UNIVERSITI TEKNOLOGI MARA

CSC 128

TOPIC 6 : ARRAY

By : MOHD SAIFULNIZAM ABU BAKAR

COURSE OUTLINE

At the end of this chapter, you should be able to:

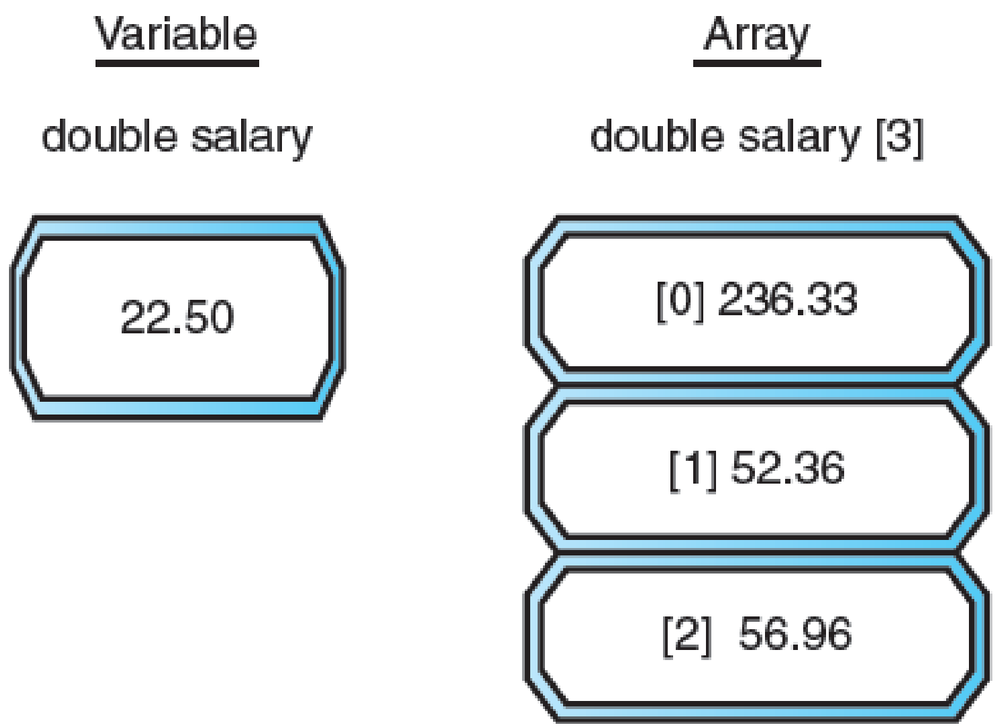
- Understand the concept of array.
- Perform the array declaration and initialization.
- Apply on how to access elements in an array.
- Perform operations (summation, maximum and minimum) in an array.
- Perform passing array to function.

INTRODUCTION

- In C++ programming, an array is a data structure. It is process of group the same categories of data.
- **Array** is a collection of data of the same data type. An array allows us to group and store same variables which have a same data type.
- Usually a variable can hold one value at a time, but by using an array, we can store more than one value in a single variable.

