# آرایه‌ها ۲

## مبانی برنامه‌نویسی

## (۱۳۹۹-۱۳-۱۱)

## جلسه‌ی بیست‌ودوم


دانشگاه شهید بهشتی

پاییز ۱۳۹۹

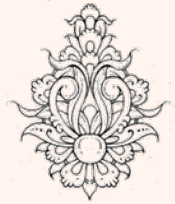
دانشکده‌ی مهندسی برق و کامپیوتر

احمد محمودی ازناوه

http://faculties.sbu.ac.ir/~a_mahmoudi/

# فهرست مطالب

- ارسال آرایه به تابع
- آرایه‌های چند بعدی

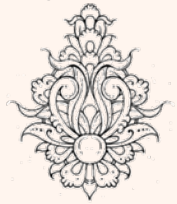


# تمرین

- اعدادی از یک آرایه را خوانده اطلاعات را به صورت هیستوگرام فراوانی رسم کنید.
  - برای نشان دادن هر عنصر از «*» استفاده نمایید.

**{12,3,4,6,13,16,2,7,0,9}**

```
Element Value   Histogram
0       12      ************
1       3       ***
2       4       ****
3       6       ******
4       13      *************
5       16      ****************
6       2       **
7       7       *******
8       0
9       9       *********
```



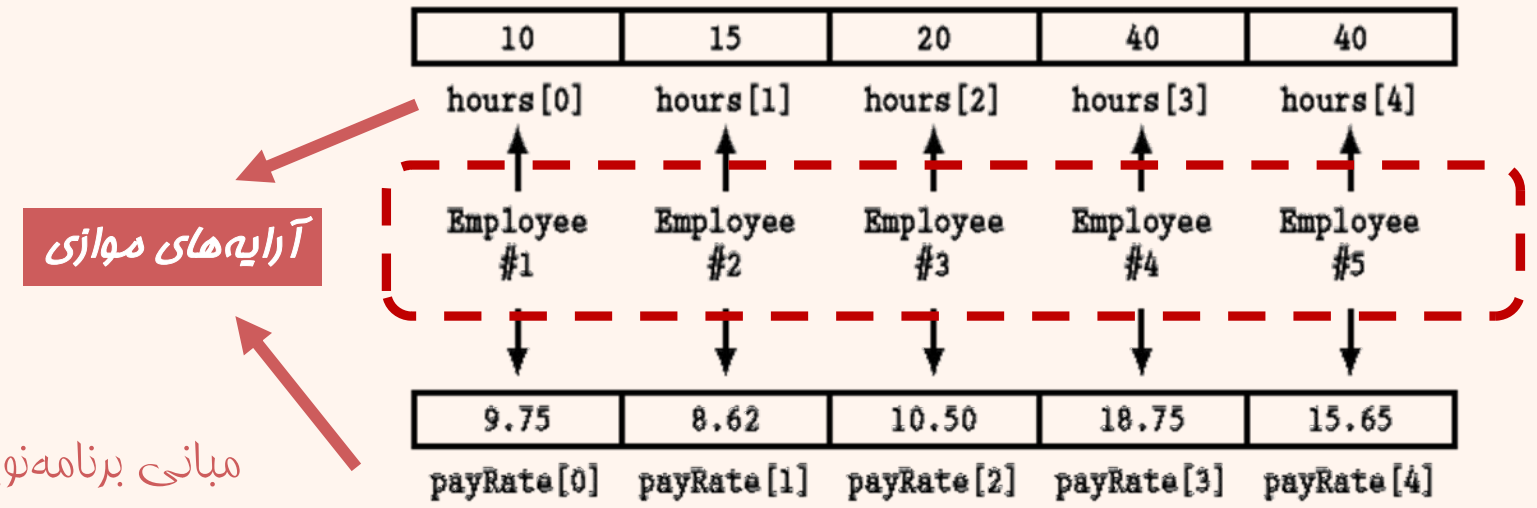
# هیستوگرام فراوانی

```
 #include <iostream>
 using namespace std;
int main()
{
const int arraySize = 10;
int n[ arraySize ]={12,3,4,6,13,16,2,7,0,9}; // array s has 10 elements



   cout << "Element\t" << "Value\t"<< "Histogram" << endl;


   for ( int i= 0; i < arraySize; i++ )
       {
          cout << i << "\t" << n[ i ]<<"\t";
             for ( int j = 0; j < n[i]; j++ )
                cout<< '*';
          cout << endl;
       }
   return 0; // indicates successful termination
} // end main
```

```
Element Value   Histogram
0       12      ************
1       3       ***
2       4       ****
3       6       ******
4       13      *************
5       16      ****************
6       2       **
7       7       *******
8       0
9       9       *********
```

