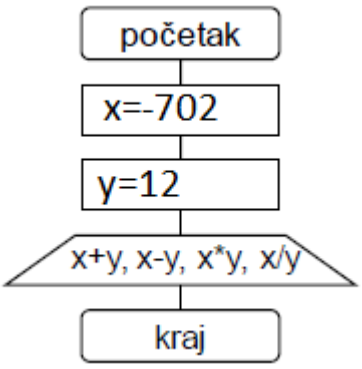
2. Napisati Pascal program (NPP) koji će na standardni izlaz u orvom redu ispisati poruku *Zdravo svete, a* potom u sledećem redu ispisati poruku *Hello, world.*

Rešenje:

```
Program Drugi;
BEGIN
    writeln('Zdravo svete');
    writeln('Hello, world');
END.
```

3. NPP koji će za dva cela broja ispisati njihov zbir, razliku, proizvod i količnik.

Rešenje:




```
Program Zad_3;
Var x,y:integer;
Begin
x:=-702;
y:=12;
write(x+y,x-y,x*y,x/y)
End.
```

Rešenje sa formatiranjem ispisa

```
Program Zad3_1;
Var x,y:integer;
Begin
x:=-702;
y:=12;
write(x+y:10,x-y:10,x*y:10,x/y:10:2)
End.
```

Elementi programskog jezika Pascal

1. **Osnovni simboli: slova, cifre, specijalni znaci**

Tip	Znaci
Slova	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 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
Cifre	0 1 2 3 4 5 6 7 8 9
Posebni znaci	+ - * / := < > = <= >= <> , . : ; .. _ ' [ ] { } (* *) (. .), space, tab, newline

2. Ključne (rezervisane) reči

Ključne reči u **PASCAL**-u

and	**end**	**nil**	**shr**
asm	**file**	**not**	**string**
array	**for**	**object**	**then**
begin	**function**	**of**	**to**
case	**goto**	**or**	**type**
const	**if**	**packed**	**unit**
constructor	**implementation**	**procedure**	**until**
destructor	**in**	**program**	**uses**
div	**inline**	**record**	**var**
do	**interface**	**repeat**	**while**
downto	**label**	**set**	**with**
else	**mod**	**shl**	**xor**

Ključne reči u **PASCAL**-u koje se mogu predefinisati

absolute	**external**	**forward**	**near**
assembler	**far**	**interrupt**	**Private**
virtual			

3. Tipovi podataka: prosti, složeni, pokazivački

Celobrojni tip – Integer

Za rad sa celim brojevima koristi se **celobrojni tip** podataka. Ovaj tip podatka se u programskom jeziku Pascal označava rezervisanom rečju **Integer**.

Primeri celobrojnih vrednosti mogu biti:

28	728	6495	6
-56	+34	-9	-4825

Znak + ispred celobrojne vrednosti nije obavezan. Ukoliko ispred celobrojne vrednosti ne postoji predznak, smatra se da je vrednost pozitivna.

Vrednost najmanjeg i najvećeg celog broja zavisi od sistema na kome se radi, tako da na jednom tipu računara raspon celih brojeva može biti od -32768 do 32767, dok na drugom može biti od -2147483648 do 2147483647.

Da li možda znate...

Zašto je minimalni ceo broj po apsolutnoj vrednosti za jedan veći od maksimalnog celog broja?

Realan tip – Real

Brojevi koji pored celobrojnog sadrže i decimalni deo nazivaju se realnim brojevima. U programskom jeziku Pascal ovi brojevi se označavaju rezervisanom rečju **Real**. Prilikom pisanja realnih brojeva, celobrojni i decimalni deo se razdvajaju **decimalnom tačkom**[1].

