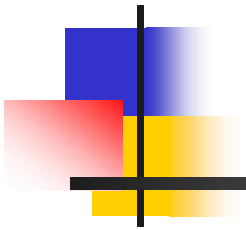
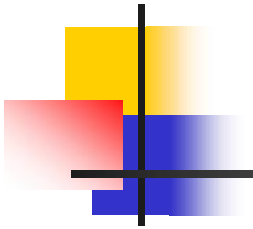


آرایه ها

آرایه ها



- آرایه مجموعه ای از عناصر همنوع است.
- هر آرایه دارای نامی است که مانند متغیرهای معمولی نامگذاری میشود
- آرایه یک مجموعه از محلهای متوالی حافظه است
- هر محل یک عنصر آرایه به شمار می رود.
- برای دسترسی به عناصر آرایه از متغیری بنام اندیس (index) استفاده میگردد. به همین دلیل، آرایه را متغیر اندیس دار میگویند.
- برای دسترسی به هر عنصر آرایه باید محل آنرا در آرایه بدانیم یعنی یک اندیس به محل آن داشته باشیم.

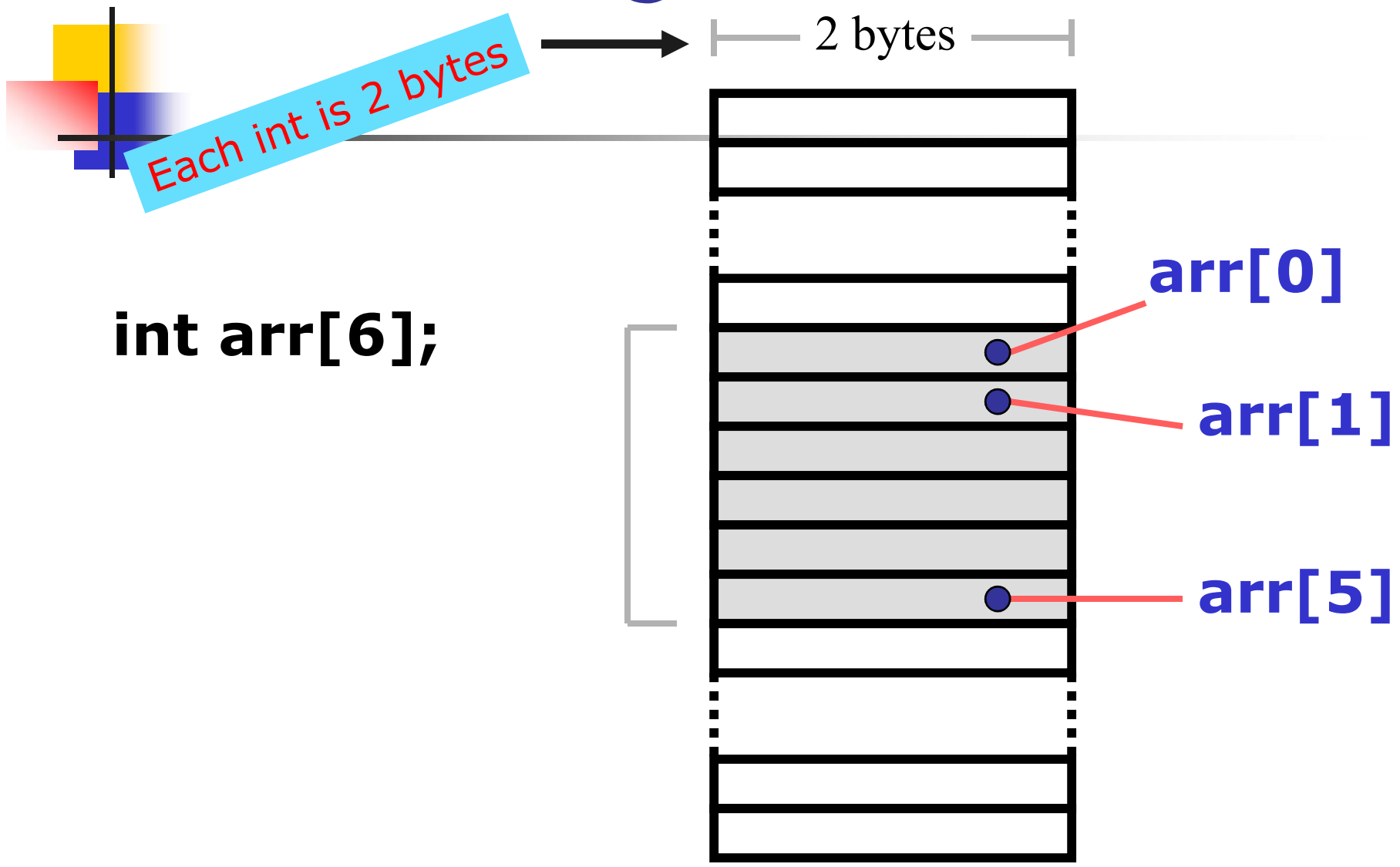
تعریف آرایه یک بعدی

- در آرایه یک بعدی که نام دیگر آن لیست است میتوان با یک اندیس به عناصر آن دست یافت. آرایه های یک بعدی در c++ به شکل زیر تعریف میشوند:

نوع آرایه نام آرایه [طول آرایه];

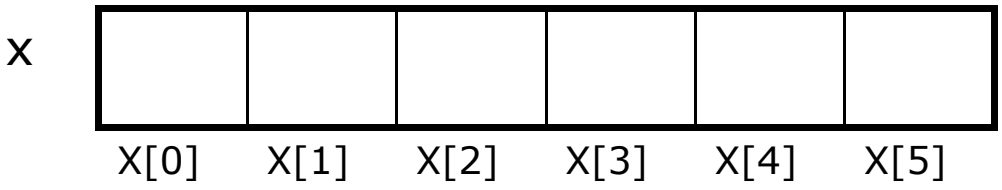
- نوع آرایه (element type) می تواند یکی از انواع داده ای قابل استفاده در زبان C++ باشد.
- نام آرایه (array name) برای دسترسی به عناصر آرایه مورد استفاده قرار میگیرد و از قوانین اسم گذاری متغییرها پیروی می کند..
