

## ZAPIS – STRUCT

Kot podatkovni tip je strukturiran na več različnih podatkovnih tipov. Zelo kompleksen podatkovni tip in ima največje izrazne možnosti.

Vsebinska razlika:

### STRING

char	char	char	char	char
------	------	------	------	------

Vsi segmenti so tipa character

ENOLIČNO DOLOČENO!!!

### ARRAY

int	int	int	int	int
-----	-----	-----	-----	-----

Segmenti so vsi istega tipa, ki ga mi pri deklaraciji določimo.

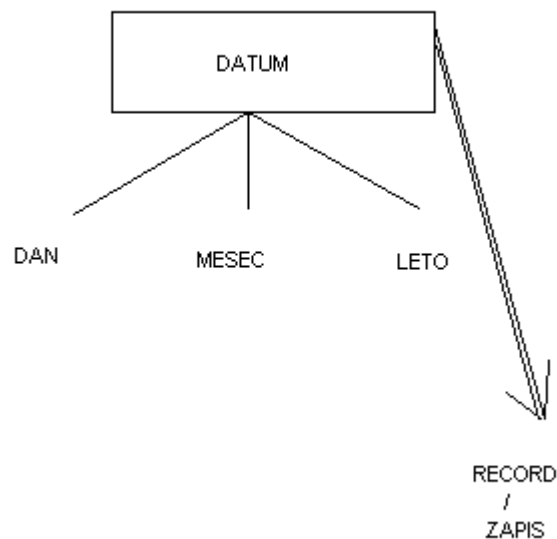
### STRUCT

char	int	char[20]	float	int
------	-----	----------	-------	-----

V tem podatkovnem tipu so lahko vsi segmenti različni. Določimo jih pri deklaraciji.

Predstavitveni model:

array	float	string	float	int
-------	-------	--------	-------	-----



## DEKLARACIJA

Deklaracija podatkovnega tipa zapis – STRUCT je vezana skoraj vedno na posredni koncept naslavljanja. Najprej izdelamo strukturo in šele nato deklariramo spremenljivko. Lahko pa tudi direktno!

```
/* primer 1: deklaracija */
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct datum {
```

```
    int dan;
```

```
    int mesec;
```

```
    int leto;
```

```
};
```

```
int main(void)
```

```
{
```

```
    struct datum danes;
```

```
    danes.dan = 6;
```

```
    danes.mesec = 4;
```

```
    danes.leto = 2005;
```

```
    printf("%d.%d.%d\n",danes.dan,danes.mesec,danes.leto);
```

```
    return 0;
```

```
}
```

```
/* primer 2: prireditev ene strukturne spremenljivke drugi spremenljivki */
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct datum {
```

```
    int dan;
```

```
    int mesec;
```

```
    int leto;
```

```
};
```

```
int main(void)
```

```
{
```

```
    struct datum a,b;
```

```
    a.dan = 1;
```

```
    a.mesec = 1;
```

```
    a.leto = 2005;
```

```
    b = a; /* stavek pomeni enako kot če bi naredili
```

```
            b.dan = a.dan; b.mesec = a.mesec; b.leto = a.leto; */
```

```
    printf("%d.%d.%d\n",b.dan,b.mesec,b.leto);
```

```
    return 0;
```

```
}
```

```
/* primer 3: inicializacija ob deklaraciji */
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct datum {
```

```
    int dan;
```

```
    int mesec;
```

```
    int leto;
```

```
};
```

```
int main(void)
```

```
{
```

```
    struct datum danes = {20,1,2005};
```

```
    printf("%d.%d.%d\n",danes.dan,danes.mesec,danes.leto);
```

```
    return 0;
```

```
}
```

## POLJA STRUKTUR

Mnogokrat moramo hraniti več podatkov, ki so istega tipa. Primer je vnos dijakov v določenem razredu. Vsi dijaki imajo iste lastnosti, le da jih je maksimalno 32.

```
#include <stdio.h>
#include <stdlib.h>
struct dijak {
    char ime[20];
    char priimek[20];
    int leto;
    int povp_ocena;
};

int main(void)
{
    struct dijak dijaki[32];
    int i;
    for (i=0;i<32;i++)
    {
        printf("Dijak stevilka %d\n",i+1);
        printf("Vnesi ime: ");
        scanf("%s",dijaki[i].ime);
        printf("Vnesi priimek dijaka: ");
        scanf("%s",dijaki[i].priimek);
        printf("Vnesi leto dijaka: ");
        scanf("%d",&dijaki[i].leto);
        printf("Vnesi povprecno oceno: ");
        scanf("%d",&dijaki[i].povp_ocena);
    }

    // izpis podatkov 5-ega dijaka
    printf("%s %s %d %d\n",dijaki[4].ime,dijaki[4].priimek,dijaki[4].leto,dijaki[4].povp_ocena);

    return 0;
}
```

