

Masīvi valodā C

tips <masīva_vārds>[izmērs 1][izmērs 2] ... [izmērs N]

tips <masīva_vārds>[izmērs]

tips <masīva_vārds>[izmērs 1][izmērs 2]

int a[100]; → satur **100** elementus → no a[**0**] līdz a[**99**]

Piemēri:

int mas[10];

float vid[5];

char text[5][10];

Piemēri:

int mas[10]={ 1,2,3,4,5,6,7,8,9,10};

float vid[5]={ 1.1,2.2,3.3,4.4,5.5};

Masīvi valodā C

`int a[3][4];`

Otrais indekss

Pirmais indekss

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]

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

`int a[3][5]={ 1,2,3,4,5,6,7,8,9,10,11 };`

`int a[3][5]={ { 1,2,3 }, { 4,5,6,7,8 }, { 9,10,11 } };`

1	2	3	4	5
6	7	8	9	10
11				

1	2	3		
4	5	6	7	8
9	10	11		

Masīvu sakārtošana. Piemērs

```
#include <stdio.h>
void main (void)
{
    int arr[10]={ 1,23,4,7,8,0,1,9,4,7};
    int i,j,tmp;
    printf(" Ievadītais masīvs: \n");
    for (i=0; i<10;i++) printf("%d ",arr[i]);           // 1 23 4 7 8 0 1 9 4 7
    printf("\n");
    for (i=0; i<9;i++)
    for (j=0; j<10;j++)
        if (arr[j] < ar[j+1])
        {
            tmp=arr[j];
            arr[j]=arr[j+1];
            arr[j+1]=tmp;
        }

    printf(" Sakārtotais masīvs: \n");
    for (i=0; i<10;i++) printf("%d ",arr[i]);           // 23 9 8 7 7 4 4 1 1 0
    printf("\n");
}
```

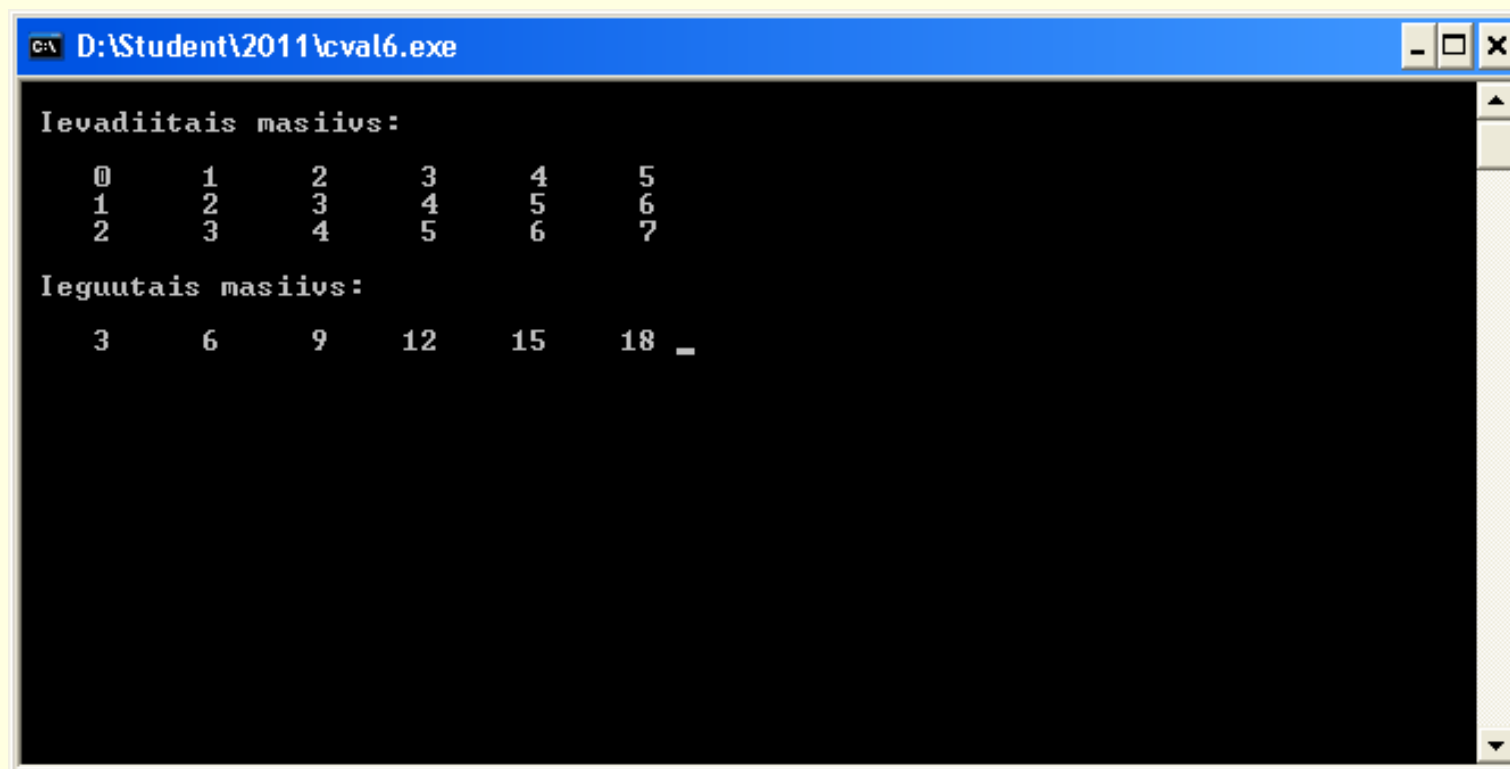
Masīvu apstrāde.Piemērs

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define n 3
#define k 6
void main()
{
    randomize();
    clrscr();
    int mas[n][k],i,j,sum[k],su;
    for (i=0; i<n; ++i)
    for (j=0; j<k; ++j)
        mas[i][j]=random(10);

    for (i=0; i<n;++i)
    {
        printf("\n");
        for ( j=0;j<k;++j)
            printf("%5d ",mas[i][j]);
        }
        su=0;
        for (j=0;j<k;++j)
        {
            for (i=0;i<n;++i)
                su+=mas[i][j];
            sum[j]=su;
            su=0;
        }

