

Programming fundamentals

Lecture 11: Sorting in arrays, different data type arrays, exploring cin, char array vs strings, getline function

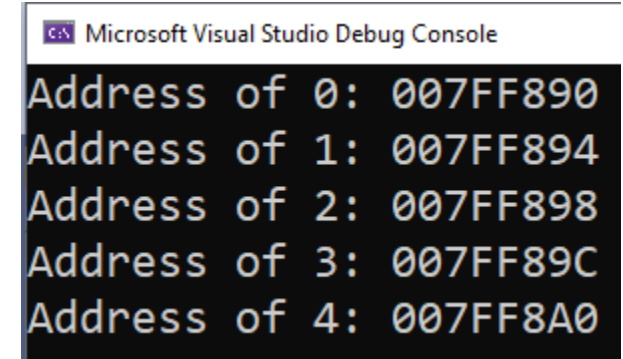
Recap

- Constants
- Introduction to arrays
- Basic input/output using arrays

Example: Printing the address of array cells

```
#include<iostream>
using namespace std;
int main()
{
    int C[5] = {5,7,9,6,5};
    for (int i = 0; i < 5; i++)
    {
        cout << "Address of "<<i<<": " << &C[i] << endl;
    }

    return 0;
}
```

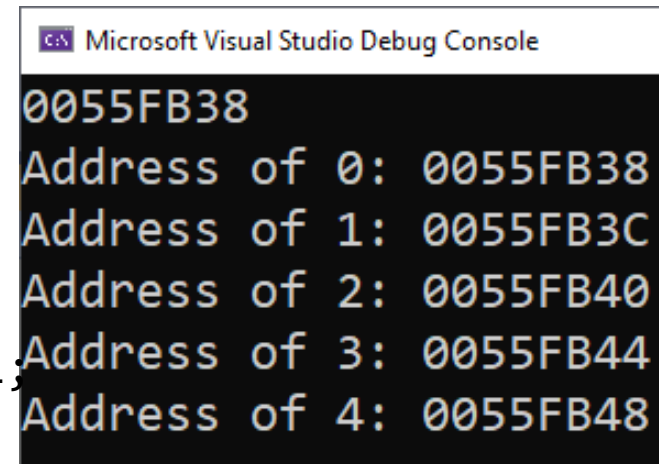
A screenshot of the Microsoft Visual Studio Debug Console showing the output of the program. The text is displayed on a black background with a white border. The output shows the memory addresses of the array elements C[0] through C[4].

Index	Address
0	007FF890
1	007FF894
2	007FF898
3	007FF89C
4	007FF8A0

Example: Base address

- Base address refers to the address of first cell of an array i.e. the cell at index 0 (arr[0]).
- Name of an array acts as a base address. That is, if you print it you get base address displayed

```
#include<iostream>
using namespace std;
int main()
{
    int C[5] = {5,7,9,6,5};
    //base address:
    cout << C << endl;
    for (int i = 0; i < 5; i++)
    {
        cout << "Address of "<<i<<": " << &C[i] << endl;
    }
    return 0;
}
```



Microsoft Visual Studio Debug Console

```
0055FB38
Address of 0: 0055FB38
Address of 1: 0055FB3C
Address of 2: 0055FB40
Address of 3: 0055FB44
Address of 4: 0055FB48
```

Example: Postfix and prefix increment with arrays

- Rules of prefix/postfix are applied accordingly on array's index
- What if an index of array and variable on which increment operator is applied are the same?