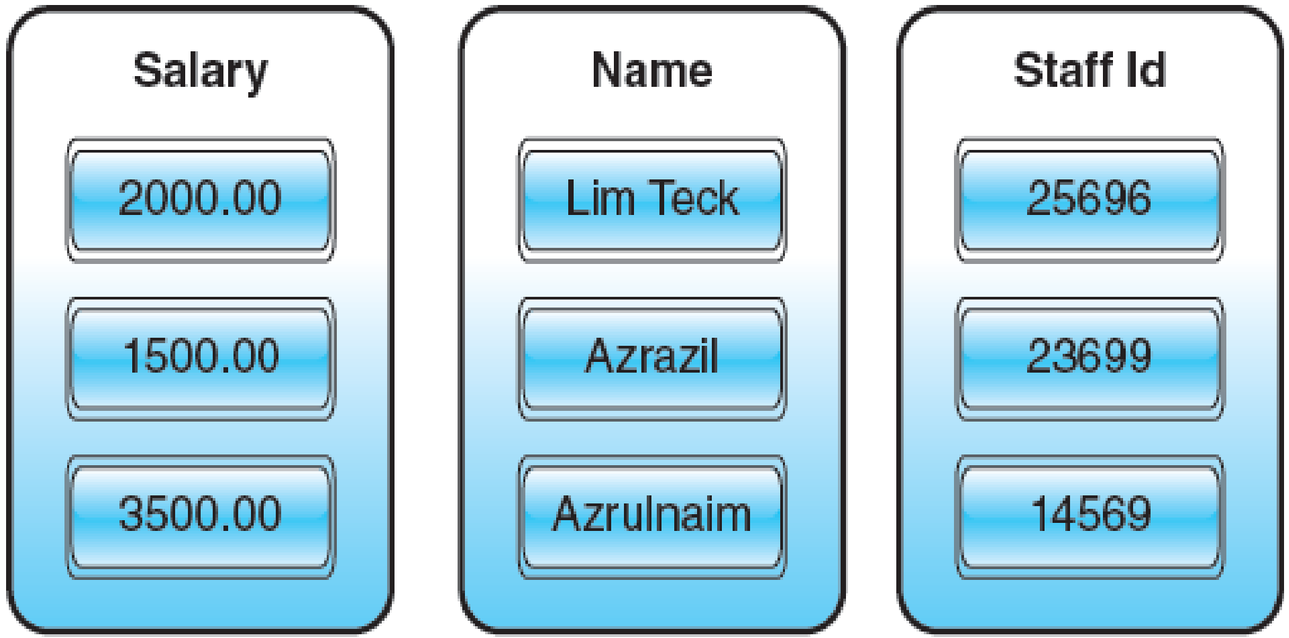
INTRODUCTION (CONT.)

- Figure 6.1 shows the difference between variable and array.



INTRODUCTION (CONT.)

- Figure 6.2 shows the arrays for variables-**salary**, **name** and **staff ID**.



INTRODUCTION (CONT.)

- An array can store many values, and each of the values is known as an **element** and has its own **array index** that can be used to access it.
- A variable can be easily retrieved by referring to the index.
- An **array index** is also known as **subscripts**.
- A subscript is a number that indicates the position of a particular array element that is being used.

ARRAY DECLARATION

- As in normal programming steps, we have to declare the entire variable before we use it. Like normal variables, arrays also have to be declared.
- An array can be declared as:

```
datatype arrayName[size];
```

  - For array declarations, firstly, type any simple data type, continue with arrayName, with any legal identifier, and the size (in square bracket).
  - The size represents the number of elements the array contains.

ARRAY DECLARATION (CONT.)

- Example of array declaration statement:

a) int listNumber [100];

- the array named listNumber which reserve 100 elements of integer values.

b) char codeId [6];

- the array reserves 6 element of characters named codeId.

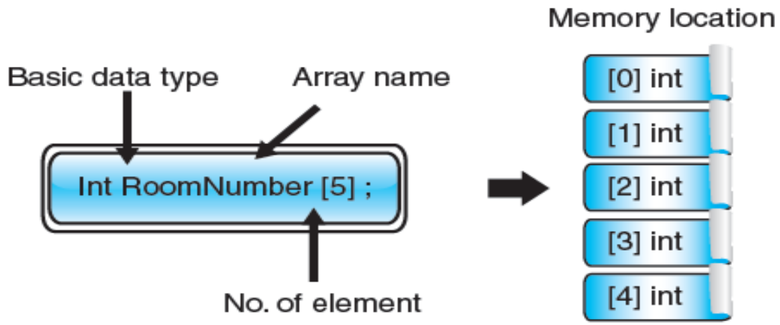


Figure 6.5 Illustration of one dimensiona

ARRAY INITIALIZATION

- In C++ programming, we can assign or initialize values to the array. Initialization means assigning values to an array.
- The format to write the array initialization is:

```
datatype arrayName[size]={elements};
```