# استفاده از آرایه‌ی موازی

- در بسیاری موارد ذخیره‌ی داده‌های مرتبط در دو یا چند آرایه می‌تواند بسیار مفید باشد.
- در این‌صورت به واسطه‌ی استفاده از اندیس یکسان می‌توان به داده‌های مرتبط دست‌یافت.
- مثال
  - برنامه‌ی حقوق کارکنان را به گونه‌ای بنویسید که برای هر پرسنل حقوق ساعتی متفاوتی در نظر گرفته شود:

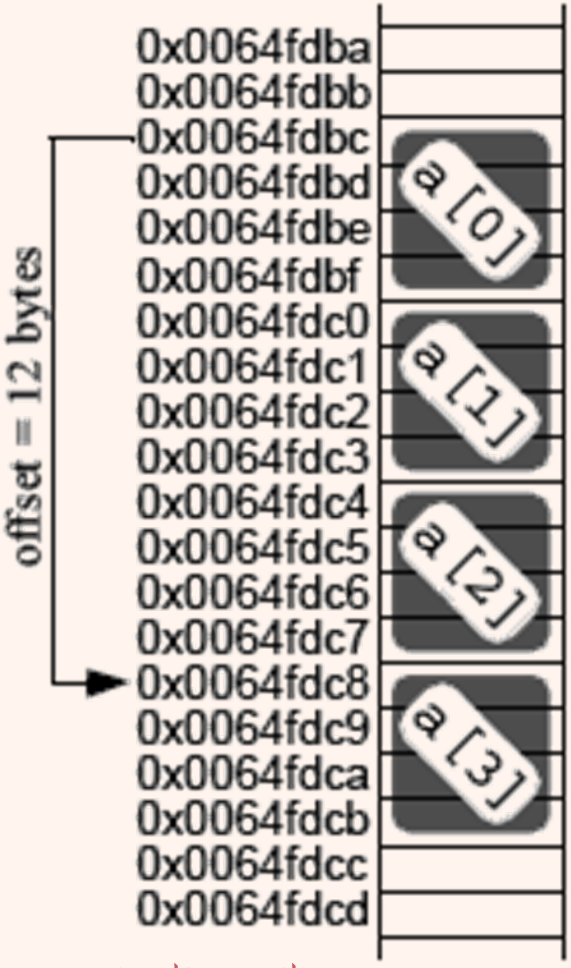


```
int main()
 {
 const int NUM_EMPS = 5;
 int index;
 int hours[NUM_EMPS]; // Define 2 paralle
 double payRate[NUM_EMPS];
 double grossPay;
 // Get employee work data
  cout << "Enter the hours worked and hourly pay rates of "
  << NUM_EMPS << " employees. \n";
for (index = 0; index < NUM_EMPS; index++)
 {
    cout << "Hours worked by employee #" << (index + 1) << ": ";
    cin >> hours[index];
    cout << "Hourly pay rate for employee #" << (index + 1) << ": ";
    cin >> payRate[index];
 }
 // Display the data
 cout << "\nHere is the gross pay for each employee:\n";
 for (index = 0; index < NUM_EMPS; index++)
 {
    grossPay = hours[index] * payRate[index];
     cout << "Employee #" << (index + 1);
     cout << ": $" <<  grossPay << endl;
 }
 return 0;
}
```

```
Enter the hours worked and hourly pay rates of 5 employees.
Hours worked by employee #1: 4
Hourly pay rate for employee #1: 15
Hours worked by employee #2: 2
Hourly pay rate for employee #2: 56
Hours worked by employee #3: 33
Hourly pay rate for employee #3: 6
Hours worked by employee #4: 23
Hourly pay rate for employee #4: 22
Hours worked by employee #5: 3
Hourly pay rate for employee #5: 44

Here is the gross pay for each employee:
Employee #1: $60
Employee #2: $112
Employee #3: $198
Employee #4: $506
Employee #5: $132
```

# آدرس عناصر آرایه

```
cin >> a[n];
```




- اگر مقدار n برابر با ۳ باشد، به مقدار ۳×۴ از a[0] جلورفته، به آدرس مورد نظر دسترسی پیدا می‌کنیم.

