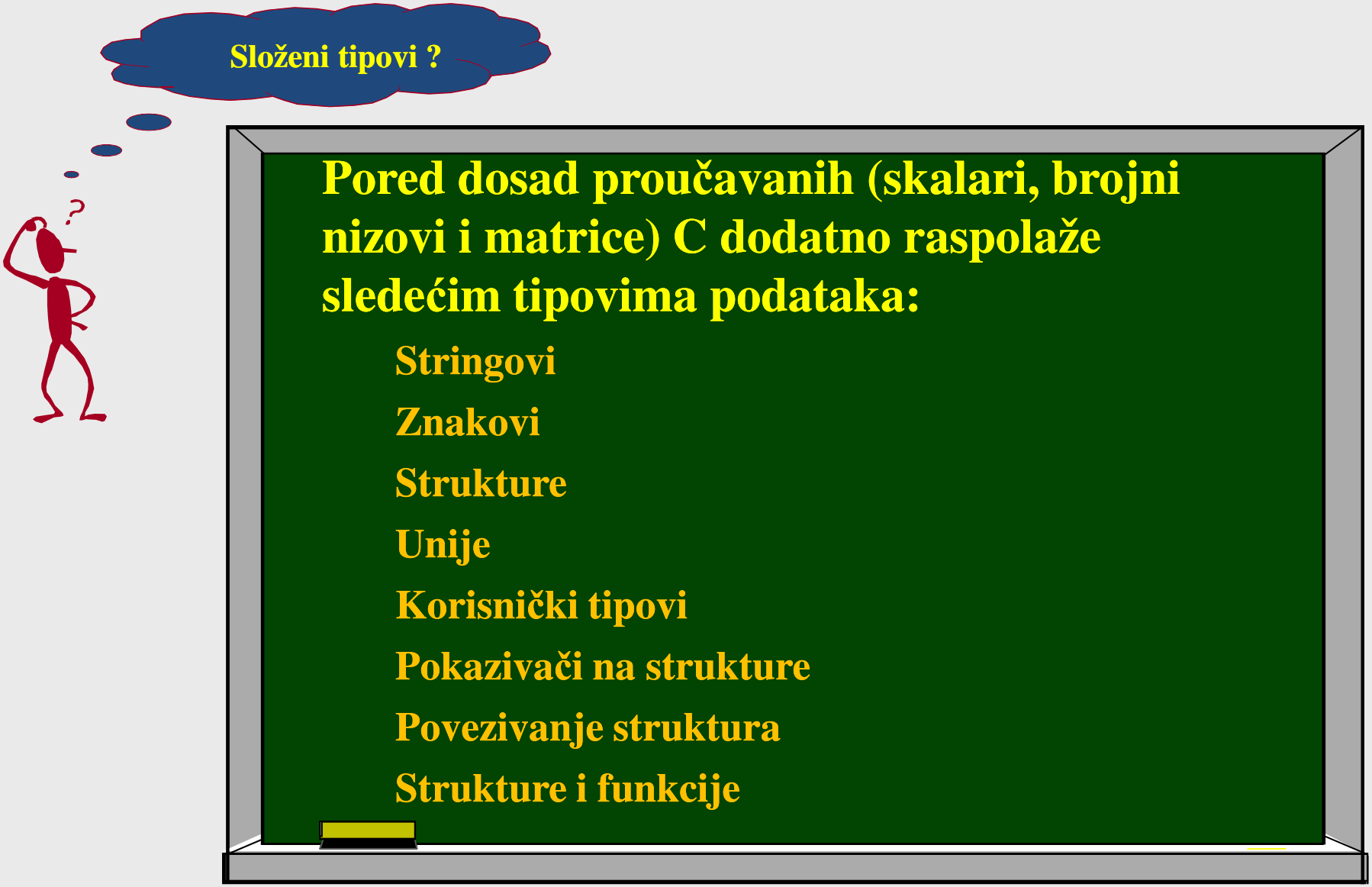
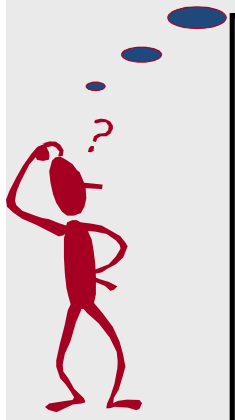
Složeni tipovi podataka – II dio



Stringovi



Stringovi

String je jednodimenzionalni znakovni niz

To je niz znakova koji završava nul-znakom (znak čiji je kod 0) `'\0'`

Deklaracija stringa:

```
char ime[duzina] = "string" ;
```

ili

```
char ime[duzina] = { znak, znak, … , znak } ;
```

Primjer:

```
char grad1[] = "PODGORICA" ;
char grad2[] = { 'B','E','R','A','N','E','\0' };
```

Označavanje znakovnih i string konstanti:

```
'D'   - ovo je znak D
"D"   - ovo je string 'D','\0'
```

Stringovi


STRING?
Manipulacija stringom
Ispisivanje stringa
ispisivanje pomoću funkcije
printf("%s", string);
Primjer:
#include <stdio.h>
main()
{
char grad[]= "PODGORICA";
printf("%s\n", grad);
}
PODGORICA
ispisivanje pomoću funkcije
puts(string);
Primjer:
#include <stdio.h>
main()
{
char grad[]= "PODGORICA";
puts(grad);
}
PODGORICA

Stringovi

STRING?

Manipulacija stringom

Učitavanje stringa

učitavanje pomoću funkcije

```
scanf("%s", string);
```

Primjer:

```
#include <stdio.h>
main()
{
   char tekst[50];
   printf("Unesi recenicu:\n");
   scanf("%s", tekst);
   printf("%s", tekst);
}
```

```
Unesi recenicu:
NOVI SAD
NOVI
```

učitavanje pomoću funkcije

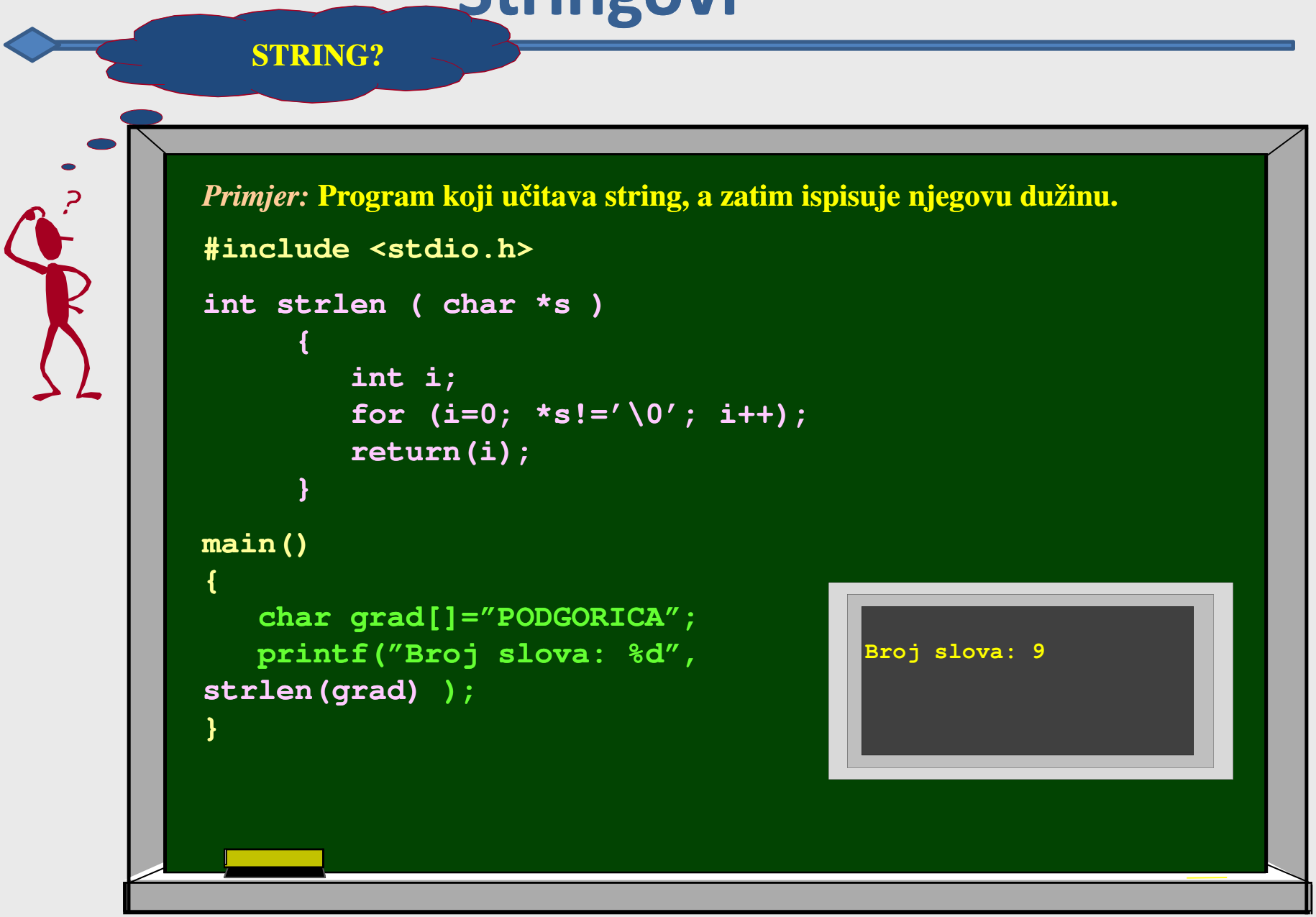
```
gets(string);
```

Primjer:

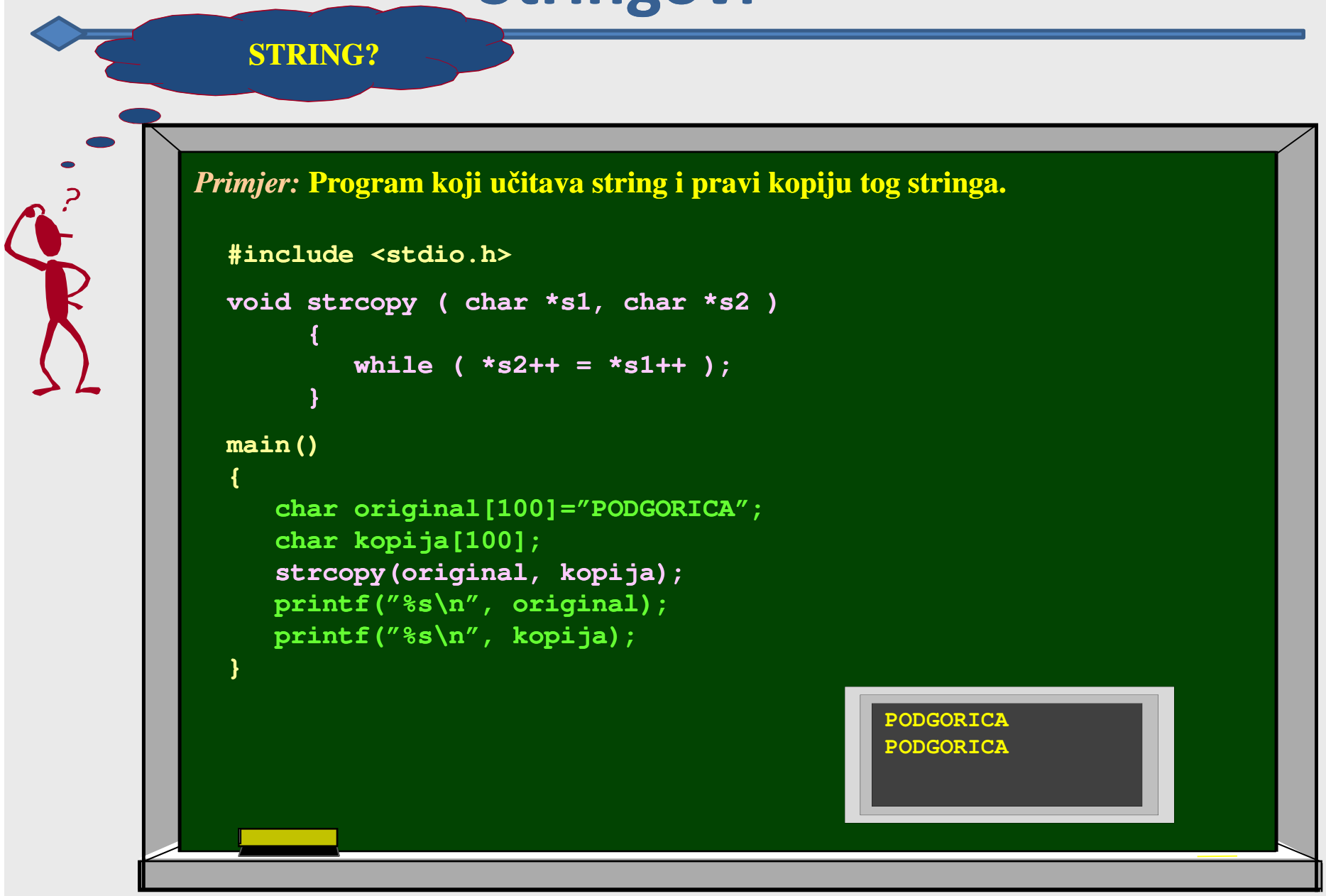
```
#include <stdio.h>
main()
{
   char tekst[50];
   printf("Unesi recenicu:\n");
   gets(tekst);
   printf("%s", tekst);
}
```

```
Unesi recenicu:
NOVI SAD
NOVI SAD
```

Stringovi


STRING?
Primjer: **Program koji učitava string, a zatim ispisuje njegovu dužinu.**
#include <stdio.h>
int strlen (char *s)
{
int i;
for (i=0; *s!='\0'; i++);
return(i);
}
main()
{
char grad[]="PODGORICA";
printf("Broj slova: %d",
strlen(grad));
}
Broj slova: 9

Stringovi


STRING?
Primjer: **Program koji učitava string i pravi kopiju tog stringa.**
#include <stdio.h>
void strcopy (char *s1, char *s2)
{
while (*s2++ = *s1++);
}
main()
{
char original[100]="PODGORICA";
char kopija[100];
strcopy(original, kopija);
printf("%s\n", original);
printf("%s\n", kopija);
}
PODGORICA
PODGORICA