Primeri realnih vrednosti mogu biti:

7.6	-4.62	7.584	-3000.0
-565.24	+34.81	-9.0	0.085

Pored predstavljenog načina pisanja realnih brojeva, programski jezik Pascal omogućava i pisanje realnih brojeva sa **pokretnom tačkom**[2] (floating point). Ovakav način pisanja je naročito prikladan za vrlo velike i vrlo male brojeve. Tako na primer broj 32 100 000, može biti zapisan kao 32.1E+6, što je ekvivalentno zapisu $32.1 \cdot 10^6$. Sa druge strane, broj 0.000658 može biti zapisan kao 6.58E-4, što je isto kao i $6.58 \cdot 10^{-4}$.

Ovakvo pisanje realnih brojeva se naziva još i eksponencijalna notacija. Eksponent (broj naveden iza slova "E") označava za koliko mesta treba pomeriti decimalnu tačku. Pozitivan eksponent označava da decimalnu tačku treba pomeriti u desno, dok negativan znak pomera decimalnu tačku u levo. Drugačije gledano, eksponent predstavlja stepen broja 10 kojim treba pomnožiti navedeni realan broj.

Ukoliko je eksponent pozitivan broj, predznak + iza slova E je moguće izostaviti. Takođe, ukoliko realni broj ne sadrži decimalni deo i on se može izostaviti.

Evo nekih primera pisanja realnih brojeva sa pokretnim zarezom:

Eksponencijalni zapis (pokretna tačka)	Standardni zapis
6.3E+4	63000
2.526E2	252.6
4.72E-2	0.0472
782.8E-1	78.28
45E+2	4500
0.25E-5	0.0000025

Neke bitne napomene

Brojevi 342 i 342.0 u matematici imaju istu vrednost, međutim u radu na računaru oni često označavaju različite tipove podataka. Tako, u nekim Pascal kompajlerima 342 predstavlja celobrojni tip, dok će 342.0 biti protumačen kao realni tip podatka. Da biste uvek bili sigurni da je broj koji ste napisali pravilno prtumačen bez obzira na kompajler koji koristite, najbolja praksa je da broj za koji želite da bude tretiran kao realan uvek pišete sa decimalnom tačkom, bez obzira da li on ima ili nema decimalnih cifara.

Neki Pascal kompajleri dozvoljavaju skraćeno pisanje realnih brojeva koji nemaju decimalnih cifara tako da se oni završavaju decimalnom tačkom. Na taj način, iako broj nema decimalnih cifara, kompajleru se stavlja do znanja da je u pitanju realan, a ne celobrojni tip. Tako se broj "342" može zapisati kao "342.", što je ekvivalentno zapisu "342.0". Takođe, ukoliko realan broj nema celobrojnog dela (po apsolutnoj vrednosti je manji od 1), u nekim kompajlerima on se može zapisati bez cifara levo od decimalne tačke. Na primer, broj "0.12" se može zapisati i kao ".12". Iako veliki broj kompajlera podržava ovakve načine zapisivanja realnih brojeva, preporuka je da se uvek koristi neskraćeno zapisivanje kako bi napisani program bio kompatibilan sa svim kompajlerima.

Evo nekih primera pomenutih načina zapisivanja realnih brojeva:

Skraćeni zapis	Potpuni zapis
23.	23.0
.756	0.756
-.5	-0.5
6.E+3	6.0E+3
.48E-3	0.48E-3

Prilikom rada sa realnim brojevima uvek treba imati u vidu da tačnost zavisi od broja cifara pomoću kojih je broj zapisan. Broj značajnih cifara, a samim tim i preciznost računanja, zavisi od broja bajtova koji se koriste za zapisivanje realnih brojeva. Pored toga, operacije nad realnim brojevima u zavisnosti od preciznosti troše više procesorskog vremena od operacija nad celim brojevima.