*نام آرایه در حقیقت آدرس ابتدای آرایه است که آدرسی ثابت و غیرقابل تغییر است*

# ارسال آرایه به تابع

- در بعضی موارد نیاز است آرایه‌ای را به تابعی ارسال نماییم.
- اگر در هر مرحله ارسال تنها یک متغیر از آرایه نیاز باشد همانند ارسال یک متغیر عمل می‌شود.

```
void showValue(int); // Function prototype

int main()
{
const int ARRAY_SIZE = 8;
int collection[ARRAY_SIZE] = {5, 10, 15, 20, 25, 30, 35, 40};

for (int cycle = 0; cycle < ARRAY_SIZE; cycle++)
    showValue(collection[cycle]);
cout << endl;
return 0;
}
void showValue(int num)
{
   cout << num << " ";
}
```

```
5 10 15 20 25 30 35 40
```

# ارسال آرایه به تابع

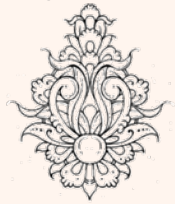
- برای این‌که یک آرایه را به عنوان آرگومان ورودی به تابعی ارسال نماییم لازم است کامپایلر نوع عناصر آرایه و نام آرایه را بداند.
- آن‌چیز که در نظر گرفته می‌شود آدرس شروع آرایه و نوع عناصر است.

```
int myarray [40];
```

```
void procedure(int arg[])
```

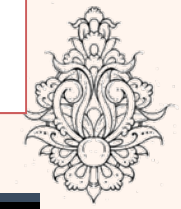
```
procedure(myarray)
```

← فراخوانی تابع



```
void showValues(int [], int);// Function prototype

int main()
{
 const int ARRAY_SIZE = 8;
 int collection[ARRAY_SIZE] = {5, 10, 15, 20, 25, 30, 35, 40};
 showValues(collection, ARRAY_SIZE);
 return 0;
 }
void showValues (int nums[], int size)
 {
 for (int index = 0; index < size; index++)
 cout << nums[index] << " ";
 cout << endl;
 }
```

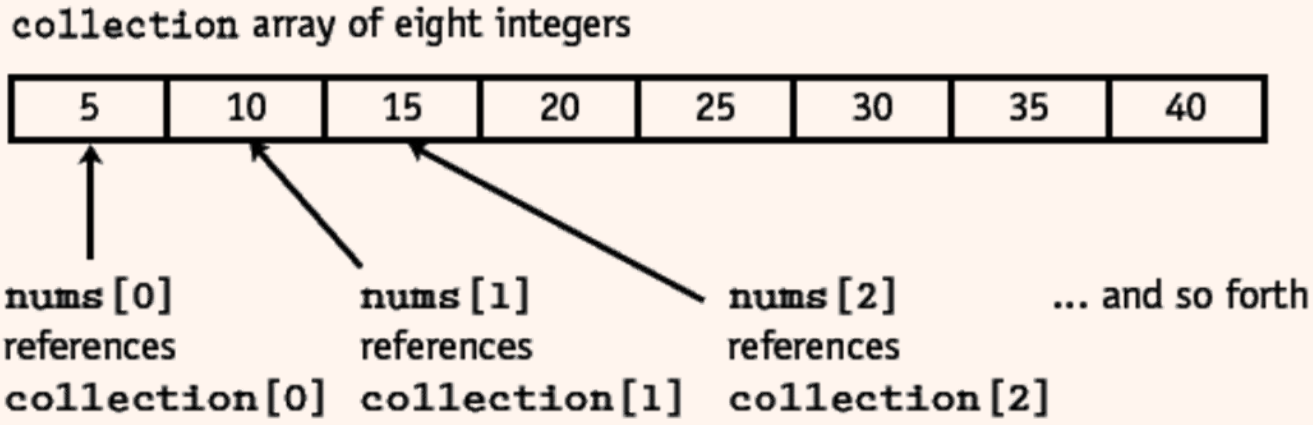
```
5 10 15 20 25 30 35 40
```



ارسال نام آرایه در حقیقت ارسال نشانی اولین عنصر آرایه است
پس ارسال آرایه به تابع شبیه ارسال به طریق ارجاع عمل می‌کند

