ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Полиња

Структурно програмирање

ФИНКИ 2014

Задача

- Да се напише програма која ќе го пресмета просекот на 10 броеви внесени од тастатура.

Задача

- Да се напише програма која ќе го пресмета просекот на 10 броеви внесени од тастатура.

```
#include <stdio.h>
int main() {
    int number, n, avg=0;
    printf("Vnesuvaj broevi:\n");
    for(n=0; n<10; n++)
    {
        scanf("%d",&number);
        avg+=number;
    }
    printf("Srednata vrednost na vnesenite broevi e: %f",
                                                (float)avg/n);
    return 0;
}
```

Задача

- Да се напише програма која ќе го пресмета просекот на 10 броеви внесени од тастатура.

```
#include <stdio.h>
int main() {
    int number, n, avg=0;
    printf("Vnesuvaj broevi:\n");
    for(n=0; n<10; n++)
    {
        scanf("%d",&number);
        avg+=number;
    }
    printf("Srednata vrednost na vnesenite broevi e: %f",
                                                (float)avg/n);
    return 0;
}
```

За секој број да се отпечати дали е под или над просекот.

Организација на податоци од ист тип

мал број:

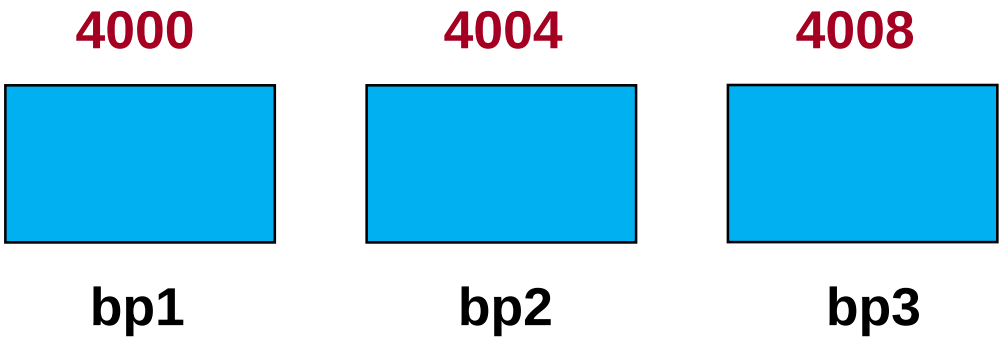
```
int bp1, bp2, bp3;
int total;

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

Организација на податоци од ист тип

мал број:

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

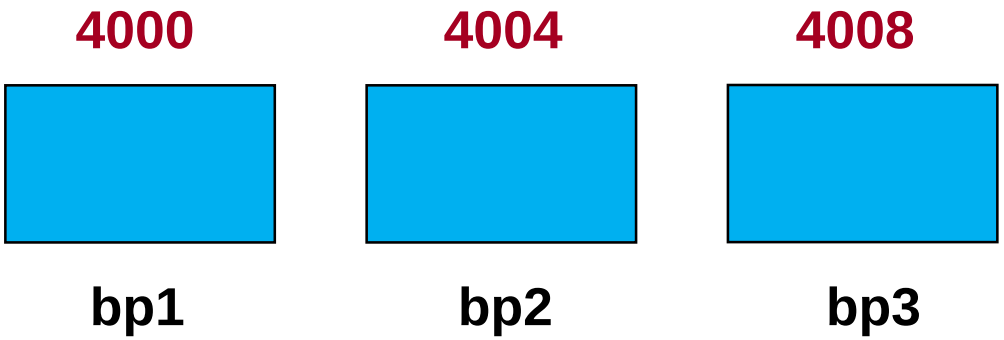