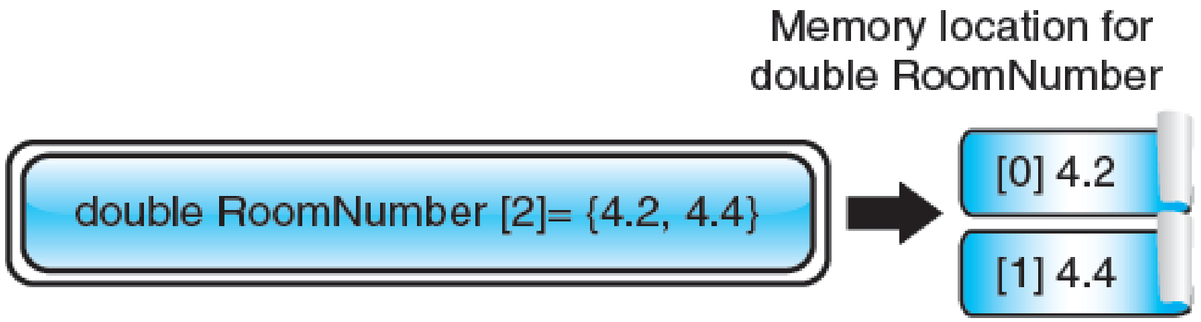
- An array's initial values are written using curly brackets
- { } and commas (,) to differentiate each element in the array as shown in the following example.

ARRAY INITIALIZATION (CONT.)

EXAMPLE 6.2

```
(a) int NoCode [5] = {5, 2, 6, 8, 7};
(b) double salary [3] = {12.5,  15.6,  25.6};
(c) char codeId [4] = {'b',  'o',  'o',  'k'};
(d) char codeId [6] = "book"; or char code [7] = {'book'};
```

Note : a string is terminated with a special sentinel,"\0" the null character.



ARRAY INITIALIZATION

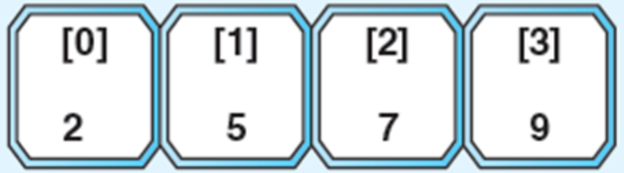
- We can also initialize the values to an array without declaring the size of the array.
- The number of elements in an array can define the size of an array as shown in Example 6.3.
- This means we can declare an array with or without size, but we must initialize the values to it.
- There can also be a situation whereby we have declared the size of an array with 6 and initialized the values for 3 elements only—the compiler will assume the next three elements are zero as shown in Example 6.4.

ARRAY INITIALIZATION (CONT.)

EXAMPLE 6.3

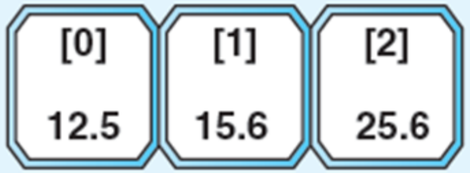
(a) `int OddList [ ] ={2,5, 7,9};`

Memory Location for OddList [ ]



(b) `double salary [ ] ={12.5, 15.6, 25.6};`

Memory Location for salary [ ]

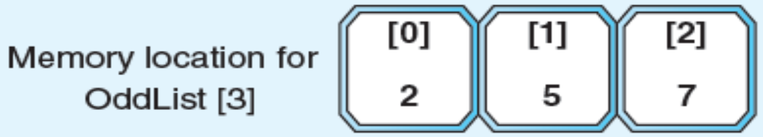


ARRAY INITIALIZATION (CONT.)

EXAMPLE 6.4

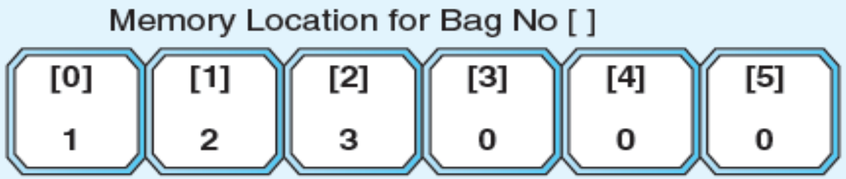
(a) `int OddList [3] ={2,5,7,9};`

The size defined is 3 but the values initialized are for 4 elements, so the compiler will ignore the last value based on the sequence. So, the actual values in the array are: {2,5,7}



(b) `int BagNo [6] ={1,2,3};`

The size defined is 6 but the value initialized is only for 3 elements, so the compiler will assume the remaining 3 elements will hold zero values:
{1,2,3,0,0,0};



RESTRICTIONS IN ARRAY OPERATION

- There are some restrictions in array operation that we should consider in a few situations.
- In the first situation, an error message will appear once we want to copy one array to another array as shown in Example 6.5.