## VGNEZDENE STRUKTURE

Elementi v strukturi so lahko sami po sebi tudi strukture – tako pridemo do struktur v strukturi. Poglejmo si primer:

```
#include <stdio.h>
#include <stdlib.h>

struct X {
    int data;
};

struct Y {
    struct X data;
    struct Z {
        int link;
    } in_data;
    int temp;
};
```

```

int main(void)
{
    struct Y podatek;
    podatek.data.data = 5;
    podatek.in_data.link = 10;
    podatek.temp = 4;
    printf("%d %d %d\n",podatek.data.data,
           podatek.in_data.link,
           podatek.temp);
    return 0;
}

```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct podatki {
    int starost; // v letih
    int visina; // v metrih
    int teza; // v tonah
};
struct datum {
    int dan;
    int mesec;
    int leto;
};
struct ladja {
    char ime[20];
    struct datum dan_odhoda;
    struct podatki lastnosti;
};

```

```

int main(void)
{
    struct ladja anek;
    strcpy(anek.ime,"Tine");
    anek.dan_odhoda.dan=5;
    anek.dan_odhoda.mesec=9;
    anek.dan_odhoda.leto=2005;
    anek.lastnosti.starost = 45;
    anek.lastnosti.visina = 180;
    anek.lastnosti.teza = 75;

    return 0;
}

```

## STRUKTURE IN FUNKCIJE

Strukturo lahko kot vsak drugi podatkovni tip posredujemo v funkcijo za nadaljno uporabo. Če želimo spreminjati vrednosti v strukturi je potrebno podatke posredovati po referenci (tako kot int, float, char,...). Če nas zanima samo en podatek za izpis strukture (ne spreminjamo – torej posredujemo podatek v funkcijo po vrednosti), je priporočljivo, da se posreduje samo ta podatek in ne celotne strukture. Problem nastane, ko je potrebno izdelati kopijo celotne strukture in to zahteva določen čas. Zato pri strukturah raje podajamo spremenljivke po referenci.

Za strukture, katere kličemo po referenci dostopamo do podatkov v strukturi z uporabo znaka -> in ne .

**struktura->podatek**

```

#include <stdio.h>
#include <stdlib.h>
struct oseba {
    int teza;
    int velikost;
};
void vnos(struct oseba *a)
{
    printf("Vnesi tezo: ");
    scanf("%d",&a->teza);
    printf("Vnesi velikost: ");
    scanf("%d",&a->velikost);
}
void izpis(struct oseba a)
{
    printf("Teza: %d\n",a.teza);
    printf("Velikost: %d\n",a.velikost);
}
int main(void)
{
    struct oseba sosed;
    vnos(&sosed);
    izpis(sosed);
    return 0;
}

```

## **TABELE STRUKTUR V FUNKCIJAH**

```

#include <stdio.h>
#include <stdlib.h>
struct oseba {
    int teza;
    int velikost;
};
void vnos(struct oseba *a)
{
    printf("Vnesi tezo: ");
    scanf("%d",&a->teza);
    printf("Vnesi velikost: ");
    scanf("%d",&a->velikost);
}
void izpis(struct oseba a)
{
    printf("Teza: %d\t",a.teza);
    printf("Velikost: %d\n",a.velikost);
}
void izpis_dijakov(struct oseba *dijaki, int n)
{
    int i;
    for (i=0;i<n;i++)
        izpis(dijaki[i]);
}

```

```

int main(void)

```

```

{
    int i;
    struct oseba dijaki[32];
    for (i=0;i<32;i++)
        vnos(&dijaki[i]);
    izpis_dijakov(dijaki,32);
    return 0;
}

```

## DEFINIRANJE LASTNIH PODATKOVNIH TIPOV

typedef uporabimo, da skrajšamo (naredimo nam bolj prijazne) ime podatkovne tipe. Z uporabo typedef ne ustvarimo nobenega novega podatkovnega tipa, ampak ustvarimo samo bližnjico do že ustvarjenega/obstoječega podatkovnega tipa. Bližnjico lahko uporabimo nad osnovnimi podatkovnimi tipi (int, float, char), kot tudi nad strukturiranimi podatkovnimi tipi (nizi, tabele, strukture).

Primeri izdelave naših podatkovnih tipov za osnovne podatkovne tipe:

