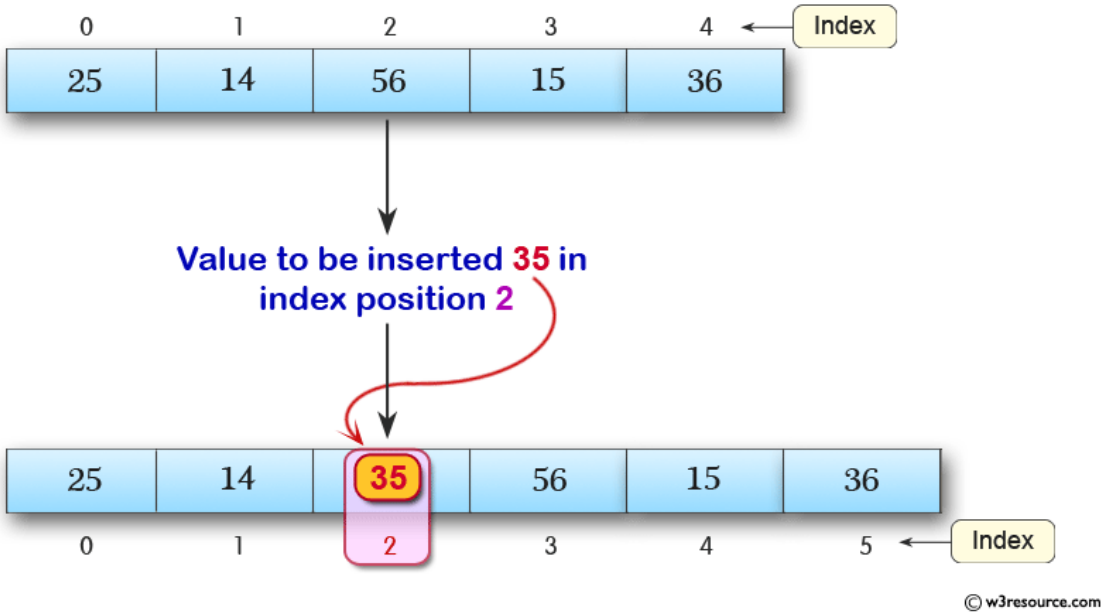
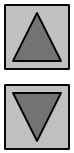
To insert new element in 1D array,

- shift elements from the given insert position to one position right.
- Hence, run a loop in descending order from size to pos to insert.
- The loop structure should look like for(i=size; i>=pos; i--) .
- Inside the loop copy previous element to current element by arr[i] = arr[i - 1];





Outline

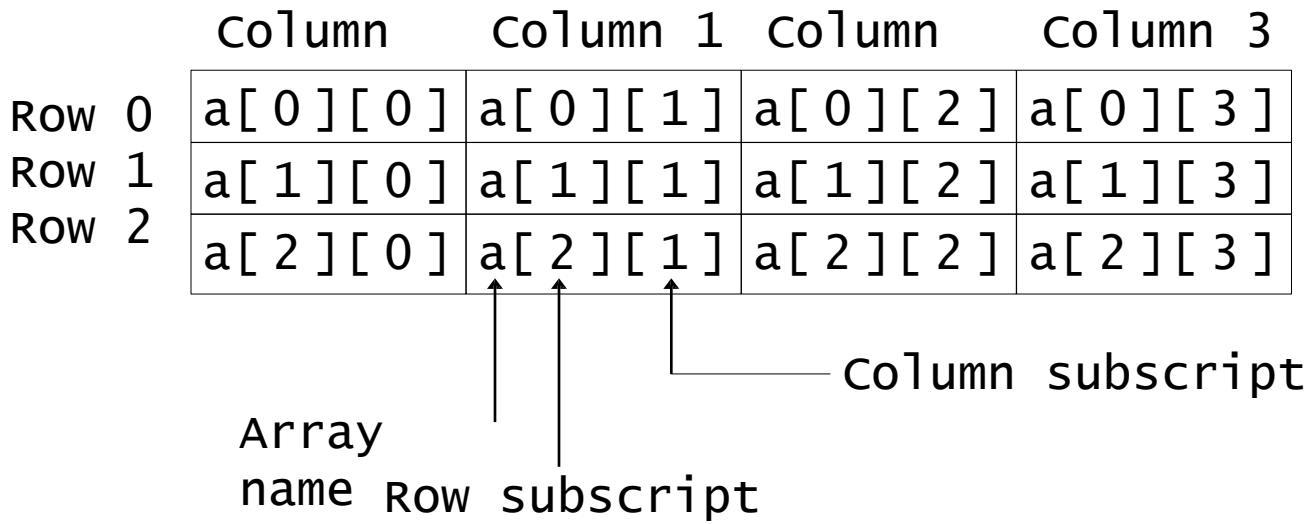
```
#include "stdafx.h"
#define SIZE 10
void main()
{
   int a[SIZE] = { 25,14,56,15,36 }, x, loc, i;
   for (i = 0; i < 5; i++)
     printf("%3d", a[i]);

   printf("\n\nenter element:");
   scanf_s("%d", &x);
   printf("enter location:");
   scanf_s("%d", &loc);
   printf("\n\n");

   for (i = 5; i >loc; i--)
     a[i] = a[i - 1];

   a[loc] = x;
   for (i = 0; i <=5; i++)
      printf("%3d", a[i]);
printf("\n\n");
}
```