Postfix

```
#include<iostream>
using namespace std;
int main()
{
    int C[5] = {5,7,9,6,5};int j = 0;
    cout << "Postfix example" << endl;
    cout << "contents before:" << endl;
    for (int i = 0; i < 5; i++)
        cout << C[i] << " ";
    cout << endl;

    C[j] = j++; //postfix increment, index
    is taken as older j value
    cout << "contents after:" << endl;
    for (int i = 0; i < 5; i++)
        cout << C[i] << " ";
    cout << endl;
    return 0;
}
```

Microsoft Visual Studio Debug Console

```
Postfix example
contents before:
5 7 9 6 5
contents after:
0 7 9 6 5
```

Prefix

```
#include<iostream>
using namespace std;
int main()
{
    int C[5] = {5,7,9,6,5};int j = 0;
    cout << "Prefix example" << endl;
    cout << "contents before:" << endl;
    for (int i = 0; i < 5; i++)
        cout << C[i] << " ";
    cout << endl;

    C[j] = ++j;//prefix increment, index is
    taken as a new updated j value
    cout << "contents after:" << endl;
    for (int i = 0; i < 5; i++)
        cout << C[i] << " ";
    cout << endl;
    return 0;
}
```

Microsoft Visual Studio Debug Console

```
Prefix example
contents before:
5 7 9 6 5
contents after:
5 1 9 6 5
```

Agenda

- Unsized array initialization
- Searching an array
- Array with different data types
- Sorting
- How cin works?
- Char array
- Char array as a string
- Getline function
- Multi dimensional arrays

Unsize array initialization

```
#include<iostream>
using namespace std;
int main()
{
    int C[] = { 5,7,9,6,5 };
    for (int i = 0; i < 5; i++)
    {
        cout << C[i] << " ";
    }

    return 0;
}
```

You can skip mentioning the size, compiler will guess the size by counting the number of elements inside brackets.

Searching an array

```
#include<iostream>
using namespace std;
int main()
{
    int C[5] = { 5,7,9,6,5 };int element = 5;
    for (int i = 0; i < 5; i++)
    {
        if (C[i] == element)
        {
            cout << "Element found at index " << i;
            break;
        }
    }

    return 0;
}
```

Microsoft Visual Studio Debug Console

Element found at index 0

C[i] is taken just as we deal with variable. Thus, we can compare each cell (one by one) with any variable or constant

Array of various data types

- There can be different types of arrays
- Note: you cannot have different data type arrays but with the same name!

```
#include<iostream>
using namespace std;
int main()
{
    double A[] = { 2.4,5.6,7.8 };
    char B[] = { 'a','b','c' };
    bool C[] = { true,false,false };
    int D[] = { 3,5,6 };
    string E[] = { "amna", "usama"};
    return 0;
}
```

General norm is that while initialization, we put the respective cell contents inside curly braces {}

Sorting

- sorting a list of values, such as a list of sales figures that must be sorted from lowest to highest or from highest to lowest, or a list of words that must be sorted into alphabetical order.

	Sales[0]	Sales[1]	Sales[2]	Sales[3]	Sales[4]
Sales	1	2	3	4	5
Index	0	1	2	3	4

- An array is sorted only if
 - $\text{Sales}[0] \leq \text{Sales}[1] \leq \text{Sales}[2] \leq \text{Sales}[3] \leq \text{Sales}[4]$

Check if array is sorted

```
#include<iostream>
using namespace std;
int main()
{
    int D[] = { 1,2,3,4,5 };
    for (int i = 0; i < 4; i++)
    {
        if (D[i] > D[i + 1])
        {
            cout << "The array is
            not sorted" << endl;
            break;
        }
    }
    return 0;
}
```

Compare... **i=0**

1	2	3	4	5
0	1	2	3	4

i=1

1	2	3	4	5
0	1	2	3	4

i=2

1	2	3	4	5
0	1	2	3	4

i=3

1	2	3	4	5
0	1	2	3	4

Swap two numbers

- Before sorting, let's first discuss about swapping.
- Swapping is heavily used in sorting numbers.

```
#include<iostream>
using namespace std;
int main()
{
    int a,b,temp;
    a = 4;
    b = 5;
    cout << "a:" << a << endl;
    cout << "b:" << b << endl;
    temp = a;
    a = b;
    b = temp;
    cout << "a:" << a << endl;
    cout << "b:" << b << endl;
    return 0;
}
```

4	5
a	b

To swap any two numbers, you must need a temporary container. Otherwise:

If we write:

a=b; this will update the contents as:

5	5
a	b

b=a; this will assign value of a (that is 5 now!) to b

How to sort?