- طول آرایه (number of elements) که یک عدد ثابت صحیح مثبت است، تعداد عناصر آرایه را تعیین مینماید.

تعریف متغیر آرایه ای از نوع int با ٦ عنصر



دسترسی به عناصر آرایه

- در c++ عنصر اول هر آرایه با اندیس صفر مشخص میشود.
- توجه داشته باشید که عناصر آرایه در خانه های متوالی حافظه قرار دارند.
- پس، آخرین عنصر یک آرایه n عنصری در محل n-1 ام قرار دارد.
- لذا، اگر سعی کنید عنصر n ام را بخوانید با خطای اجرا مواجه خواهید شد.
- برای دسترسی به هر عنصر از اندیس آن استفاده میشود مثلاً x[2] = 5 عنصر دوم را برابر با 5 قرار میدهد.



میزان حافظه و آدرس عناصر

- میزان حافظه ای که به آرایه اختصاص داده میشود به طریق زیر محاسبه میگردد:

میزان حافظه آرایه = (تعداد عناصر آرایه) × (طول نوع داده آرایه)

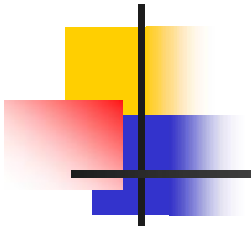
- اگر عنصر اول آرایه در آدرس A قرار داشته باشد و نوع داده آرایه n بایت باشد. آدرس عنصر i ام از رابطه زیر محاسبه میشود:

آدرس عنصر iام = A + n × i

- برای مشخص کردن اندیس هر عنصر می توان از عبارات طبیعی زبان C++ استفاده کرد.

x[17] , x[i+3] , x[a+b+c]

مثال:

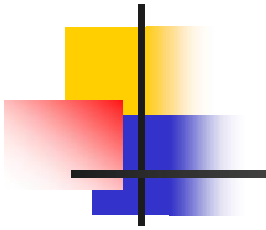


- برنامه ای که با تعریف آرایه ای بطول ۱۰ مقدار هر عنصر آرایه را برابر با اندیس آن عنصر قرار دهد و سپس مجموع مربعات عناصر آرایه را محاسبه کرده و به خروجی ببرد

```
void main()
{
   int array[10] , i , sum = 0 ;
   for (i=0 ; i < 10 ; i++)
        array[i] = i;
   for (i=0 ; i < 10 ; i++)
        sum += array[i] * array[i];
   printf(" the sum square of array: %d " , sum);
   getch();
}
```

```
the sum square of array: 285
```

مثال:



- برنامه ای که معدل ۵ دانشجو را از ورودی گرفته، در آرایه ای قرار دهد و بیشترین معدل و محل وجود آن را پیدا کرده و به خروجی میبرد

```
void main()
{
    const int n=5;
    float ave[n];
    int i , p ;
    for (i=0 ; i < n ; i++)
    {
          printf( "Enter an average:");
          scanf( "%f" , &ave[i]);
    }
    p = 0;
    for (i=1 ; i < n ; i++)
          if (ave[i] > ave[p] )
          {
              p = i;
          }
    printf("\n max = %5.2f, position = %d" , ave[p] , p+1);
    getch();
}
```

```
Enter an average: 12
Enter an average: 15
Enter an average: 12
Enter an average: 17
Enter an average: 16
max = 17.00 , position = 4
```

جهت تسلط به بحث آرایه ها باید موارد زیر مورد توجه قرار گیرند:

- الف) تعریف یک آرایه
- ب) نحوه دستیابی به یک المان از آرایه
- ج) نحوه مقداردهی اولیه آرایه
- د) نحوه ارسال آرایه به تابع
- ه) چند کاربرد از آرایه ها

مقدار دهی اولیه به آرایه ها

- می توان آرایه ها را هنگام تعریف آنها مقدار دهی اولیه نیز کرد.

```
int iArray[5] = { 1,8,3,6,12};
double dArray[2] = { 0.707, 0.707};
char s[] = { 'R', 'P', 'I' };
```

اگر آرایه را مقدار دهی اولیه کنید نیازی به مشخص کردن سایز آرایه نیست.
در مثال فوق طول آرایه S برابر با ۳ در نظر گرفته میشود

آرایه یک بعدی بعنوان آرگومان تابع

- اگر آرایه بعنوان آرگومان تابع باشد، پارامتر معادل آن به سه صورت زیر تعریف میشود:

۱- آرایه ای با طول مشخص

۲- آرایه ای با طول نامشخص که در اینصورت بهتر است طول آرایه نیز بعنوان آرگومانی دیگر منتقل شود

۳- استفاده از اشاره گر که مشابه به آرایه با طول نامشخص است

مثال: چاپ یک آرایه در خروجی

می توان آرایه را به این صورت به تابع فرستاد و نیازی به مشخص کردن سایز آرایه نیست

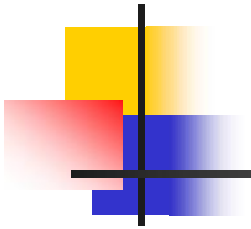
```
void print_array (int arr[], int len)
{
   for (int i=0;i<len;i++)
        printf( " Arr[%d] = %d\n" , i , arr[i] );
}
void main()
{
   int test[4]={1,5,7,4};
   print_array (test , 4);
}
```

```
Arr[0]= 1
Arr[1]= 5
Arr[2]= 7
Arr[3]= 4
```

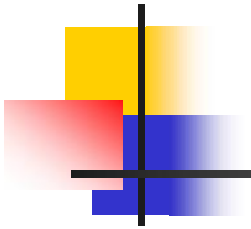
تمرین

- برنامه ای که تعداد ۱۰ عدد صحیح را از ورودی میخواند و ابتدا اعداد منفی و سپس اعداد مثبت را به خروجی میبرد تعداد اعداد مثبت و منفی را نیز مشخص میکند
- برنامه ای بنویسید که تعداد ۵ عدد از ورودی خوانده و سپس آنها را به ترتیب معکوس درون آرایه دیگری کپی کند نتیجه را نیز در خروجی نمایش دهد