EXAMPLE 6.5

Given:

```
int  ListA [3] ={1,2,3};
int  ListB [3];
ListB = ListA;   //will show an error message
```

To overcome it, we will use the looping method to copy each element in ListA to ListB:

```
for (int index=0; index<3;index++)
     ListB [index] = ListA [index];
```

RESTRICTIONS IN ARRAY OPERATION (CONT.)

- We will also have a problem once we want to insert values into an array. We cannot insert or read data to an array using the normal way.
- For the solution, we have to insert data to an array using the looping method as shown in Example 6.6.

EXAMPLE 6.6

Given :

```
cin>>ListA;    // will show an error
```

To overcome it, we will use the looping method to insert elements into ListA:

```
for (int index=0;index<3;index++)
cin>>ListA[index];
```

RESTRICTIONS IN ARRAY OPERATION (CONT.)

- In another situation, we have to specify the index number once to retrieve the value from an array or it will display an error message. Example 6.7 shows the solution for the situation.

EXAMPLE 6.7

Given :

```
cout<<ListA;   //will show an error message
```

To overcome it, we will use the looping method to insert elements into ListA:

```
for  (int index=0; index<3;index++)
     cout<<ListA [index];
or
cout<<ListA [index];
```

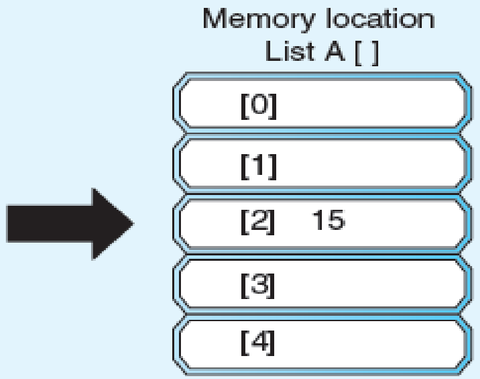
ACCESSING ARRAY ELEMENTS

- The **for loop** is commonly used to manipulate all the operations in arrays. Usually, with the iteration or loop process, it will make the process easier because it allows us to access the elements by referring to the subscript in the sequence.
- Besides using **for loop** to perform the accessing process, we also can access and initialize the elements using the various methods as shown in Example 6.8.

ACCESSING ARRAY ELEMENTS (CONT.)

EXAMPLE 6.8

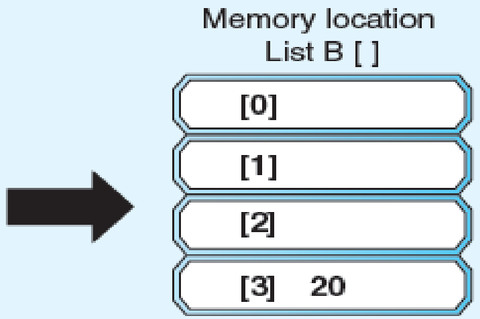
(a) `listA[2]=15;`



Value 15 is assigned to the subscript index 2 in ListA [2].

(b) `ListB[3]= 10*2;`

The total 20 is calculated and stored into element subscript 3 in the array.



ACCESSING ARRAY ELEMENTS (CONT.)

- Besides all the methods shown, we can use other ways such as the **for loop** to initialize or access arrays. Example 6.9 shows the general form of how to read elements into an array.
- As in a normal iteration process, we have to define the **for loop** argument based on the array size.
- As usual, we have to start the iteration with zero (0), and stop the loop once we reach **the arraysize-1.**
- The counter for the loop should be increased by one with each iteration.

ACCESSING ARRAY ELEMENTS (CONT.)

EXAMPLE 6.9

```
for   (int index=0; index<arraysize; index++)
      cin>>listA [index];
```

- Example 6.10 shows the general form to display the array's contents. Here, we will also use the **for loop** method to display all the elements of an array by sequence

EXAMPLE 6.10

```
for   (int index=0; index<arraysize; index++)
      cout<< listA [index]<<endl;
```

ACCESSING ARRAY ELEMENTS (CONT.)

- Example 6.11 shows a complete C++ program that creates an array and display its contents. This program will have an array with 10 elements that will be initialized with values and then display them.