Program Output

```
 25 14 56 15 36

enter element:35
enter location:2

 25 14 35 56 15 36
```

6.9 Multiple-Subscripted Arrays

- Multiple subscripted arrays
  - Tables with rows and columns (`m` by `n` array)
  - Like matrices: specify row, then column



6.9 Multiple-Subscripted Arrays

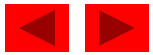
- C programming language allows to create arrays of arrays known as multidimensional arrays. For example:

```
float a[2][6];
```

- here, *a* is an array of two dimension, which is an example of multidimensional array. This array has 2 rows and 6 columns
- For better understanding of multidimensional arrays, array elements of above example can be thinked of as below:

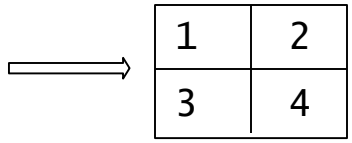
	col 1	col 2	col 3	col 4	col 5	col 6
row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]
row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]

Figure: Multidimensional Arrays



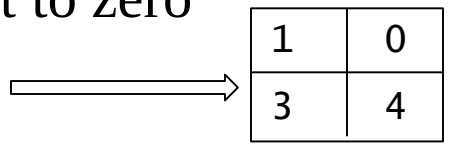
6.9 Multiple-Subscripted Arrays

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

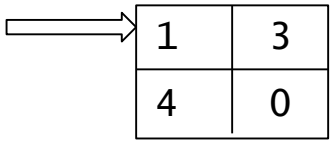


  - Initializers grouped by row in braces
  - If not enough, unspecified elements set to zero

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

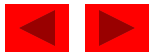


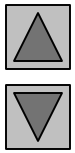

```
int b[ 2 ][ 2 ] = { 1, 3, 4 };
```



- Referencing elements
  - Specify row, then column

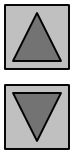
```
printf( "%d", b[ 0 ][ 1 ] );
```



Outline

```
/* EX1: printing array elements*/
#include "stdafx.h"
 void printArray(int a[][3]);
    int main()
    {
     /* initialize array1, array2, array3 */
       int array1[ 2 ][ 3 ] = { { 1, 2, 3 }, { 4, 5, 6 } };
       int array2[ 2 ][ 3 ] = { 1, 2, 3, 4, 5 };
       int array3[ 2 ][ 3 ] = { { 1, 2 }, { 4 } };

        printf("Values in array1 by row are:\n");
      printArray(array1);

      printf("Values in array2 by row are:\n");
      printArray(array2);

      printf("Values in array3 by row are:\n");
      printArray(array3);

      return 0;

     }
```



Outline

```
void printArray(int a[][3])
{
   for (int i = 0; i < 2; i++)
   {
      for (int j = 0; j < 3; j++)
          printf("%3d", a[i][j]);
      printf("\n");

   }

}
```

fig06_21.c (Part 2 of 2)

```
Values in array1 by row are:
1 2 3
4 5 6
Values in array2 by row are:
1 2 3
4 5 0
Values in array3 by row are:
1 2 0
4 0 0
```

Program Output

jimmy

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

```c
/* EX2: printing array elements*/
#include "stdafx.h"
#define WIDTH 5
#define HEIGHT 3
void main()
{
  int jimmy[HEIGHT][WIDTH];
  int n, m;
  for (n = 0; n < HEIGHT; n++)
  {
     for (m = 0; m < WIDTH; m++)
     {
       jimmy[n][m] = (n + 1)*(m + 1);
       printf("%3d", jimmy[n][m]);
     }
     printf("\n");
  }
}
```

Program Output

```
  1  2  3  4  5
  2  4  6  8 10
  3  6  9 12 15
```

```c
// EX3: filling an array with a random number
#include "stdafx.h"
#include "stdlib.h"
#include "time.h"
void main()
{
   int a[3][2], r, c;
   srand(time(NULL));
   for (r = 0; r<3; r++)
   {
     for (c = 0; c<2; c++)
     {
       a[r][c] = rand() % 9 + 2;
       printf("%3d", a[r][c]);
     }
     printf("\n");
   }
}
```

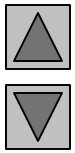
```
  8  6
  9  9
  2  3
```

Program Output

```c
// EX4: filling an array with a random number and finding sum of each row
#include "stdafx.h"
#include "stdlib.h"
#include "time.h"
void main()
{
   int a[2][4], sum[2] = { 0 }, r, c;
   srand(time(NULL));
   for (r = 0; r<2; r++)
   {
     for (c = 0; c<4; c++)
     {
        a[r][c] = rand() % 10 + 1;
        printf("%3d", a[r][c]);
        sum[r] += a[r][c];
     }
     printf(" = %d", sum[r]);
     printf("\n");
   }
}
```

Program Output

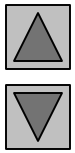
```
 10  5  6  3 = 24
  7  5  6 10 = 28
```



