

---

# Struktura programa

---

nastavak

---

# Automatske varijable

- Varijable deklarirane unutar blokova su implicitno automatske te se stoga ključna riječ `auto` obično ne koristi.
- Ekvivalentno je:

{	{
char <b>c</b> ;	auto char <b>c</b> ;
int i, j, k;	auto int i, j, k;
.....	.....
}	}

# Identifikator `register`

- Identifikator `register` sugerira (ne znači da će to biti i napravljeno) prevodiocu da varijablu smjesti u registar mikroprocesora umjesto u memoriju.
- Uporabom te oznake trebalo bi se dobiti na brzini izvođenja. Oznaka `register` uobičajeno se koristi za najčešće korištene varijable poput npr. kontrolne varijable petlje ili pak brojača.
- Ne može se primijeniti na globalne varijable.
- Registri nemaju adrese – ne može se
  - koristiti pokazivač na varijable čija je oznaka memorijske klase `register`;
  - primijeniti na te varijable adresni operator (`&`).

# Statičke varijable

- Inicijaliziraju se na početku izvršavanja programa i uništavaju se tek nakon izvođenja programa. Time je omogućeno lokalnoj varijabli da zadrži vrijednost nakon napuštanja bloka.
- Ukoliko nisu inicijalizirane eksplicitno, prevodilac će ih inicijalizirati nulom.
- Statičke varijable je moguće inicijalizirati samo konstantnim izrazima:

```
int f(int j)
{
    static int i = j;    /* greška */
    .....
}
```

---

# Lokalne statičke varijable

- Dostupne su samo unutar funkcije u kojoj su definirane.
  - Sadržaj se ne gubi nakon završetka funkcije.
  - Statička varijabla inicijalizira se samo jednom i to prilikom prvog ulaza u blok.
  - Ponovnim pozivom funkcije sadržaj lokalne statičke varijable se obnavlja te je njena vrijednost jednaka vrijednosti koju je imala u prethodnom pozivu funkcije.
-

---

# Primjer:

```
void f(){
    int a = 10;
    static int sa;

    a += 5;
    sa += 5;
    printf("a = %d, sa = %d\n", a, sa);
}
...
for (i = 0; i < 4; ++i)
    f();
...
```

---

# Primjer: Fibonaccijevi brojevi

- 0, 1, 1, 2, 3, 5,... primjer s predavanja uz malu izmjenu:

```
int fibonacci(int n)
{  int i;
   int f;
   static int f0=0, f1=1;
   .....
}
int main(void)
{
  printf("%d\n", fibonacci(4));    /* 3 */
  printf("%d\n", fibonacci(4));    /* 13 */
  .....
```

---

---

# Doseg varijable

- Doseg varijable je područje u kojemu je varijabla dostupna (možemo joj pristupati i mijenjati sadržaj); ponekad se kaže da je ime varijable vidljivo.
  - Prema dosegu varijable se dijele na vanjske (globalne) i unutarnje (lokalne).
  - Svaka varijabla definirana unutar nekog bloka je unutarnja za taj blok. Ona nije definirana izvan tog bloka čak i kad je statička.
  - Statička unutarnja varijabla postoji za cijelo vrijeme izvršavanja programa, ali se može dohvatiti samo iz bloka u kojemu je definirana.
-



---

# Vanjske varijable

- Varijabla definirana izvan svih funkcija naziva se vanjska varijabla.
  - Vanjska varijabla je vidljiva od mjesta definicije do kraja datoteke u kojoj se nalazi.
  - Svaka funkcija može doseći vanjsku varijablu u svom dosegu i promijeniti njenu vrijednost.
  - Na taj način funkcije mogu komunicirati bez uporabe formalnih argumenata.
-

---

# Globalne statičke varijable

- Oznaka memorijske klase `static` ispred vanjske i unutarnje varijable ima različito značenje!
- `static` je implicitni identifikator memorijske klase za globalne varijable:

```
static int brojac;  
int max;  
/* obje varijable imaju identifikator memorijske klase static.*/  
int main(void) { ...}
```



---

# Primjer:

```
int x;  
int main (void)  
{  
    void f(void);  
    int a, b;  
    .....  
}
```

```
int y;  
void f (void)  
{  
    int c, d;  
    .....  
}
```