```
scanf("%d%d%d",&bp1,&bp2,&bp3);
total = bp1 + bp2 + bp3;
```

Организација на податоци од ист тип

мал број:

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



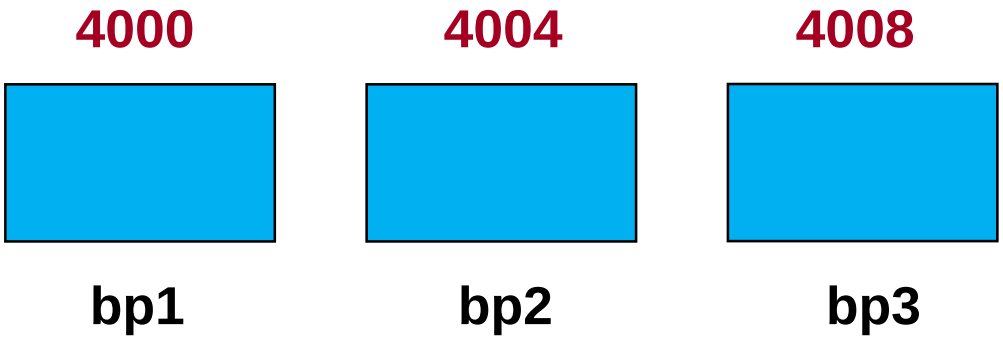

```
scanf("%d%d%d",&bp1,&bp2,&bp3);
total = bp1 + bp2 + bp3;
```

Но, што ако се потребни 100 променливи од ист тип?

Организација на податоци од ист тип

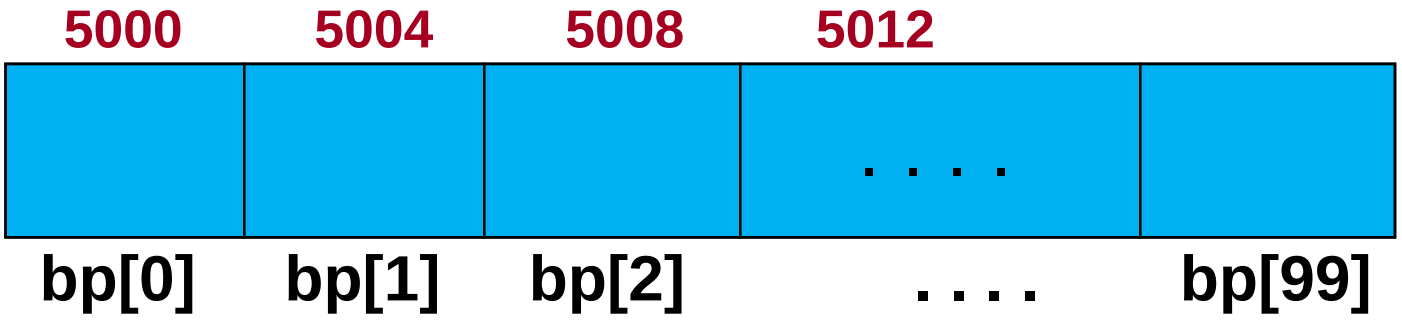
мал број:

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



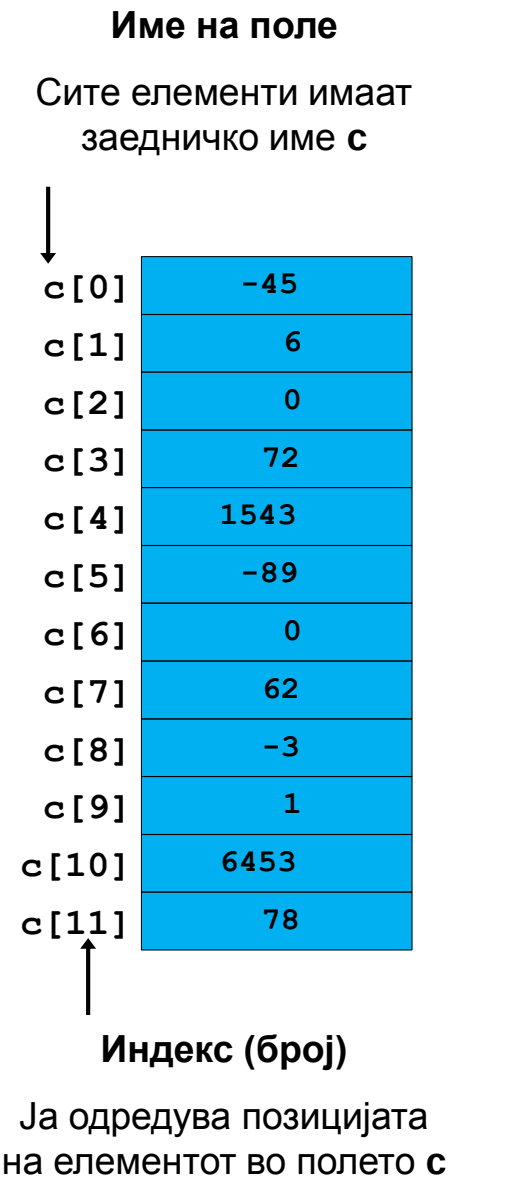