EXAMPLE 6.11

```
//Program to create an array named ListNumber that can hold 10 initialized
values and display them.
#include <iostream>
using namespace std;
int main( )

{
const int ArraySize = 10;
double ListNumber[ArraySize];   //array declaration
int x;

//enter numbers into the array
for  (x = 0; x<ArraySize; x++)
{
     ListNumber [x] = x;//initialized
}

cout<<"List of the array's contents: \n";

//print the numbers in the array
for (x = 0; x<ArraySize; x++) {

     cout<<x+1<<"."<<ListNumber[x]<<endl;
}
return  0;
}
```

ACCESSING ARRAY ELEMENTS (CONT.)

- Example 6.12 shows a complete C++ program that creates an array and displays its contents.
- The program will have two arrays, named **StudentNo** and **Student_Marks**. Both arrays will store 3 values each.
- The program will prompt the user to insert the values for the arrays.
- Then, the program will display the contents of the arrays in table form as shown in the output.

ACCESSING ARRAY ELEMENTS (CONT.)

EXAMPLE 6.12

```
//Program to create two arrays, named StudentNo and Student_Marks, that
can read 5 values each and display them
#include <iostream>
using namespace std;
int main()
{
//array declaration
int StudentNo [5];
double Student_Mark [5];
int i,x;

//enter numbers into the array
for (i = 0; i<5; i++)
{          cout<<"Enter Student No: ";
           cin>>StudentNo[i]; // read data into array
           cout<"Enter Student marks: ";
           cin>>Student_Mark[i]; // read data into array
}

cout<<"List of student marks: "<<endl;
cout<<"Student No\t\t\t Marks"<<endl;
//print the numbers in the array
for (x = 0; x<5; x++){
     i=x;

 cout<<i+1 <<"."<<StudentNo[x]<<"\t\t\t\t\t"<<Student_Mark[x]<< endl;
}
return  0;
}
```

```
OUTPUT
Enter   Student   No:   12345
Enter   Student   marks: 45
Enter   Student   No:   2636
Enter   Student   marks: 60
Enter   Student   No:   9969
Enter   Student   marks: 90
Enter   Student   No:   2636
Enter   Student   marks: 60
Enter   Student   No:   9970
Enter   Student   marks: 99
```

```
List of student marks:
     Student No          Marks
 1.  12345               45
 2.  2636                60
 3.  9969                90
 4.  2636                60
 5.  99070               99
```

ACCESSING ARRAY ELEMENTS (CONT.)

- The program in Example 6.13 will declare an array named EvenList [ ] with 10 elements in it.
- The array will store 10 values consisting of even numbers that are less than 20, then display them.
- Then, the program will identify the even numbers from the list of numbers entered by the user.

ACCESSING ARRAY ELEMENTS (CONT.)

EXAMPLE 6.13

```
//Program will store even numbers from list in an array

##include <iostream>
using namespace std;
int main()
{
//array declaration
int List[10]={1,2,3,4,5,6,7,8,9,10};
int i;

cout<<"List of all numbers from the array list: "<<endl;

//print the all numbers in the array
for  (int x = 0; x<10; x++){

      cout<<List[x] <<",n";
}

cout<<"List of even numbers from the array list: "<<endl;

//print the even numbers in the array
for (int x = 0; x<10; x++){

      if (List[x]%2==0) {
      cout<<List[x] <<",n";
      }
}
return  0;
}
```

OUTPUT

List of numbers from the array list:
1, 2, 3, 4, 6, 6, 7, 8, 9, 10

List of even numbers from the array list:
2, 4, 6, 8, 10

ARRAY OPERATIONS

- In programming, we can perform a few operations to solve problems in arrays.
- These operations include summation, finding the minimum and maximum values, searching, deleting, merging, sorting, traversing, and inserting.
- These operations are shown in Figure 6.7.

Figure 6.7
List of array operations

TRAVERSING

- Traversing is an **operation whereby each element in an array is accessed for a specific purpose.**
- Actually, in the traversal operation, we will use the values in many programs because it is a basic operation that we have discussed before.
- The process of printing the elements, and calculations between the elements are all examples of the traversal operation.

INSERTING

- Inserting is a **process of adding a new element into an existing array.**
- We can add the element in the beginning, middle, or at the end of the array.
- The inserting process can be applicable only if the size that has been declared is large enough. For example, an array with size 5 has only 3 elements initialized to it. In this case, we can apply the insertion process.
- However, if there is not enough space (size) available, then we cannot proceed with the operation.

SEARCHING

- Searching means the **process of finding an element in the array**.
- If the element is in the array, the location or the subscript of the element will be displayed and if the element is not available in the array, an appropriate message will be displayed.
- There are two types of searching methods available—linear search and binary search.