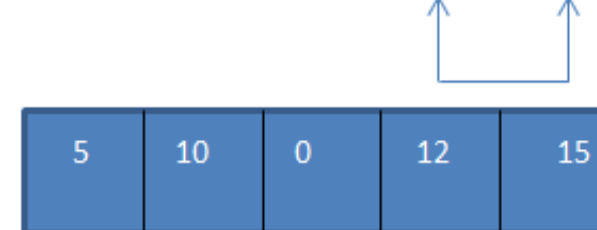
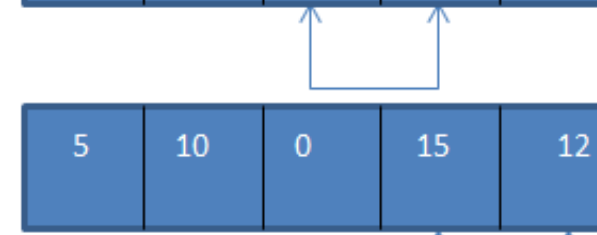
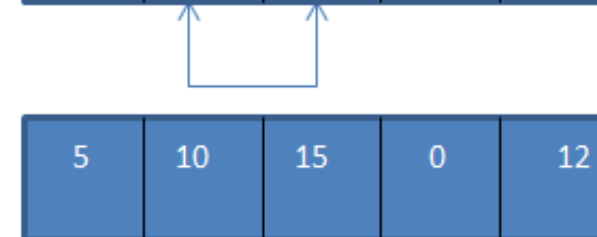
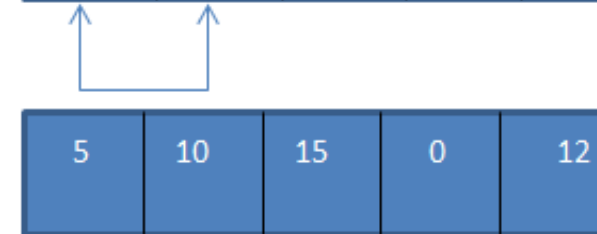
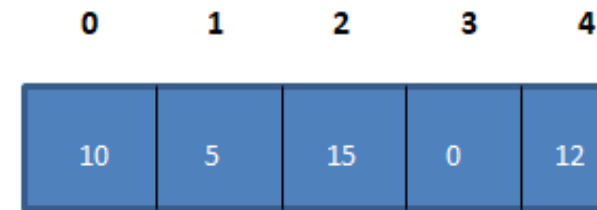
- There are many sorting algorithms, insertion sort, bubble sort, selection sort, etc.
- We will discuss only the simple algorithm (bubble sort) as other algorithms are not in our scope.

Bubble sort

(Optional to study for practice)

