Универзитет “Св. Кирил и Методиј” во Скопје
Факултет за електротехника и информациски технологии

ПРОГРАМИРАЊЕ И АЛГОРИТМИ

ПОЛИЊА

- Програмски јазик C -

Учебна 2018/19 година

НОВ ПРОБЛЕМ

Проблем: Да се внесат **25 цели броеви** од тастатура и да се пресмета нивниот **производ**. Потоа, **секоја** од внесените 25 вредности **да се измени** на следниот начин: редоследно да се зголеми вредноста за 2*n, каде n=1, 2, 3, ... По менувањето на сите вредности, да се пресмета **сумата од сите 25 изменети** вредности. На крај да се прикажат пресметаниот производ и сума, како и изменетата вредност на 3-тиот по ред внесен број.

Чекори:

25 променливи ????

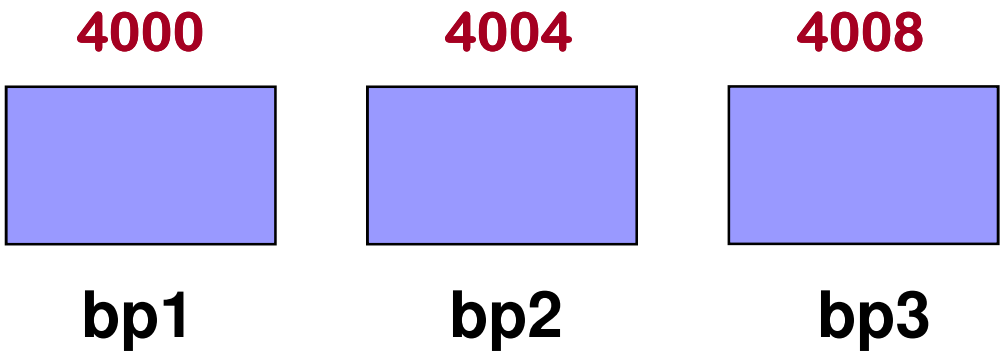
Пристап до внесените вредности

1. Внеси 25 цели броеви
2. Пресметај го нивниот производ
3. Зголеми ги внесените вредности за 2*n, каде n=1,2,3, ...
4. Пресметај ја сумата на изменетите вредности
5. Прикажи ги пресметаниот производ и сума
6. Прикажи ја изменетата вредност на 3-от по ред внесен број
7. Крај

Организација на податоци од ист тип

мал број:

```
int bp1, bp2, bp3;
int total;
```




```
scanf("%d%d%d",&bp1,&bp2,&bp3);
total = bp1 + bp2 + bp3;
```

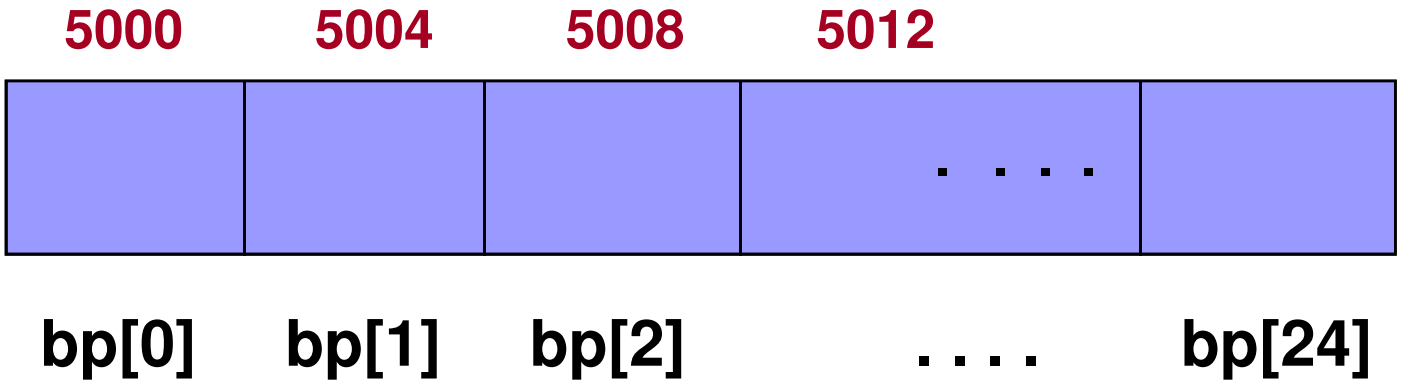
Но, 25 променливи од ист тип?

РЕШЕНИЕ

ПОЛИЊА/ВЕКТОРИ (ARRAY)