مرتب سازی آرایه ها



- اعمال مرتب سازی و جستجو، عمومی ترین اعمالی هستند که در برنامه نویسی انجام میشوند.
- اگر یک لیست مرتب باشد در اینصورت عملیات جستجو سریعتر انجام میگیرد
- مرتب سازی آرایه ها به دو صورت صعودی و نزولی انجام میگیرد.
- مرتب سازی صعودی: x[0] < x[1] < x[2] <...<x[n]
- مرتب سازی نزولی: x[0] > x[1] > x[2] >...>x[n]



مرتب سازی حبابی

- این روش مرتب سازی بسیار ساده و قابل درک میباشد
- در این روش، باید چندین مرتبه در طول آرایه حرکت نمود و دوبه دوی عناصر با هم مقایسه میشوند و در صورت لزوم جای عناصر با هم عوض میشود.

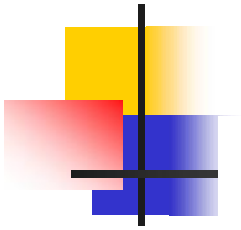
مرتب سازی حبابی یک آرایه

```
void bubble (int arr[] , int len)
{
    int i,j,tmp;
    for (i=0 ; i<len-1 ; i++)
          for (j=i+1 ; j < len ; j++)
                if (arr[i]>arr[j])
                {
                      tmp = arr[i];
                      arr[i] = arr[j];
                      arr[j] = tmp;
                }
}
void main()
{
    int test[4]={1,5,7,4};
    bubble (test , 4);
    print_array (test , 4);
}
```

```
1
4
5
7
```

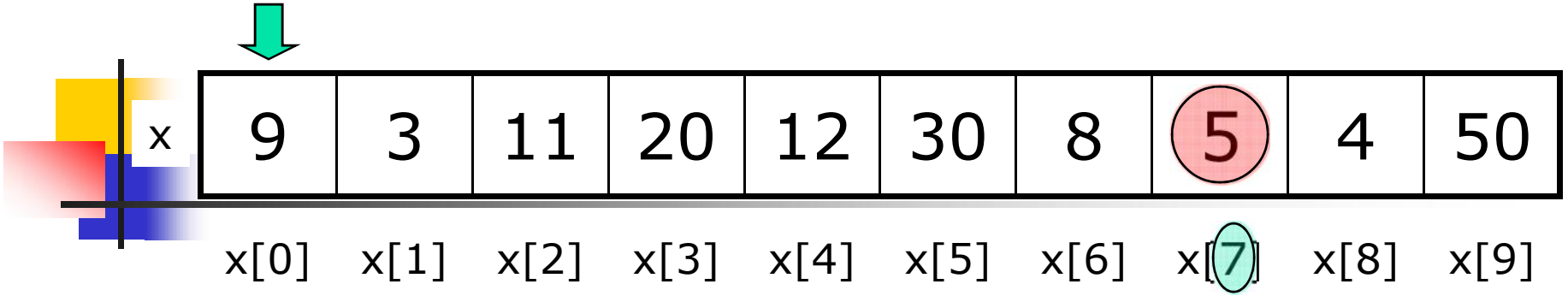
جستجوی یک مقدار در آرایه

جستجوی ترتیبی



- این تابع در آرایه حرکت کرده اگر به مولفه مساوی با مورد جستجو برسد اندیس آن و در غیر اینصورت عدد منفی برمیگرداند.

```
int lsearch (int arr[] , int len, int val)
{
  for (int i=0;i<len;i++)
      if (arr[i] == val)
            return i;
  return -1;
}
```

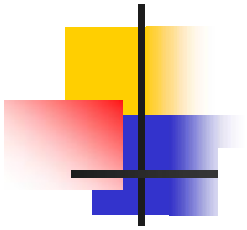


lsearch(x , 10 , 5)

```
int lsearch (int arr[] , int len, int val)
{
   for (int i=0;i<len;i++)
        if (arr[i] == val)
                return i;
   return -1;
}
```