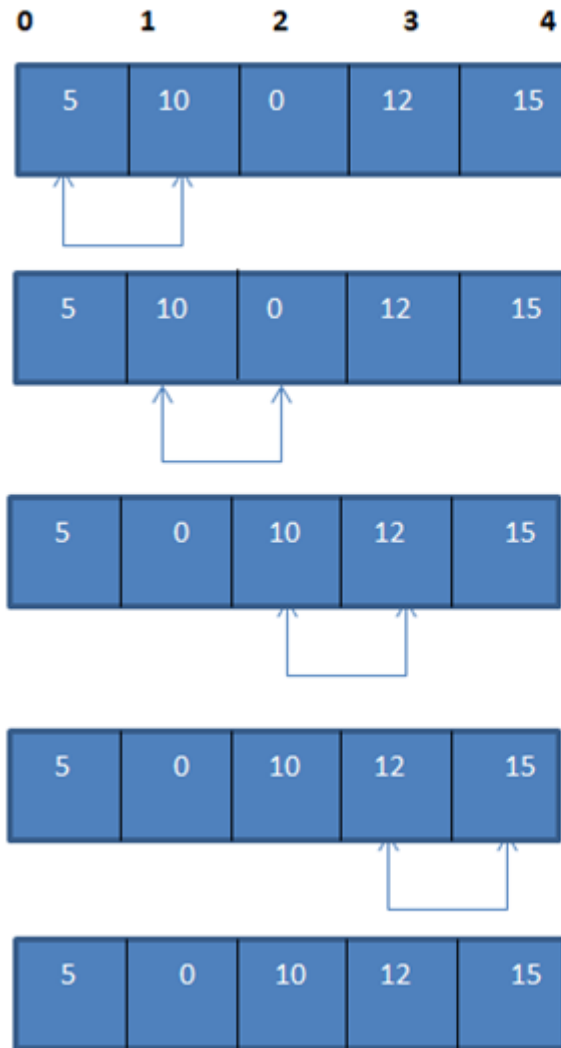
- Works by repeatedly swapping the adjacent elements if they are in wrong order
- Multiple passes to an array
- In each pass, iterate and compare consecutive element pairs..e.g.
 - Index 0 and index 1
 - Index 1 and index 2
 - Index 2 and index 3
 - Index 3 and index 4
- After each pass, if the comparison is done on the basis of $>$ operator the largest element is shifted to the end of an array

Pass 1:



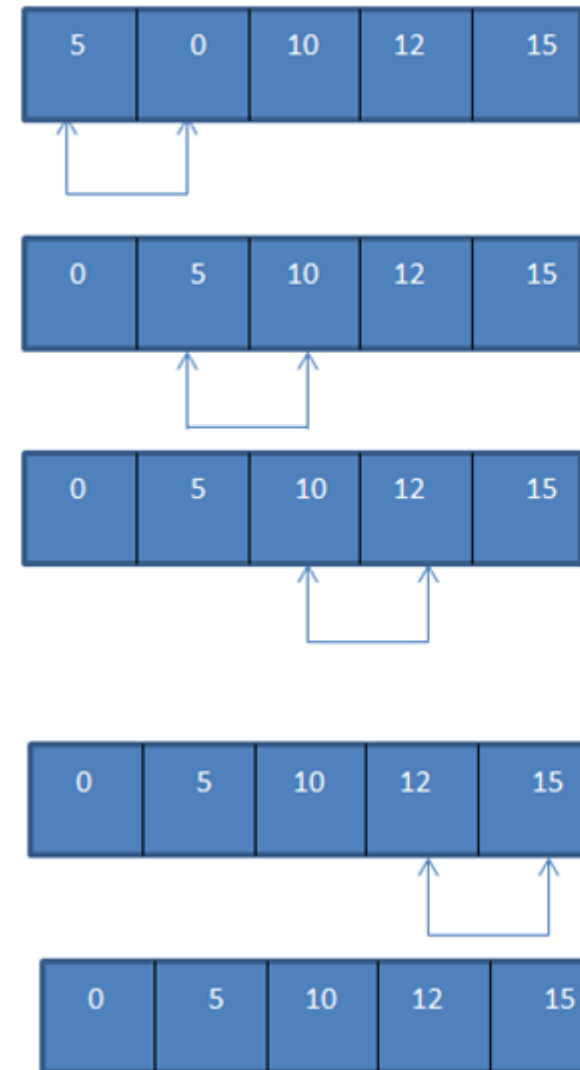
=>the largest element bubbled up

Pass 2:



=> second largest element bubbled

Pass 3:



=> third element bubbled up, list sorted

```
#include<iostream>
using namespace std;
int main()
{
    int n, i, arr[50], j, temp;
    cout << "Enter total number of elements :";
    cin >> n;
    cout << "Enter " << n << " numbers :";
    for (i = 0; i < n; i++)
        cin >> arr[i];
    cout << "Sorting array using bubble sort technique...\n";
    for (i = 0; i < (n - 1); i++)
    {
        for (j = 0; j < (n - i - 1); j++)
        {
            if (arr[j] > arr[j + 1])
            {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    return 0;
}
```

```
cout << "Sorted list in ascending order :\n";
for (i = 0; i < n; i++)
{
    cout << arr[i] << " ";
}
return 0;
}
```