Outline

fig06_22.c (Part 1 of 6)

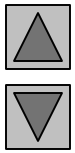
```
1  /*  EX5:
2     Double-subscripted array example */
3  #include <stdafx.h>
4  #define STUDENTS 3
5  #define EXAMS 4
6
7  /* function prototypes */
8  int minimum(int grades[][ EXAMS ], int pupils, int tests );
9  int maximum(int grades[][ EXAMS ], int pupils, int tests );
10 double average(int setOfGrades[], int tests );
11 void printArray(int grades[][ EXAMS ], int pupils, int tests );
12
13 /* function main begins program execution */
14 int main()
15 {
16    int student; /* counter */
17
18    /* initialize student grades for three students (rows) */
19    int studentGrades[ STUDENTS ][ EXAMS ] =
20       { { 77, 68, 86, 73 },
21         { 96, 87, 89, 78 },
22         { 70, 90, 86, 81 } };
23
```



Outline

fig06_22.c (Part 2 of 6)

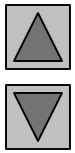
```
24    /* output array studentGrades */
25    printf( "The array is:\n" );
26    printArray( studentGrades, STUDENTS, EXAMS );
27
28    /* determine smallest and largest grade values */
29    printf( "\n\nLowest grade: %d\nHighest grade: %d\n",
30       minimum( studentGrades, STUDENTS, EXAMS ),
31       maximum( studentGrades, STUDENTS, EXAMS ) );
32
33    /* calculate average grade for each student */
34    for ( student = 0; student <= STUDENTS - 1; student++ ) {
35       printf( "The average grade for student %d is %.2f\n",
36          student, average( studentGrades[ student ], EXAMS ) );
37    } /* end for */
38
39    return 0; /* indicates successful termination */
40
41 } /* end main */
42
43 /* Find the minimum grade */
44 int minimum( int grades[][ EXAMS ], int pupils, int tests )
45 {
46    int i;              /* counter */
47    int j;              /* counter */
48    int lowGrade = 100; /* initialize to highest possible grade */
49
```



Outline

fig06_22.c (Part 3 of 6)

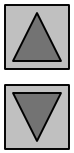
```
50    /* loop through rows of grades */
51    for ( i = 0; i < pupils; i++ ) {
52
53       /* loop through columns of grades */
54       for ( j = 0; j < tests; j++ ) {
55
56          if ( grades[ i ][ j ] < lowGrade ) {
57             lowGrade = grades[ i ][ j ];
58          } /* end if */
59
60       } /* end inner for */
61
62    } /* end outer for */
63
64    return lowGrade; /* return minimum grade */
65
66 } /* end function minimum */
67
68 /* Find the maximum grade */
69 int maximum(int grades[][ EXAMS ], int pupils, int tests )
70 {
71    int i;             /* counter */
72    int j;             /* counter */
73    int highGrade = 0; /* initialize to lowest possible grade */
74
```



Outline

fig06_22.c (Part 4 of 6)

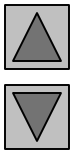
```
75    /* loop through rows of grades */
76    for ( i = 0; i < pupils; i++ ) {
77
78       /* loop through columns of grades */
79       for ( j = 0; j < tests; j++ ) {
80
81          if ( grades[ i ][ j ] > highGrade ) {
82             highGrade = grades[ i ][ j ];
83          } /* end if */
84
85       } /* end inner for */
86
87    } /* end outer for */
88
89    return highGrade; /* return maximum grade */
90
91 } /* end function maximum */
92
93 /* Determine the average grade for a particular student */
94 double average(int setOfGrades[], int tests )
95 {
96    int i;          /* counter */
97    int total = 0; /* sum of test grades */
98
```



Outline

fig06_22.c (Part 5 of 6)

```
99    /* total all grades for one student */
100   for ( i = 0; i < tests; i++ ) {
101      total += setOfGrades[ i ];
102   } /* end for */
103
104   return ( double ) total / tests; /* average */
105
106} /* end function average */
107
108/* Print the array */
109void printArray( int grades[][ EXAMS ], int pupils, int tests )
110{
111   int i; /* counter */
112   int j; /* counter */
113
114   /* output column heads */
115   printf( "                [0]  [1]  [2]  [3]" );
116
117   /* output grades in tabular format */
118   for ( i = 0; i < pupils; i++ ) {
119
120      /* output label for row */
121      printf( "\nstudentGrades[%d] ", i );
122
```



Outline

```
123       /* output grades for one student */
124       for ( j = 0; j < tests; j++ ) {
125          printf( "%-5d", grades[ i ][ j ] );
126       } /* end inner for */
127
128    } /* end outer for */
129
130} /* end function printArray */
```

```
The array is:
                 [0]  [1]  [2]  [3]
studentGrades[0] 77   68   86   73
studentGrades[1] 96   87   89   78
studentGrades[2] 70   90   86   81

Lowest grade: 68
Highest grade: 96
The average grade for student 0 is 76.00
The average grade for student 1 is 87.50
The average grade for student 2 is 81.75
```

fig06_22.c (Part 6 of 6)