## Using Parallel Arrays

- Parallel arrays: two or more arrays that contain related data
- A subscript is used to relate arrays: elements at same subscript are related
- Arrays may be of different types

## Parallel Array Example

```
const int SIZE = 5;    // Array size
int id[SIZE];          // student ID
double average[SIZE]; // course average
char grade[SIZE];      // course grade
...
for(int i = 0; i < SIZE; i++)
{
   cout << "Student ID: " << id[i]
        << " average: " << average[i]
        << " grade: " << grade[i]
        << endl;
}
```

## Parallel Array Example

**Program 7-12**

```
// This program stores, in an array, the hours worked by 5
// employees who all make the same hourly wage.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   const int NUM_EMPLOYEES = 5;
   int hours[NUM_EMPLOYEES];          // Holds hours worked
   double payRate[NUM_EMPLOYEES];     // Holds pay rates

   // Input the hours worked.
   cout << "Enter the hours worked by " << NUM_EMPLOYEES;
   cout << " employees and their\n";
   cout << "hourly pay rates.\n";
   for (int index = 0; index < NUM_EMPLOYEES; index++)
   {
      cout << "Hours worked by employee #" << (index+1) << ": ";
      cin >> hours[index];
      cout << "Hourly pay rate for employee #" << (index+1) << ": ";
      cin >> payRate[index];
   }

```

*(Program Continues)*

## Parallel Array Example

Program 7-12 *(Continued)*

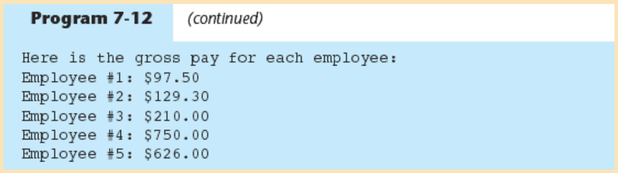
```
   // Display each employee's gross pay.
   cout << "Here is the gross pay for each employee:\n";
   cout << fixed << showpoint << setprecision(2);
   for (index = 0; index < NUM_EMPLOYEES; index++)
   {
      double grossPay = hours[index] * payRate[index];
      cout << "Employee #" << (index + 1);
      cout << ": $" << grossPay << endl;
   }
   return 0;
}
```

**Program Output with Example Input Shown in Bold**

```
Enter the hours worked by 5 employees and their
hourly pay rates.
Hours worked by employee #1: 10 [Enter]
Hourly pay rate for employee #1: 9.75 [Enter]
Hours worked by employee #2: 15 [Enter]
Hourly pay rate for employee #2: 8.62 [Enter]
Hours worked by employee #3: 20 [Enter]
Hourly pay rate for employee #3: 10.50 [Enter]
Hours worked by employee #4: 40 [Enter]
Hourly pay rate for employee #4: 18.75 [Enter]
Hours worked by employee #5: 40 [Enter]
Hourly pay rate for employee #5: 15.65 [Enter]
```

*(program output continues)*

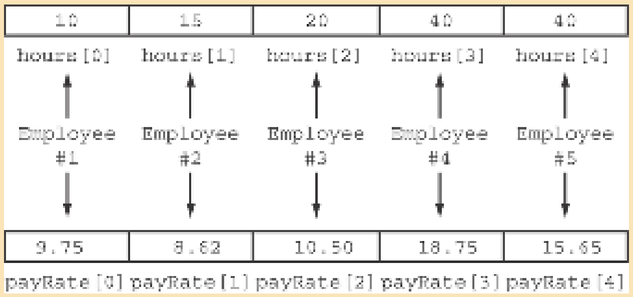
## Parallel Array Example



**Program 7-12** *(continued)*

```
Here is the gross pay for each employee:
Employee #1: $97.50
Employee #2: $129.30
Employee #3: $210.00
Employee #4: $750.00
Employee #5: $626.00
```

The `hours` and `payRate` arrays are related through their subscripts:



## Arrays as Function Arguments

- To pass an array to a function, just use the array name:

  `showScores(tests);`

- To define a function that takes an array parameter, use empty `[]` for array argument:

```
void showScores(int []); // function prototype
void showScores(int tests[])// function header
```