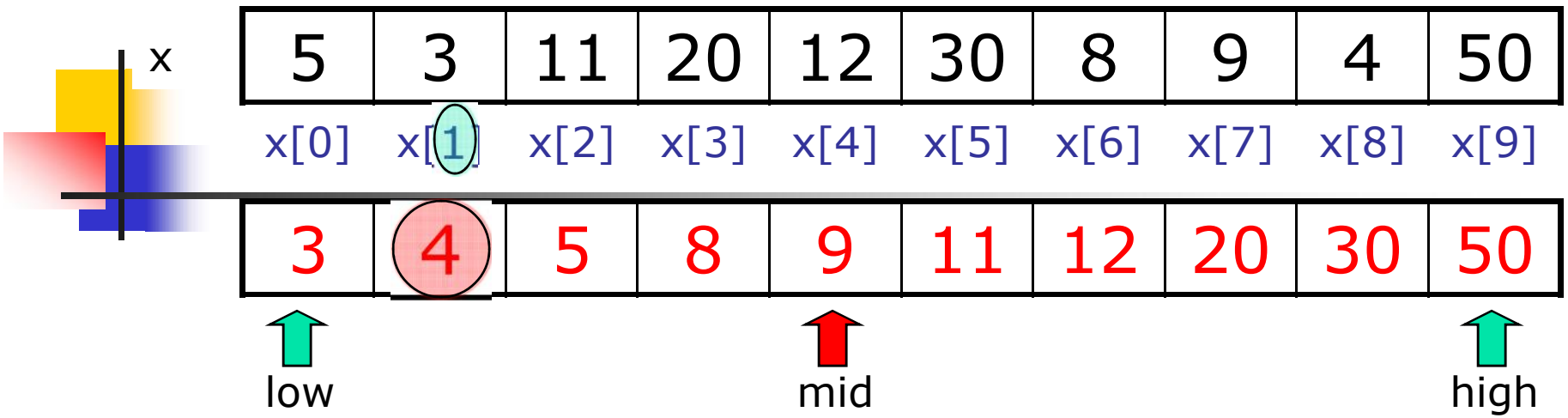
جستجوی یک مقدار در آرایه (ادامه...)

جستجوی دودویی



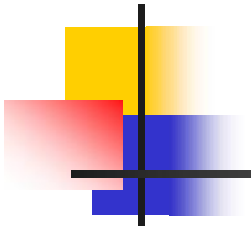
- این جستجو در آرایه مرتب انجام میشود

```
int bsearch (int arr[] , int len, int val)
{
    int mid , low = 0 , high = len -1;
    while (low<=high)
    {
        mid = (low+high)/2;
        if (arr[mid] == val)
            return mid;
        else if (arr[mid] > val)
            high = mid-1;
        else
            low = mid+1;
    }
    return -1;
}
```



bsearch(x , 10 , 4)

آرایه های چند بعدی



- می توان آرایه ای از آرایه ها درست کرد
- در c++ آرایه هایی با بیش از یک بعد میتوان درست کرد ولی اغلب برنامه نویسان از آرایه های ۲ بعدی استفاده میکنند.
- تعریف یک آرایه ۲ بعدی به شکل زیر انجام میگیرد
- نوع آرایه نام آرایه[تعداد سطرها][تعداد ستونها];
- یک آرایه n بعدی به شکل زیر تعریف میگردد.
- نوع آرایه نام آرایه[بعد ۱][بعد ۲]...[بعد n];

موقعیت عناصر آرایه در یک آرایه ۲ بعدی

int A[4][3]

