

PODATKI TIPA STRING

NIZ ali STRING prikažemo kot vektor, ki ima največ 255 segmentov,

ali pa kot tabelo, ki ima 255 prostorčkov.

[illegible]

Splošne lastnosti:

- predstavitev podatkovnega tipa: kot vektor, kot tabela
- maksimalna vrednost je 255 (ta vrednost je indeksna) – indeks je tipa integer, celo število
- kot večina podatkovnih tipov, ima tudi string svoje standardne funkcije in procedure.
- Deklaracijske zakonitosti:
 - STRING[dolžina niza]
 - dolžina niza je nepredznačeno celo število od 1 – 255

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

- niz je vektorska veličina, ima začetek, smer in dolžino.
- Vsak niz ima pred podatki en blok, v katerem je zapisana veličina niza – dolžina znakov v nizu

10	1	2	3	4	5	6	7	8	9	0
-----------	---	---	---	---	---	---	---	---	---	---

Zgled:

```
program ccc;
```

```
uses crt;
```

var

```
a:string[20]; (* niz je sestavljen iz 20 znakov *)
```

a - dvajset znakov

[illegible]

Deklaracija konstante nad podatkovnim tipom string;

konstanta je dolžina niza:

const

n = 10;

var

```
a:string[n];
```

konstanta je sam niz:

const

```
a='Lepo vreme';
```

a='12'; (* to ni celo število, ampak niz z dvema črkama 1 in 2 *)

DEFINIRANJE LASTNEGA PODATKOVNEGA TIPA

Zgled:

type

```
OSNOVA = STRING[25];
```

```
var
a,b,c:OSNOVA;
s:string[250];
```

Type je rezervirana beseda, ki nakazuje, da bomo izdelali lasten podatkovni tip;

OSNOVA je ime novega podatkovnega tipa;

Lastne tipe podatkov se vedno deklarira pred spremenljivkami.

Spremenljivke lahko definiramo na dva načina

```
a) neposredno      s:string
b) posredno        type
                    osnova = string[12];
                    var
                    a,b:osnova;
```

primer:

```
const
c=5;
type
ocena=c..20;
var
a,b:ocena;
```

naloga:

program Project2;

{\$APPTYPE CONSOLE}

uses

SysUtils,crt32;

const

g = 9.81;

var

v0, alfa, Vx, Vy, T,D,H,R:real;

begin

clrscr;

writeln('vnesi začetno hitrost (h/s) in kot (stopinje)');

readln(v0,alfa);

r := alfa * pi / 180;

Vx := v0 * cos(r);

Vy := v0 * sin(r);

t := 2* Vy/g;

d := T * Vx;

h := Vy * Vy / (2 * g);

writeln;

writeln('zacetna hitrost je ',v0:5:5,'m/s');

writeln('kot meta je ',alfa:5:5,'stopinj');

writeln('cas letenja je ',t:5:5,'sekund');

writeln('dolzina meta je ',d:5:5,'metrov');

writeln('najvisja tocka meta je ',h:5:5,'metrov');

readln;

end.

NIZOVNE FUNKCIJE IN PROCEDURE

Funkcije:

COPY - IZREŽE DOLOČEN PODNIZ

COPY(=>nizovni izraz, =>pozicija, =>dolžina);
string podatka tipa integer

Zgled:

x := 'ABCDEFGH';
copy(x,1,2); => funkcija vrne vrednost 'AB'

X

A	B	C	D	E	F	G
---	---	---	---	---	---	---

Začetek in za dve mesti naprej kopiramo

copy(x,3,10); => funkcija vrne vrednosti 'CDEFGH'
copy(a,b,c); => vrne podniz dolžine c niza a, ki se začne z znakom na lokaciji b.

CONCAT - funkcija združuje – spaja več nizov v enega

CONCAT(=>niz1, =>niz2, =>niz3,...);
enostavno izberemo nize, ki bi jih radi združili.