```
scanf("%d%d%d",&bp1,&bp2,&bp3);
total = bp1 + bp2 + bp3;
```

Но, што ако се потребни 100 променливи од ист тип?



Вовед во полиња (низи)

- Поле
  - структура со релациски поврзани податоци
  - бројот на елементи е однапред познат
  - сите елементи се променливи од ист тип
  - колекција од променливи од ист тип сместени во низа последователни меморискилокации, 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**



Поле

Формална (прецизна) дефиниција:

- Поле е хомогена, линеарна, податочна структура со директен пристап

Неформална дефиниција

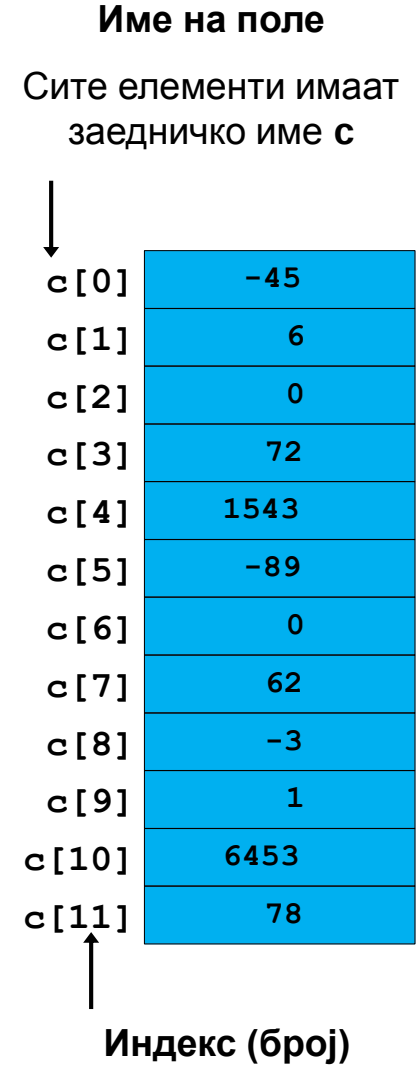
- Променлива која содржи повеќе елементи од ист податочен тип
- Колекција од повеќе индивидуални елементи со заедничко име (овозможен е пристап до индивидуалните елементи)

Вовед во полиња

- Според бројот на индекси полињата се делат на
  - едноиндексни (вектори),
  - двоиндексни (матрици), итн...
- Елементите на полето се променливи

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

x е променлива од целоброен тип



Декларирање на едноиндексно поле

- За декларирање на еден вектор потребно е неговото име, типот на елементите (променливите) и бројот 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];
```

Декларирање на едноиндексно поле

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

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

број на елементи – мора да биде константа

temps[0] temps[1] temps[2] temps[3] temps[4]

индекси на елементите

Декларирање на едноиндексно поле

Погрешно:

```
int i;

static int j;

double a[i];

int b[j];

int b[];
```

Иницијализација на вектори

- Во декларацијата може и да се иницијализираат елементите на векторот

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

- Ако не се наведени доволно вредности за иницијализирање, најдесните елементи се иницијализираат на 0

```
int jf[5] = {0,1,3}; /* preostanatite elementi se 0 */
int area[10]={0}; /* najednostaven nacin za inicijalizacija na site vrednosti na 0 */
int year[80] = {97}; /* prviot element ima vrednost 97, a site ostanati 0 */
```

Иницијализација на вектори

- Ако се врши иницијализација, големината на полето може да се изостави во декларацијата (ја одредува компјутерот)

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

Погрешно:

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

Пристап до елемент на вектор

- Да се пристапи до елемент од полето, потребно е да се зададе името на полето и позицијата на елементот во полето
  - формат:

    imePole[*IndeksPozicija*]
  - за поле 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.
  - индекс на поле може да биде само целобројна вредност
  - одговорност на програмерот е да обезбеди индексот да биде во интервалот [0,n-1] !!!