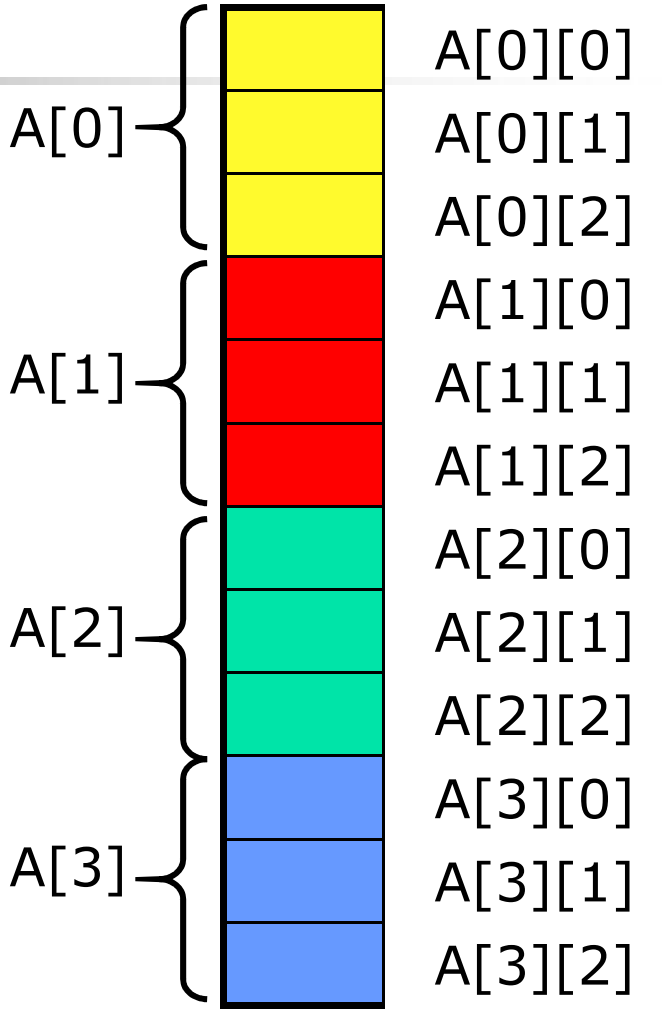
	Col 0	Col 1	Col 2
Row 0	A[0][0]	A[0][1]	A[0][2]
Row 1	A[1][0]	A[1][1]	A[1][2]
Row 2	A[2][0]	A[2][1]	A[2][2]
Row 3	A[3][0]	A[3][1]	A[3][2]

موقعیت عناصر آرایه دو بعدی در حافظه

char A[4][3];

- اگر A[0][0] در آدرس M حافظه باشد، آنگاه A[i][j] در آدرس زیر خواهد بود:

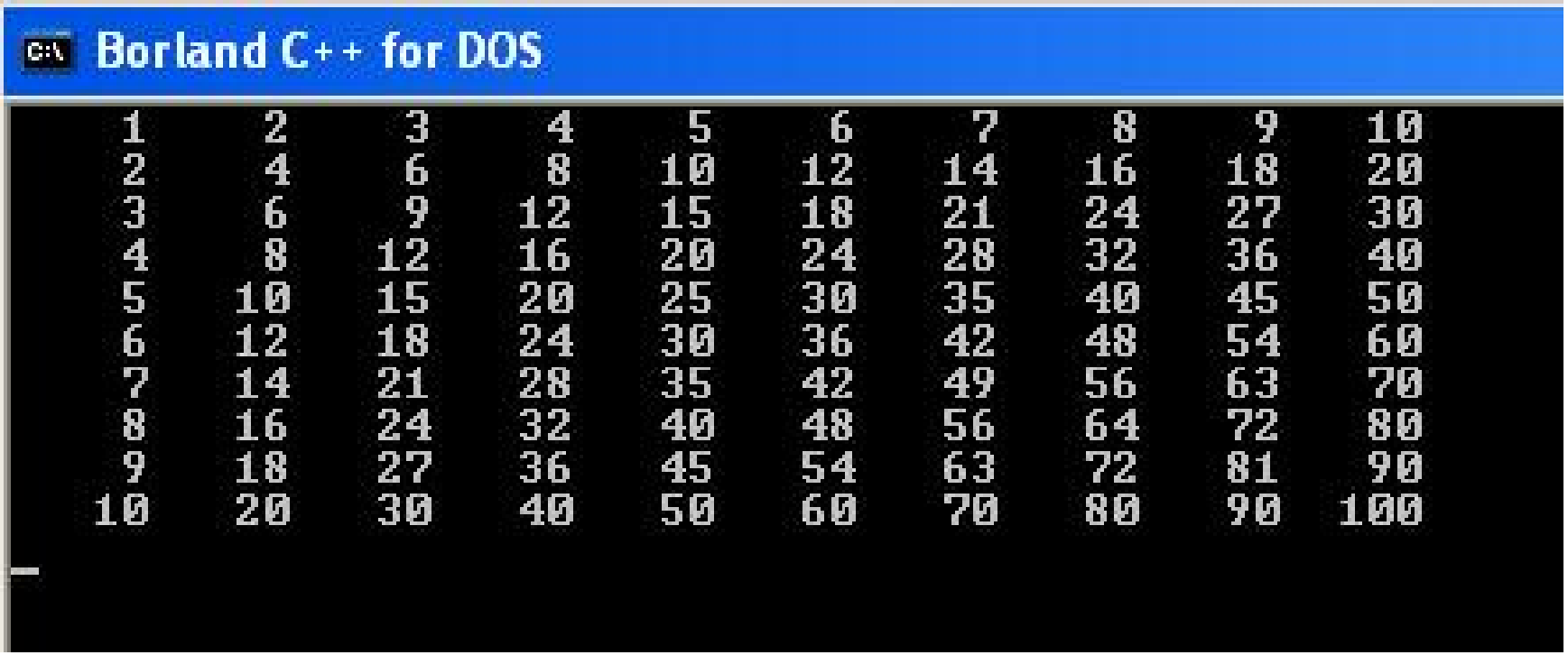
M+(i*3+j)*sizeof(char)