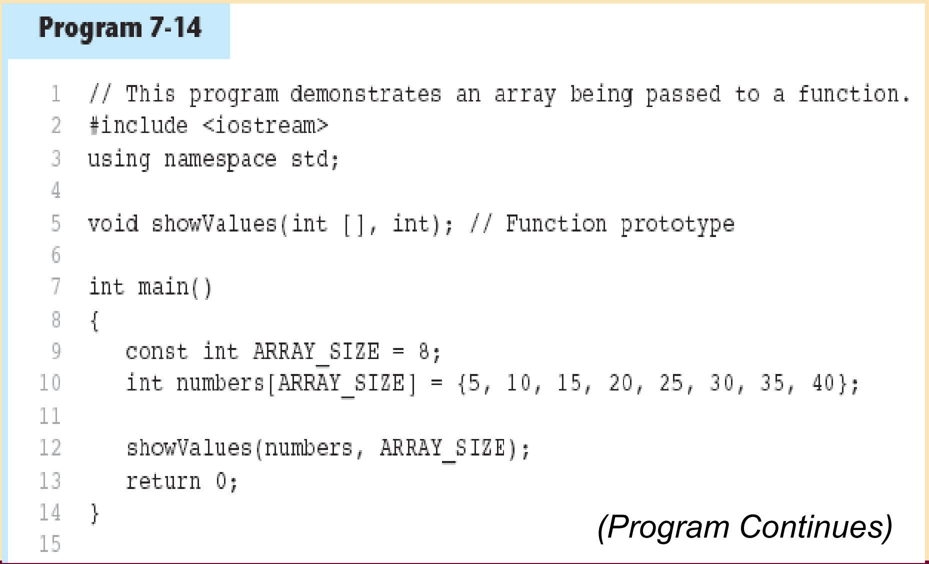
## Arrays as Function Arguments

- When passing an array to a function, it is common to pass array size so that function knows how many elements to process:

  ```
  showScores(tests, ARRAY_SIZE);
  ```
- Array size must also be reflected in prototype, header:

  ```
  void showScores(int [], int);
             // function prototype
  void showScores(int tests[], int
   size)
             // function header
  ```

## Arrays as Function Arguments - example



**Program 7-14**

```
// This program demonstrates an array being passed to a function.
#include <iostream>
using namespace std;

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

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

   showValues(numbers, ARRAY_SIZE);
   return 0;
}

