DHA Suffa University
Department of Computer Science
CS 2001L – Data Structures and Algorithms Lab
Spring 2021

Lab 04 – Array and String Operations

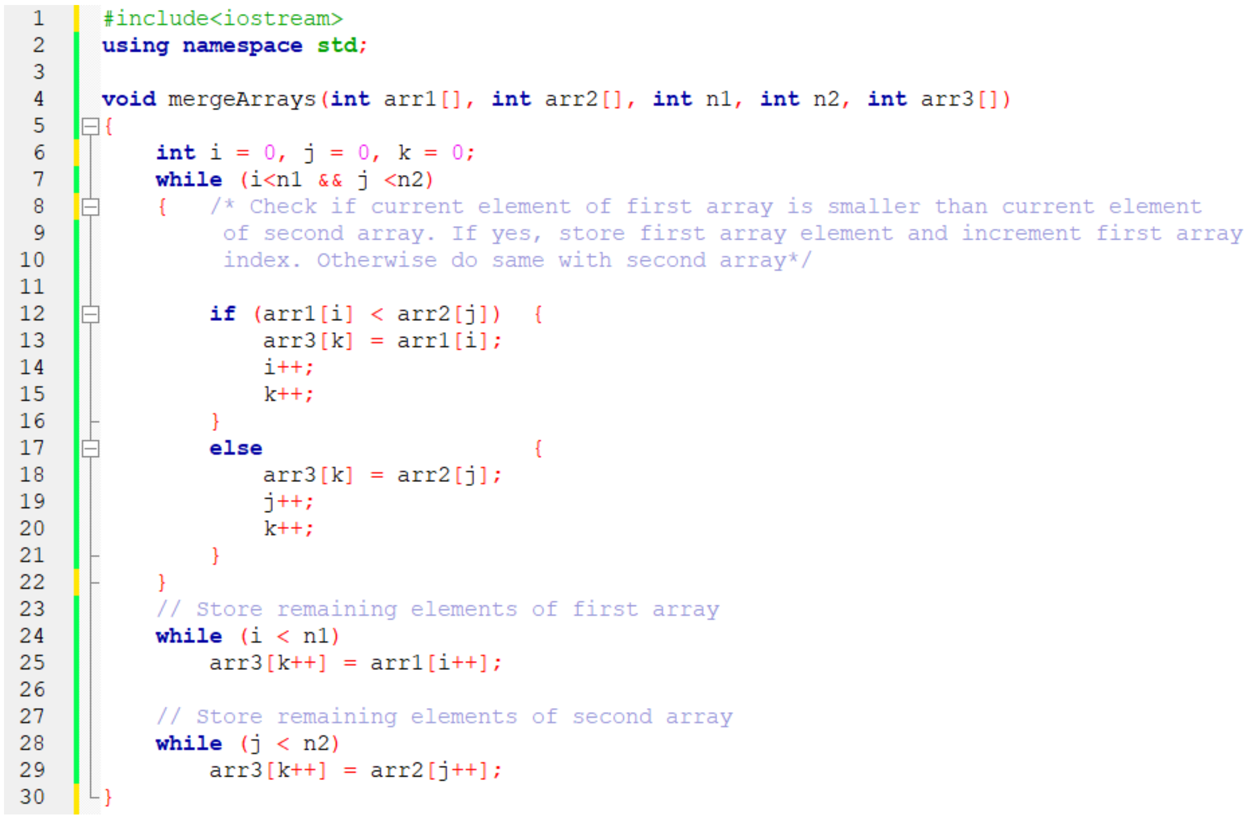
Objective:

To learn about

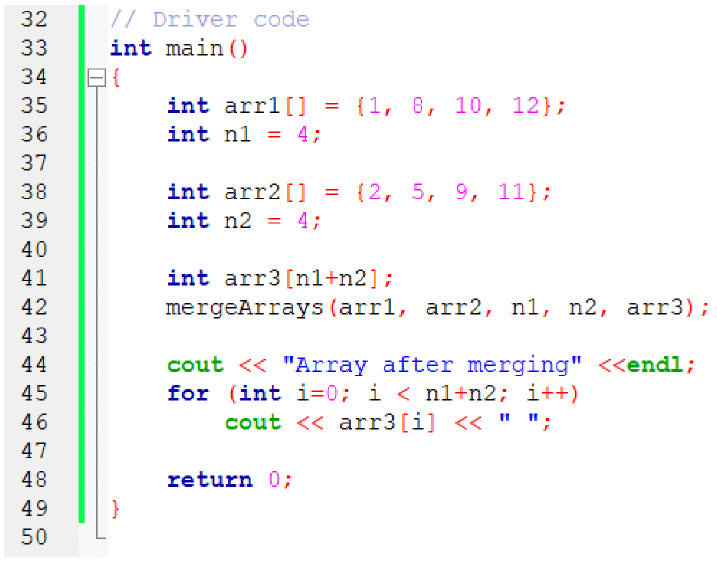
- Merging arrays
- Mapping 2D array to 1D array
- Length
- Concatenation
- Substring
- Pattern Matching