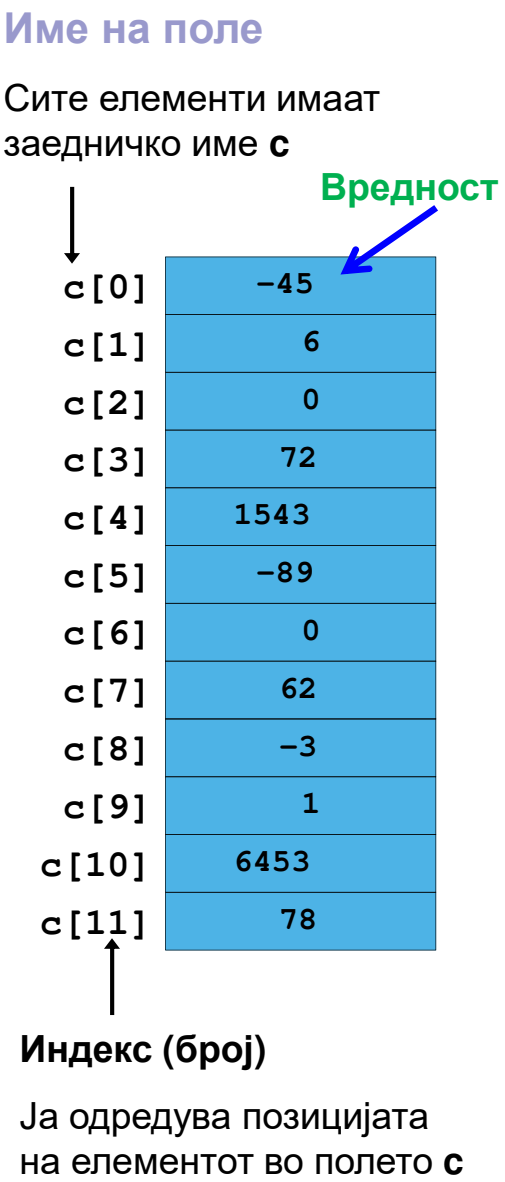
- Структурирани податочни типови
  - колекција од повеќе индивидуални елементи со заедничко име
  - пристап до индивидуалните елементи

25 променливи од ист тип



Вовед во полиња

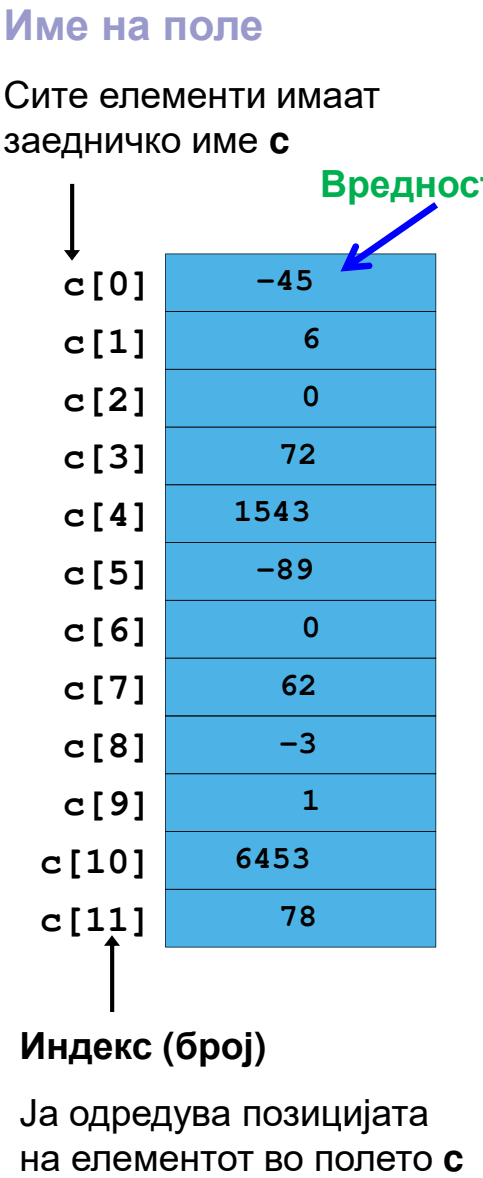
- Карактеристики:
  - структура со релациски **поврзани податоци**
  - **бројот** на елементи е **однапред познат**
  - **сите** елементи се **променливи** од **ист тип**



Вовед во полиња (2)

- **Поле (дефиниција)**:
  - колекција од променливи од ист тип сместени во низа последователни мемориски локации, на кои им е доделено единствено име

- За пристап до кој и да е елемент од полето се користи **името и позицијата** на елементот во полето
  - позицијата се одредува со индекс



Вовед во полиња (3)

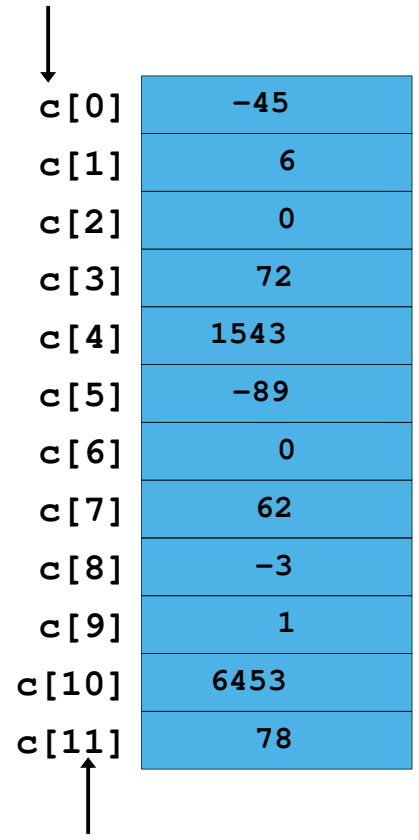
- Според бројот на индекси :
  - едноиндексни (вектори),
  - двоиндексни (матрици), итн...