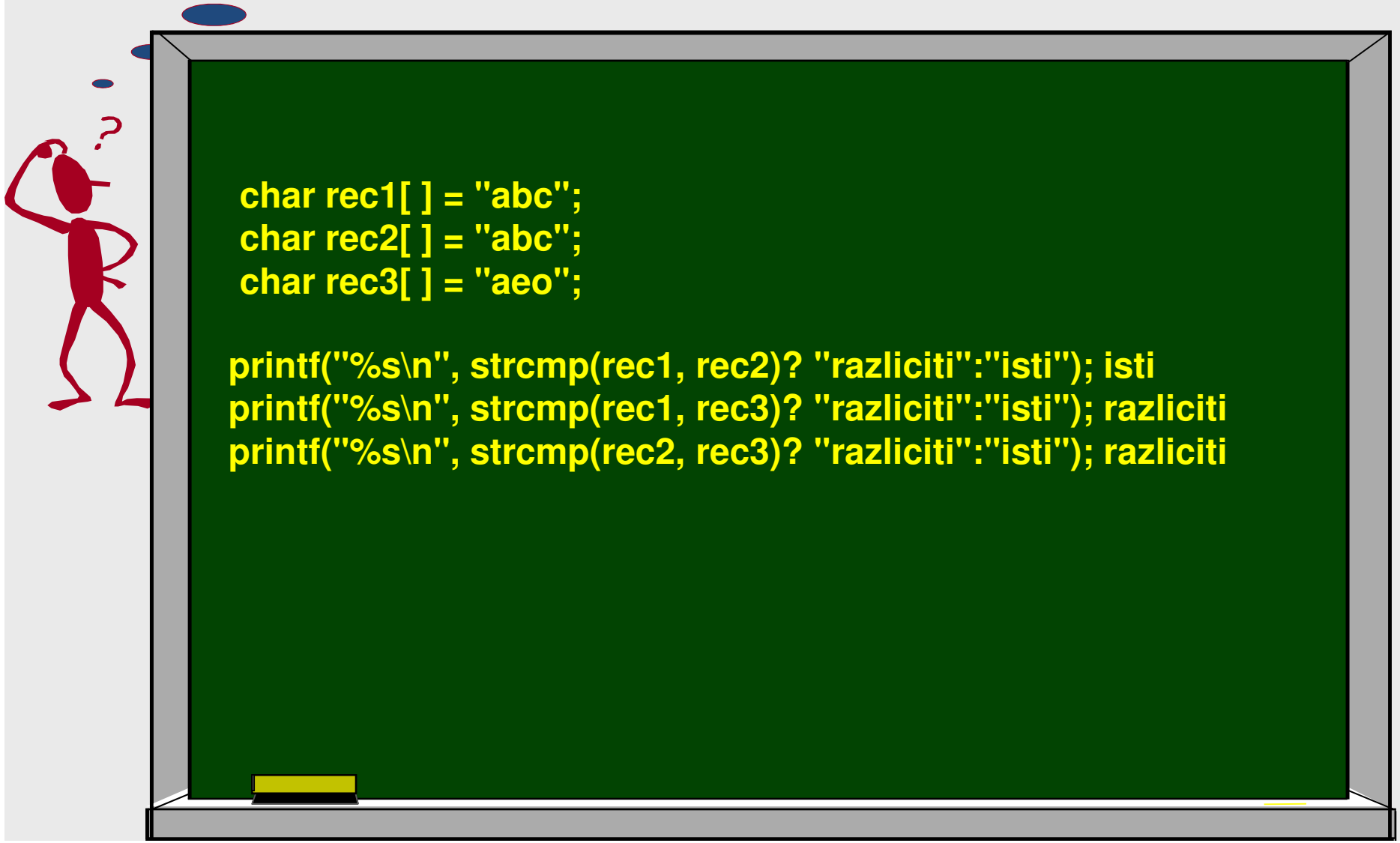
Stringovi

(biblioteka: string.h)

strlen() - određivanje dužine stringa bez znaka null na kraju
strcmp() - poređenje kompletnih sadržaja dva stringa
strncmp() - poređenje djelova sadržaja dva stringa
strcpy() - kopiranje jednog stringa u drugi, od početka drugog
strncpy() - kopiranje dijela jednog stringa u drugi, od početka drugog
strcat() - kopiranje jednog stringa u drugi, u produžetku sadržaja drugog
strncat() - kopiranje dela jednog stringa u drugi, u produžetku sadržaja drugog

Stringovi

(biblioteka: string.h)

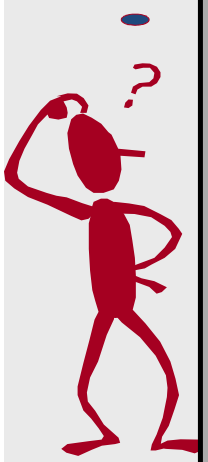


```
char rec1[ ] = "abc";
char rec2[ ] = "abc";
char rec3[ ] = "aeo";

printf("%s\n", strcmp(rec1, rec2)? "razliciti":"isti"); isti
printf("%s\n", strcmp(rec1, rec3)? "razliciti":"isti"); razliciti
printf("%s\n", strcmp(rec2, rec3)? "razliciti":"isti"); razliciti
```

Stringovi

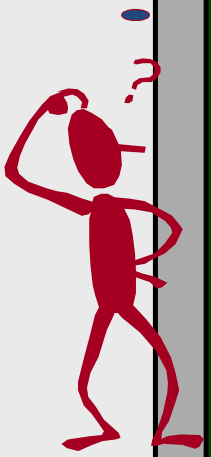
(biblioteka: string.h)



```
// KOPIRANJE STRINGA
#include <stdio.h>
#include <string.h>
int main(void)
{
char bafer[6];
char rec[ ] = "abc";
strcpy(bafer, rec);
puts(bafer);
getchar();
return(0);
}
```

Stringovi

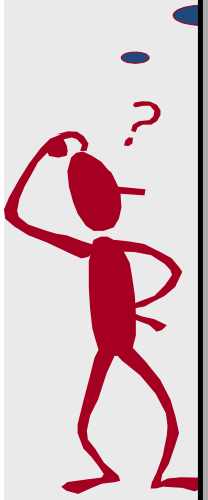
(biblioteka: string.h)



```
// KOPIRANJE STRINGA
#include <stdio.h>
#include <string.h>
int main(void)
{
char bafer[6];
char rec[ ] = "abc";
strcpy(bafer, rec);
puts(bafer);
getchar();
return(0);
}
```

Stringovi

biblioteka: stdlib.h)
(biblioteka: string.h)



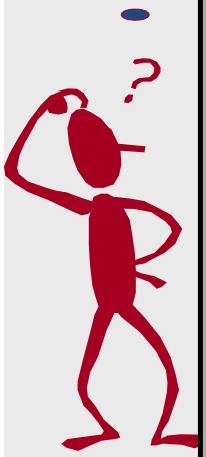
- atoi() za konverziju stringa u ceo broj, ako je to moguće
- atof() za konverziju stringa u realan broj, ako je to moguće

```
//PRIMJER atof()
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(void)
{
char tekst1[ ] = " 2.5";
char tekst2[ ] = "0005";
char tekst3[ ] = "10.25 i neki tekst";
char tekst4[ ] = "5e+2";
char tekst5[ ] = "neki tekst 200.30";
printf("\nbroj: %f", atof(tekst1)); //2.500000
printf("\nbroj: %f", atof(tekst2)); //5.000000
printf("\nbroj: %f", atof(tekst3)); //10.250000
printf("\nbroj: %f", atof(tekst4)); //500.000000
printf("\nbroj: %f", atof(tekst5)); //0.000000
getchar();
return(0);
}
```

Znakovi

znakovi



Znak potreban programu može biti: slovo, cifra, ostali...

- Tip promenljive za čuvanje znaka – samo jedan: char
- Svaki znak je kodiran u ASCII tabeli – pomoću broja
- Nema posebnih operatora za znakove!

Znakovi

ASCII TABELA

Dec	Hex	Oct	Char
0	0	000	NUL (null)
1	1	001	SOH (start of heading)
2	2	002	STX (start of text)
3	3	003	ETX (end of text)
4	4	004	EOT (end of transmission)
5	5	005	ENQ (enquiry)
6	6	006	ACK (acknowledge)
7	7	007	BEL (bell)
8	8	010	BS (backspace)
9	9	011	TAB (horizontal tab)
10	A	012	LF (NL line feed, new line)
11	B	013	VT (vertical tab)
12	C	014	FF (NP form feed, new page)
13	D	015	CR (carriage return)
14	E	016	SO (shift out)
15	F	017	SI (shift in)
16	10	020	DLE (data link escape)
17	11	021	DC1 (device control 1)
18	12	022	DC2 (device control 2)
19	13	023	DC3 (device control 3)
20	14	024	DC4 (device control 4)
21	15	025	NAK (negative acknowledge)
22	16	026	SYN (synchronous idle)
23	17	027	ETB (end of trans. block)
24	18	030	CAN (cancel)
25	19	031	EM (end of medium)
26	1A	032	SUB (substitute)
27	1B	033	ESC (escape)
28	1C	034	FS (file separator)
29	1D	035	GS (group separator)
30	1E	036	RS (record separator)
31	1F	037	US (unit separator)

Dec	Hex	Oct	Char
32	20	040	Space
33	21	041	!
34	22	042	"
35	23	043	#
36	24	044	$
37	25	045	%
38	26	046	&
39	27	047	'
40	28	050	(
41	29	051	)
42	2A	052	*
43	2B	053	+
44	2C	054	,
45	2D	055	-
46	2E	056	.
47	2F	057	/
48	30	060	0
49	31	061	1
50	32	062	2
51	33	063	3
52	34	064	4
53	35	065	5
54	36	066	6
55	37	067	7
56	38	070	8
57	39	071	9
58	3A	072	:
59	3B	073	;
60	3C	074	<
61	3D	075	=
62	3E	076	>
63	3F	077	?

