



Programming Fundamentals

Aamina Batool

Arrays as Parameters to Functions

- Arrays are passed by reference only
- The symbol `&` is *not* used when declaring an array as a formal parameter

Arrays as Parameters to Functions (continued)

- If the size of one-dimensional array is specified when it is declared as a formal parameter
 - It is ignored by the compiler
- The reserved word `const` in the declaration of the formal parameter can prevent the function from changing the actual parameter

EXAMPLE 9-5

```
void funcArrayAsParam(int listOne[], double listTwo[])
{
    .
    .
    .
}
```

Constant Arrays as Formal Parameters

```
void example(int x[], const int y[], int sizeX, int sizeY)
{
    .
    .
    .
}
```



Passing Arrays to Functions

- Array size is usually passed to function
- Arrays passed call-by-reference
- Value of name of array is address of the first element
- Function knows where the array is stored
 - Modifies original memory locations
- Individual array elements passed by call-by-value
 - pass subscripted name (i.e., **myArray**[3 1]) to function



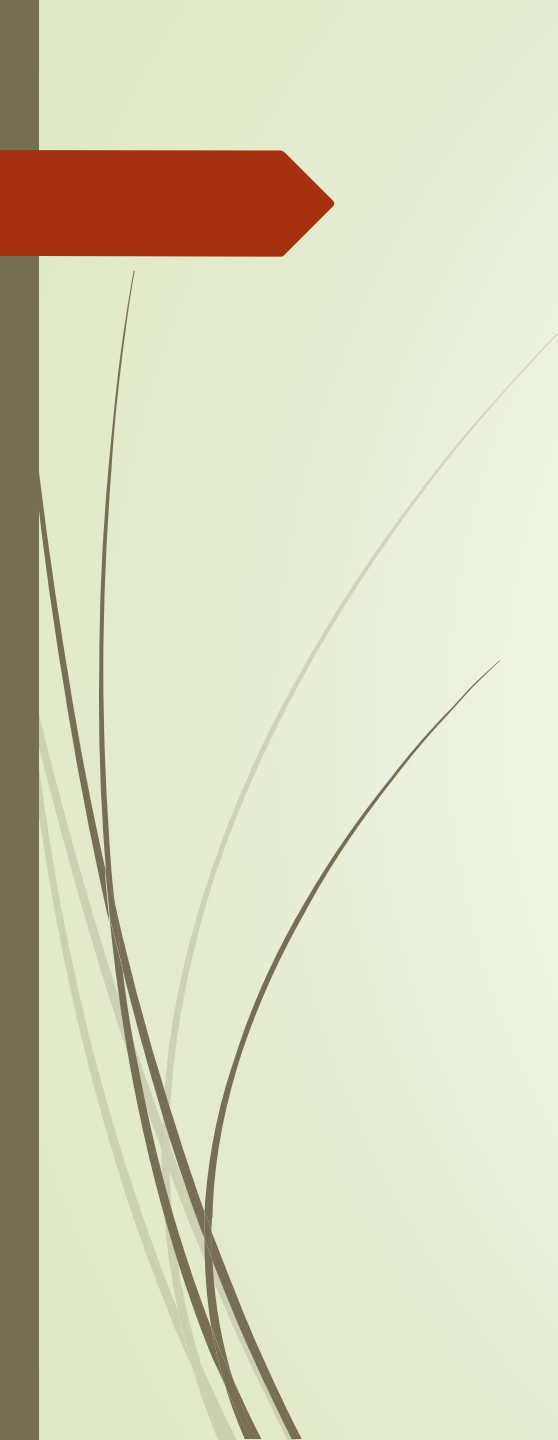
Passing Arrays to Functions

➡ Function prototype:

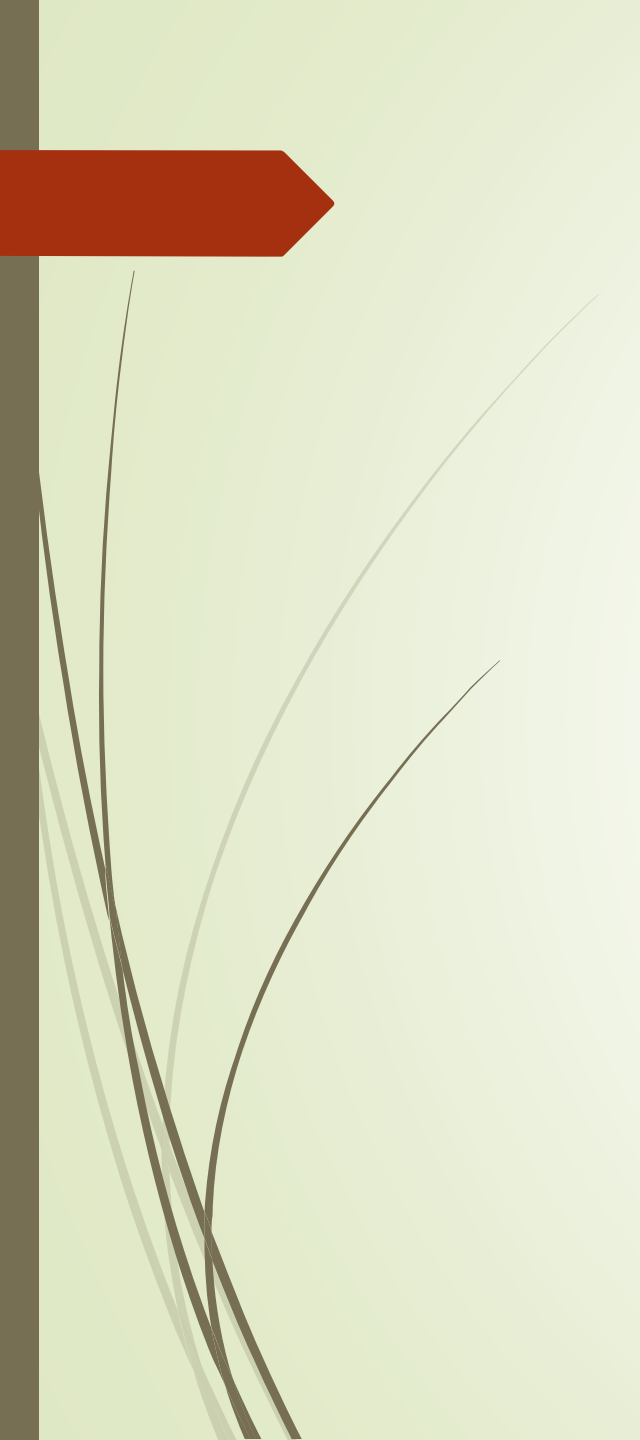
➡ `void modifyArray(int b[], int arraySize);`

EXAMPLE 9-6

```
//Function to initialize an int array to 0.  
//The array to be initialized and its size are passed  
//as parameters. The parameter listSize specifies the  
//number of elements to be initialized.  
void initializeArray(int list[], int listSize)  
{  
    int index;  
  
    for (index = 0; index < listSize; index++)  
        list[index] = 0;  
}  
  
//Function to read and store the data into an int array.  
//The array to store the data and its size are passed as  
//parameters. The parameter listSize specifies the number  
//of elements to be read.  
void fillArray(int list[], int listSize)  
{  
    int index;  
  
    for (index = 0; index < listSize; index++)  
        cin >> list[index];  
}
```



```
//Function to print the elements of an int array.  
//The array to be printed and the number of elements  
//are passed as parameters. The parameter listSize  
//specifies the number of elements to be printed.  
void printArray(const int list[], int listSize)  
{  
    int index;  
  
    for (index = 0; index < listSize; index++)  
        cout << list[index] << " ";  
}  
  
//Function to find and return the sum of the  
//elements of an int array. The parameter listSize  
//specifies the number of elements to be added.  
int sumArray(const int list[], int listSize)  
{  
    int index;  
    int sum = 0;  
  
    for (index = 0; index < listSize; index++)  
        sum = sum + list[index];  
  
    return sum;  
}
```

```
//Function to find and return the index of the first
//largest element in an int array. The parameter listSize
//specifies the number of elements in the array.
int indexLargestElement(const int list[], int listSize)
{
    int index;
    int maxIndex = 0; //assume the first element is the largest

    for (index = 1; index < listSize; index++)
        if (list[maxIndex] < list[index])
            maxIndex = index;

    return maxIndex;
}

//Function to copy one array into another array.
//The elements of listOne are copied into listTwo.
//The array listTwo must be at least as large as the
//number of elements to be copied. The parameter
//listOneSize specifies the number of elements of
//listOne to be copied into listTwo.
void copyArray(const int listOne[], int listTwo[],
               int listOneSize)
{
    int index;

    for (index = 0; index < listOneSize; index++)
        listTwo[index] = listOne[index];
}
```

Base Address of an Array

- The base address of an array is the address, or memory location of the first array component
- If `list` is a one-dimensional array
 - base address of `list` is the address of the component `list[0]`
- When we pass an array as a parameter
 - base address of the actual array is passed to the formal parameter
- Functions cannot return a value of the type array



Exercises

- Functions for:
 - for input, output, search, reverse,
 - Shifting and Rotation of elements: right and left
 - Insert and delete elements from ordered list using shifting.



References



1. C++ Programming: From Problem Analysis to Program Design, Third Edition
2. <https://www.just.edu.jo/~yahya-t/cs115/>