- Елементите на полето се променливи

```
c[0]= 3;
x=3;
printf("%d", c[0]);
c[5-2] == c[3] == c[x];
```

Име на поле

Сите елементи имаат заедничко име **c**



Индекс (број)

Ја одредува позицијата на елементот во полето **c**

Декларација на едноиндексно поле

- Декларирање на **вектор (едноиндексно поле)**
  - име
  - тип на елементите (променливите)
  - број на елементи во полето

- Формат на наредбата:
  - tip ImePole[BrojNaElementi];

Примери:

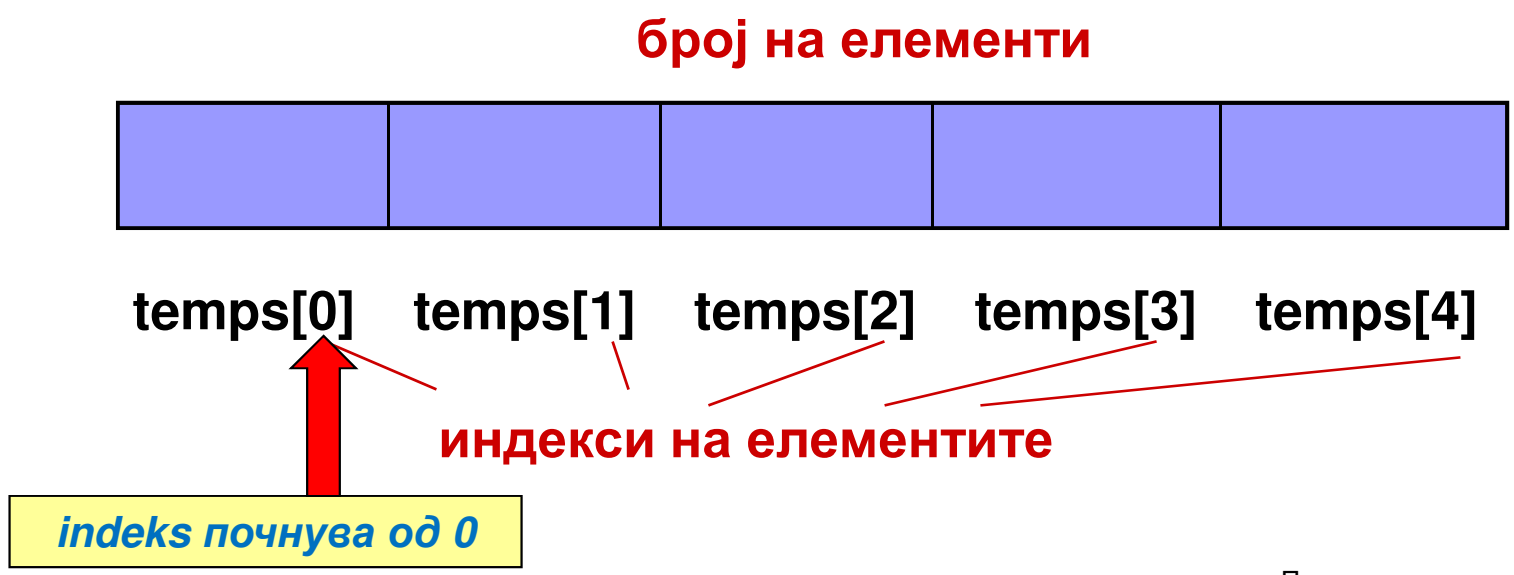
```
int c[10];
float moePole[3284];
double Tez[100];

int b[100], x[27];
```

Декларација на едноиндексно поле (2)

Пример: да се дефинира едноиндексно поле **temps** што содржи 5 индивидуални реални вредности.

`float temps[5];` - декларацијата значи **резервирање мемориски простор**



Декларација на едноиндексно поле (3)

Погрешно:

```
int i;
static int j;
double a[i];   ???
int b[j];      ???
int b[];
```

Иницијализација на вектори

- **Декларација и иницијализација** на елементите на векторот

```
int primer1[10] = { 1,2,3,4,5,6,7,8,9,10 },
n[5] = {1, 2, 3, 4, 5 };
float primer2[4] = {1.0, 0.3, 2.25, 4.5};
```

- Ако нема доволно вредности, најдесните елементи се иницијализираат на 0

```
int pole_so_pet[5] = {0,1,3};/*preostanatite elementi se 0*/

float pole[10]={0.0};/* najednostaven nacin za inicijalizacija na site vrednosti na 0.0 */

int year[80] = {97};/* prviot element ima vrednost 97, a site ostanati 0 */
```

Иницијализација на вектори (2)

- Ако не е определена големината на полето, тогаш истата ја определува компајлерот

```
int n[] = { 1, 2, 3, 4, 5 };  /* 5 почетни вредности, согласно полето ќе биде декларирано како вектор со 5 елемента */
int kf[] = {1,3,5,7,11,13}; /* вектор со 6 елементи */
int size[] = {3,1,5,99,18,-1}; /* вектор со 6 елементи */
```

Погрешно:

```
int a[10], b[10];
a=0; b=a;
```

Пристап до елемент на вектор

- **Пристап:** **име** на полето **и позиција** на елементот во полето
- **Формат:**

  `imePole[`*`IndeksPozicija`*`]`
- **Индекс:** само целобројна вредност
- За вектор со n елементи и име **c** важи:

  `c[0], c[1]...c[n-1]`

  - **првиот елемент се наоѓа на позиција (има индекс) 0**
  - вториот на позиција (има индекс) 1, итн.