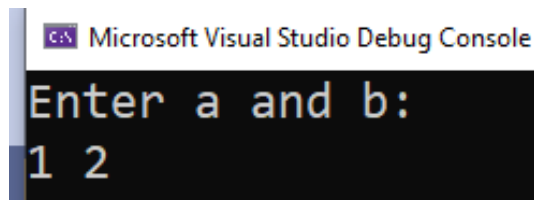
Reading

- Walter savitch, problem solving with C++
 - Pg 421, Sorting an array

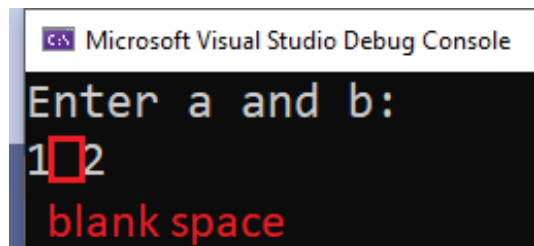
How cin works?

- Cin takes blank space and newline as a terminating characters. That is, when we enter an input, cin will get to know that you finished entering the input only if you either enter a blank space or a newline.

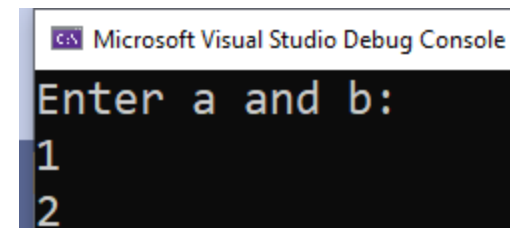
```
#include<iostream>
using namespace std;
int main()
{
    int a, b;
    cout << "Enter a and b:" << endl;
    cin >> a;
    cin >> b;
    return 0;
}
```



```
Microsoft Visual Studio Debug Console
Enter a and b:
1 2
```



```
Microsoft Visual Studio Debug Console
Enter a and b:
1  
blank space
```



```
Microsoft Visual Studio Debug Console
Enter a and b:
1
2
```

User entered 1, now the moment spacebar is pressed, the compiler will assume that you finish giving input for variable a, and thus now prompts for input in variable b

char array

```
#include<iostream>
using namespace std;
int main()
{
    char a1[5];
    for (int i = 0; i < 5; i++)
        cin>>a1[i];

    for (int i = 0; i < 5; i++)
        cout << "index " << i << " " << a1[i] << endl;
    return 0;
}
```

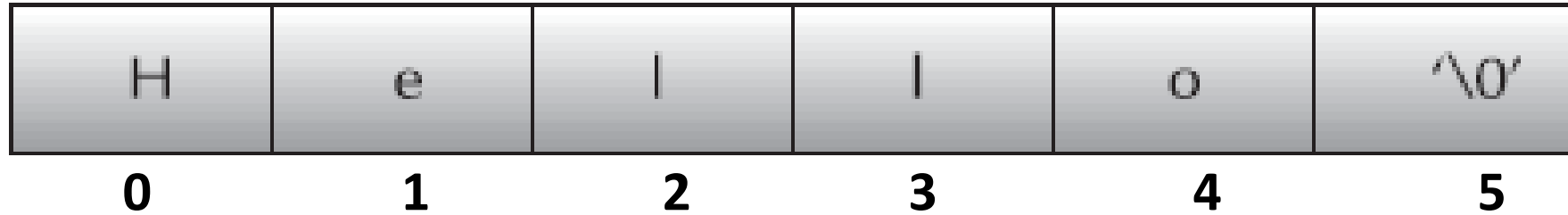
Char array can also act as a string

- In C++, a *string* is defined as a character array that is terminated by a null.
- A null character is specified using `'\0'`.
- Because of the null terminator, it is necessary to declare a character array to be one character longer than the largest string that it will hold.

```
char str[11];
```

- Specifying the size as 11 makes room for the null at the end of the string.
- Note: the string data type that we briefly covered is not the same as this character array. When we write “string name;” we are actually creating an object of string class built-in and defined in standard c++ library.
- We will cover string class in a separate lecture. So, ignore it for now.