Logički tip – Boolean

Logički tip podataka se koristi za označavanje istinitosti nekog iskaza i može imati dve vrednosti:

Vrednost	Značenje
false	netačno
true	tačno

U programskom jeziku Pascal se logički tip podatka označava rezervisanom rečju **Boolean**. Vrlo često se logičke vrednosti nazivaju i Boolovim vrednostima u čast engleskog matemaričara George Boole-a, koji je prvi razvio logičku algebru. U računaru se logičke vrednosti predstavljaju jednim bitom koji može imati vrednost 0 za netačno i 1 za tačno.

Znakovni tip – Char

Znakovni tip podataka je uređen skup znakova koji mogu biti:

- slova abecede
- numerički znakovi od 0 do 9
- znakovi interpunkcija
- specijalni znakovi

U programskom jeziku Pascal znakovne vrednosti se zapisuju kao znak između jednostrukih navodnika:

`'A'`	`'b'`	`'X'`	`'4'`	`'+'`	`'-'`	`'>'`	`'@'`

Korišćenje navodnika pri pisanju znakova je neophodno kako bi se oni razlikovali od brojeva, operatora, promenljivih, itd. Na primer, 4 je celobrojna vrednost, a '4' znakovna vrednost.

Nizovi znakova kao što su reči i rečenice se u Pascal-u nazivaju stringovima i označavaju se rezervisanom rečju **String**.

Evo nekih primera stringova:

`'Programiranje'`	`'Goran igra fudbal, a Jana vozi bicikl.'`	`'345.08'`	`'2+5=7'`	`'peraperic@yahoo.com'`

Konstante

Odeljak za definisanje konstanti počinje rezervisanom reči **const** (od eng. reči *constant=konstanta*) iza koje se navode nazivi konstanti i njihovih vrednosti. Konstante i njihove vrednosti se razdvajaju znakom **=**, a definicije pojedinih konstanti znakom **;**. Vrednost konstante u Pascal-u može biti broj (ceo ili realan), string ili prethodno definisana konstanta. Pojedini kompajleri omogućavaju da vrednost konstante bude izraz koji se sastoji od drugih konstanti. Pri definisanju konstante ne navodi se njen tip, pošto je on jednoznačno određen vrednošću koja joj se dodeljuje. Vrednost definisane konstante nije moguće menjati u programu.

Primer

```
const   PI=3.14;
        email='pera@yahoo.com';
        g=9.81;
        metar=1;
        kilometar=1000*metar;
```

4. NPP koji će za dva pozitivna cela broja veća od 100 ispisati njihov zbir, razliku, proizvod i količnik.

```
Program Zad4_1;
Var x,y,r1,r2:integer;
r3:longint;
r4:real; {deklarisanje promenljivih rezultata r1,r2,r3,r4}
Begin
x:=482;
y:=315;
r1:=x+y;
r2:=x-y;
r3:=x*y;
r4:=x/y;
writeln(r1:10,r2:10,r3:10,r4:10:2)
End.
```

5. Da li je sledeći program korektan? Ako jeste, šta je rezultat njegovog rada?

```
Program Zad4_1;
Var x,y,r1,r2, r3, r4:integer;
{deklarisanje CELOBROJNIH promenljivih rezultata r1,r2,r3,r4}
Begin
x:=482;
y:=315;
r1:=x+y;
r2:=x-y;
r3:=x*y;
r4:=x/y;
writeln(r1:10,r2:10,r3:10,r4:10:2)
End.
```

Promenljive

Promenljive ili varijable su objekti čija vrednost može biti promenjena tokom izvršavanja programa. Svaka promenljiva koja se koristi u programu mora biti prethodno deklarisana. Deklaracijom promenljive ne određuje se njena vrednost, već samo tip podatka koji će u njoj biti smešten. Na osnovu navedenog tipa podatka svakoj promenljivoj se dodeljuje memoriski prostor odgovarajuće veličine u kome će biti upisana vrednost promenljive. Dodeljivanje odgovarajućeg memorijskog prostora (alokacija) se obavlja prilikom pokretanja programa ili potprograma, a njegova pozicija i veličina se ne mogu menjati tokom izvršenja, pa se iz tog razloga ovaj način alokacije naziva **statička alokacija**. Pored ovakvog načina dodeljivanja memorije postoji i **dinamička alokacija**, koji omogućava alokaciju memorije određene veličine u bilo kom trenutku tokom izvršavanja programa.