Пристап до елемент на вектор

- Поле може да се иницијализира и со следната низа наредби

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

пример:

- собирање на вредностите на елементите на две полиња и нивно сместување во трето поле

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

Пристап до елемент на вектор

Пример: Ако важи следната дефиниција

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

тогаш можни се следните доделувања

.

.

.

Пристап до елемент на вектор

Пример: Ако важи следната дефиниција

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

тогаш можни се следните доделувања

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

.

.

Пристап до елемент на вектор

Пример: Ако важи следната дефиниција

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

тогаш можни се следните доделувања

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

Пристап до елемент на вектор

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 поле

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

a[2]++

компјутерот ќе ја зголеми вредноста на променливата што се наоѓа на третата позиција во полето a за 1.

Задачата...

Да се напише програма која ќе го пресмета просекот на 10 броеви внесени од тастатура. **За секој број да се отпечати дали е под или над просекот.**

```
#include <stdio.h>
int main() {
    int broj[10], n;
    float prosek=0;
    printf("Vnesuvaj broevi:\n");
    for(n=0; n<10; n++)
        scanf("%d",&broj[n]);
    for(n=0; n<10; n++)
        prosek+=broj[n];
    prosek/=n;
    printf("Srednata vrednost na vnesenite broevi e: %f\n", prosek);
    for(n=0; n<10; n++)
        printf("%d %s %f\n", broj[n],
                        broj[n]>prosek?"> ":"<=", prosek);
    return 0;
}
```

Пристап до елемент на поле

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 поле

Пример: Да се напише програмски код што ќе овозможи проверка дали две полиња се идентични

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;
2. Не е дефинирана наредба A==B што овозможува проверка дали две полиња имаат идентична содржина, (оваа наредба може да се напише во C, но нејзиното значење е поинакво)

Проверка дали две полиња се идентични

```
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:

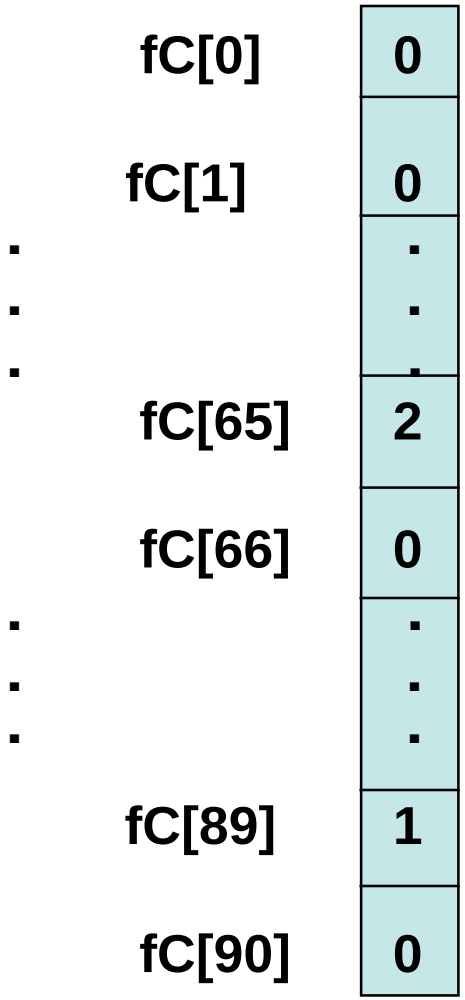
```
#include <stdio.h>
#define SIZE 10
int main() {
    int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
    int i, j;
    printf("%s%13s%17s\n", "Element", "Value", "Histogram");
    for (i = 0; i < SIZE; i++) {
        printf( "%7d%13d          ", i, n[ i ]);
        for ( j = 1; j <= n[ i ]; j++ )
           printf( "%c", '*' );
        printf( "\n" );
    }
    return 0;
}
```

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

Користење на вектори за организација на бројачи

Напиши програма што 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



не се користи

бројач за 'A' и 'a'

бројач за 'B' и 'b'

бројач за ' Y' и 'y'

бројач за 'Z' и 'z'

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

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]++;
   }
 }
 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; }
```