  - одговорност на програмерот е да обезбеди индексот да биде во интервалот [0,n-1] !!!!!!!!!!!!

Пристап до елемент на вектор (2)

- Вектор може да се иницијализира и со следната низа наредби

```
int i, n[10];
for (i=0; i<10; i++) n[i] = i;  ??? Вредности?
```

- **Пример:** собирање на вредностите на елементите на еден вектор

```
int i, a[10], n=5, suma=0;
for(i=0; i<n; i++)  suma=suma+a[i];
```

- **Пример:** собирање на вредностите на елементите на два вектора и нивно сместување во трет вектор

```
for(i=0; i<n; i++)
        c[i]=a[i]+b[i];
```

Пристап до елемент на вектор (3)

- **Пример:** Ако важи следната дефиниција

```
int array[35];
```

тогаш можни се следните доделувања:

```
array[19] = 3 * array[32];
array[-3] = array[500];
```

бидејќи во C не се проверуваат границите, оваа наредба може да се употреби во програмите, но, користење на индекс надвор од границите на полето предизвикува програмата да пристапи до локации надвор од полето!

Пристап до елемент на вектор (4)

- Операторот што го одредува индексот има најголем приоритет, поради што во следниот израз:

```
a[2]++          /* a[2]= a[2]+1; */
```

ќе се зголеми вредноста на променливата што се наоѓа на **третата позиција во полето** за 1.

Пристап до елемент на вектор (5)

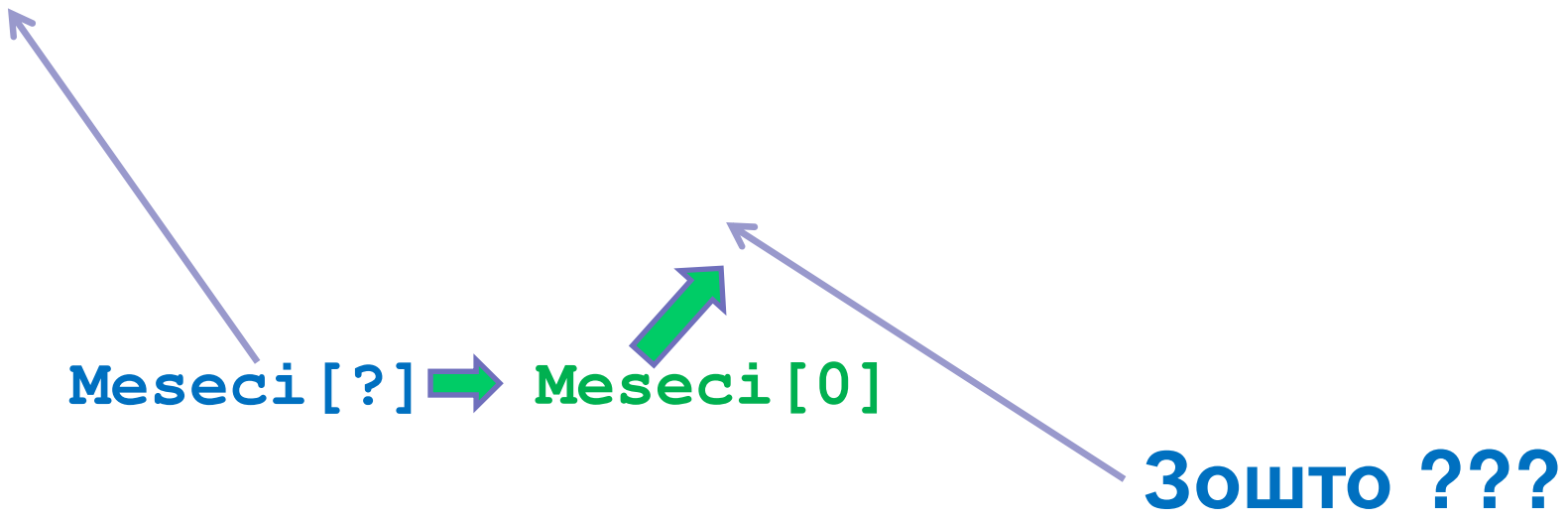
Пример: Да се определи просечната оценка на **сите студенти од прва година** по предметот Програмирање и алгоритми.

```
#include <stdio.h>
int main() {
        float suma = 0;  int i, ocenka[450];
        for(i = 0; i < 450; i++) {
                printf("Vnesete ocenka ");
                scanf("%d", &ocenka[i]); }
        for(i = 0; i < 450; i++) {
                suma += ocenka[i];
        }
        printf("Srednata ocenka e %f ", suma/450);
        return 0;
}
```

Пристап до елемент на вектор (6)

Пример: Да се прикаже бројот на деновите во сите месеци во годината.

```
#include <stdio.h>
int main() {
        int i, meseci[] = {31,28,31,30,31,30,31,31,30,31,30,31};
        for(i = 1; i < 13; i++)
                printf("Mesecot broj %d ima %d denovi\n",i,meseci[i-1]);
}
```



Пристап до елемент на вектор (7)

Пример: Да се напише програмски код кој ќе овозможи проверка на **идентичноста на два вектора**

1. Два вектора се идентични ако имаат **ист број на елементи** и ако елементите на двата вектора што се наоѓаат на **иста позиција имаат идентична вредност**;