```

typedef int cela; // cela je bližnjica za int podatkovni tip
typedef float realna;
typedef char znaki;
typedef char c;

```

Primeri izdelave podatkovnih tipov za nize, tabele in strukture

```

typedef int cela, stevila[100]; // lahko izdelamo tudi več bližnjic za en podatkovni tip – primer cela
                                // in tabelo 100 celih števil

```

```

typedef float realna[20];
typedef struct oseba osebe[100];

```

```

#include <stdio.h>
#include <stdlib.h>
struct dijak {
    char ime[20];
    char priimek[20];
    int leto;
    int povp_ocena;
};

```

```

typedef struct dijak dijaki[32]; // dijaki je tabela 32 dijakov

```

```

void vnos(dijak *s)
{
    int i;
    printf("Vnesi ime: ");
    scanf("%s",s->ime);
    printf("Vnesi priimek dijaka: ");
    scanf("%s",s->priimek);
    printf("Vnesi leto dijaka: ");
    scanf("%d",&s->leto);
    printf("Vnesi povprecno oceno: ");
    scanf("%d",&s->povp_ocena);
}

```

```

int main(void)
{
    dijaki r2;
    int i;
}

```

```

for (i=0;i<32;i++)
{
    printf("Dijak stevilka %d\n",i+1);
    vnos(&r2[i]);
}

// izpis podatkov 5-ega dijaka
printf("%s %s %d %d\n",r2[4].ime,r2[4].priimek,r2[4].leto,r2[4].povp_ocena);

return 0;
}

```

## NALOGE

1. Izdelajte podatkovno strukturo VIDEOTEKA v kateri hranite podatke o filmih. Vsak film je opisan z sledečimi podatki: naslov, leto, žanr, oceno in petimi igralci; Vsak igralec je opisan z sledečimi podatki: imenom, priimkom in datumom rojstva; Datum rojstva je opisan z sledečimi podatki: dnevom, mesecem in letom; Izdelajte podprograme za vnos in izpis podatkov, ter v glavnem programu pokažite, kako se podprogrami uporabijo.

2. Znak na zaslonu opisujejo naslednji parametri:

ASCII\_koda : char; (\* koda znaka, ki ga želimo prikazati \*)

X\_pozicija : int; (\* x-koordinata položaja znaka \*)

Y\_pozicija : int; (\* y-koordinata položaja znaka \*)

Barva : int; (\* barva znaka \*)

Napišite program, ki omogoča:

- nastavitev začetne vrednosti (inicializacijo) polja 10 znakov  
void init(X:Tabela);  
( Polje inicializirajte z velikimi črkami, za barvo pa uporabite svetlejše barve (od št. 10 do 14).)
- izpis polja  
void izpis\_znaka(Z:TZnak); {pomožna procedura za izpis enega znaka}  
void prikaz(X : Tabela);
- urejanje polja, glede na ASCII\_kodo znaka  
void uredi(Var X : Tabela);
- iskanje znaka, ki je po legi najbližji desnemu spodnjemu vogalu zaslona.  
float razdalja(X:TZnak); {pomožna funkcija za izračun razdalje}  
void isci(X : Tabela; Z : TZnak);

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <conio.h>
```

```
#include <time.h>
```

```
#include <math.h>
```

```

struct znak {
    char ASCII_koda;
    int x_pozicija,y_pozicija;
    int barva;
};

```

```
void init(struct znak *znaki, int n)
```

```

{
    int i;
    for (i=0;i<n;i++)
    {

```

```
        znaki[i].ASCII_koda = rand()%26+65;
```

```
        znaki[i].x_pozicija = rand()%80+1;
```

```

        znaki[i].y_pozicija = rand()%25+1;
        znaki[i].barva = rand()%5+10;
    }
}

void izpis_znaka(struct znak zn)
{
    gotoxy(zn.x_pozicija,zn.y_pozicija);
    textcolor(zn.barva);
    cprintf("%c",zn.ASCII_koda);
}

void prikazi(struct znak *znaki, int n)
{
    int i;
    clrscr();
    for (i=0;i<n;i++)
        izpis_znaka(znaki[i]);
}

void uredi(struct znak *zn, int n)
{
    int i,j;
    struct znak pom;
    for (i=0;i<n-1;i++)
    {
        for (j=n-1;j>0;j--)
        {
            if (zn[j].ASCII_koda < zn[j-1].ASCII_koda)
            {
                pom = zn[j];
                zn[j] = zn[j-1];
                zn[j-1]=pom;
            }
        }
    }
}

float razdalja(struct znak zn)
{
    float x,y;
    x = 80-zn.x_pozicija; // potrebujemo dolzino in ne velikost
    y = 25-zn.y_pozicija;
    return sqrt(pow(x,2)+pow(y,2));    // po pitagorovem izreku izracunamo dolzino
}

void isci(struct znak *zn, int n)
{
    int i;
    int pos;
    float max,t;
    max = razdalja(zn[0]);
    pos = 0;
    for (i=1;i<n;i++)
    {
        t = razdalja(zn[i]);
        if (t<max)