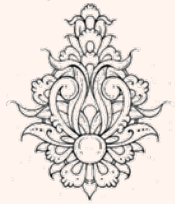
```
showValues (collection,  ARRAY_SIZE) ;
```

```
void showValues (int nums [ ] , int size)
{
      for (int index = 0; index < size ; index++)
      cout << nums [ index ] << " " ;
      cout << endl;
}
```



```cpp
#include <iostream>
using namespace std;

void printarray (int arg[], int length) {
  for (int n=0; n<length; n++)
    cout << arg[n] << " ";
  cout << "\n";
}

int main ()
{
  int firstarray[] = {5, 10, 15};
  int secondarray[] = {2, 4, 6, 8, 10};
  printarray (firstarray,3);
  printarray (secondarray,5);
  return 0;
}
```

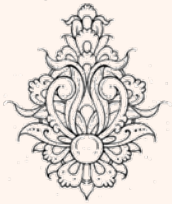
```
5 10 15
2 4 6 8 10
```



# مثال

```
#include <iostream>
using namespace std;
int sum(int[],int);
int main()
{   int a[] = { 11, 33, 55, 77 };
    int size = sizeof(a)/sizeof(int);
    cout << "sum(a,size) = " << sum(a,size) << endl;
     }
int sum(int a[], int n)
{   int sum=0;
    for (int i=0; i<n; i++)
        sum += a[i];
    return sum;
}
```

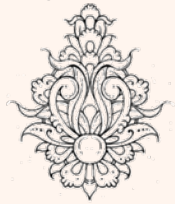
```
sum(a,size) = 176
```



# مثال

```
#include <iostream>
using namespace std;
int main()
{   int a[4] = { 44, 32,345,134};
    cout << "a = " << a << endl;      // the address of a[0]
    cout << "a[0] = " << a[0]<<endl; // the value of a[0]
}
```

```
a = 0012FF1C
a[0] = 44
```

```
#include <iostream>
using namespace std;
void read(int[],int&);
void print(int[],int);
const int MAXSIZE=100;
int main()
{
     int a[MAXSIZE]={0},size;
     read(a,size);
     cout << "The array has " << size << " elements: ";
     print(a,size);
}
```

```
void print(int a[],int n)
{
     for (int i=0; i<n; i++)
     cout << a[i] << " ";
}
```

```
Enter integers. Terminate with 0:
a[0]: 12
a[1]: 3
a[2]: 4
a[3]: 5
a[4]: 6
a[5]: 0
The array has 5 elements: 12 3 4 5 6
```

```
void read(int a[],int& n)
{
     cout << "Enter integers. Terminate with 0:\n";
     n = 0;
do
{
     cout << "a[" << n << "]: ";
     cin >> a[n];
} while (a[n++] != 0 && n < MAXSIZE);
--n; // don't count the 0
}
```

# Defined data types (typedef)

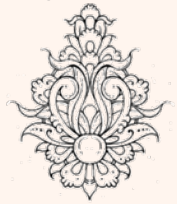
به وسیله‌ی typedef می‌توان نام انواع موجود را به نوع‌هایی جدید تغییر نام داد:

```
typedef  existing_type  new_type_name;
```

با استفاده از typedef نوع جدیدی به‌وجود نمی‌آید. بلکه نام دیگری برای نوعی که‌وجود دارد در نظر گرفته می‌شود

```
typedef  int  integer;
typedef  unsigned int  uint;
```

- در این صورت
  - برنامه به صورت خواناتر کدنویسی می‌شود.
  - اگر در آینده بخواهیم نوع داده‌ای را عوض نماییم کار ساده‌تر می‌شود.
  - برای داده‌هایی با اسامی طولانی و پیچیده کاربرد دارد.