1. **Не е дефинирана наредба A==B** што овозможува проверка дали две полиња имаат идентична содржина,
   - оваа наредба **може** да се напише во C,
   - нејзиното значење е **поинакво**

Пристап до елемент на вектор (8)

```
if(n1 == n2) {
        for(IstiSe = 1, i=0; IstiSe && (i<n1); i++)
        if(a[i]!=b[i]) IstiSe = 0;
}  else IstiSe = 0;
```

Друг начин да се направи истото:

```
for(i=0; a[i]==b[i] && i<n1; i++);
if(i>=n1) istise....
```

Пример од вектори

```
#include <stdio.h>
#define SIZE 10

int main(){
        int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
        int i, j;
        printf( "%s%13s%17s\n", "Element", "Vrednost", "Histogram" );
        for ( i = 0; i < SIZE; i++ ) {
                printf( "%7d%13d        ", i, n[ i ]) ;
                for ( j = 1; j <= n[ i ]; j++ ) printf( "%c", '*' );
                printf( "\n" );
        }
        return 0;
}
```

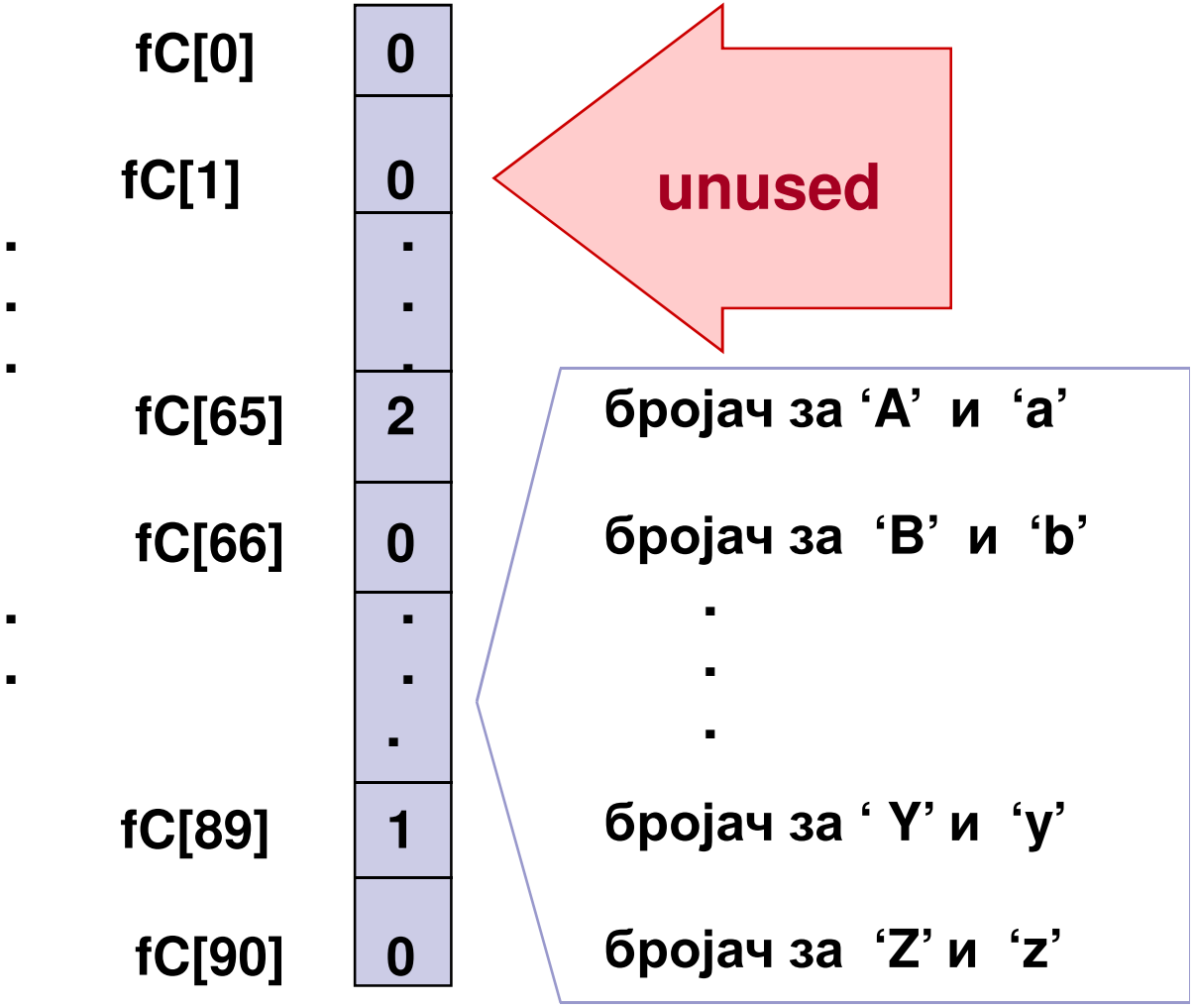
```
Element        Vrednost     Histogram
      0              19     ********************
      1               3     ***
      2              15     ***************
      3               7     *******
      4              11     ***********
      5               9     *********
      6              13     *************
      7               5     *****
      8              17     *****************
      9               1     *
```

Вектори и организација на бројачи

ПРИМЕР: Да се напише програма која ќе изброи колку пати секоја буква се појавува во текст внесен од тастатура.