```



```

        {
            max = t;
            pos = i;
        }
    }
    gotoxy(1,23);
    printf("Najblize je znak %c na poziciji %dx%d",zn[pos].ASCII_koda,

        zn[pos].x_pozicija,

        zn[pos].y_pozicija);
}

```

```

int main(void)
{
    struct znak znaki[10];
    int i;
    srand(time(NULL));
    init(znaki,10);
    prikazi(znaki,10);
    gotoxy(1,24);
    for (i=0;i<10;i++)
        printf("%c ",znaki[i].ASCII_koda);

    uredi(znaki,10);
    gotoxy(1,25);
    for (i=0;i<10;i++)
        printf("%c ",znaki[i].ASCII_koda);

    isci(znaki,10);
    getch();
    return 0;
}

```

3. Izdelajte podatkovno strukturo STUDENT v kateri hranite podatke o študentih. Vsak študent je opisan z sledečimi podatki: ime, priimek, domači naslov, telefon, vpisna številka, datumom rojstva, datumom vpisa in še naslednje podatke za število bonov za prehrano, procent popusta pri avtobusni vozovnici, št ur na teden. Datum rojstva in datum vpisa je opisan z sledečimi podatki: dnevom, mesecem in letom. Izdelajte podprograme za vnos in izpis podatkov, ter v glavnem programu pokažite, kako se podprogrami uporabijo.
4. V programu, ki obdeluje podatke o atletih, ki tekmujejo v suvanju krogle, imamo naslednje deklaracije:

```

struct metalec {
    int startnaStevilka;
    char priimek[20], ime[20];
    float dolzineMetrov[6];
};
struct metalec m;

```

Spremenljivka m hrani podatke o enem tekmovalcu, komponenta dolzineMetrov pa vsebuje dosežene daljave v posameznih serijah (vsak tekmovalec ima na voljo 6 metrov, neveljavni met ima dolžino 0).Izdelajte for stavek tako, da bo za tekmovalca m izpisal dolžine vseh šestih metrov.

5. Napiši program, ki pretvori podatke iz tabele s strukturo prva\_a v tabelo s strukturo prva\_b

#### Struktura **prva\_a**

```
struct prva_a {  
    float a;  
    char b[15];  
};
```

#### Struktura **prva\_b**

```
struct prva_b {  
    int a;  
    char b[20];  
};
```

6. Izdelajte podatkovno strukturo STUDENT v kateri hranite podatke o študentih

Vsak študent je opisan z sledečimi podatki:

ime, priimek, domači naslov, telefon, vpisna številka, datumom rojstva, datumom vpisa, število bonov za prehrano, procent popusta pri avtobusni vozovnici, število ur na teden. Datum rojstva in datum vpisa je opisan z dnevom, mesecem in letom. Izdelajte podprograme za vnos in izpis podatkov, ter v glavnem programu pokažite, kako se podprogrami uporabijo.

7. Napišite program v programskem jeziku C, ki prebere datum v obliki dan, mesec, leto in izpiše, kateri po vrsti je ta dan v letu (med 1 in 365 oz. 1 in 366).

Meseci s 31 dnevi so januar, marec, maj, julij, avgust, oktober, december; Meseci s 30 dnevi so april, junij, september, november, februar pa ma 28 dni, razen kadar je leto prestopno. Od leta 1582 velja, da so prestopna leta tista, ki so deljiva z 4 in hkrati niso deljiva s 400. (30 točk)

8. Imamo tabelo zapisov. Zapis sestavljata podatka: vzdevek osebe (niz 10 znakov) in rezultat (celo število). Napišite vse potrebne deklaracije in program, ki bo izpisal vzdevek in rezultat vseh oseb, ki imajo najslabši rezultat.
9. Napišite program, ki poišče državo, v kateri je cena goriva najnižja. Izdelajte podprograme za vnos, iskanje in izpis.