---

---

# Primjer:

```
int x;
int main(){
    void f(void);
    f();          /* 110 */
    printf("%d\n", y);    /* 'y' : undeclared identifier */
    return 0;
}
int y;
void f(void){
    x = 100;
    y = 10 + x;
    printf("%d\n", y);
}
```

---

---

# Program smješten u više datoteka

- C program može biti smješten u više datoteka. Na primjer, svaka funkcija definirana u programu može biti smještena u zasebnu .c datoteku.
  - Vanjske varijable i funkcije definirane u jednoj datoteci mogu se koristiti u drugoj ako su tamo deklarirane.
  - Za deklaraciju objekta koji je definiran u drugoj datoteci koristimo ključnu riječ `extern`.
  - `extern` ispred deklaracije objekta informira prevodilac da se radi o objektu definiranom u nekoj drugoj datoteci.
-

# Primjer:

datoteka\_1.c

```
A ()
{
    int x;
}
int y; /* definicija */
B ()
{
    static int w;
}
C () {...}
```

datoteka\_2.c

```
D () {...}
extern int y; /* deklaracija */
E () {...}
F ()
{
    int v;
}
```

---

# Dvodimenzionalna polja

---

ponavljanje

---

# Dvodimenzionalna polja

- Dvodimenzionalno polje je jednodimenzionalno polje čiji su elementi jednodimenzionalna polja.
- Dvodimenzionalno polje definira se na sljedeći način:

```
mem_klasa tip ime[izraz_1][izraz_2]
```

- Na primjer,

```
static int m[2][3];
```

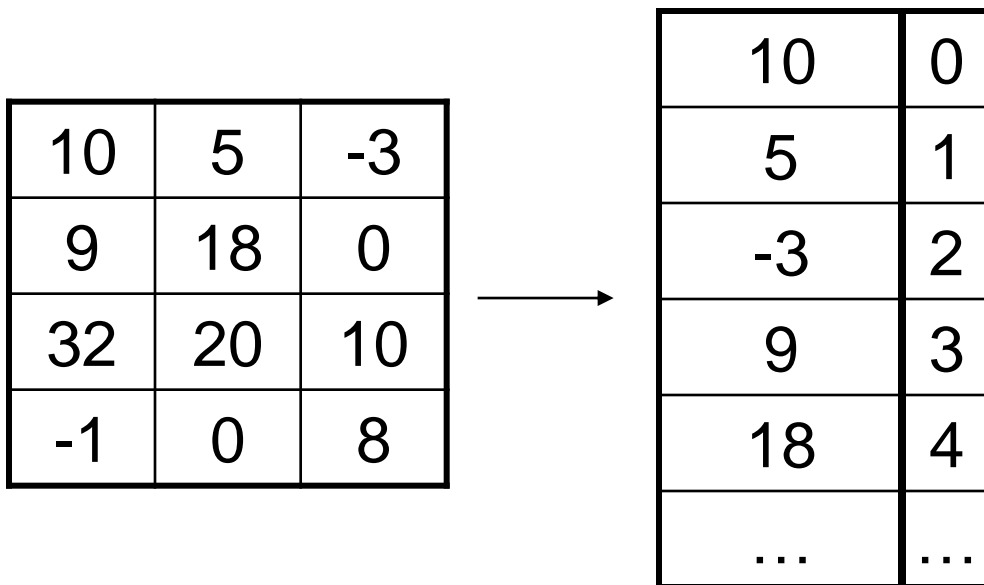
predstavlja matricu s 2 retka i 3 stupca.

---



# Dvodimenzionalna polja (2)

- Poredak elemenata polja m u memoriji:



$$M[1][1] = 18 \Rightarrow 1 * 3 + 1 = 4$$

- $m[i][j] \rightarrow \text{pozicija\_u\_memoriji} = i * \text{broj\_stupaca} + j$   
(Pozicija u memoriji ne ovisi o broju redaka!)