letter	ASCII
‘A’	65
‘B’	66
‘C’	67
‘D’	68
.	.
.	.
.	.
‘Z’	90

```
const int SIZE=91;
int fC[SIZE]={0};
```



ASCII Code табела

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Алгоритам

- Иницијализирај поле од бројачи на 0 (индекси се големите латинични букви/нивните кодови од 65 до 90)

```
const int SIZE=91;
int fC[SIZE]={0};
```

- Внесувај знак по знак се додека не се означи крај (EOF (Ctrl–C))
- Само ако знакот е буква
  - Провери дали е мала
  - Ако е мала буква претвори ја во голема
  - Добиената буква е индекс во векторот
  - Зголеми го бројачот на таа позиција (индекс) за 1
- Кога ќе се означи крај на внесување испечати ги бројачите за буквите од ‘A’ до ‘Z’

Реализација на програмата

```
#include < stdio.h >
#include <ctype.h>
#define SIZE 91
int main () {
        int m, fC[SIZE];
        char ch, index;
        for(m = 0 ; m < SIZE ; m++ ) fC[ m ]=0;
        while ( (ch = getchar()) != EOF ) {
                if ( isalpha( ch ) ) {
                        if ( islower(ch) ) ch = toupper(ch);
                        fC[ ch ] += 1; /* ili fC[ch]++; */
                }
        }
        printf("Tekstot sodrzi\n ");
        printf("Bukva      Broj na pojavi\n" );
        for( index = 'A'; index < = 'Z'; index++ )
                printf("%c \t%d\n", index, fC[ index ] );

        return 0;
}
```

Враќа 1 ако `ch` е буква

Враќа 1 ако `ch` е мала буква

Враќа ASCII код на големата буква која одговара на `ch`

Повеќеиндексни полиња

- Дозволени се полиња со повеќе индекси
- **Матрица** или **двоиндексно поле**
  - Табела со редови и колони (*m* по *n* елементи)
  - При декларација на матрица **прво** се определуваат **редиците**, а **потоа колоните**

- **Формат:** `tip Ime[Redovi][Koloni];`

	Колона 0	Колона 1	Колона 2	Колона 3
Ред 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Ред 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Ред 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Име на матрица

Индекс на ред

Индекс на колона

Повеќеиндексни полиња (2)

- Во **компјутерската меморија** сите елементи на матрицата ќе бидат сместени во **последователни мемориски локации на следниот начин (по редови):**

a[0][0], a[0][1], a[0][2], a[0][3], a[1][0], a[1][1], a[1][2], a[1][3], a[2][0], a[2][1], a[2][2], a[2][3]

Матрици

- Иницијализација во наредба за декларирање

```
int b[2][2]={{1,2},{3,4}};
```

1	2
3	4

- Ако нема доволно вредности, елементите за кои фалат вредности се поставуваат на нула

```
int b[2][2]={{1},{3,4}};
```

1	0
3	4

- **Пример:**

```
int pole[3][4] = { {26, 34, 22, 17},
                   {24, 32, 19, 13},
                   {28, 38, 25, 20}};
```

е еквивалентно со:

```
int pole[3][4] = {26,34,22,17,24,32,19,13,28,38,25,20};
```

- Првата димензија може да се изостави и матрицата да се декларира на следниот начин:

```
int pole[][4] = { {26, 34, 22, 17},
                  {24, 32, 19, 13},
                  {28, 38, 25, 20} };
```

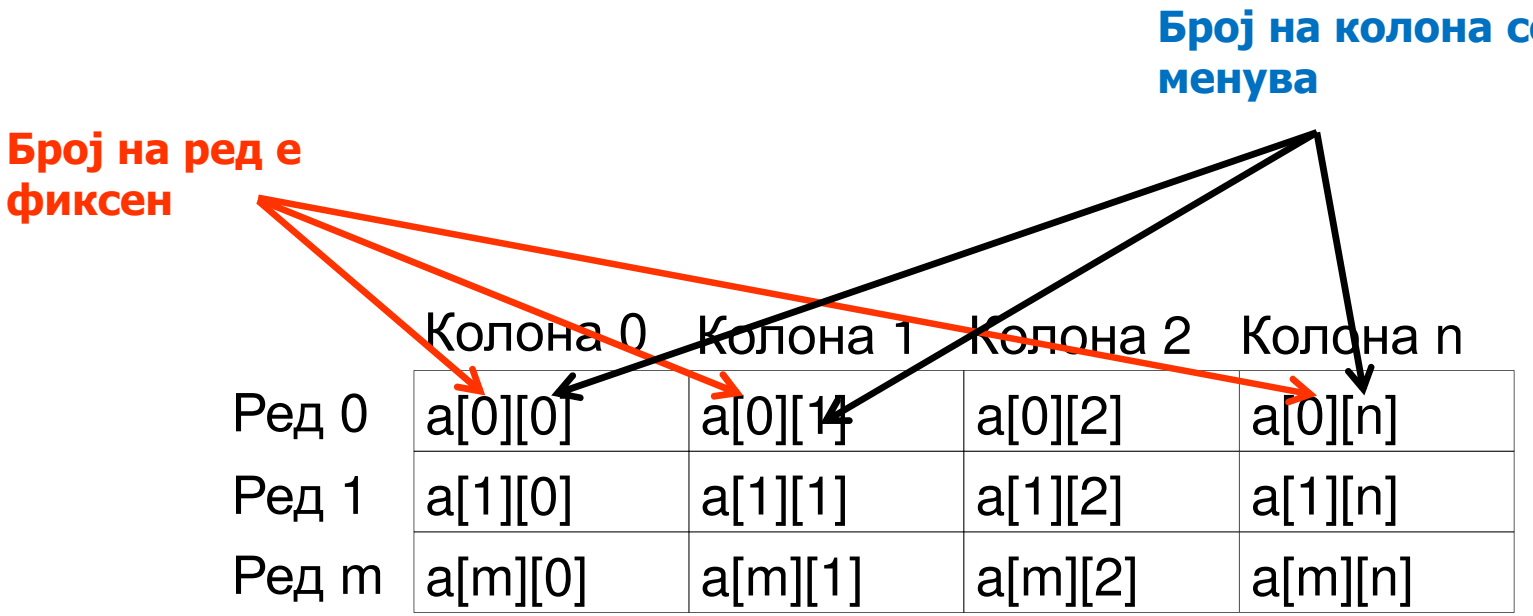
Матрици (2)