# مثال

```
typedef int arrayType[];
// Function prototypes
void doubleArray(arrayType, int);
void showValues(arrayType, int);
int main()
{
const int ARRAY_SIZE = 7;
arrayType set = {1, 2, 3, 4, 5, 6, 7};
cout << "The arrays values are:\n";

showValues(set, ARRAY_SIZE);
doubleArray(set, ARRAY_SIZE);

cout << "\nAfter calling doubleArray, the values are:\n";
showValues(set, ARRAY_SIZE);
cout << endl;
return 0;
}
```

```
The arrays values are:
1 2 3 4 5 6 7

After calling doubleArray, the values are:
2 4 6 8 10 12 14
```

```
void showValues (arrayType nums, int size)
{
for (int index = 0; index < size; index++)
  cout << nums[index] << " ";
cout << endl;
}

void doubleArray(arrayType nums, int size)
{
for (int index = 0; index < size; index++)
  nums[index] *= 2;
}
```

ارسال آرایه همانند ارسال از طریق ارجاع عمل می‌کند پس می تواند مقادیر را تغییر دهد

# چند تابع

```
double sumArray(double[], int);
double getHighest(double[], int);
double getLowest(double[], int);
int main()
{
const int NUM_DAYS = 5;
double sales[NUM_DAYS],
total, average, highest, lowest;
cout << "Enter the sales for this week.\n"
for (int day = 0; day < NUM_DAYS; day++)
{
    cout << "Day " << (day + 1) <<": ";
   cin >> sales[day];
}
// Get total sales and compute average sale
total = sumArray(sales, NUM_DAYS);
average = total / NUM_DAYS;
// Get highest and lowest sales amounts
highest = getHighest(sales, NUM_DAYS);
lowest = getLowest(sales, NUM_DAYS);

// Display results
cout << "The total sales are $" <<  total << e
cout << "The average sales amount is $"  << av
cout << "The highest sales amount is $"  << hi
cout << "The lowest sales amount is $"  << low
return 0;
}
```

```
double sumArray(double array[], int size)
{
double total = 0; // Accumulator

for (int count = 0; count < size; count++)
total += array[count];
return total;
}
```

```
 double getHighest(double array[], int size)
{
double highest = array[0];
for (int count = 1; count < size; count++)
{ if (array[count] > highest)
highest = array[count];
}
return highest;
}
```

```
double getLowest(double array[], int size)
{
double lowest = array[0];
for (int count = 1; count < size; count++)
{ if (array[count] < lowest)
lowest = array[count];
}
return lowest;
}
```

```
Enter the sales for this week.
Day 1: 23
Day 2: 4
Day 3: 5
Day 4: 41
Day 5: 2
The total sales are $75
The average sales amount is $15
The highest sales amount is $41
The lowest sales amount is $2
```

# نحوه‌ی تعریف آرایه‌های n بعدی

نوع آرایه  نام آرایه  [طول بعد اول]  [طول بعد دوم ] ..... [طول بعد nام ]

- **تعریف آرایه دو بعدی از نوع صحیح:**
- int a[2][5];
- **یک آرایه سه بعدی از نوع صحیح:**
- int k[3][2][5];
- **میزان حافظه‌ی مصرفی یک آرایه n بعدی:**

*میزان حافظه مصرفی= sizeof (نوع آرایه) طول بعد اول x طول بعد دوم x ... xطول بعد nام*

برای مثال نخست، میزان حافظه مورد نیاز را حساب کنید:

**Sizeof(int)*2*5=40**

# آرایه‌ی دوبعدی

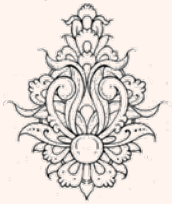
- آرایه‌ی دو بعدی از کنار هم گذاردن تعدادی آرایه‌ی یک بعدی به دست می‌آید.

| | Column 0 | Column 1 | Column 2 | Column 3 |
|---|---|---|---|---|
| Row 0 | score[0][0] | score[0][1] | score[0][2] | score[0][3] |
| Row 1 | score[1][0] | score[1][1] | score[1][2] | score[1][3] |
| Row 2 | score[2][0] | score[2][1] | score[2][2] | score[2][3] |

```
double score[3][4];
```

Rows Columns

| | Column 0 | Column 1 |
|---|---|---|
| Row 0 | 8 | 5 |
| Row 1 | 7 | 9 |
| Row 2 | 6 | 3 |



```
int hours[3][2] = {{8, 5}, {7, 9}, {6, 3}};
```

```
int hours[3][2] = {8, 5, 7, 9, 6, 3};
```

# انواع مقداردهی

- آرایه دو بعدی:

```
int a[2][3]={{3,1,2},{3,5,7}};
```

- ننوشتن بعد اول:

```
int a[ ][3]={{3,1,2},{3,5,7}};
```

```
int table[3][2] = {{1}, {3, 4}, {5}};
```