        {
            printf("\n\n");
            for (j=0;j<k;++j)
                printf("%5d ",sum[j]);
            }
            getch();
        }
```

Rezultāts



```
C:\ D:\Student\2011\cval6.exe

Ievadiitais masiivs:
  0      1      2      3      4      5
  1      2      3      4      5      6
  2      3      4      5      6      7

Ieguutais masiivs:
  3      6      9     12     15     18 _
```

Rādītāji

C valoda

```
#include <stdio.h>
void main()
{
    int *p1;
    float *p2;
    .....
}
```

Pascal

```
Program my;
Var  p1: ^integer;
      p2: ^real;
Var  p: pointer;
Begin
    ....
End.
```

Rādītāji. Piemērs

```
main()
```

```
{
```

```
int dates[4], *pti, index;
```

```
float bills[4], *ptf;
```

```
pti = dates;
```

```
ptf = bills;
```

```
for (index = 0; index < 4; index++)
```

```
printf ("Rādītāji + %d: %10u %10u \n",
```

```
index, pti + index, ptf + index);
```

```
}
```

Rādītāji + 0: 56014 56026

Rādītāji + 1: 56016 56030

Rādītāji + 2: 56018 56034

Rādītāji + 3: 56020 56038

Masīvi un rādītāji

`dates+ 2 == &dates[2]` `/* viena un tā pati adrese */`

`*(dates+ 2) == dates[2]` `/* viena un tā pati vērtība */`

`*dates + 2` `/* masīva 1.elementa vērtība papildināta ar 2 */`

`*(dates + 2)` `/* masīva 3.elementa vērtība */`

`__(*zippo + n*i + j)` `/* masīva zippo[i][j] elements, kur`
`n – kolonnu skaits */`

Masīvu apstrāde ar rādītājiem

<pre>#include<stdio.h> #include<stdlib.h> #include<conio.h> #define el_sk 6 void main() { clrscr(); randomize(); int mas[el_sk][el_sk],i,j; for(i=0;i<el_sk;++i) for(j=0;j<el_sk;++j) mas[i][j]=random(10);</pre>	<pre>for(i=0;i<el_sk;++i) { printf("\n"); for(j=0;j<el_sk;++j) <u>printf("%2d ",mas[i][j]);</u> } printf("\n");</pre>	<pre>for(i=0;i<el_sk;++i) { printf("\n"); for(j=0;j<el_sk;++j) <u>printf("%2d ",*(*mas+el_sk*i+j));</u> } getche(); }</pre>
---	---	---

Masīvu apstrāde funkcijās. Matricas reizināšana

```
#include <stdio.h>

void multiply (int U[3][3], int V[3][3], int W[3][3]);

main (void)
{
    int A[3][3]={0,1,2,3,4,5,6,7,8};
    int B[3][3]={ 1,2,3,4,5,6,7,8,9};
    int i,j,C[3][3];
    multiply (A,B,C);
    printf ("Masīvs C: \n");
    for(i=0;i<3;i++)
        printf ("%4d %4d %4d \n", C[i][0], C[i][1], C[i][2]);
    return 0;
}
```

Matricas reizināšana (turpinājums)

```
void multiply (int U[3][3], int V[3][3], int W[3][3])
{
    int i,j,k;
    for(i=0;i<3; i++)
        for(j=0;j<3; j++)
            {
                W[i][j]=0;
                for(k=0;k<3; k++)
                    W[i][j] += U[i][k] *V[k][j];
            }
}
```

Masīvu apstrāde funkcijās. Atrast divdimensiju masīva lielāko elementu

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

int max_ind(int masivs[SIZE][SIZE], int n);
int max_rad(int*, int);
int i, j, max;

void main()
{
    int max_elem, masivs[SIZE][SIZE];

    clrscr();

    randomize();

    for(i=0;i<SIZE;i++)
        for(j=0;j<SIZE;j++)
            masivs[i][j]=random(20);
```

```
    printf("Sakummasivs: \n");
    for(i=0;i<SIZE;i++)
    {
        for(j=0;j<SIZE;j++)
            printf("%4d", masivs[i][j]);
        printf("\n");
    }

    max_elem=max_ind(masivs, SIZE);
    printf("Max elements ar indeksiem = %d\n", max_elem);
    max_elem=max_rad(*masivs, SIZE);
    printf("Max elements ar raditajiem = %d", max_elem);
    getch();
}
```

Atrast divdimensiju masīva lielāko elementu (turpinājums)

```
int max_ind(int masivs[SIZE][SIZE], int n)
{
    max=masivs[0][0];
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            if(masivs[i][j]>max)
                max=masivs[i][j];
    return max;
}
```

Atrast divdimensiju masīva lielāko elementu (turpinājums)

```
int max_rad(int *masivs, int n)
{
    max=*masivs;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            if(*(masivs+i*n+j)>max)
                max=*(masivs+i*n+j);
    return max;
}
```