- Пристап до елементи на матрицата
  - Име на матрица
  - Ред
  - Колона

```
printf( "%d", b[0][1] );
```

Изминување матрица

- Два циклуси
  - Еден за редици (m редици)
  - Еден за колони (n колони)
  - Надворешниот (првиот) цилкус кажува **како** се изминува матрицата



Изминување матрица (2)

- Ако **m** е бројот на редици
- Изминување по **редици**
  - Внесување елементи во матрица
  - Печатење елементи на матрица
  - Обработка на елементи на матрица

Изминување по колони ???

```
for(i = 0 ; i < m ;  i++ ) {
        for(j = 0 ; j < n ;  j++ )
                printf("A[%d][%d]=%d ",i,j, A[i][j]);
        printf("\n");
}
```

Пример за употреба на матрици

```
#include <stdio.h>
int main(){
        int day_tab[2][13] = {      {0,31,28,31,30,31,30,31,31,30,31,30,31},
                        {0,31,29,31,30,31,30,31,31,30,31,30,31} };
        int i, prest, den, mesec, godina;
        printf("Vnesi datum");
        scanf("%d%d%d", &den, &mesec, &godina);
        prest = godina%4==0 && godina%100!=0 || godina%400==0;
        for(i=1; i < mesec; i++) den+=day_tab[prest][i];
        printf ("Vneseniot datum e %d – iot den vo godinata", den);
        return 0;
}
```

Пример 2 за употреба на матрици

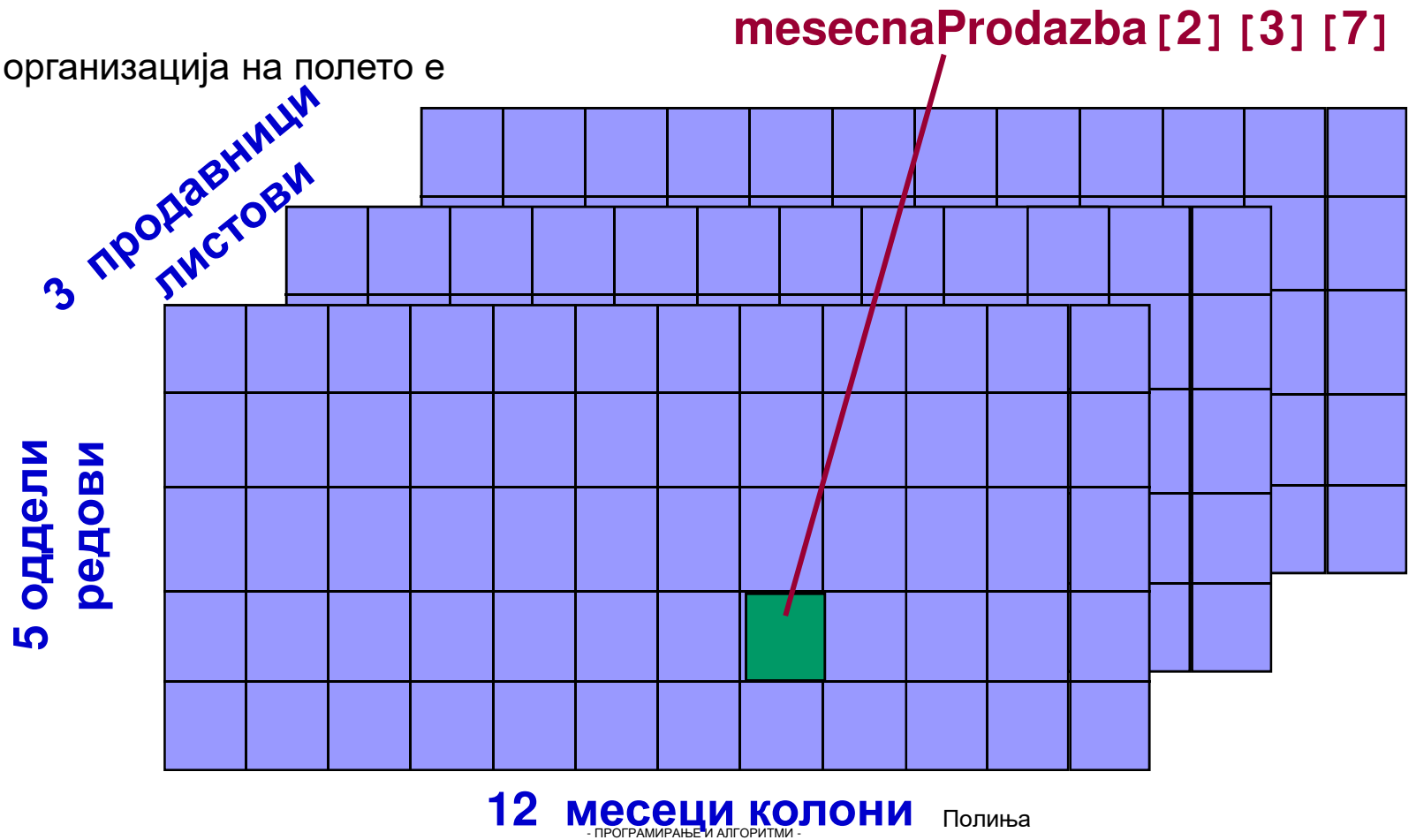
```
#include <stdio.h>
#define RED 10
#define KOLONA 3
int main(){
        int red, kol, vek[RED][KOLONA];
        for(red=0; red<RED; red++)
                for(kol=0; kol<KOLONA; kol++) vek[red][kol]=red+kol;
                        for(red=0; red<RED; red++)
        for(kol=0; kol<KOLONA; kol++)
                printf("Vektor[%d][%d] = %d", red, kol, vek[red][kol]);
        return 0;
}
```