Odeljak za deklarisanje promenljivih počinje rezervisanom reči **var** (od eng. reči *variable=promenljiva*) iza koje se navode nazivi promenljivih i njihovi tipovi. Promenljive istog tipa mogu biti odvojene zapetom, iza čega sledi dvotačka i tip podatka.

Prilikom deklarisanja promenljivih tipa **string** moguće je u uglastim zagradama iza rezervisane reči **string** definisati i maksimalnu dužinu stringa. Ukoliko prilikom deklarisanja nije naveden ovaj podatak, podrazumeva se da je maksimalna dužina stringa 255.

Primer

```
var   x,y,z:real;
      ime:string;
      adresa:string[30];
      i,j:integer;
      pritisak:real;
      iskaz1,iskaz2:boolean;
      slovo:char;
```

Izrazi

Izrazi su kombinacije simbola koje definišu poredak i način izračunavanja neke vrednosti korišćenjem:

- operanda (konstante, promenljive, funkcije)
- operatora
- oblih zagrada

Operatori definišu koje je operacije potrebno izvriti nad operandima, a zagrade definišu redosled vršenja tih operacija. U slučajevima kada redosled vršenja operacija nije određen zagradama, primenjuju se pravila o prioritetu operacija:

Operatori	Prioritet
not	1 (najveći prioritet)
and * / div mod	2
or + -	3
in = > < >= <= <>	4 (najmanji prioritet)

Operatori

Nad svim tipovima podataka moguće je izvršiti određene operacije. Operacija preslikava konačan skup podataka (operande) u konačan skup podataka (rezultate). **Operatori** definišu operacije koje je potrebno izvršiti nad operandima da bi se dobio rezultat.

Operacije nad celobrojnim (integer) tipom podataka

Nad celobrojnim operandima se mogu izvoditi sledeće operacije koje daju celobrojni rezultat:

Operator	Operacija
*	množenje
div	celobrojno deljenje
mod	ostatak pri celobrojnom deljenju
+	sabiranje
-	oduzimanje

Operacije *, **div** i **mod** imaju viši prioritet od operacija **+** i **-**. Operacije istog prioriteta se izvršavaju sa leva na desno.

Primeri

```
3 * 7 = 21
15 div 2 = 7
15 mod 2 = 1
4 + 9 = 13
4 - 9 = -5
-4 + 9 = 5
3 * 7 div 5 mod 3 = 1
3 + 2 * 4 - 2 * 5 = 1
```

Operacije nad realnim (real) tipom podataka

Operator	Operacija
*	množenje
/	deljenje
+	sabiranje
-	oduzimanje

Pri radu sa realnim brojevima treba voditi računa o tome da se oni u memoriji ne beleže apsolutno tačno, već sa određenom tačnošću (određenim brojem decimalnih mesta). Iz tog razloga su i rezultati operacija nad realnim brojevima približni.

Primeri

```
3.47 * 7.21 = 25.0187
8.0 / 2.0 = 4.0
1.0 / 3.0 = 0.3333333
1.0 / 3.0 * 3.0 = 0.9999999
2.71 + 6.525 = 9.235
-7.812 - 0.1 = -7.912
6.17 + 2.14 * 3.81 - 4.5 = 9.8234
```

Operacije nad logičkim (boolean) tipom podataka

Nad logičkim operandima se mogu izvoditi sledeće operacije koje daju logički rezultat:

Operator	Operacija
not	negacija
and	konjukcija
or	disjunkcija
xor	ekskluzivna disjunkcija

Operacija **not** ima viši prioritet od operacija **and** i **or**. Operacije istog prioriteta se izvršavaju sa leva na desno.