مثال: جدول ضرب با استفاده از آرایه دوبعدی

```
void main ()
{
   int table[10][10] , i , j;
   for ( i=0 ; i<10 ; i++)
        for( j=0 ; j<10 ; j++)
              table[i][j] = (i+1) * (j+1);

   for ( i=0 ; i<10 ; i++)
   {
        for( j=0 ; j<10 ; j++)
              printf("%5d" , table[i][j] );
        printf("\n" );
   }
   getch();
}
```


Borland C++ for DOS
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100

آرایه دو بعدی بعنوان آرگومان تابع

ارسال آرایه دو بعدی همانند آرایه یک بعدی است با این تفاوت که شما میتوانید تعداد سطرها را ذکر نکنید

```
void f1(int x[][10],int len);
void f2(int x[5][10]);
```

مثال: برنامه‌ای که عناصر ماتریس ۲×۳ را از ورودی خوانده، بزرگترین عنصر هر سطر را پیدا کند و به خروجی ببرد.

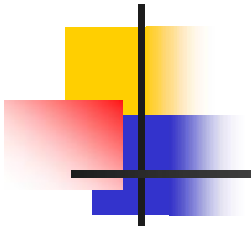
```
void minput (int [][2],int);
void mcal (int mat[][2],int);
void main ()
{
   int mat[3][2];
   clrscr();
   minput ( mat , 3);
   mcal ( mat , 3);
   getch();
}
```

```
void minput (int m[][2],int r)
{
    for (int i=0 ; i < r ; i++)
          for(int j=0 ; j < 2 ; j++)
          {
                printf("Enter mat[%d][%d]: ", i , j);
                scanf("%d" , &m[i][j] );
          }
}
void mcal (int mat[][2],int r)
{
    int i , j , rmax;
    printf("\nROW \t\t MAX");
    printf("\n--------------------------");
    for ( i=0 ; i< r ; i++)
    {
          rmax = mat[i][0];
          for( j=1 ; j< 2 ; j++)
                if (mat[i][j] > rmax)
                      rmax = mat[i][j];
          printf("\n%3d \t\t %3d:", i+1 , rmax);
    }
}
```

خروجی

```
Enter mat[0][0]: 3
Enter mat[0][1]: 4
Enter mat[1][0]: 5
Enter mat[1][1]: 6
Enter mat[2][0]: 7
Enter mat[2][1]: 8

ROW           MAX
--------------------
  1             4
  2             6
  3             8
```



نکته مهم

ارسال آرایه ها به توابع بصورت فراخوانی با مرجع میباشد.

یعنی هر تغییری که در تابع روی آرایه انجام پذیرد در مرجع اولیه آن اعمال میشود.

مقداردهی اولیه به آرایه های دو بعدی

- آرایه های دو بعدی همانند آرایه های یک بعدی میتوانند مقداردهی اولیه شوند.