- باقی عناصر اتوماتیک صفر می‌شوند.
- آرایه سه بعدی:

```
int [2][3][4]={{{1,2,3,4},{3,2,3,4},{3,2,2,1}},{{2,3,5,6},{1,2,4,2},{1,2,2,1}}}
```

# مثال

```
int main()
{
   const int NUM_ROWS = 3; // Number of rows
   const int NUM_COLS = 5; // Number of columns
   int total = 0; // Accumulator
   int numbers[NUM_ROWS][NUM_COLS] = {{2, 7, 9, 6, 4},
   {6, 1, 8, 9, 4},
   {4, 3, 7, 2, 9}};
   // Sum the array elements
   for (int row = 0; row < NUM_ROWS; row++)
   { for (int col = 0; col < NUM_COLS; col++)
   total += numbers[row][col];
   }
   // Display the sum
   cout << "The total is " << total << endl;
    return 0;
}
```

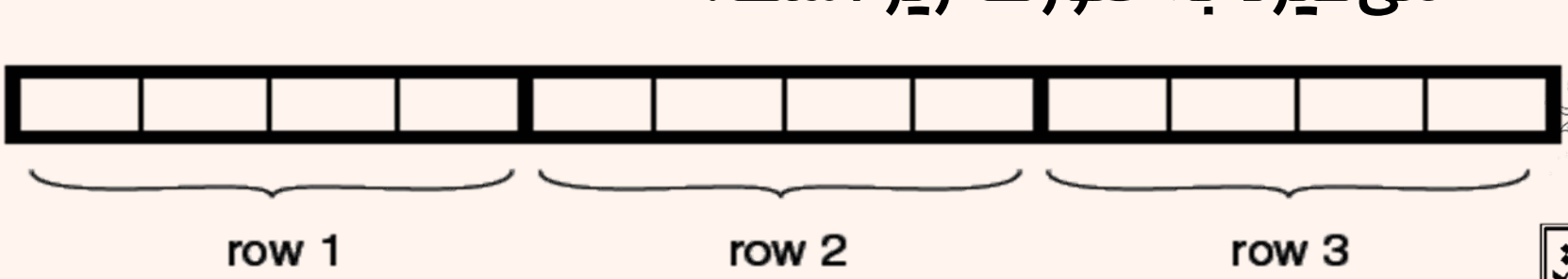
```
The total is 81
```

# ارسال آرایه‌ی دو بعدی به تابع

- برای ارسال یک آرایه‌ی دو بعدی به تابع می‌باید ستون ها مشخص باشد.

```
void showArray(int array[][NUM_COLS],int numRows)
```

- در c++ ساختاری که یک آرایه‌ی دوبعدی قرار می‌گیرد به صورت زیر است:




- در این حالت هر ردیف پس از ردیف دیگر قرار می‌گیرد.

# مثال

```
#include <iostream>
#include <iomanip>
 using namespace std;

 const int NUM_COLS = 4; // Number of columns in each array
 const int TBL1_ROWS = 3; // Number of rows in table1
 const int TBL2_ROWS = 4; // Number of rows in table2

 void showArray(int [][NUM_COLS], int); // Function prototype

 int main()
 {
 int table1[TBL1_ROWS][NUM_COLS] = {{1, 2, 3, 4},
 {5, 6, 7, 8},
 {9, 10, 11, 12}};
 int table2[TBL2_ROWS][NUM_COLS] = {{ 10, 20, 30, 40},
 { 50, 60, 70, 80},
 { 90, 100, 110, 120},
 {130, 140, 150, 160}};

 cout << "The contents of table1 are:\n";
 showArray(table1, TBL1_ROWS);
 cout << "\nThe contents of table2 a
 showArray(table2, TBL2_ROWS);
 return 0;
 }
```

```
The contents of table1 are:
    1     2     3     4
    5     6     7     8
    9    10    11    12

The contents of table2 are:
   10    20    30    40
   50    60    70    80
   90   100   110   120
  130   140   150   160
```

```
void showArray(int array[][NUM_COLS], int numRows)
{
for (int row = 0; row < numRows; row++)
 {
     for (int col = 0; col < NUM_COLS; col++)
      {
        cout << setw(5) << array[row][col] << " ";
      }
cout << endl;
  }
}
```