SUMMATION

- Summation is a **process of adding two or more values together**.
- So, in arrays, summation can be performed to find the total of the values in an array.
- Besides that, summation can also be done between any elements in an array as will be discussed in Example 6.15.

SUMMATION (CONT.)

EXAMPLE 6.15

```
#include <iostream.h>
void main()
{
//array declaration
int list1 [4] = {1, 5, 9, 8};
int list2 [5] = {1, 5, 9, 8, 6};

//print the numbers after calculate them
      cout<<list1[3]+list2[3];
      cout<<list1[2]+list1[3];
      cout<<list2[4]+list1[3];
      cout<<list2[1]+list1[2];
}
```

SUMMATION (CONT.)

- Example 6.16 shows a program segment for summation operation to find the total of an array price[ ].
- Here, for calculating the total of the values in an array, we have declared a variable to accumulate the total.
- The variable should be initialized to zero (0) before we start to accumulate the value.

SUMMATION (CONT.)

EXAMPLE 6.16

```
#include <iostream>
using namespace std;
int main( )
{
      int total=0;double price [5] = {5.5, 6.5, 1.2, 3.5, 5.2};
      for (int index=0; index<5; index++)
      {

      total=total + price [index];

      }
      cout <<"The total of prices in the array : "<<total;
      return 0;
}
```

Index	price[ ]	total
0	5.5	5.5
1	6.5	12.0
2	1.2	13.2
3	3.5	16.7
4	5.2	21.9

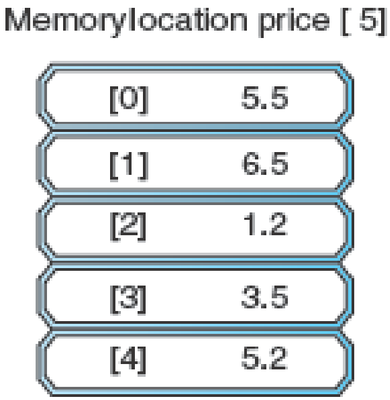


Figure 6.8
Process of calculating the summation in an array

FINDING THE MAXIMUM

- For finding the largest value in an array, we can use the maximum operation, which will compare each element in an array to find the largest value.
- Example 6.18 shows the general form to find the largest value. We will use for loops and if methods to check and compare each of the values in the array to find the maximum value.
- In the operation, firstly, we have to define a variable named MaxIndext and assign zero (0) to it.
- Then, we will start to compare the values in the array using the sequence until we fulfil the objective to find the largest value.

FINDING THE MAXIMUM

- Example 6.18 shows the general form to find the largest value.

EXAMPLE 6.18

```
MaxIndex=0;
for (index=1; index<5; index++))
{
     if (marks [MaxIndex]<marks [index]
          MaxIndex=index;
}
largest=marks [MaxIndex];
```

FINDING THE MAXIMUM

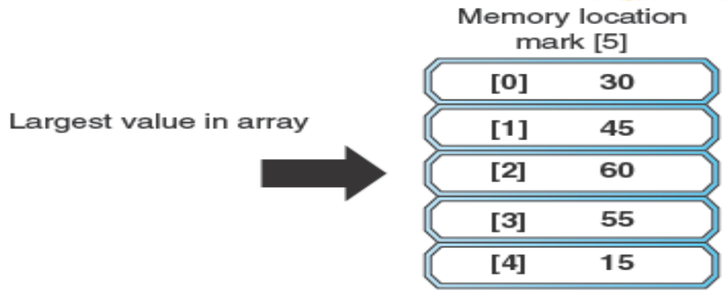
EXAMPLE 6.19

```
#include <iostream>
using namespace std;
int main()
{
   double marks [5] = {30, 45, 60, 55, 15};
   int MaxIndex=0;
   int largest;

   for  ( int index = 1; index<5; index++)
{
      if (marks [MaxIndex]<marks [index])
           MaxIndex=index;
}
largest=marks [MaxIndex];

cout <<"Largest value in array is : "<<largest;

return  0;
}
```



MaxIndex	index	marks [MaxIndex]	marks [index]
0	1	30	45
1	2	45	60
2	3	60	55
2	4	60	15

Figure 6.9
Table and memory allocation that illustrates the process of identifying the maximum value

FINDING THE MINIMUM

- We can find the lowest value in an array using the same steps for finding the maximum value. Here also, the operation will compare each element in an array to find the lowest value.
- Example 6.20 shows the general form to find the lowest value in an array. We will use for loops and if methods to check and compare each of the values in the array to find the minimum value.