- int y[2][3] ={1,3,4,7,6,15};
- int y[2][3] ={{1,3,4},{7,6,15}};
- int y [3] ={1};

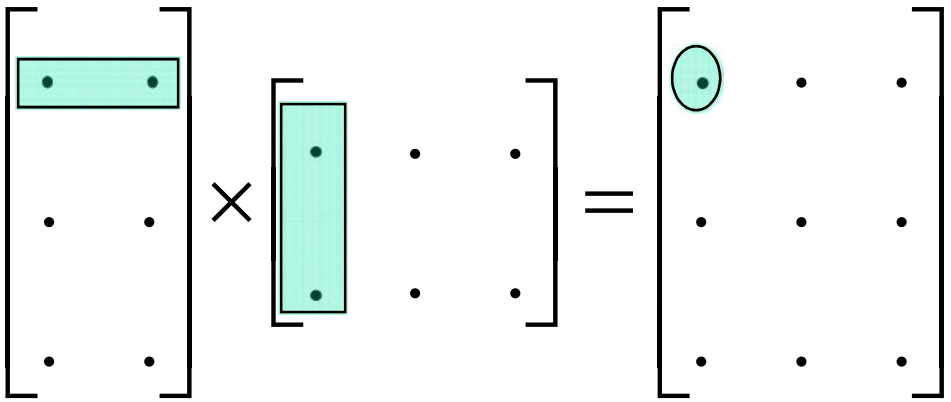
- اگر تعداد عناصری که مقدار دهی اولیه شده اند از تعداد عناصر مشخص شده کمتر باشند در اینصورت مابقی عناصر مقدار صفر میگیرند.

مثال:

- برنامه ای بنویسید که که دو ماتریس را از ورودی خوانده و ماتریس حاصلضرب را در خروجی نمایش دهد.

$$C_{m\times p} = A_{m\times n} \times B_{n\times p}$$

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}$$

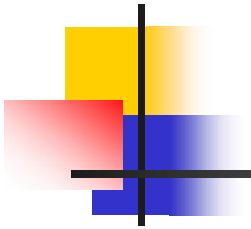


```
void Read_Mat1 (int m[][3] , int len);
void Read_Mat2 (int m[][4] , int len);
void main ()
{
    int mat1[2][3] , mat2[3][4] , mat3[2][4]={0};
    Read_Mat1 ( mat1 , 2);
    Read_Mat2 ( mat2 , 3);
    int i , j , k;
    for ( i=0 ; i < 2 ; i++)
          for( j=0 ; j < 4 ; j++)
          {
                    mat3[i][j]=0;
                    for( k=0 ; k < 3 ; k++)
                              mat3[i][j] += mat1[i][k] * mat2[k][j];
          }
    for ( i=0 ; i < 2 ; i++)
    {
          printf( "\n");
          for( j=0 ; j < 4 ; j++)
                    printf( "%5d" , mat3[i][j] );
    }
    getch();
}
```

خروجی

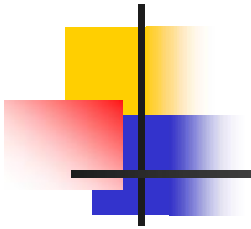
```
Enter mat1[0][0]: 1
Enter mat1[0][1]: 2
Enter mat1[0][2]: 3
Enter mat1[1][0]: 5
Enter mat1[1][1]: 4
Enter mat1[1][2]: 7
Enter mat2[0][0]: 8
Enter mat2[0][1]: 9
Enter mat2[0][2]: 5
Enter mat2[0][3]: 8
Enter mat2[1][0]: 10
Enter mat2[1][1]: 0
Enter mat2[1][2]: 12
Enter mat2[1][3]: 21
Enter mat2[2][0]: 0
Enter mat2[2][1]: 5
Enter mat2[2][2]: 9
Enter mat2[2][3]: 12

28      24      56      86
80      24     136     208
```



نکته

- اگر تعداد عناصر آرایه مشخص نباشد، باید حداکثر تعداد مورد انتظار را در نظر گرفت.



تمرین

- برنامه ضرب ماتریسها را به گونه ای اصلاح کنید که بتواند هر دو ماتریس مجاز را که ابعاد آنها کمتر از 6×6 است را در هم ضرب کند.
- توجه: ابعاد و درایه‌های هر ماتریس از کاربر گرفته شود.