Programming fundamentals

Lecture 12: 2D arrays, char arrays and strings

Recap

- Basic idea of multi-dimensional arrays
- Finding element in an array
- Sorting
- Working of cin

Agenda

- 2D arrays
- Char array
- Strings

Copying one array **A** to another array **B**

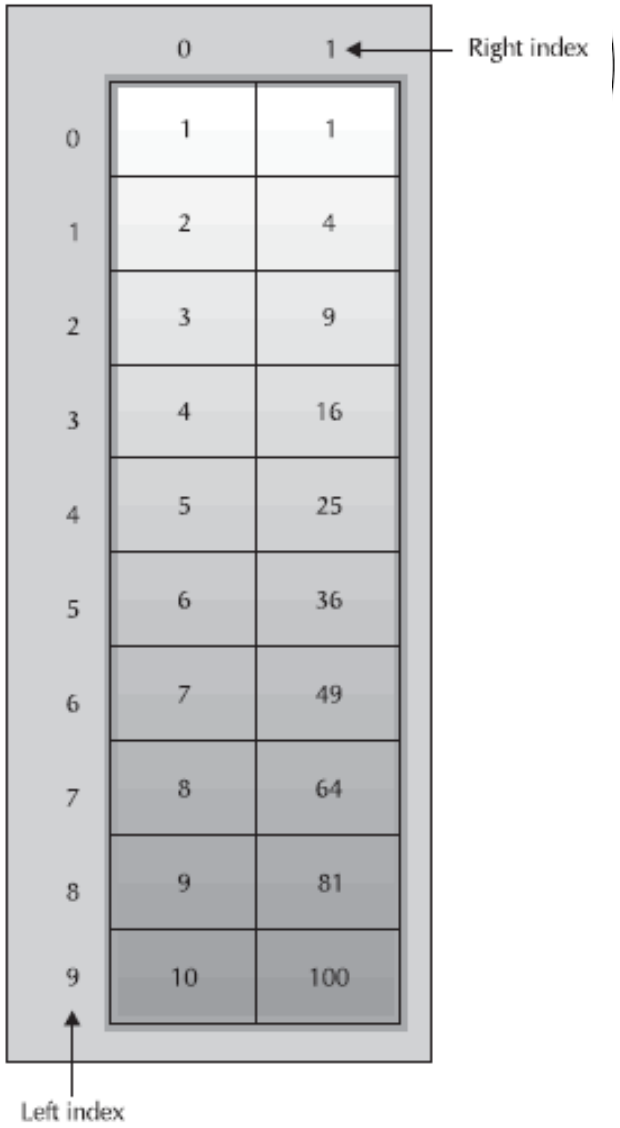
```
#include<iostream>
using namespace std;
int main()
{
    int A[5],B[5];
    for (int i = 0; i < 5; i++)
    {
        cin[ >> A[i];
    }
    for (int i = 0; i < 5; i++)
    {
        B[i]=A[i];
    }
    for (int i = 0; i < 5; i++)
    {
        cout << "A: " << A[i] << " ";
        cout << "B: " << B[i];
        cout << endl;
    }
    return 0;
}
```

2D array

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

Two Ways to initialize 2D arrays



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;
}
```

	0	1	2	3
0	**1**	**2**	**3**	**4**
1				
2				

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

String vs char array

- A character array is simply an **array of characters** can terminated by a null character.
- A string is a **class which defines objects** that be represented as stream of characters.
- Size of the character array has to **allocated statically**, more memory cannot be allocated at run time if required. Unused allocated **memory is wasted** in case of character array.
- In case of strings, memory is **allocated dynamically**. More memory can be allocated at run time on demand. As no memory is preallocated, **no memory is wasted**.
- Implementation of **character array is faster** than std:: string. **Strings are slower** when compared to implementation than character array.
- Character array **do not offer** much **inbuilt functions** to manipulate strings. String class defines **a number of functionalities** which allow manifold operations on strings.

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

H	e	l	l	o	'\0'
0	**1**	**2**	**3**	**4**	**5**

```
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 '\o'). 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 might 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().

```
#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;
}
```