# Inicijalizacija

```
int M[4][3] = {{10, 5, -3},  
               {9, 18, 0},  
               {32, 20, 10},  
               {-1, 0, 8}};
```

```
M[2][1] = 20;
```

M    [0]    [1]    [2]

[0]	10	5	-3
[1]	9	18	0
[2]	32	20	10
[3]	-1	0	8

- Inicijalne vrijednosti ne moraju biti grupirane:

```
int M[4][3] = {10, 5, -3, 9, 18, 0, 32, 20, 10, -1, 0, 8};
```

---

# Inicijalizacija (2)

## Primjer:

```
int M[4][3] = {{10, 5, -3}, {9, 18, 0},  
              {32, 20, 1}, {-1, 0, 8}};
```

```
printf ("%d\n", sizeof(M));
```

```
printf ("%d\n", sizeof(M[0]));
```

```
printf ("%d\n", sizeof(M[0][0]));
```



# Polje kao argument funkcije

- Kada je dvodimenzionalno polje argument funkcije ono se može deklarirati sa svim svojim dimenzijama:

```
int mat[MAX_m][MAX_n];
```

```
.....
```

```
void funkcija(int mat[MAX_m][MAX_n], int m, int n);
```

gdje su  $m$  i  $n$  stvarni brojevi redaka i stupaca.

- Drugi način deklariranja je:

```
void funkcija(int mat[][MAX_n], int m, int n);
```

---

# Primjer: Funkcija euclid

```
double euclid(double
    a[10][10], int m, int n)
{
    int i,j;
    double suma = 0;
    for(i = 0; i < m; i++)
        for(j = 0; j < n; j++)
            suma += a[i][j] * a[i][j];
    return sqrt(suma);
}
```

```
double euclid(double a[10][10],
    int m, int n)
{
    int i, j;
    double suma = 0;
    for(j = 0; j < n; j++)
        for(i = 0; i < m; i++)
            suma += a[i][j] * a[i][j];
    return sqrt(suma);
}
```

Poziv: `printf("%g\n", euclid(a, 3, 5));`

---

# Množenje matrice i vektora: $y=Ax$

```
#include <stdio.h>
#define MAX_m 10
#define MAX_n 10

int main(void)
{
    int A[MAX_m][MAX_n], x[MAX_n], y[MAX_m];
    int m, n; /* Stvarne dimenzije matrice A. */
    int i, j;
    void umnozак(int, int, int mat1[][MAX_n], \
    int mat2[MAX_n], int mat3[MAX_m]);
    .....
    umnozак(m, n, A, x, y);
```

# Funkcija umnozaka

```
void umnozaka(int m, int n, int mat1[][MAX_n], \
    int mat2[MAX_n], int mat3[MAX_m])
{
    int i, j;

    /* Množenje matrice i vektora */
    for(i = 0; i < m; i++)
    {
        mat3[i] = 0;
        for(j = 0; j < n; j++)
            mat3[i] += mat1[i][j] * mat2[j];
    }
}
```

---

# Primjer: Množenje matrica

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

```
int main( )
```

```
{
```

```
    int a[10][10], b[10][20], c[10][20];          /* c = a*b */
```

```
    int m, n, p;
```

```
    int i, j;
```

```
    void umnozak_m(int, int, int, int mat1[10][10], \
    int mat2[][20], int mat3[][20]);
```

```
    printf("Učitajte dimenzije matrica\n");
```

```
    scanf("%d %d %d", &m, &n, &p);
```

---



# Množenje matrica (2)

```
/* Učitavanje matrica */
for(i = 0; i < m; i++)
    for(j = 0; j < n; j++)
        scanf("%d", &a[i][j]);
for(i = 0; i < n; i++)
    for(j = 0; j < p; j++)
        scanf("%d", &b[i][j]);
umnozak_m(m, n, p, a, b, c);
/* Ispis rezultata */
for(i = 0; i < m; i++){
    for(j = 0; j < p; j++)
        printf("%d ", c[i][j]);
    printf("\n");}
return 0;
}
```

# Funkcija umnozак\_m

```
void umnozак_m(int m, int n, int p, int mat1[10][10], \
    int mat2[][20], int mat3[][20])
{
    int i, j, k;

    /* Množenje matrica */
    for(i = 0; i < m; i++)
        for(j = 0; j < p; j++)
        {
            mat3[i][j] = 0;
            for(k = 0; k < n; k++)
                mat3[i][j] += mat1[i][k] * mat2[k][j];
        }
}
```