Zgled:

X := CONCAT('AB','DC','XP'); x => 'ABDCXP'

LENGTH => funkcija vrne celo številčno vrednost, ki pomeni, koliko znakov je
zasedenih v nizu (vrne nam dolžino uporabljenega niza v nizu).

LENGTH(=> niz);
string

Zgled:

x := 'SAM_SI_POMAGAM';
i := length(x); i => 14

14	S	A	M	_	S	I	_	P	O	M	A	G	A	M
----	---	---	---	---	---	---	---	---	---	---	---	---	---	---

PROGRAM VAJA;

```
var
a:string[10];
b:string[20];
begin
a := 'abc';
b := '12345';
end.
```

<i>X</i>	<i>Length(x)</i>
A	3
B	5
A[1]	1

```

Program reverz;
var
  s:string[80];
  i:integer;
begin
  clrscr;
  writeln('vnesi niz');
  readln(s);
  for i := length(s) downto 1 do
    write(s[i]);
  readln;
end.

```

POS => funkcija nam vrne pozicijo, na kateri se nahaja iskani niz. Če niza ne najde, funkcija vrne 0.

Funkcija deluje na principu vzorčenja.

```

POS(=> a, => c);
a .. iskani podniz
c .. niz

```

Zgled:

x =>

A	C	M	A	M	A	1
---	---	---	---	---	---	---

```
I := pos('MAMA',x);
```

I => 3;

Procedure

DELETE => procedura nam izbriše željene znake iz niza

```
delete(=> ime nizovne spremenljivke, => pozicija, => dolžina);
```

Zgled:

```
x := 'ABCDEFGG';
```

```
delete(x,2,4);                      x => 'AFG'
```

Procedura DELETE briše zahtevano število znakov niza, začenši od označene pozicije.

INSERT => procedura nam vstavi željene znake v niz.

```
Insert(=> nizovni izraz, => nizovna spremenljivka, => pozicija);
```

Zgled:

```
x := 'ABCDEFGG';
```

```
insert('12',x,4);                      x => 'ABC12DEFG';
```

STR => STR je koren besedice string. Procedura je pomembna za toliko, ker je sposobna konvertirati številčno vrednost, ki jo dobimo kot rezultat nekega računanja, v niz znakov. Podobni podprogram `inttostr/floattostr..`

`Str(=>numerični izraz(:format), ime nizovne spremenljivke)`

Zgled:

`s:string[20]; x: real=5.56;`

`str(x:5:5,s);` `s => ' 5.56'`

VAL => Val je koren besedice valutacija. Procedura nam iz niza pretvori v število. Potrebno je paziti, ker če v nizu nimamo števila, pride do napake. Zato imamo kontrolno spremenljivko! Podobni podprogram `strtoint/strtofloat/....`

`Val(=> nizovni izraz, => numerična spremenljivka - pretvorba, => numerična spr. kontrola);`
celoštevilska spremenljivka,
ki nam pove, ali je
konverzija uspela ali ne.

Zgled:

`i,c:integer;`

`val('123',i,c);` `i => 123, c => 0`

`val('12Z',i,c);` `i = 0, c = 3`

`val('123Z',i,c);` `i = 0, c = 4`

primer programa:

`program Project1;`

`uses`

`Classes;`

`var`

`s:string;`

`i,k:integer;`

`begin`

`writeln('Vnesi število za pretvorbo: ');`

`readln(s);`

`val(s,i,k);`

`writeln('Pretvorjeno število: ',i);`

`writeln('napaka: ',k);`

`readln;`

`end.`

Pri uspešni konverziji nizovne vsebine v numerično pridobi tretji argument vrednost 0.

Pri neuspešni konverziji pa je vrednost celoštevilčne spremenljivke mesto, na katerem pride do napake.

FUNKCIJE: funkcije računajo matematične izraze.

PROCEDURE: delajo več kot matematične izraze, delajo tudi postopke.