FINDING THE MINIMUM

EXAMPLE 6.21

```
#include <iostream>
using namespace std;
int main()
{
      double marks [5] = {30, 45, 60, 55, 15}; int MinIndex=0;
      int lowest;
      for ( int index=1; index<5; index++)
      {
      if (marks [MinIndex]>marks [index])
            MinIndex=index;
      }
      lowest=marks[MinIndex];
cout <<"Lowest value in array is : "<<lowest;
return  0;
}
```

OUTPUT

Lowest value in array is: 15

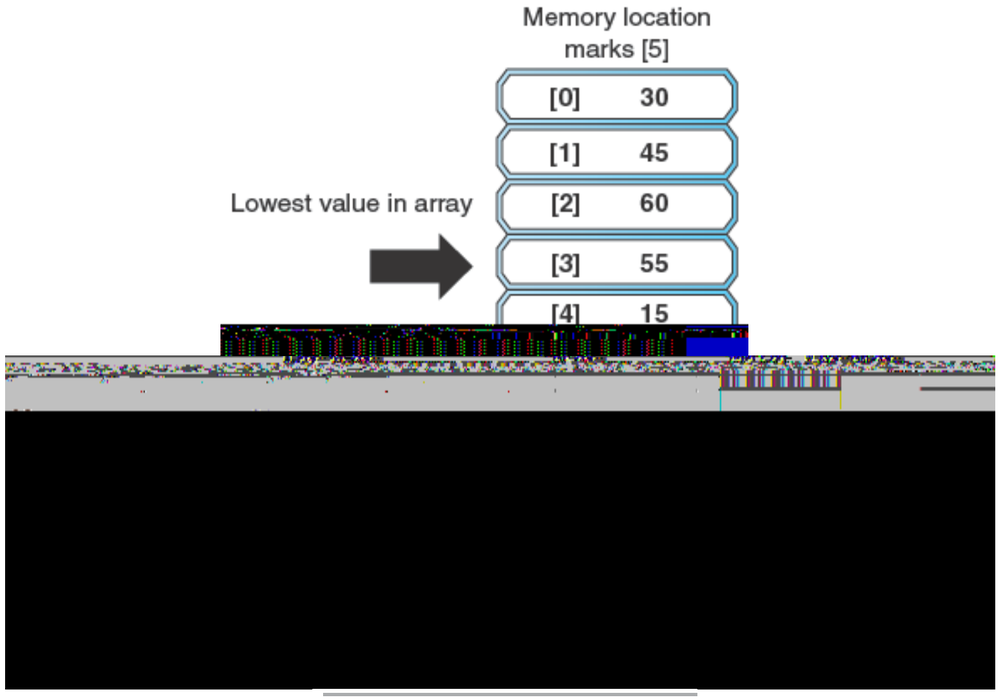


Figure 6.10
Illustration of the process of identifying the minimum value

PASSING ARRAY TO FUNCTION

- Arrays can also apply in functions in C++ program.
- An array can be passed as a parameter in a function. Example 6.22 shows the array usage in a function.
- The program shows that a function named **`calculateTotal()`** receives a parameter of type array.
- The function will receive the values in the array, then calculate the total. It returns the total to the main function.

PASSING ARRAY TO FUNCTION

EXAMPLE 6.22

```
#include<iostream>
using namespace std;
//function prototype
int calculateTotal (int [ ]);
int main( )
{
      int Num [ ] = {3, 5,7, 9};
      cout<<"Total is "<< calculateTotal (Num)<<endl;
      return 0;
}

int calculateTotal (int x [ ] )
{     int total=0;
      for (int index=0; index<4; index++)
      {
            total=total+x [index];
      }
      return total;
}
```

OUTPUT

```
Total is 24
```

CONCLUSION

- An array is a collection of similar data type value.
- In C++ programming, array is also known as a simple data structure.
- Array will store the group data for a temporary basis in memory.
- Each data in the array is stored in a sequence and has its own address that known as index. Index is also called as subscript.
- Summation, and finding the maximum and minimum values, are some common operations of an array method.