Microsoft Visual Studio Debug Console

```
4
You entered:
```

The program terminated without taking input in char array

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;
}
```

Microsoft Visual Studio Debug Console

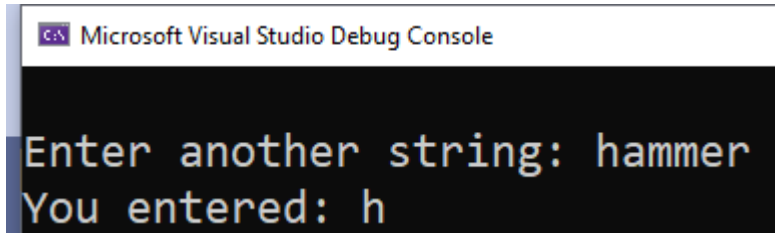
```
4
hello
You entered: hello
```

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.



Practice problems

character array

Changing a character from upper case to lower case

ASCII value of A is 65 and ASCII value of a is 97

65+32 = 97 = a = equivalent value of A in lowercase i.e., a

Decimal	Character	Decimal	Character
65	A	97	a
66	B	98	b
67	C	99	c
68	D	100	d
69	E	101	e
70	F	102	f
71	G	103	g
72	H	104	h
73	I	105	i
74	J	106	j

```
#include<iostream>
using namespace std;
int main()
{
    char ch;
    cout << "Enter a character in uppercase :
    ";
    cin >> ch;
    ch = ch + 32;
    cout << "Character in lowercase = " << ch;
    return 0;
}
```

Finding length of a string

```
#include<iostream>
using namespace std;
int main()
{
    char str[100]; /* Declares a string of size 100 */
    int l = 0;

    cin >> str;
    while (str[l] != '\0')
    {
    l++;
    }
    cout << "Length of the string is " << l;
    return 0;
}
```

Microsoft Visual Studio Debug Console

```
Usama
Length of the string is 5
```

Reverse a string

Length: 5

U	S	A	M	A	\0
0	1	2	3	4	5

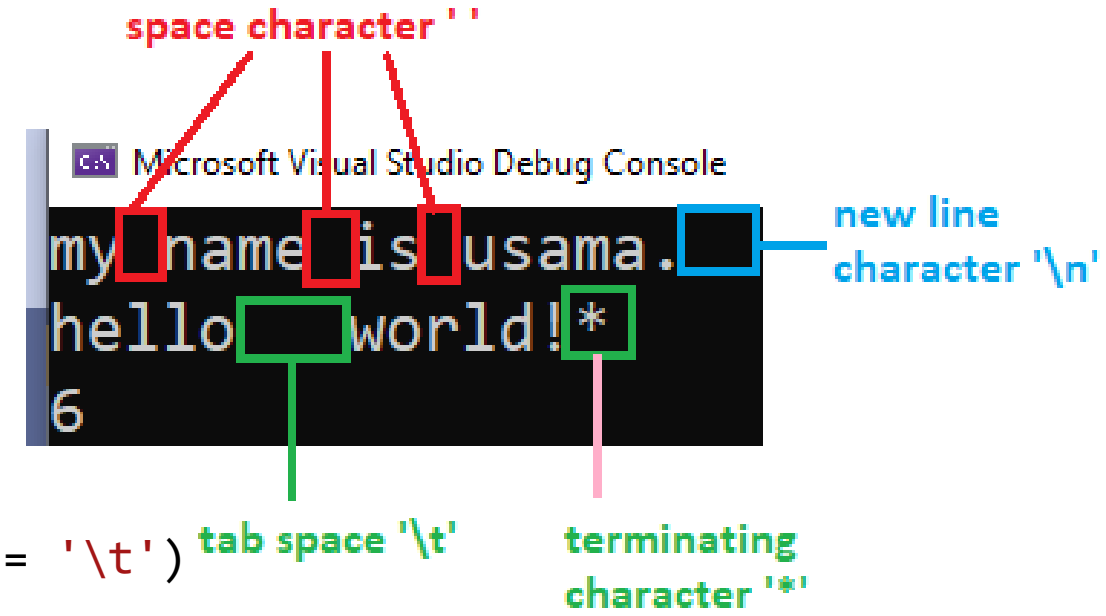
```
#include<iostream>
using namespace std;
int main()
{
    char str[100];
    int l = 0;
    cin >> str;
    while (str[l] != '\0')
    {
        l++;
    }
    for (int i = l-1; i >= 0; i--)
    {
        cout<<str[i];
    }
    return 0;
}
```

Microsoft Visual Studio Debug Console

```
usama
amasu
```

Count the total number of words in a string.

