BSCS-Spring-2021

CL118-Programming Fundamentals

Lab Manual 12

LAB 12 Manual

Programming Fundamentals

Topic: Arrays-I (1D arrays)

Arrays:

An array is a data structure for storing more than one data item that has a similar data type. Values are stored in adjacent memory locations.

Declared using [] operator:

int tests[5];

In the definition int tests[5];

1. int is the data type of the array elements

2. tests is the name of the array

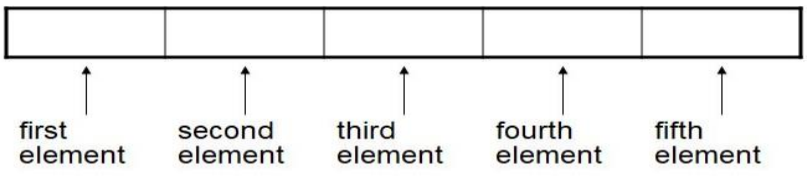
3. 5, in [5], is the size declarator. It shows the number of elements in the array.

4. The size of an array is the total number of bytes allocated for it. The size of an array is (number of elements) * (size of each element) e.g., **int tests[5] is an array of 20 bytes, assuming 4 bytes for an int**

Array in memory:

The definition: int tests[5];

allocates the following memory:

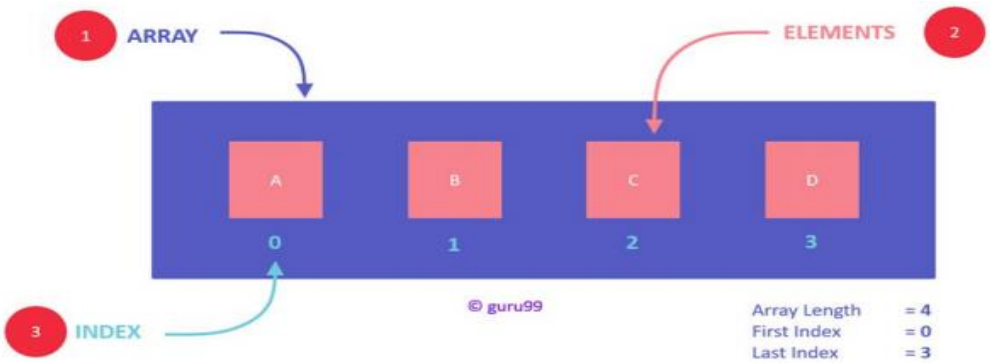


Arrays Terminology:

•Each element in an array is assigned a unique subscript.

•Subscripts start at 0

•The last element's subscript is n-1 where n is the number of elements in the array.



Size Declarators:

• Named constants are commonly used as size declarators.

const int SIZE = 5;

int tests[SIZE];

Accessing Array Elements:

• Array elements can be used as regular variables:

tests[0] = 79;

cout << tests[0];

cin >> tests[1];

tests[4] = tests[0] + tests[1];

• Arrays must be accessed via individual elements: **cout << tests; // not legal, it only works with character arrays**

• Can access element with a constant or literal subscript: **cout << tests[3] << endl;**

• Can use integer expression as subscript:

int i = 5;

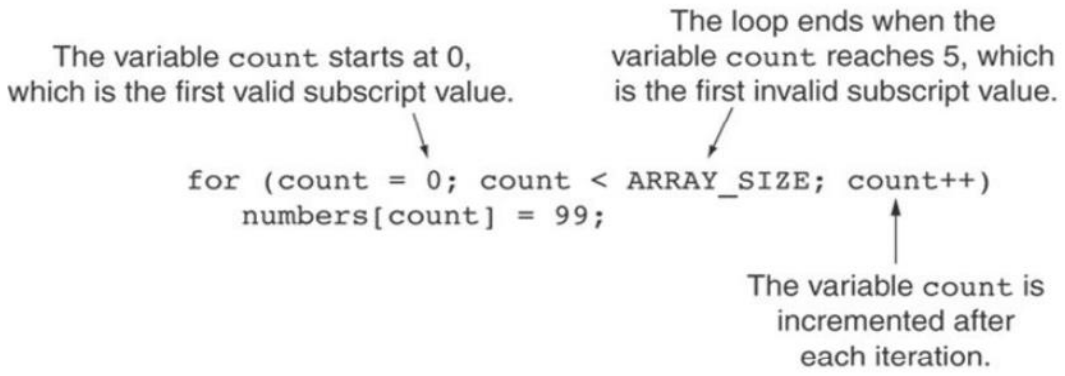
cout << tests[i] << endl;

Looping over an array:

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

int main()
{
    const int ARRAY_SIZE = 5;
    int numbers[ARRAY_SIZE];
    for (int count = 0; count < ARRAY_SIZE; count++)
     numbers[count] = 99;

    return 0;
}
```

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

int main()
{
    const int ARRAY_SIZE = 5;
    int numbers[ARRAY_SIZE];
    for (int count = 0; count < ARRAY_SIZE; count++)
     numbers[count] = count;

    return 0;
}
```