Primeri

p	q	not p	p and q	p or q	p xor q
false	false	true	false	false	false
true	false	false	false	true	true
false	true	true	false	true	true
true	true	false	true	true	false

Relacijski operatori

Relacijski operatori omogućavaju poređenje dva operanda istog tipa, a dobijeni rezultat je logičkog tipa. lako su celi i realni brojevi formalno različiti tipovi podataka, korišćenjem relacijskih operatora moguće ih je međusobno porediti.

Relacijski operatori

Operator	Relacija
=	jednako
<>	različito
>	veće
<	manje
>=	veće ili jednako
<=	manje ili jednako
Operatori nad logičkim tipom	
=	ekvivalencija
<>	ekskluzivna disjunkcija
<=	implikacija
Operatori nad skupovima	
=	jednakost skupova
<>	različitost skupova
<=	podskup
>=	nadskup
in	element skupa

Operatori = i <> mogu se primeniti i na promenljive tipa **array** i **record** ukoliko su one istog tipa.

Relacijski operatori su najnižeg prioriteta i obavljaju se tek pošto se prethodno obave svi aritmetički operatori.

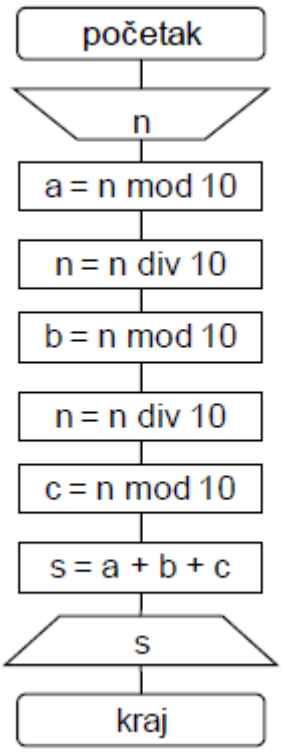
U slučaju primene navedenih operatora na znakovni tip podatka, rezultat se dobije poređenjem njihovih ASCII kodova, tj. njihovih pozicija u ASCII tabeli. Tako je rezultat relacije 'A'<'C' jednak **true**, pošto je ASCII kod znaka 'A' 65, a znaka 'C' 67.

Primeri

Izraz	Vrednost
`5 <= 0`	`false`
`(2*2) <> (3*3)`	`true`
`true < false`	`false`
`(5 > 0) > (5 < 0)`	`true`
`(5 > 0) = (5 <> 0)`	`true`
`(5 < 7) and (7 < 9) <= (9 < 5)`	`false`
`'C' < 'M'`	`true`

6. NPP koji za uneti trocifreni ceo broj ispisuje zbir cifara tog broja.

Rešenje 1:
Suma(654) = 4+5+6= (654 mod 10) + (65 mod 10) + (6 mod 10)




```
Program Zad6_1;
Var n,a,b,c,s:integer;
Begin
readln(n);
a:=n mod 10;
n:=n div 10;
b:=n mod 10;
n:=n div 10;
c:=n mod 10;
s:=a+b+c;
writeln(s)
End.
```

Rešenje 2:
Suma(654) = 6+5+4= (654 div 100) + (654 div 10 mod 10) + (654 mod 10)

```
Program Zad6_2;
Var n,s:integer;
Begin
readln(n);
s:=(n div 100)+ (n div 10 mod 10)+ (n mod 10);
writeln(s)
End.
```

7. NPP koji sa standardnog ulaza učitava prirodan broj i ispisuje taj broj bez cifre desetica

ULAZ	IZLAZ
158	18
12	2
7	7
108	18

8. NPP koji permutuje cifru jedinica i desetica učitanom prirodnom broju i ispisuje permutovani broj na standardni izlaz.

ULAZ	IZLAZ
158	185
12	21
7	70
108	180