```

*(Program Continues)*

## Arrays as Function Arguments - example

### Program 7-14 *(Continued)*

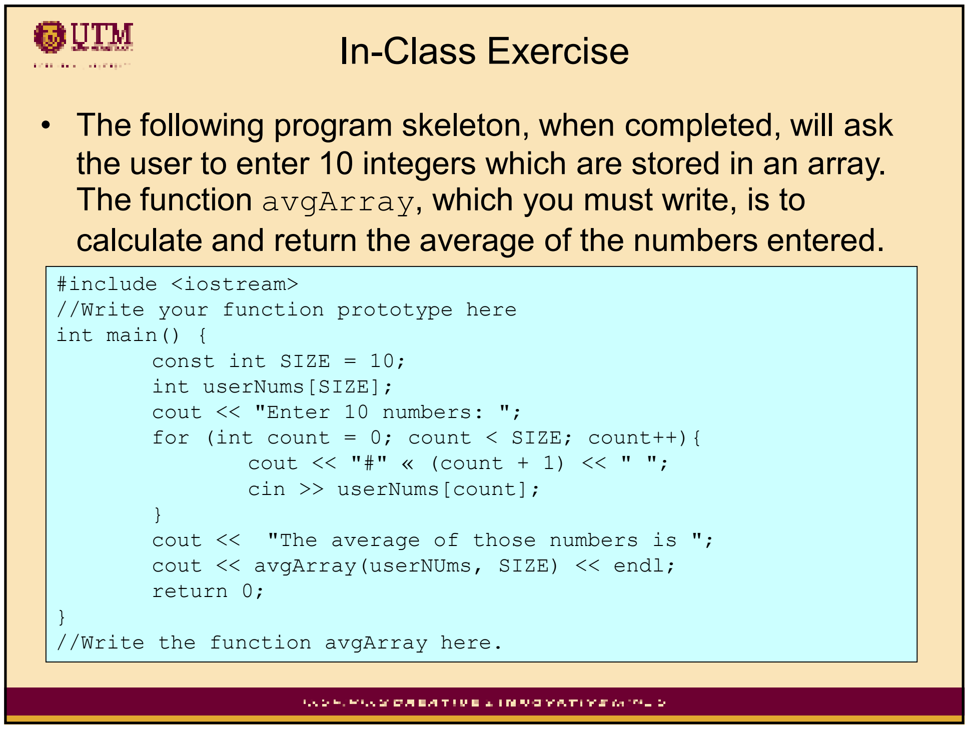
```
//*****************************************************
// Definition of function showValue.                  *
// This function accepts an array of integers and     *
// the array's size as its arguments. The contents    *
// of the array are displayed.                        *
//*****************************************************

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

**Program Output**

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

## Modifying Arrays in Functions

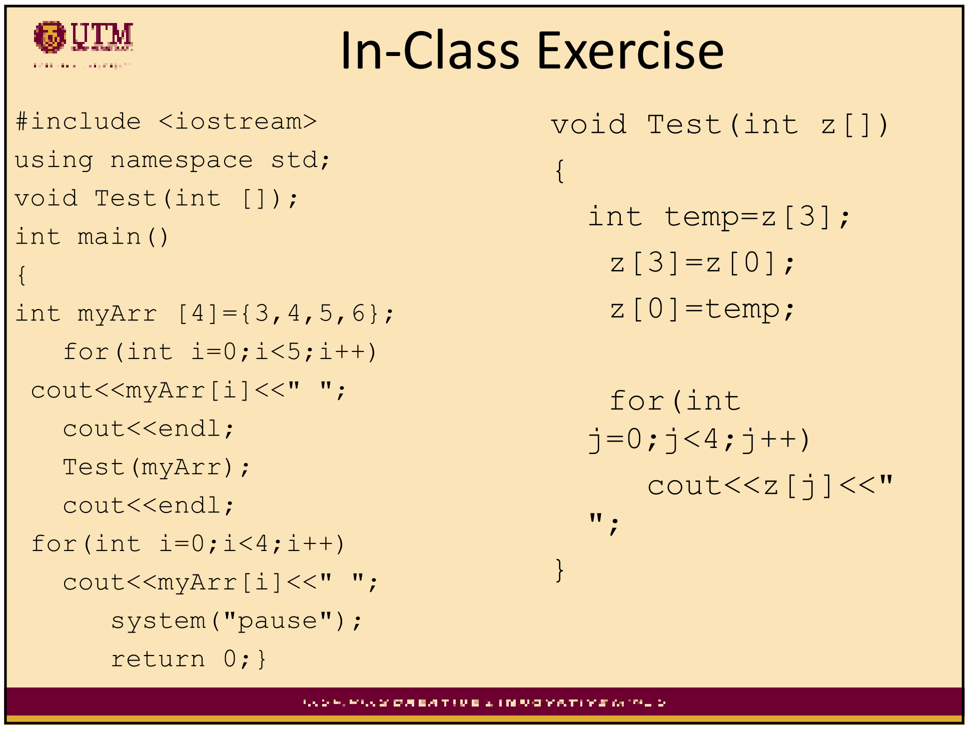
- Array names in functions are like reference variables – changes made to array in a function are reflected in actual array in calling function
- Need to exercise caution that array is not inadvertently changed by a function



## In-Class Exercise

- The following program skeleton, when completed, will ask the user to enter 10 integers which are stored in an array. The function `avgArray`, which you must write, is to calculate and return the average of the numbers entered.

```
#include <iostream>
//Write your function prototype here
int main() {
        const int SIZE = 10;
        int userNums[SIZE];
        cout << "Enter 10 numbers: ";
        for (int count = 0; count < SIZE; count++){
                cout << "#" « (count + 1) << " ";
                cin >> userNums[count];
        }
        cout <<  "The average of those numbers is ";
        cout << avgArray(userNUms, SIZE) << endl;
        return 0;
}
//Write the function avgArray here.
```



## In-Class Exercise

```
#include <iostream>
using namespace std;
void Test(int []);
int main()
{
int myArr [4]={3,4,5,6};
   for(int i=0;i<5;i++)
 cout<<myArr[i]<<" ";
   cout<<endl;
   Test(myArr);
   cout<<endl;
 for(int i=0;i<4;i++)
   cout<<myArr[i]<<" ";
      system("pause");
      return 0;}
```

```
void Test(int z[])
{
  int temp=z[3];
   z[3]=z[0];
   z[0]=temp;

   for(int
  j=0;j<4;j++)
      cout<<z[j]<<"
  ";
}
```

## In-Class Exercise

```
#include <iostream>
using namespace std;
void Test(int , int,int[]);
int main()
{  int x = 1;
   int y[3];
   y[0]=1;
   Test(x,y[0],y);
   cout<<"x is: " << x<<
   endl;
   cout<<"y[0] is: " <<y[0]
   << endl;
   for(int i=0;i<3;i++)
             cout<<y[i]<<endl;
        system("pause");
        return 0;}
```

```
void Test(int num, int num1,
   int z[])
{
   num=1001;
   num1=290;
   z[1]=34;
   z[2]=35;
}
```

## In-Class Exercise

- What is the output of the following code? (You may need to use a calculator.) .