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]++;
   }
 }
 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

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]++;
   }
 }
 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

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]++;
   }
 }
 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

Реализација на програмата

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 91
int main () {
 int m, fC[SIZE] ={0};
 char ch, index;

 while ( (ch = getchar()) != EOF ) {
   if ( isalpha(ch) ) {
      if ( islower(ch) ) ch = toupper(ch);
   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

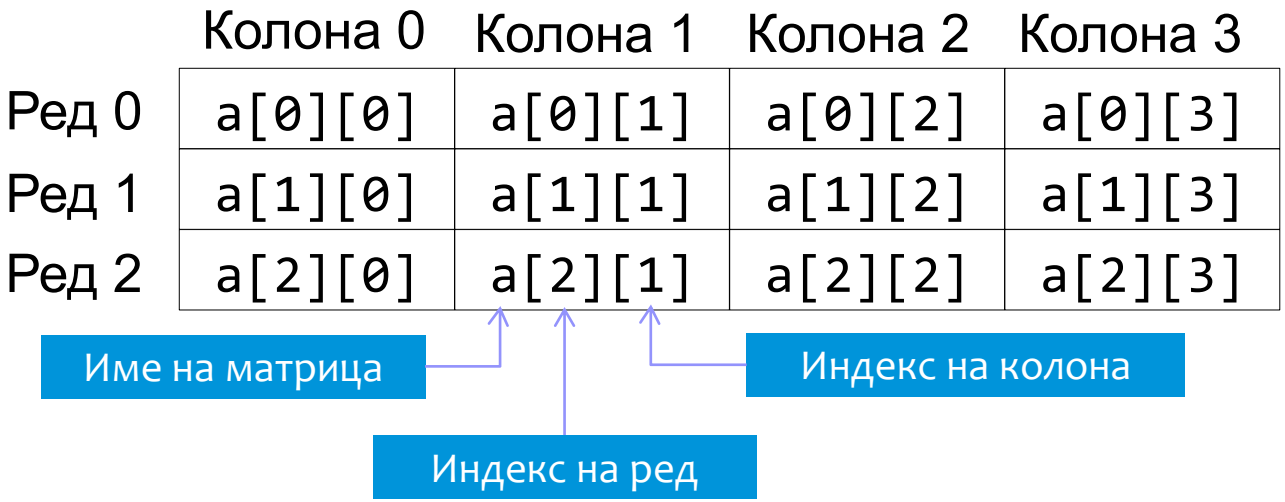
Враќа ASCII код на големата буква која одговара на ch

Повеќеиндексни полиња

- Полињата можат да имаат и повеќе индекси (повеќе димензии)
- Така зборуваме за
  - Едноиндексни полиња (вектори)
  - Двоиндексни полиња (матрици)
  - Повеќеиндексни полиња (3, 4, ...)

Матрици

- 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];



Матрици

- Иницијализација во наредба за декларирање

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

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

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

1	2
3	4

Пример:

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

1	0
3	4

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

- Пристап до елементи на матрицата - по името се задава редот, а потоа колоната во која припаѓа елементот

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

b[1][2] = b[2][1];

scanf( "%d", &b[2][0] );
```

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;
}
```

Задача

Да се напише програма која матица ќе пополни со 1 и 0 во форма на шаховска табла и ќе ја испечати.

```
#include <stdio.h>
#define SIZE 8
int main()
{
    int board[SIZE][SIZE];
    int i,j;

    for(i=0; i<SIZE; i++)
        for(j=0; j<SIZE; j++)
            board[i][j]=(i+j)%2;
    for(i=0; i<SIZE; i++)
    {
        for(j=0; j<SIZE; j++)
            putchar(board[i][j]?'1':'0');
        putchar('\n');
    }
    return 0;
}
```

```
01010101
10101010
01010101
10101010
01010101
10101010
01010101
10101010
```