```
#include<iostream>
using namespace std;
int main()
{
    char str[50];
    int i, wrd;
    cin.getline(str,50,'*');//considering * as a
    terminating character
    i = 0;
    wrd = 1;
    while (str[i] != '\0')
    {
    if (str[i] == ' ' || str[i] == '\n' || str[i] == '\t')
    wrd++;
    i++;
    }
    cout<< wrd;
    return 0;
}
```




```
my name is usama.
hello   world!*
6
```

Compare two strings with respect to length

```
#include<iostream>
using namespace std;
int main()
{
char str1[10], str2[10]; /* Declares a string of size
100 */
int l1 = 0, l2 = 0; int i;
cin >> str1;
cin >> str2;
i = 0;
while (str1[i] != '\0')
{
        l1++;i++;
}
cout << "Length of first string is: " << l1<<endl;
i = 0;
while (str2[i] != '\0')
{
        l2++; i++;
}
cout << "Length of second string is: " << l2<<endl;
```

Microsoft Visual Studio Debug Console

```
natalia
zara
Length of first string is: 7
Length of second string is: 4
First string is larger in length than the second string
```

```
if (l1 == l2)
        cout << "Both strings are equal";
else if (l1 > l2)
        cout << "First string is larger in length
        than the second string";
else
        cout << "First string is smaller in length
        than the second string";

return 0;
}
```

Extract a substring

Microsoft Visual Studio Debug Console

```
natalia
Input the position to start extraction :
4
alia
```

```
#include<iostream>
using namespace std;
int main()
{
    char str[10], sstr[10];
    int pos, c = 0; int i; int l = 0;
    cin >> str;
    i = 0;
    while (str[i] != '\0')
    {
        l++; i++;
    }
    cout<<"Input the position to start extraction :"<<endl;
    cin>>pos;
    while (c < l)
    {
        sstr[c] = str[pos- 1+c];
        c++;
    }
    sstr[c] = '\0';
    cout << sstr;
    return 0;
}
```

Some more problems..

- **Copy one string into another string**
- **Count total number of vowels**
- **Extract a substring with starting position as *x* and ending position as *y***
- **Change the letters in input string from upper case to lower case**

Practice problems

String

String Concatenation

+ operator helps in appending a character or a string

```
#include <string>
#include<iostream>
using namespace std;
int main()
{
   string firstName = "Natalia ";
   string lastName = "Chaudhry";
   string fullName = firstName + lastName;
   cout << fullName;
return 0;
}
```

String Concatenation

Use append() function

```
#include <string>
#include<iostream>
using namespace std;
int main()
{
   string firstName = "Natalia ";
   string lastName = "Chaudhry";
   string fullName = firstName.append(lastName);
   cout << fullName;
   return 0;
}
```

+ operator acts according to the situation

- + is used for both addition and concatenation

z will be 30

```
#include <string>
#include<iostream>
using namespace std;
int main()
{
    int x = 10;
    int y = 20;
    int z = x + y;
    cout << z;
    return 0;
}
```

z will be '1020'

```
#include <string>
#include<iostream>
using namespace std;
int main()
{
    string x = 10;
    string y = 20;
    string z = x + y;
    cout << z;
    return 0;
}
```

Find string length: use `length()` function

```
#include <string>
#include<iostream>
using namespace std;
int main()
{
   string txt = "Natalia";
   cout << "The length string is: " << txt.length();
   return 0;
}
```

Microsoft Visual Studio Debug Console

```
The length string is: 7
```

Access Strings

- You can access the characters in a string by referring to its index number inside square brackets []

```
string myString = "Hello";
cout << myString[0];
// Outputs H
myString[0] = ’h';
```

Input in strings

- cin considers a space (whitespace, tabs, etc) as a terminating character, which means that it can only display a single word (even if you type many words)
- Cin.getline() function only works for char array as its first argument
- Use another version of getline():
  - Getline(first argument, second argument);
  - First argument is **cin**
  - Second argument is **string** variable name

Input in strings

```
#include <string>
#include<iostream>
using namespace std;
int main()
{
   string fullName;
   cout << "Type your full name: ";
   getline(cin, fullName);
   cout << "Your name is: " << fullName;
      return 0;
}
```

Microsoft Visual Studio Debug Console

```
Type your full name: Natalia Chaudhry
Your name is: Natalia Chaudhry
```

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