Strings or char array



```
char str[6] = "hello";
```

This is the same as writing

```
char str[6] = { 'h', 'e', 'l', 'l', 'o', '\0' };
```

If we use character array as a string and initialize it with a string, then we must consider a space for '\0' character. Otherwise, compiler will generate an error. Like in this example, hello is of 5 characters so you need to declare a char array of 6 size. One extra cell for '\0' character.

```
#include<iostream>
using namespace std;
int main()
{
    char str[5] = "hello";
    return 0;
}
```

a value of type "const char [6]" cannot be used to initialize an entity of type "char [5]"

[Search Online](#)

Alternative ways of defining a string

```
char str[4] = "C++";
```

```
char str[] = {'C','+', '+', '\0'};
```

```
char str[4] = {'C','+', '+', '\0'};
```

For initialization using curly brackets, we cannot put string inside braces, this is not allowed. For example, see the code below.

```
#include<iostream>
using namespace std;
int main()
{
    char a[5] =
    {'h','e','l','l','o'};
    char a[5] = { "hello" };
    return 0;
}
```

```
int main()
{
    char a[5] = {'h','e','l','l','o'};
    char a[5] = { "hello" };
    return 0;
}
```

a value of type "const char [6]" cannot be used to initialize an entity of type "char [5]"

[Search Online](#)


```
#include<iostream>
using namespace std;
int main()
{
    char str[5];
    cin >> str;
    cout << str;
    return 0;
}
```

If we want to use character array as a string, then there is no need to append square brackets with the name of array.

Note: for int, double, bool, and other data types, array name depicts base address. But in case of char array, the array name does not act as a base address.

Like in this example, we have declared an array of 5 characters. Thus, it is mandatory for user to enter only 4 characters (4 plus 1 for '\0'). If the user enters "hello" which is 5 characters long then the compiler will throw exception.

```
#include<iostream>
using namespace std;
int main()
{
    char str[5];
    cin >> str;

    int i = 0;
    while (1)
    {
        if (str[i++] == '\0')
            break;
    }
    return 0;
}
```

We can apply conditional logic on null terminator character to see if we reach the end of our character array or not. This will help in designing a logic of “finding the number of characters in a string”

In this program, an infinite loop is created that keeps on accessing elements of an array one by one using statement `str[i++]`. We want to end the loop, once end of a character array is reached. Therefore we introduced if statement in it.



Giving input to char array *in string form*

```
#include<iostream>
using namespace std;
int main()
{
    char a[6];
    cin >> a;
    for(int i=0;i<7;i++)
        cout << "index "<<i<<" "<<a[i]<<endl;
    return 0;
}
```

```
Microsoft Visual Studio Debug Console
hello
index 0 h
index 1 e
index 2 l
index 3 l
index 4 o
index 5 
index 6 |
```

Giving input to char array *in characters form*

```
#include<iostream>
using namespace std;
int main()
{
    char a[6];
    cin >> a;
    for (int i = 0; i < 7; i++)
        cout << "index " << i << " " << a[i] << endl;
    return 0;
}
```

Once you enter a space after character 'h', cin will assume that you have finished giving input. It thus stores 'h' to array. Any other characters that you write after that will be discarded. You will notice that by default the compiler places blanks and special characters into the empty places

```
Microsoft Visual Studio Debug Console
h e l l o
index 0 h
index 1 
index 2 |
index 3 |
index 4 |
index 5 |
index 6 |
```

What if we want to have a sentence recorded in our char array

Char array takes a space as a terminating character

- “Hello World” is taken as “Hello”... space is taken as terminating character

```
#include<iostream>
using namespace std;
int main()
{
    char str[20];
    cout << "\nEnter another string: ";
    cin >> str;
    cout << "You entered: " << str <<
    endl;
    return 0;
}
```

‘world’ is discarded because cin has taken space as a terminating character.