Задача

Алтернативно решение

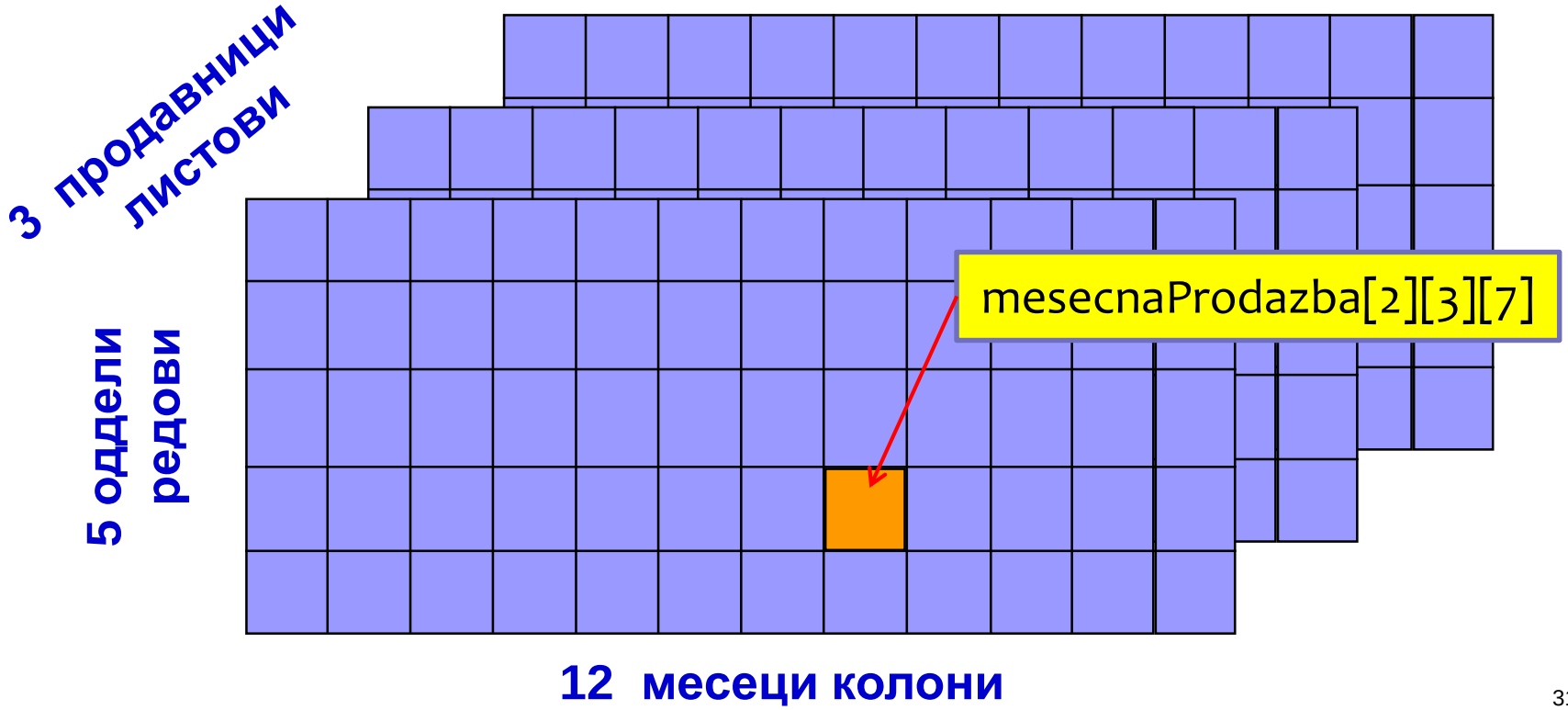
```
#include <stdio.h>
#define SIZE 8
int main()
{
    int board[SIZE][SIZE];
    int i, j, color=0;

    for(i=0; i<SIZE; color=!color, i++)
        for(j=0; j<SIZE; color=!color, j++)
            board[i][j]=color;

    for(i=0; i<SIZE; putchar('\n'), i++)
        for(j=0; j<SIZE; j++)
            putchar(board[i][j]?'1':'0');
    return 0;
}
```

Пример за декларирање на троиндексно поле

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



Прашања?