Dec	Hex	Oct	Char
64	40	100	@
65	41	101	A
66	42	102	B
67	43	103	C
68	44	104	D
69	45	105	E
70	46	106	F
71	47	107	G
72	48	110	H
73	49	111	I
74	4A	112	J
75	4B	113	K
76	4C	114	L
77	4D	115	M
78	4E	116	N
79	4F	117	O
80	50	120	P
81	51	121	Q
82	52	122	R
83	53	123	S
84	54	124	T
85	55	125	U
86	56	126	V
87	57	127	W
88	58	130	X
89	59	131	Y
90	5A	132	Z
91	5B	133	[
92	5C	134	\
93	5D	135	]
94	5E	136	^
95	5F	137	_

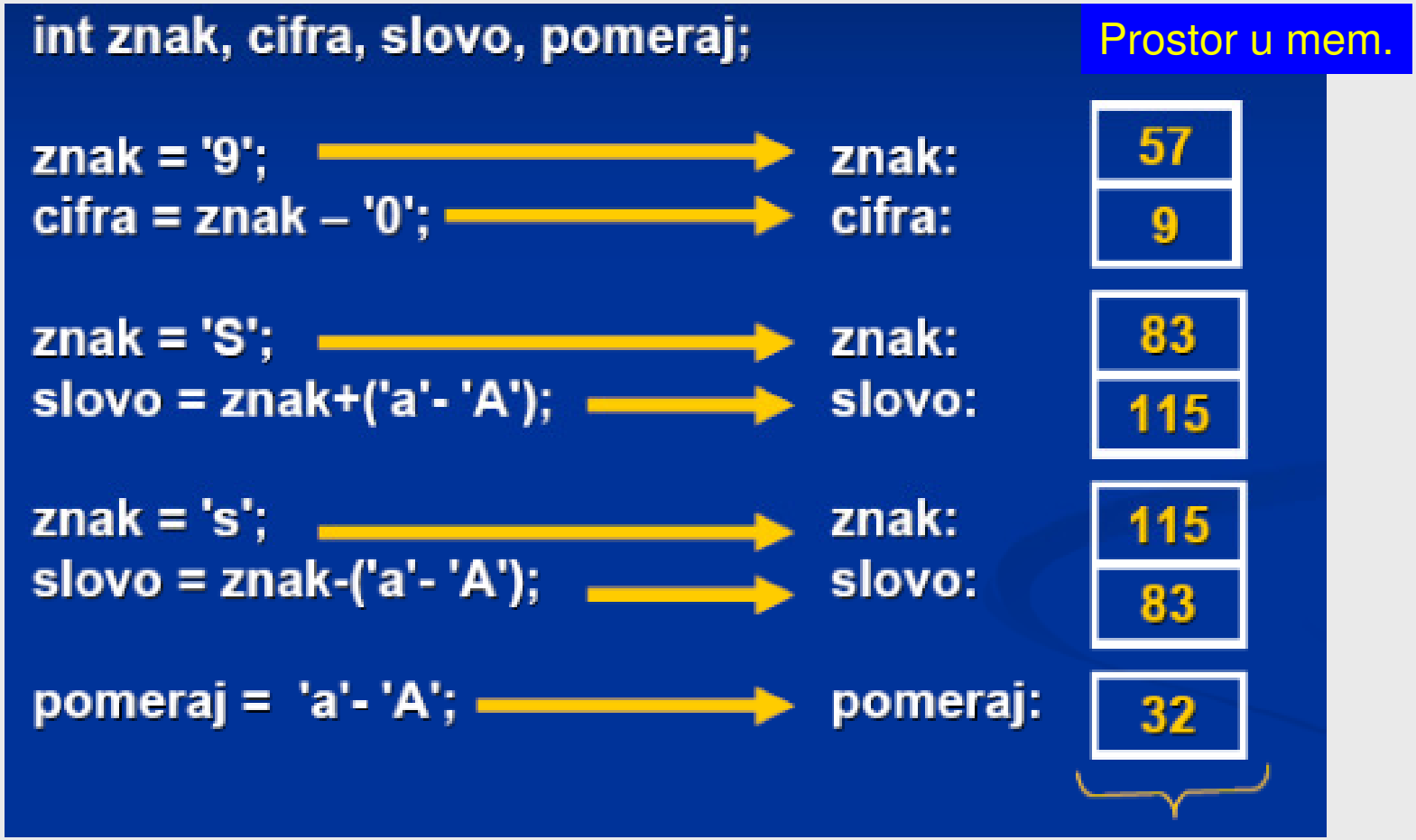
Dec	Hex	Oct	Char
96	60	140	`
97	61	141	a
98	62	142	b
99	63	143	c
100	64	144	d
101	65	145	e
102	66	146	f
103	67	147	g
104	68	150	h
105	69	151	i
106	6A	152	j
107	6B	153	k
108	6C	154	l
109	6D	155	m
110	6E	156	n
111	6F	157	o
112	70	160	p
113	71	161	q
114	72	162	r
115	73	163	s
116	74	164	t
117	75	165	u
118	76	166	v
119	77	167	w
120	78	170	x
121	79	171	y
122	7A	172	z
123	7B	173	{
124	7C	174	|
125	7D	175	}
126	7E	176	~
127	7F	177	DEL

Znakovi

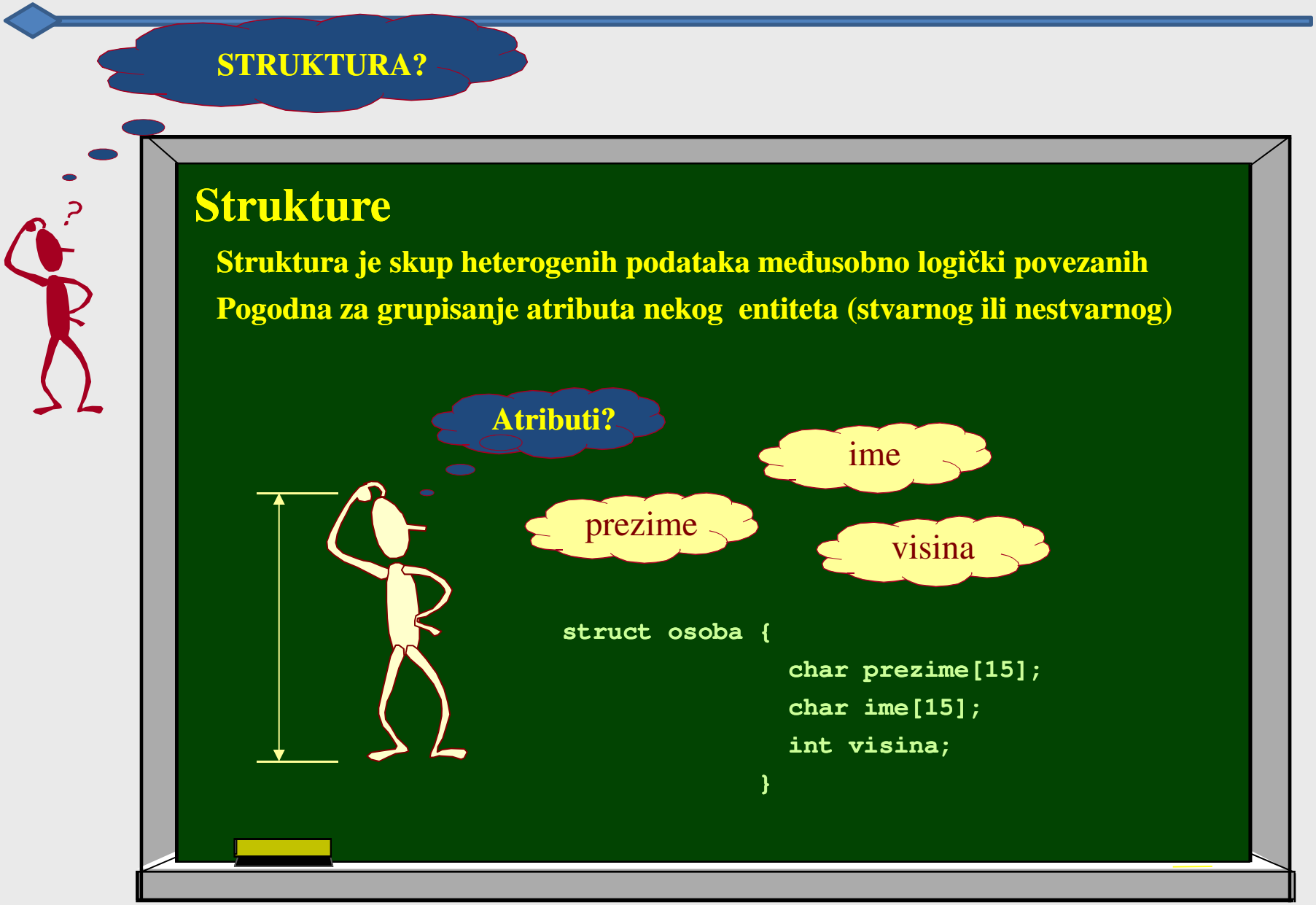
- **UPRAVLJAČKI (NEKI)**
- **13**, Povratak na početak reda, **'\r'**
- **12**, Prelaz na novu stranu, **'\f'**
- **11**, Vertikalni pomeraj, **'\v'**
- **10**, Prelaz u nov red, **'\n'**
- **9**, Horizontalni pomeraj, **'\t'**
- **8**, Povratak za jedno mesto, **'\b‘**
- **7, Zvono**

Znakovi

- Prepoznavanje slova ili cifre u znaku



Strukture


STRUKTURA?
Strukture
Struktura je skup heterogenih podataka međusobno logički povezanih
Pogodna za grupisanje atributa nekog entiteta (stvarnog ili nestvarnog)
Atributi?
ime
prezime
visina
struct osoba {
char prezime[15];
char ime[15];
int visina;
}

Strukture

STRUKTURA?