PODATEK TIPI TABELA – ARRAY

Imenovanje podatkovnega tipa:

- Tabela, vektor (1D)
- Polje (2D)
- Matrika (3D)

1.) Predstavitev

Podatek tipa ARRAY ima 3 stanja

a) ENODIMENZIONALNA TABELA

a

1	2	3	4
1 .. 4			

Če želimo poklicati tretjo razpredelnico naredimo takole: **a[3]**

b) DVOZIMENZIONALNA TABELA

a

1	2	3	4	
				1
				2
1..4,1..2				

Če želimo poklicati tretjo razpredelnico v prvi vrstici naredimo takole: **a[3,1]**

c) TRIDIMENZIONALNA TABELA ali PROSTORSKA TABELA

a

	1	2	3	4	
1					1
					2
1..4,1..2					
	1	2	3	4	
2					1
					2
1..4,1..2					
	1	2	3	4	
3					1
					2
1..4,1..2					

Če želimo poklicati tretjo razpredelnico v prvi vrstici druge tabele naredimo takole: **a[3,1,2]**

Vsebine posameznih segmentov:

- vsi primitivni podatkovni tipi
- string
- tabela

Vsi segmenti v tabeli so enakega podatkovnega tipa.

Indeksi tabele so lahko:

- integer
- char
- boolean
- intervalni podatkovni tipi
- naštevni podatkovni tipi

- Deklaracija

Tudi podatek tipa ARRAY je intervalni podatkovni tip.

Var:

a: array[1..10] of string[20]; - enodimenzionalna tabela

- Deklaracijsko pravilo v pascalu

V pascalu navadno deklariramo podatek tipa tabela v posrednem načinu.

TYPE

MAT = array[1..10] of string[20];

VAR

MATRIKA:MAT;

Zgledi deklaracij:

array[char] of integer;

array[-10..10] of real;

array['A'..'Z'] of set of char;

 |_ nabor znakov

array[1..3,1..2,'A'..'H'] of integer;

array[boolean,boolean] of boolean;

array[1..10] of array[char] of string[20];

 je enako kot

array[1..10,char] of string[20];

DEKLARACIJA TABELARIČNE KONSTANTE

const

m:array[1..12] of 1..31 = (31,28,31,30,31,30,31,31,30,31,30,31);

d:array[1..7] of string[20]

=('PONEDELJEK','TOREK','SREDA','ČETRTEK','PETEK','SOBOTA','NEDELJA');

m[11] => 30

d[3] => 'SREDA'

```
const
EQW:array[boolean,boolean] of boolean = ((true,false),(false,true));
```

<i>TRUE</i>	<i>FALSE</i>	FALSE
FALSE	TRUE	TRUE
FALSE	TRUE	

```
Const
x:array[1..2,0..3,'a'..'c'] of integer =
(((0,1,2),
(3,4,5),
(6,7,8),
(9,10,11)),
((12,13,14),
(15,16,17),
(18,19,20),
(21,22,23)));
```

	<i>'A'</i>	<i>'B'</i>	<i>'C'</i>
x[1,0	0	1	2
x[1,1	3	4	5
x[1,2	6	7	8
x[1,3	9	10	11
x[2,0	12	13	14
x[2,1	15	16	17
x[2,2	18	19	20
x[2,3	21	22	23

A	1	2
0	0	12
1	3	15
2	6	18
3	9	21

B	1	2
0	1	13
1	4	16
2	7	19
3	10	22

C	1	2
0	3	14
1	5	17
2	8	20
3	11	23

VSTOP, IZSTOP V TABELO (pisanje, branje)

V tabelarični tip najlepše vstavljamo s pomočjo FOR zanke.

A

--	--	--	--

1..4 - POSAMEZNI INDEKSI SO TIPA INTEGER, ZATO LAHKO UPORABIMO FOR ZANKO.

Klicanje:

a[1] – kličemo prvi segment

a[2] – kličemo drugi segment

a[3] – tretji segment

a[4] – četrti segment

Če je tabela enodimenzionalna potrebujemo eno for zanko.

Za dvodimenzionalne tabele potrebujemo dve for zanki.

Za trodimenzionalne tabele potrebujemo tri for zanke.

FOR ZANKE SE NE SMEJO SEKATI.