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

int main()
{
    const int ARRAY_SIZE = 5;
    int numbers[ARRAY_SIZE];
    for (int count = 0; count < ARRAY_SIZE; count++)
        cin>>numbers[count];

    return 0;
}
```



Array Initialization:

• Arrays can be initialized with an **initialization list**:

const int SIZE = 5;

int tests[SIZE] = {79, 82, 91, 77, 84};

• The values are stored in the array in the order in which they appear in the list.

• The initialization list cannot exceed the array size.

Implicit Array Sizing:

• Can determine array size by the size of the initialization list:

int quizzes[]={12, 17, 15, 11};

12	17	15	11

• Must use either array size declarator or initialization list at array definition

- Once the array is declared (it's size is set for fixed size arrays), you cannot change the size again. That is, you cannot add more elements to it

Printing out array using loops:

```
7       const int MONTHS = 12;
8       int days[MONTHS] = { 31, 28, 31, 30,
9                            31, 30, 31, 31,
10                           30, 31, 30, 31};
11
12      for (int count = 0; count < MONTHS; count++)
13      {
14         cout << "Month " << (count + 1) << " has ";
15         cout << days[count] << " days.\n";
16      }
```

Program Output

```
Month 1 has 31 days.
Month 2 has 28 days.
Month 3 has 31 days.
Month 4 has 30 days.
Month 5 has 31 days.
Month 6 has 30 days.
Month 7 has 31 days.
Month 8 has 31 days.
Month 9 has 30 days.
Month 10 has 31 days.
Month 11 has 30 days.
Month 12 has 31 days.
```

Processing Array Contents:

- Array elements can be treated as ordinary variables of the same type as the array
- When using ++, -- operators, don't confuse the element with the subscript:

tests[i]++; // add 1 to tests[i]

tests[i++]; // increment i, no effect on tests

- To copy one array to another, don't try to assign one array to the other:

newTests = tests; // Won't work

- Instead, assign element-by-element:

```
for (i = 0; i < ARRAY_SIZE; i++)
    newTests[i] = tests[i];
```

Comparing Arrays:

• To compare two arrays, you must compare element-by-element

```
const int SIZE = 5;
int firstArray[SIZE] = { 5, 10, 15, 20, 25 };
int secondArray[SIZE] = { 5, 10, 15, 20, 25 };
bool arraysEqual = true; // Flag variable
int count = 0;           // Loop counter variable
// Compare the two arrays.
while (arraysEqual && count < SIZE)
{
   if (firstArray[count] != secondArray[count])
      arraysEqual = false;
   count++;
}
if (arraysEqual)
   cout << "The arrays are equal.\n";
else
   cout << "The arrays are not equal.\n";
```

Example: The following example displays the sum of elements of array.

Example	OUTPUT

#include <iostream>
using namespace std;
 	
int foo [] = {16, 2, 77, 40, 12071};
int n, result=0;
 	
int main ()
{
for (n=0 ; n<5 ; ++n)
{
result += foo[n];
}
cout << result;
return 0;
}
	12206

Example: The following example shows how to Find the Highest and Lowest Values in a Numeric Array

HIGHEST

```
int count;
int highest;

highest = numbers[0];
for (count = 1; count < SIZE; count++)
{
   if (numbers[count] > highest)
      highest = numbers[count];
}
```

LOWEST

```
int count;
int lowest;

lowest = numbers[0];
for (count = 1; count < SIZE; count++)
{
   if (numbers[count] < lowest)
      lowest = numbers[count];
}
```

Lab Tasks

Q1) Write a C++ program to find and print all unique elements of a given array of integers.

Sample Output:

```
Original array: 1 5 7 5 8 9 11 11 2 5 6
Unique elements of the said array: 1 5 7 8 9 11 2 6
```

Q2) Write a C++ program to find the number of pairs of integers in a given array of integers whose sum is equal to a specified number taken as user input.

Sample Output:

```
Original array: 1 5 7 5 8 9 11 12
Array pairs whose sum equal to 12:
1,11
5,7
7,5
Number of pairs whose sum equal to 12: 3
```

Q3) Take 10 integer inputs from user and store them in an array. Now, copy all the elements in another array but in reverse order and show the output. Write a C++ program for it.

Q4) Write a C++ program to sort an array in ascending order.

Sample Output:

```
Enter any 10 num in array:
2 5 1 7 5 3 8 9 11 4
Data After Sorting: 1 2 3 4 5 7 8 9 11
```

Q5) Write a C++ program to sort an array in descending order.

Sample Output:

```
Enter any 10 num in array:
2 5 1 7 5 3 8 9 11 4
Data After Sorting: 11 9 8 7 5 4 3 2 1
```

Submission Instructions:

1. Save all .cpp files with your roll no and task number e.g. i20XXXX_Task01.cpp

2. Now create a new folder with name ROLLNO_LAB12 e.g. i20XXXX_LAB12

3. Move all of your .cpp files to this newly created directory and compress it into .zip file.

4. Now you have to submit this zipped file on Google Classroom.