```
const int SIZE = 5;
int time[SIZE] = {1, 2, 3, 4, 5},
speed[SIZE] = {18, 4, 27, 52, 100},
dist[SIZE];

for (int count = 0; count < SIZE; count++)
        dist[count] = time[count] * speed[count];
for (int count = 0; count < SIZE; count++) {
        cout << time[count] << " ";
        cout << speed[count] << " ";
        cout « dist[count] << endl;
}
```

## In-Class Exercise

- Write a program that store the populations of 12 countries. Define 2 arrays that may be used in parallel to store the names of the countries and their populations. Write a loop that uses these arrays to print each country's name and its population.

## In-Class Exercise

- Each of the following definitions and program segments has errors. Locate as many as you can and correct the errors.

```
a)  void showValues(int nums)
       {
            for(int i = 0; i<8; i++)
                    cout<<nums[i];
       }
```

```
b)  void showValues(int nums [4])
{
     for(int i = 0; i<8; i++)
             cout<<nums[i];
}
```

## In-Class Exercise

- Consider the following function prototypes:

```
void funcOne(int [], int);
int findSum(int, int);
```

And the declarations:

```
int list[50];
int num;
```

Write a C++ statements that:

a) Call the function `funcOne` with the actual parameters, `list` and 50 respectively.

b) Print the value returned by the function `funcSum` with the actual parameters, 50, and the fourth element of `list` respectively.

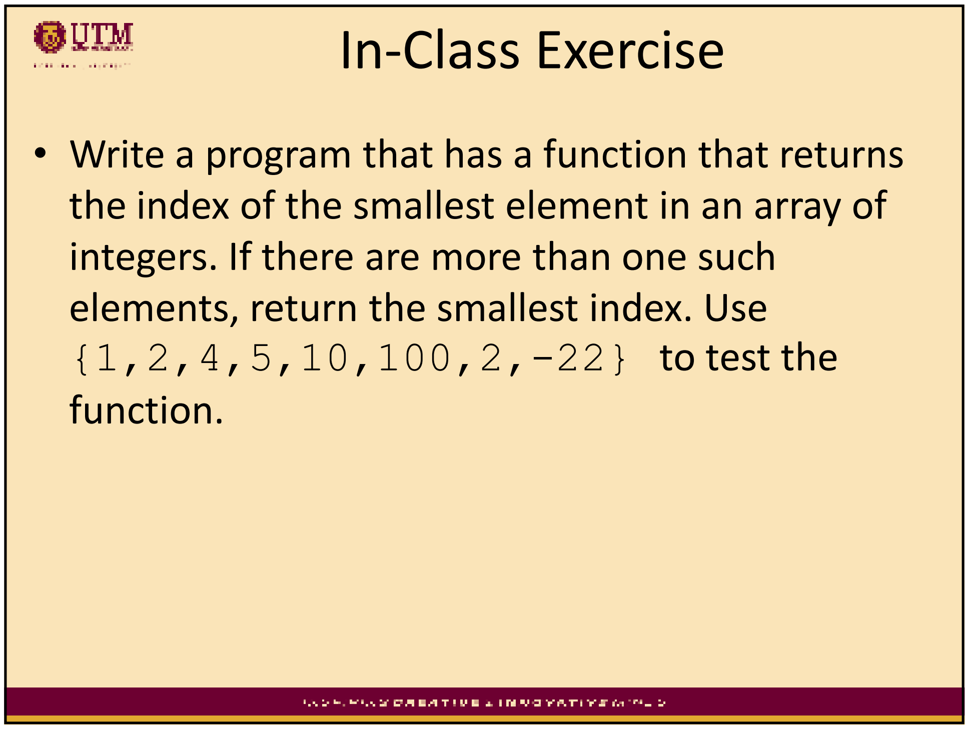
c) Print the value returned by the function `funcSum` with the actual parameters, the thirtieth and tenth elements of `list`, respectively.

## In-Class Exercise

- Write a program that has two overloaded functions that return the average of an array with the following headers:

```
int average(int array[], int size)
double average(int array[], int size
```

Use `{1,2,3,4,5,6}` and `{6.0,4.4,1.9,2.9,3.4,3.5}` to test the functions.



## In-Class Exercise

- Write a program that has a function that returns the index of the smallest element in an array of integers. If there are more than one such elements, return the smallest index. Use `{1,2,4,5,10,100,2,-22}` to test the function.