Deklaracija strukture

Opšti oblik deklaracije:

```
struct ime {
    tip1 element1;
    tip2 element2;
    ...
    tipN elementN;
} lista_promjenljivih;
```

- `struct` — ključna riječ
- `ime` — ime strukture
- `tip1 element1; ...` — Deklaracija elementa (polja)
- `lista_promjenljivih` — lista promjenljivih ovog tipa

Primjer:

```
struct osoba {
        char prezime[15];
        char ime[15];
        int visina;
      } student1, student2, klub[100];
```

Strukture

STRUKTURA?

Pristup elementima (poljima) strukture

Pristup pomoću operatora

"." = ELEMENT STRUKTURE

Primjer:

```
struct osoba {
              char prezime[15];
              char ime[15];
              int visina;
             } student, klub[100];
```

Pristup poljima:

```
student.prezime = "Markovic";    klub[1].prezime = "Jankovic";
student.ime = "Marko";           klub[1].ime = "Janko";
student.visina = 190;            klub[1].visina = 202;
```

Strukture

STRUKTURA?

Primjer: Ilustracija struktura

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

struct evid {
                    char prezime[50];
                    char ime[50];
                    int maticni;
                  } trazena = {"Stojanovic","Radovan",234}, osoba;

int main()
{
int a,b,c;
printf("Upisi Prezime Ime osobe i maticni broj ");
scanf("%s %s %d", osoba.prezime, osoba.ime, &osoba.maticni);
//printf("%s %s %d",osoba.prezime, osoba.ime, osoba.maticni);
a=strcmp(osoba.prezime, trazena.prezime);
b=strcmp(osoba.ime, trazena.ime);
if((!a && !b)&&(osoba.maticni=trazena.maticni))
printf("WANTED");
else
printf("PASS");
getchar();
getchar();}
```

Unije

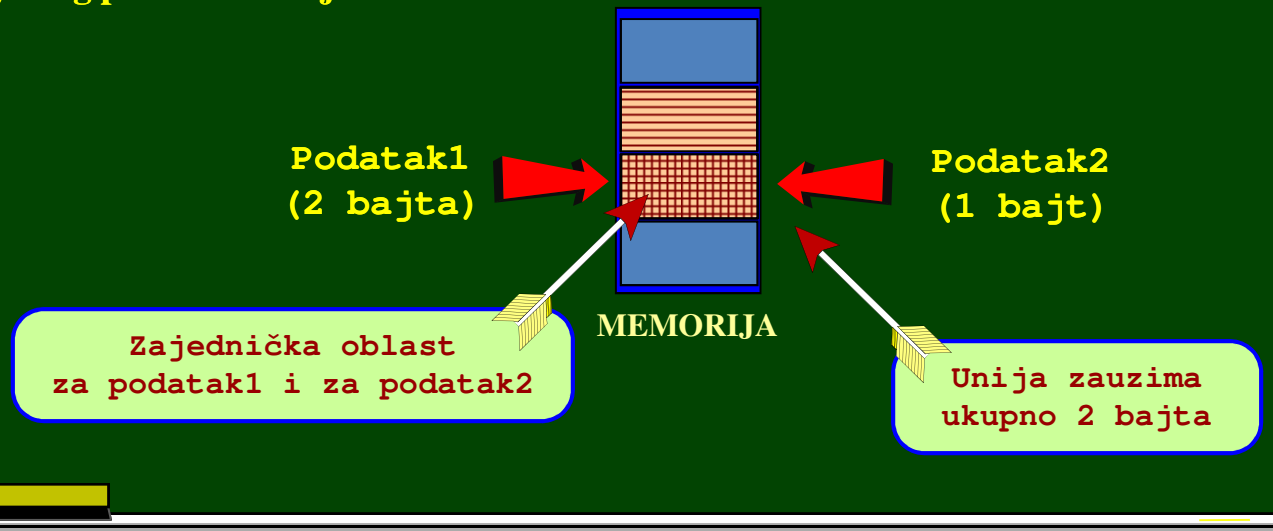
Unije

Unija je skup podataka različitih tipova smještenih u istom memorijskom prostoru.

Podaci u uniji nisu međusobno nezavisni, jer koriste isti memorijski prostor.

Promjena jednog podatka može dovesti i do promjene drugog podatka u uniji.

Unija zauzima u memoriji onoliko bajtova koliko je potrebno za memorisanje najvećeg podatka u uniji



Unije

Deklaracija unije

Opšti oblik deklaracije:

```
union ime {
    tip1 element1;
    tip2 element2;
    ...
    tipN elementN;
} lista_promjenljivih;
```

ključna riječ

ime unije

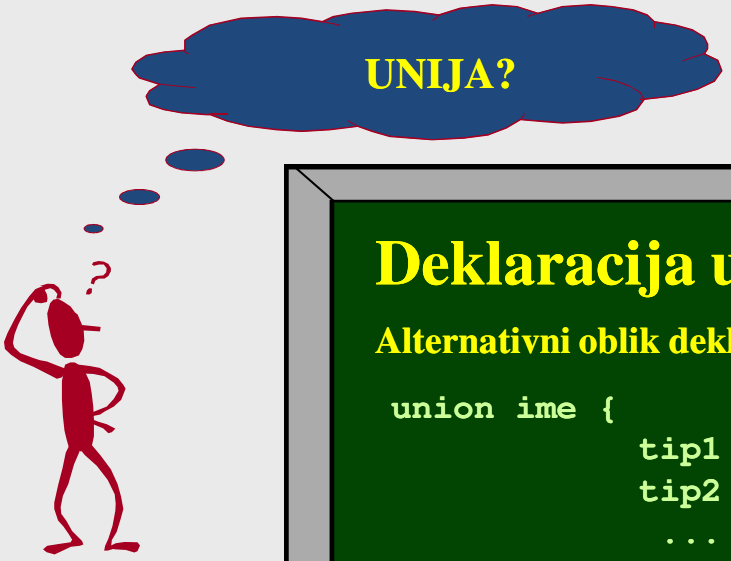
Deklaracija elementa (polja)

lista promjenljivih ovog tipa

Primjer:

```
union unija {
        char c;
        int i;
      } x;
```

Unije



Deklaracija unije

Alternativni oblik deklaracije:

```
union ime {
            tip1 element1;
            tip2 element2;
             ...
            tipN elementN;
          };
union ime lista_prom;
```

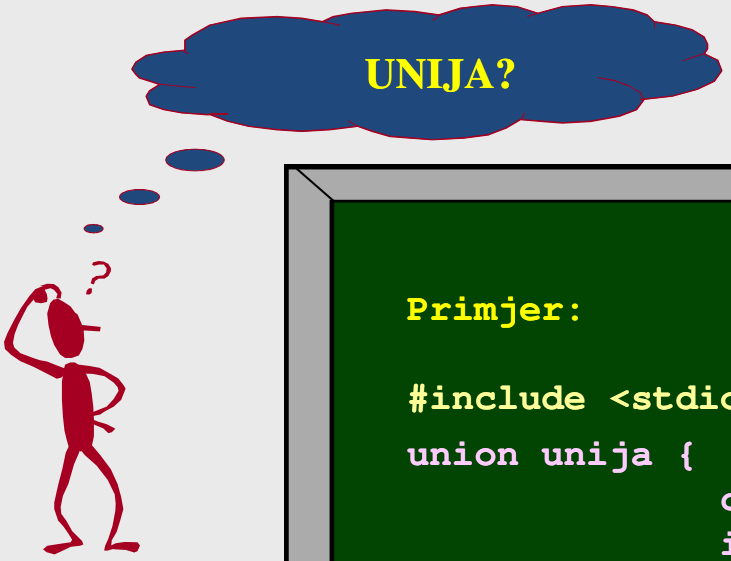
Primjer:

```
union unija {
                char c;
                int i;
                float f;
              };
union unija x;
```

Pristup elementima (poljima) unije

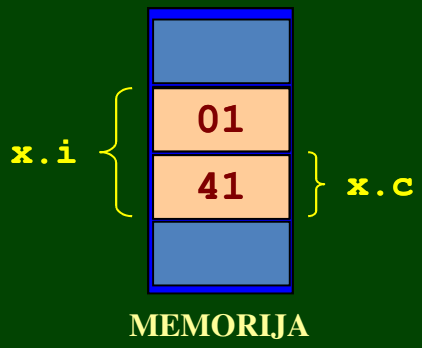
```
x.i = 100;
x.c = 'A';
```

Unije



Primjer:

```
#include <stdio.h>
union unija {
                char c;
                int i;
              };
main()
{
    union unija x;
    x.i = 0x141;
    printf("x.i = %d\n", x.i);
    printf("x.c = %c\n", x.c);
}
```




```
x.i = 321
x.c = A
```

Korisnički tipovi

Kako definisati vlastiti tip?

Definisanje korisničkih tipova

C omogućava:

- definisanje korisničkih tipova
- promjenu imena postojećim tipovima

Opšti oblik:

```
typedef staritip novitip;
```

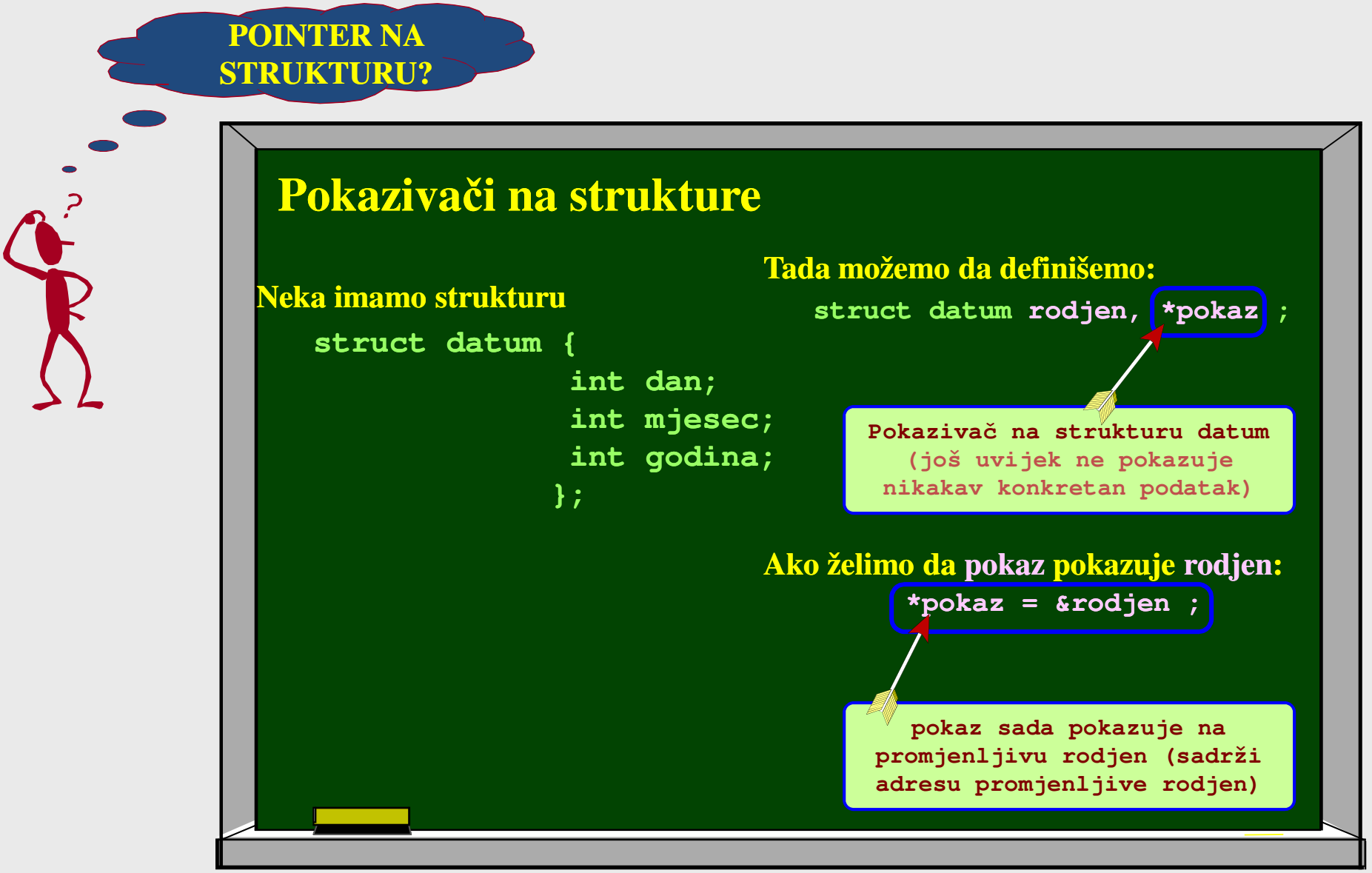
Primjeri:

```
typedef int INTEGER;
INTEGER i,j;
```

```
typedef float NIZ[100];
NIZ niz1, niz2;
```

```
typedef struct {
                  int i;
                  char c;
                } TIP;
TIP niz[10];
```

Pokazivači na strukture



Pokazivači na strukture

POINTER NA STRUKTURU?

Pristup elementima strukture

Neka imamo

```
struct datum {
              int dan;
              int mjesec;
              int godina;
             } rodjen, *pokaz = &rodjen ;
```

Pristup elementima strukture:

```
rodjen.dan = 1;
rodjen.mjesec = 4;
rodjen.godina = 1985;
```

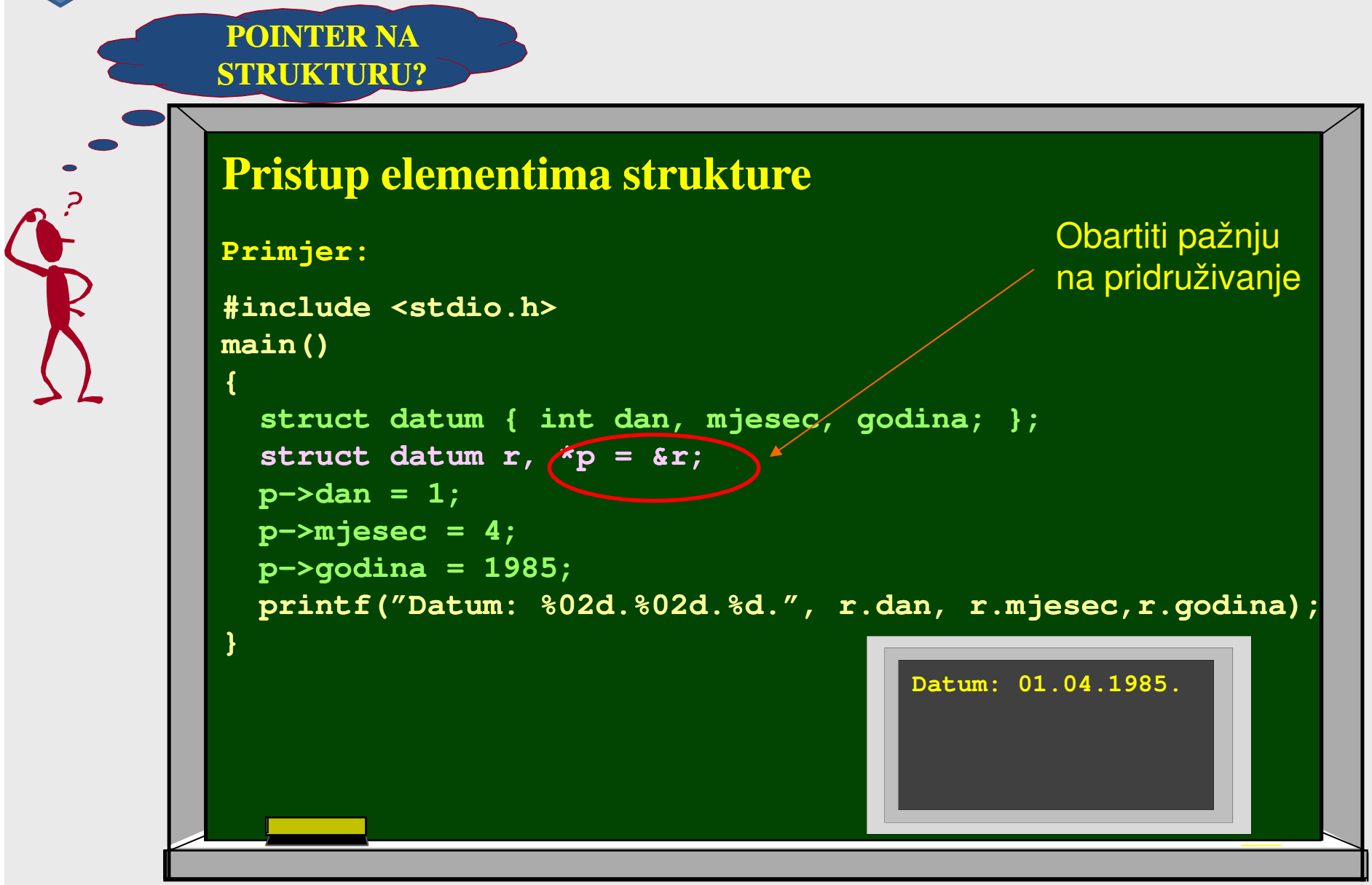
Pristup elementima strukture preko pokazivača:

```
(*pokaz).dan = 1;
(*pokaz).mjesec = 4;
(*pokaz).godina = 1985;
```

ili

```
pokaz->dan = 1;
pokaz->mjesec = 4;
pokaz->godina = 1985;
```

Pokazivači na strukture



Pokazivači na strukture

POINTER U STRUKTURI?

Pokazivač u strukturi

Pokazivač može da bude element strukture

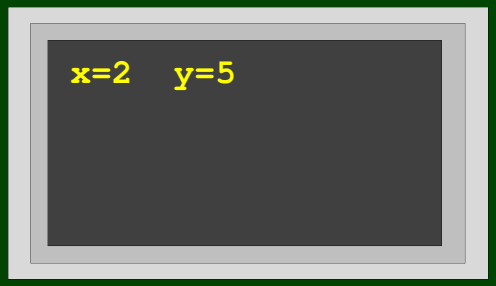
```
struct primjer { int *p1, *p2; } clan;
```

Pristup pokazivaču u strukturi

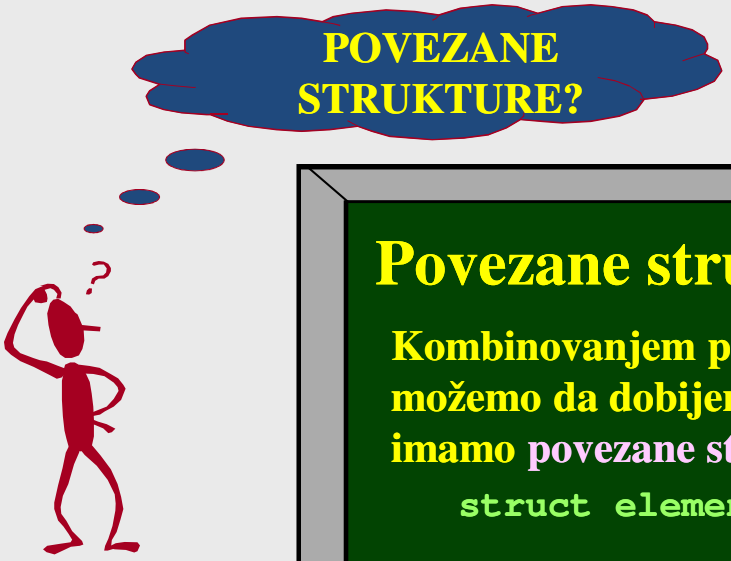
```
*clan.p1    odnosno    *clan.p2
```

Primjer:

```
#include <stdio.h>
main()
{
  int x,y;
  struct primjer { int *p1, *p2; };
  struct primjer proba, *pok = &proba;
  proba.p1 = &x; *proba.p1 = 2;
  pok->p2 = &y; *pok->p2 = 5;
  printf("x=%d  y=%d", x,y);
}
```



Pokazivači na strukture



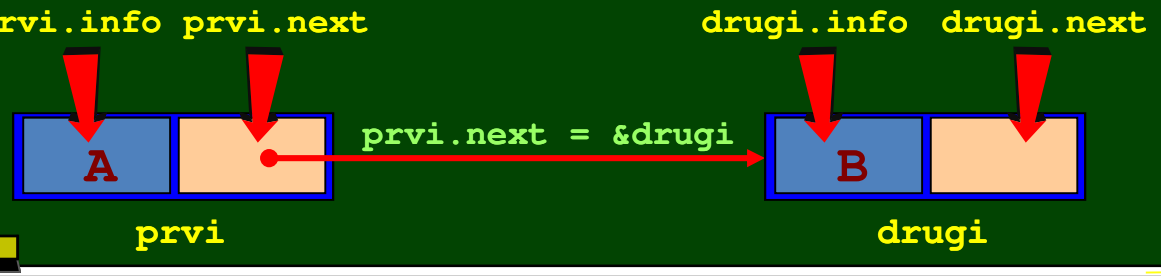
Povezane strukture

Kombinovanjem pokazivača na strukturu i strukture sa pokazivačem možemo da dobijemo strukturu sa pokazivačem na istu strukturu tj. imamo povezane strukture

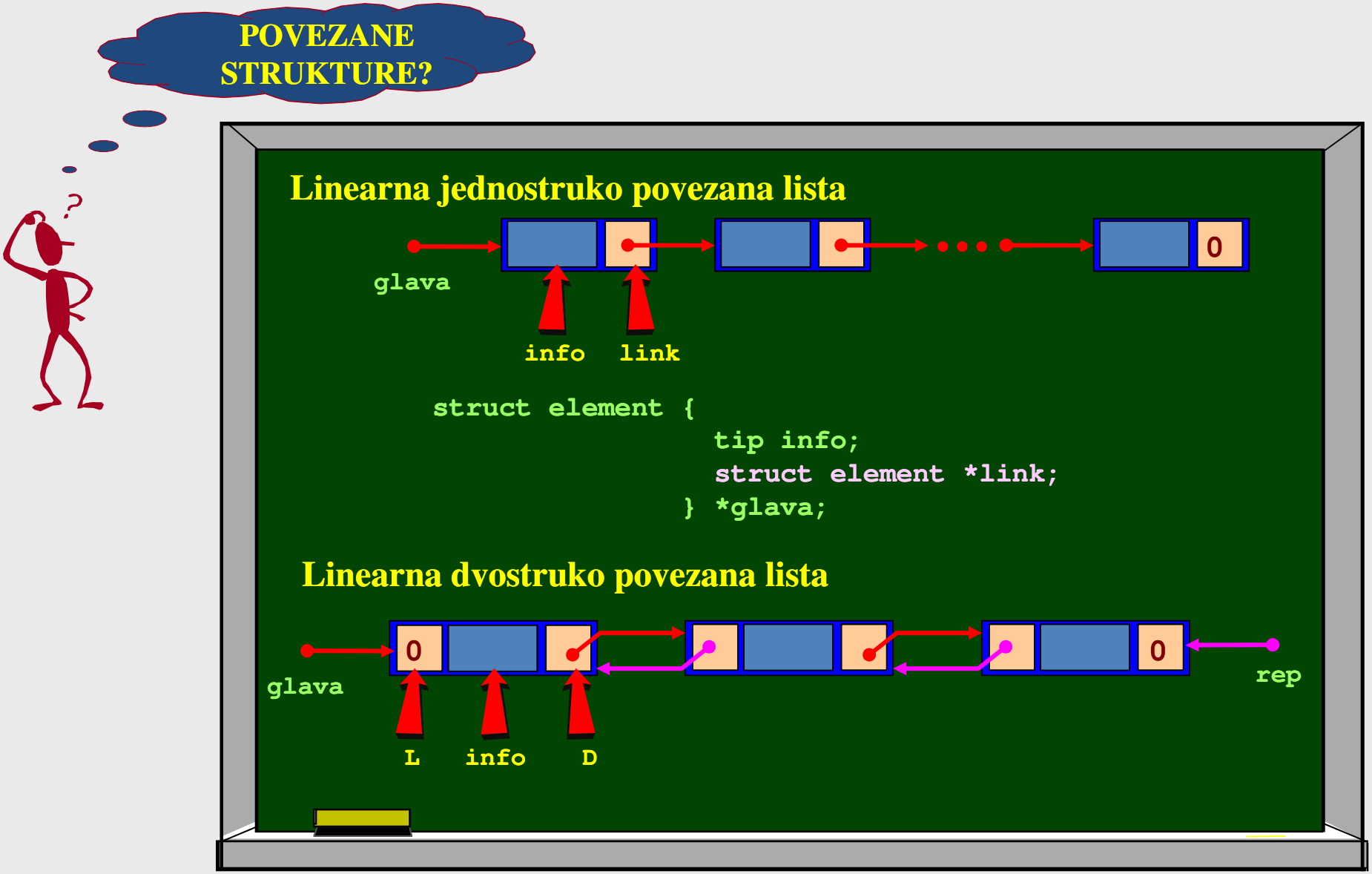
```
struct element {
    tip info;
    struct element *next;
} prvi, drugi;
```

informaciono polje u strukturi

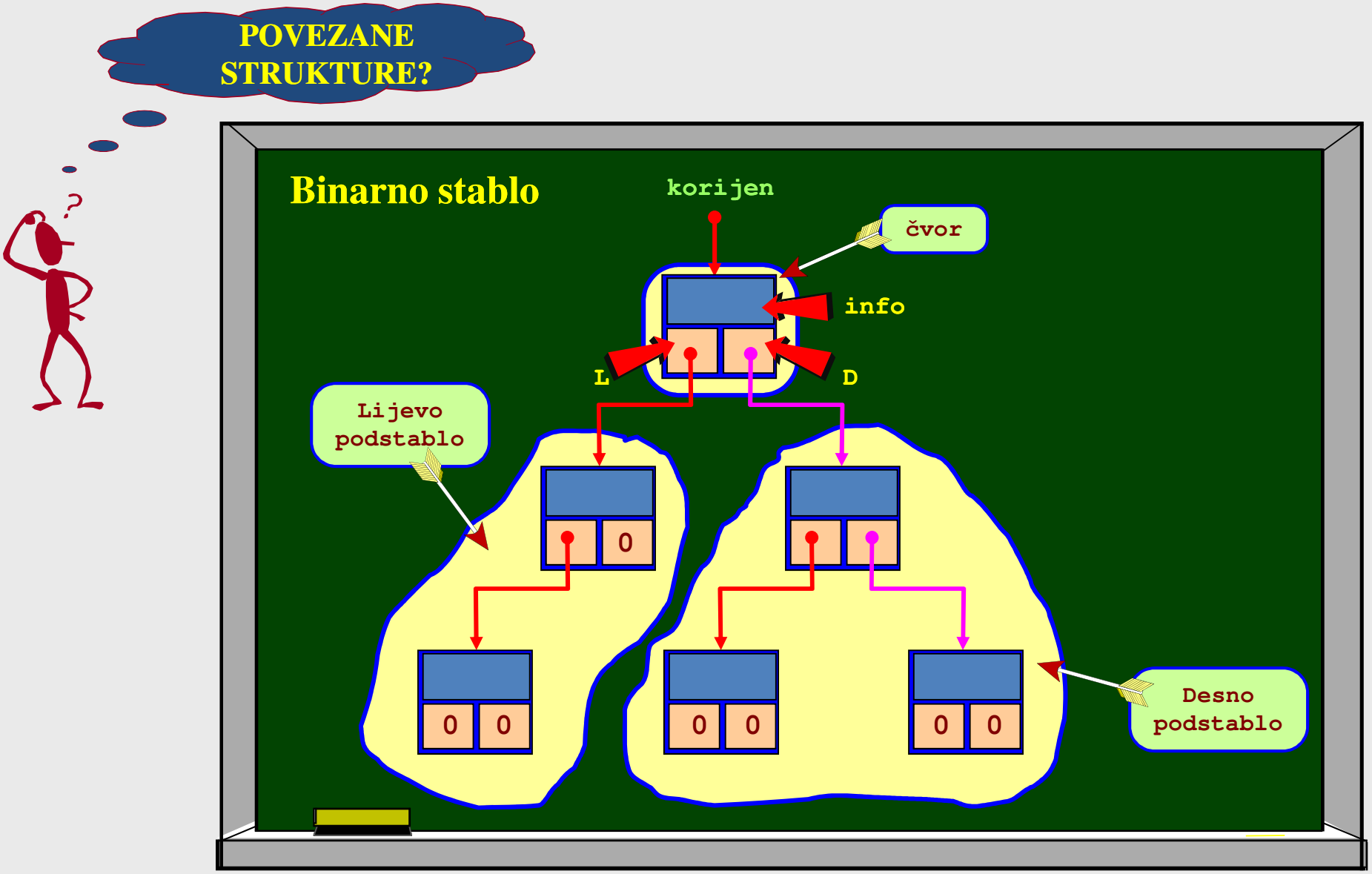
pokazivač na istu strukturu (omogućava povezivanje struktura i pravljenje liste – dinamičkog niza)



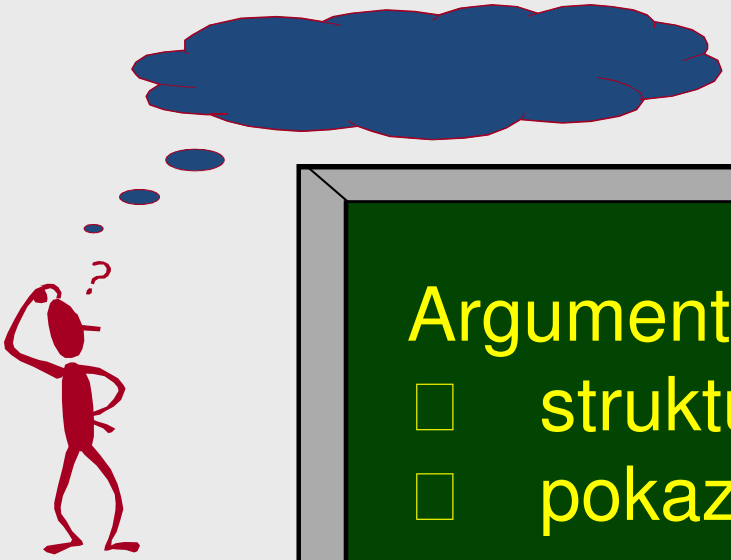
Povezivanje struktura



Povezivanje struktura



Strukture i funkcije



Argument funkcije može biti:

- struktura – po vrijednosti
- pokazivač na strukturu – po referenci

Povratna vrijednost od funkcije može biti:

- struktura
- pokazivač na strukturu

Strukture i funkcije

Struktura- argument funkcije

Struktura se može predati funkciji kao argument, a funkcija može vratiti strukturu onome ko ju je pozvao. Predajom strukture kao argumenta u funkciju se prenose kopije svakog njenog elementa tj. vrši se prenos po vrijednosti.

```
struct adresa{
char ime_ulice[31];
int broj_ulaza; };
void stampa(struct adresa x);
main()
{
struct adresa a1={“Ime”, 1};
stampa(a1);
}
void stampa(struct adresa x)
{
printf(“%s ”, x.ime_ulice);
printf(“%d ”, x.broj_ulaza
}
```

Deklaracija funkcije čiji je argument struktura

Pri pozivu funkcije po vrijednosti strukture formalna struktura x dobija sadržaj stvarne strukture a1

Strukture i funkcije

Pokazivač na niz struktura – argument funkcije

```
struct adresa{
char ime_ulice[31];
int broj_ulaza; };
void stampa(struct adresa niz[ ], int n);
main()
{ struct adresa spisak[ ] = {“Ime1”, 1, “Ime2”, 2, “Ime3”, 3};
stampa(spisak, 3);
}
void stampa(struct adresa niz[ ], int n)
{ int i;
for(i=0; i<n; i++)
{ printf(“%s ”, niz[i].ime_ulice);
printf(“%d ”, niz[i].broj_ulaza); }
}
```

Prenosi se preko adrese „spisak”

Strukture i funkcije

Struktura rezultat od funkcije

```
struct adresa{
char ime_ulice[31];
int broj_ulaza; };
struct adresa izmena(struct adresa x);
main()
{
struct adresa a1 = {“Ime1”, 1}, a2;
a2 = izmena(a1);
printf("%s %d ", a2.ime_ulice, a2.broj_ulaza);
}
struct adresa izmena(struct adresa x)
{
struct adresa y;
strcpy(y.ime_ulice,x.ime_ulice);
puts(“Novi broj ulaza”); scanf(“%d”, &y.broj_ulaza);
return y;}
```

Prima strukturu a1 preko strukture x, kopira x u strukturu y, vraca y, struktura a2 poprime vrijednost y