Што прави програмата?

ПРИМЕР за декларирање на ТРОИНДЕКСНО ПОЛЕ

```
#define ODDELI = 5;
#define MESECI = 12;
#define PRODAVNICI = 3;
int mesecnaProdazba [ PRODAVNICI ][ ODDELI ][ MESECI ];
```

Логичката организација на полето е



ПРИМЕР 1 (Полиња и Рекурзија)

- Напомена: да се види примерот после поминување на темата со покажувачи!

- Да напише рекурзивна функција која како аргумент добива низа од цели броеви. Функцијата треба да провери дали низата е строго растечка (секој елемент да е помал од следниот елемент во низата). Доколку низата е строго растечка, тогаш функцијата враќа 1, инаку враќа 0.

```
#include <stdio.h>
#include <stdlib.h>
int rek(int *niza,int n)
{           if(n==1) return 1;
            else if(*niza<*(niza+1))
                        return rek(niza+1,n-1);
            else return 0; }
int main(){             int n,i;
            printf("Vnesete dolzina na niza\n");
            scanf("%d",&n);  int a[n];
            printf("Vnesi elementi na niza\n");
            for(i=0;i<n;i++)
            {scanf("%d",&a[i]);
            } if(rek(a,n)==1) printf("Nizata e rastechka");
                else printf("Nizata ne e rastechka");    return 0; }
```

```
Пример:
intNiza1[]={1,2,8,10,12,15,20};
intNiza2[]={1,2,2,3,5,6,8,9};
int z1=rek(Niza1,7); // враќа 1
int z2=rek(Niza2,8);// враќа 0
```

ПРИМЕР 2 (Полиња и Рекурзија)

- Напомена: да се види примерот после поминување на темата со покажувачи!

- Да се напише рекурзивна функција која што како аргумент добива низа од цели броеви и среда вредност на низата. Функцијата треба да изброи и врати колку од елементите се поголеми од средната вредност на низата (таа што се праќа како аргумент на функцијата).

```
Пример:
intNiza[]={1,2,2,3,5,6,8,9};
int srv=4;
int z1=rek(Niza,srv,8); // враќа4
```

```
#include <stdio.h>
#include <stdlib.h>
int rek(int *a,int sr,int n){
if(n<1)      return 0;
             else { if (*a>sr)
                          return 1+rek((a+1),sr,n-1);
                          else
                          return 0+rek((a+1),sr,n-1);}
}
int main(){              int n,i; int srv=4; printf("Vnesete dolzina na niza\n");
             scanf("%d",&n);  int a[n];
             printf("Vnesi elementi na niza\n");
             for(i=0;i<n;i++)
             {scanf("%d",&a[i]);
             } printf("%d",rek(a,srv,n);        return 0; }
```

ПРИМЕР 3 (Полиња и Рекурзија)

- Напомена: да се види примерот после поминување на темата со покажувачи!

- Да се напише рекурзивна функција која како аргумент добива низа од цели броеви. Функцијата треба да изброи и врати колку парни броеви има во низата, почнувајќи од првиот елемент притоа прескокнувајќи по еден елемент (т.е. секој втор елемент или секој елемент на парна позиција во низата).

```
Пример:
intNiza[]={1,2,2,3,5,6,8,9};
int z1=rek(Niza,8,0,0); // враќа 2
```

```
#include <stdio.h>
#include <stdlib.h>
int rek(int a[], int n, int i, int br) {
        if(i==n) return br;
        else{
                if(a[i]%2==0 && i%2==0){
                        br=br+1;
                        %printf("a=%d i=%d",a[i]),i); %
                }
                return rek(a,n,i+1,br);
        }
}
```

КРАЈ