Microsoft Visual Studio Debug Console

```
Enter another string: hello world
You entered: hello
```



Solution to get a complete sentence as input in character array

- Use getline function
- Getline takes three arguments: 1) char array name, 2) number of characters to take from an input, 3) terminating character. Argument 3 is optional and if we don't specify it then by default '\n' newline character is assumed.
- Now with getline, cin will either terminate when **number of characters** is reached, or when newline (ENTER) character is encountered.

```
#include<iostream>
using namespace std;
int main()
{
    char str[20];
    cout << "\nEnter another string: ";
    cin.getline(str,20);
    cout << "You entered: " << str << endl;
    return 0;
}
```

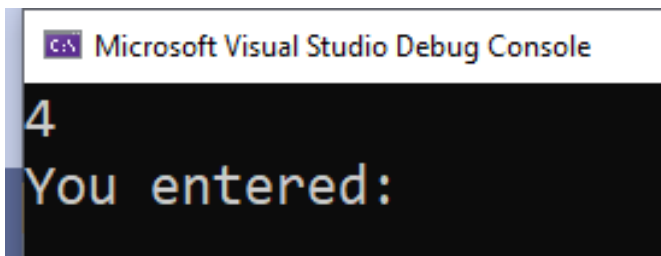
```
Microsoft Visual Studio Debug Console

Enter another string: hello World
You entered: hello World
```

Contd..

Cin integer and then cin char array will cause ambiguous behavior

- After you do `cin>>t;`, the buffer still contains the newline sequence. Then `getline()` reads an immediate newline which tricks it into thinking that the user just pressed enter without typing anything.
- In order to fix this, you need to ignore the newline before calling `getline()`.



The program terminated
without taking input in char
array

```
#include<iostream>
using namespace std;
int main()
{
    int x;
    char str[6];
    cin >> x;
    cin.getline(str, 6, '\n');
    cout << "You entered: " << str << endl;
    return 0;
}
```

Contd..

```
#include<iostream>
using namespace std;
int main()
{
    int x;
    char str[6];
    cin >> x;
    cin.ignore();
    cin.get(str, 6, '\n');
    cout << "You entered: " << str << endl;
    return 0;
}
```

CN Microsoft Visual Studio Debug Console

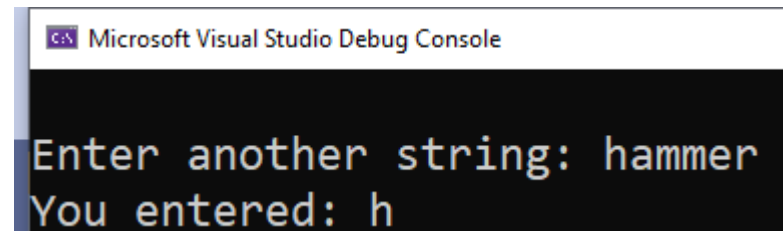
```
4
hello
You entered: hello
```

Contd..

- It also allows a third argument that specifies the terminating character for input. The default value is the newline character.

```
#include<iostream>
using namespace std;
int main()
{
    char str[20];
    cout << "\nEnter another string: ";
    cin.getline(str, 20, 'a');
    cout << "You entered: " << str << endl;
    return 0;
}
```

Since 'a' is taken as a terminating character. Thus, once the user enters a, cin will stop taking further input. Like 'a' inserted after first 'h' in below example.

A screenshot of the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output shows the text "Enter another string: hammer" on the first line and "You entered: h" on the second line, demonstrating that the input was terminated at the character 'a'.

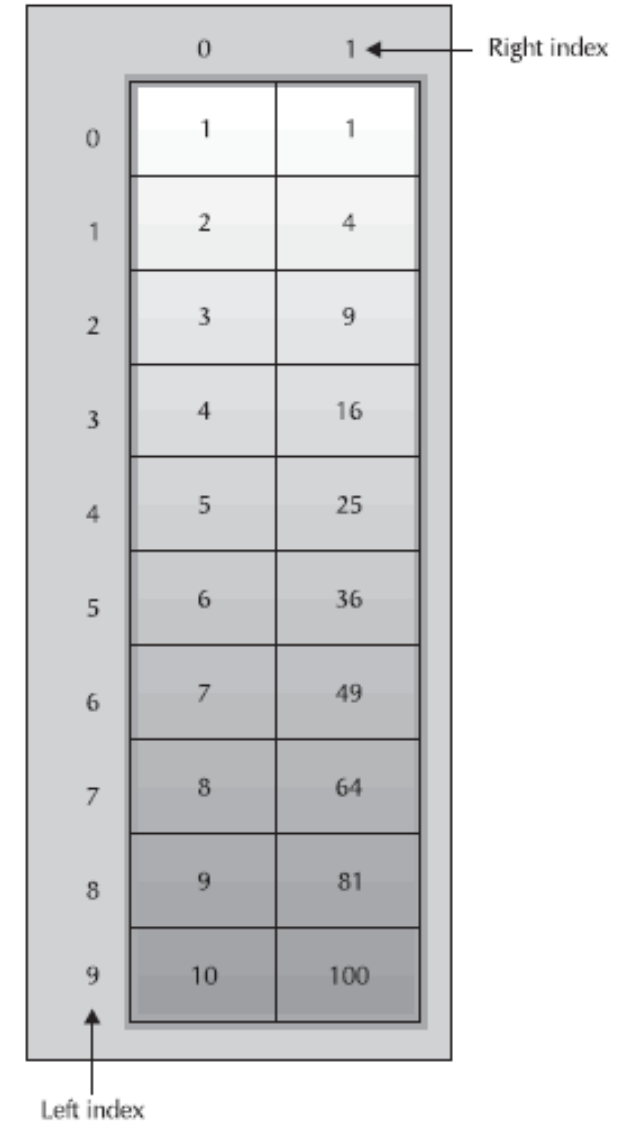
Multi-dimensional array

```
#include<iostream>
using namespace std;
int main()
```

Two Ways to initialize 2D arrays

```
{
    int sqrs1[10][2] = {1, 1, 2, 4, 3, 9, 4, 16, 5, 25, 6, 36, 7, 49, 8, 64, 9, 81, 10, 100};

    int sqrs2[10][2] = {{1, 1},{2, 4},{3, 9},
                        {4, 16},{5, 25},{6, 36},{7, 49},{8, 64},{9, 81},
                        {10, 100}};
    return 0;
}
```



The diagram illustrates a 10x2 2D array. The rows are indexed from 0 to 9 on the left, and the columns are indexed from 0 to 1 on the top. The values in the array are the squares of integers from 1 to 10, arranged in two columns. The first column contains the squares of integers 1 through 10, and the second column contains the squares of integers 2 through 11. The array is enclosed in a light gray border, and the indices are labeled with arrows pointing to the corresponding rows and columns.

	0	1
0	1	1
1	2	4
2	3	9
3	4	16
4	5	25
5	6	36
6	7	49
7	8	64
8	9	81
9	10	100

Example

	0	1
0	1	2
1	3	4

[0][0]	[0][1]
[1][0]	[1][1]

```
#include<iostream>
using namespace std;
int main()
{
    const int rows = 2; const int cols = 2;
    int arr[rows][cols] = {1,2,3,4};
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Reading

- D.S Malik C++ Programming-From Problem Analysis to Program Design, Pg. 486 - Pg. 526, Chapter 9 Array and Strings
- Walter savitch, problem solving with C++
 - Pg 425, multi-dimensional array basics

Recommended reads

- Walter Savitch, Problem Solving with C++ The Object of Programming
 - Chapter 7, page 378, section 7.1