Merging arrays

Merging two sorted arrays



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

void mergeArrays(int arr1[], int arr2[], int n1, int n2, int arr3[])
{
    int i = 0, j = 0, k = 0;
    while (i<n1 && j <n2)
    {   /* Check if current element of first array is smaller than current element
         of second array. If yes, store first array element and increment first array
         index. Otherwise do same with second array*/

        if (arr1[i] < arr2[j])  {
            arr3[k] = arr1[i];
            i++;
            k++;
        }
        else                    {
            arr3[k] = arr2[j];
            j++;
            k++;
        }
    }
    // Store remaining elements of first array
    while (i < n1)
        arr3[k++] = arr1[i++];

    // Store remaining elements of second array
    while (j < n2)
        arr3[k++] = arr2[j++];
}
```



```cpp
// Driver code
int main()
{
    int arr1[] = {1, 8, 10, 12};
    int n1 = 4;

    int arr2[] = {2, 5, 9, 11};
    int n2 = 4;

    int arr3[n1+n2];
    mergeArrays(arr1, arr2, n1, n2, arr3);

    cout << "Array after merging" <<endl;
    for (int i=0; i < n1+n2; i++)
        cout << arr3[i] << " ";

    return 0;
}
```

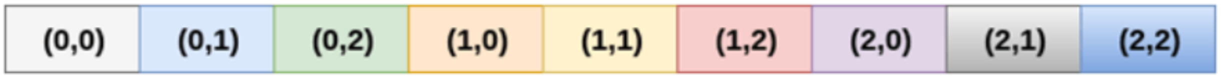
Mapping 2D array to 1D array

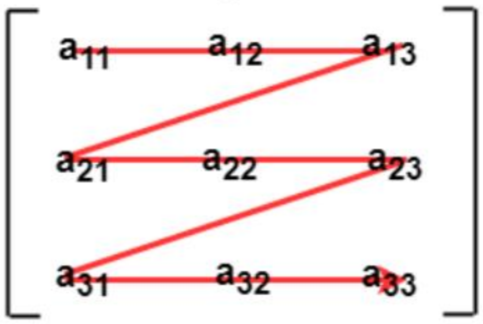
A two dimensional array needs to be mapped to a one dimensional array in order to store it into the memory. The size of the one dimensional array is equal to the multiplication of number of rows and the number of columns present in the two dimensional array.

There are two main techniques of storing 2D array elements into memory

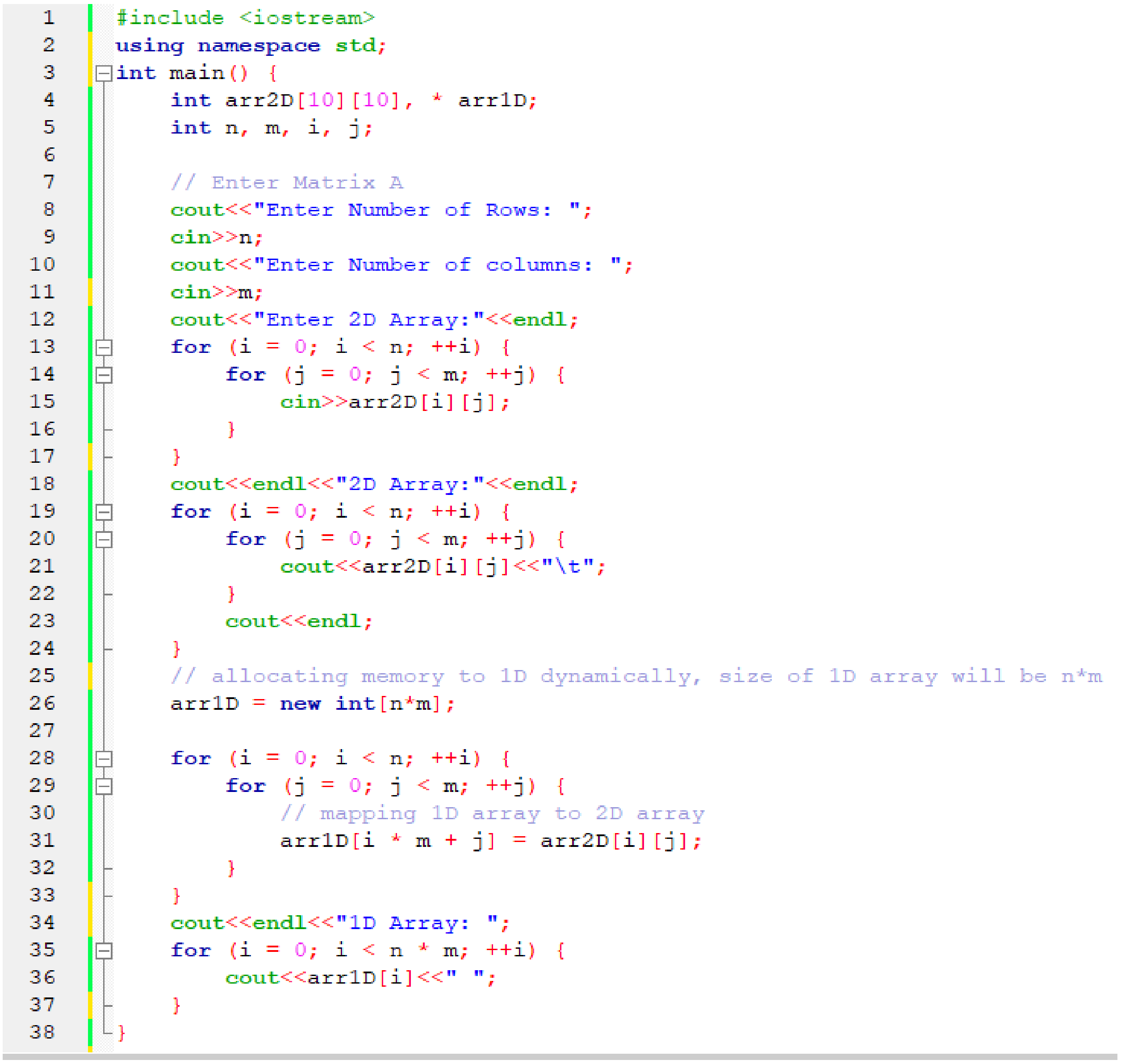
Row Major ordering

In row major ordering, all the rows of the 2D array are stored into the memory contiguously.





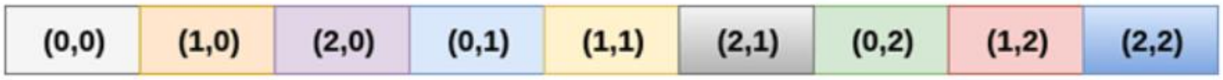
Implementation:

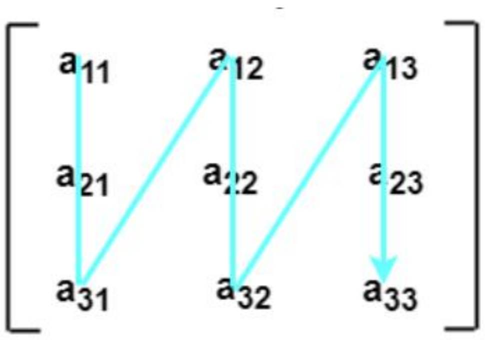


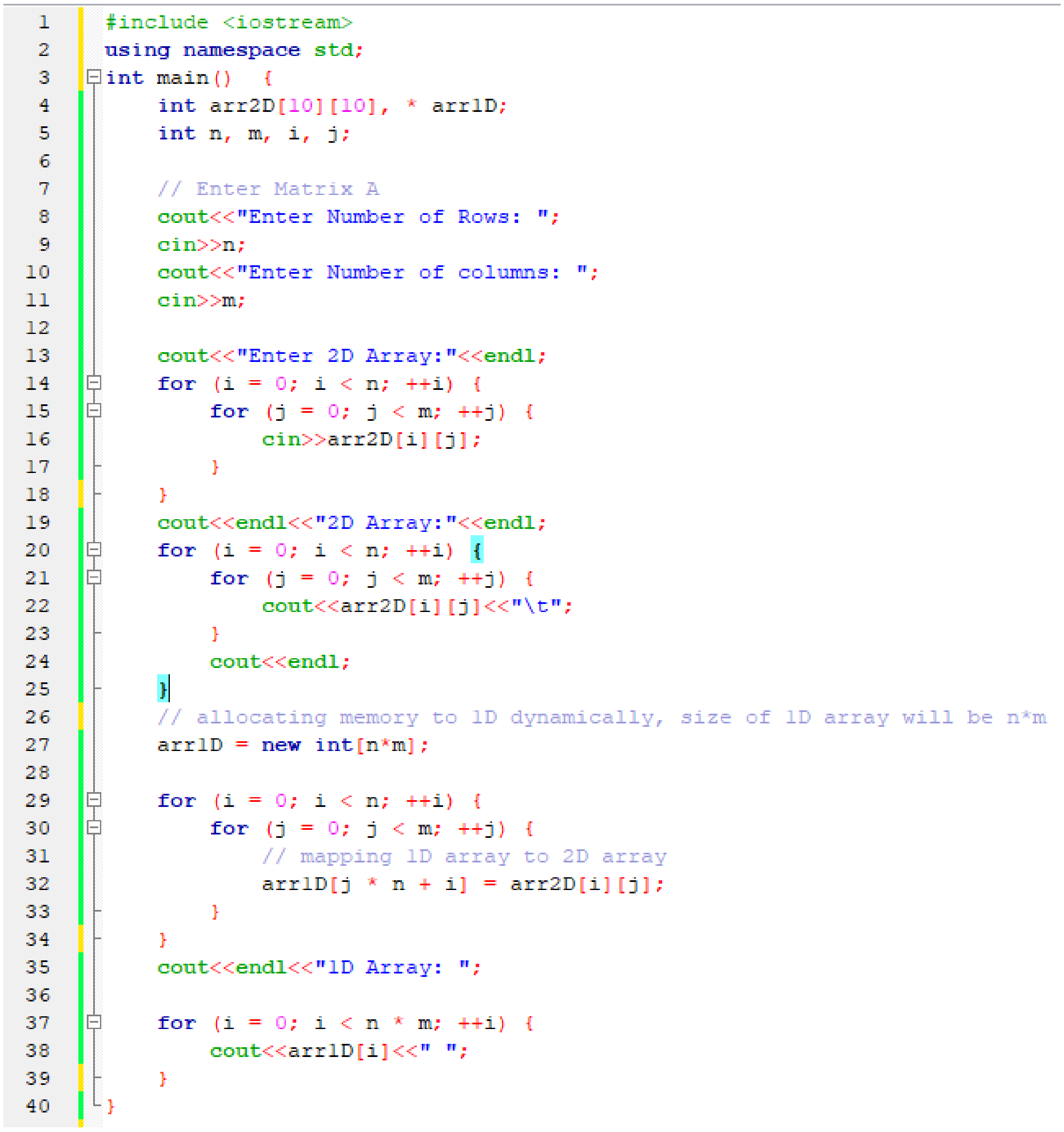
```
#include <iostream>
using namespace std;
int main() {
    int arr2D[10][10], * arr1D;
    int n, m, i, j;

    // Enter Matrix A
    cout<<"Enter Number of Rows: ";
    cin>>n;
    cout<<"Enter Number of columns: ";
    cin>>m;
    cout<<"Enter 2D Array:"<<endl;
    for (i = 0; i < n; ++i) {
        for (j = 0; j < m; ++j) {
            cin>>arr2D[i][j];
        }
    }
    cout<<endl<<"2D Array:"<<endl;
    for (i = 0; i < n; ++i) {
        for (j = 0; j < m; ++j) {
            cout<<arr2D[i][j]<<"\t";
        }
        cout<<endl;
    }
    // allocating memory to 1D dynamically, size of 1D array will be n*m
    arr1D = new int[n*m];

    for (i = 0; i < n; ++i) {
        for (j = 0; j < m; ++j) {
            // mapping 1D array to 2D array
            arr1D[i * m + j] = arr2D[i][j];
        }
    }
    cout<<endl<<"1D Array: ";
    for (i = 0; i < n * m; ++i) {
        cout<<arr1D[i]<<" ";
    }
}
```

Column Major ordering

In column major ordering, all the columns of the 2D array are stored into the memory contiguously.



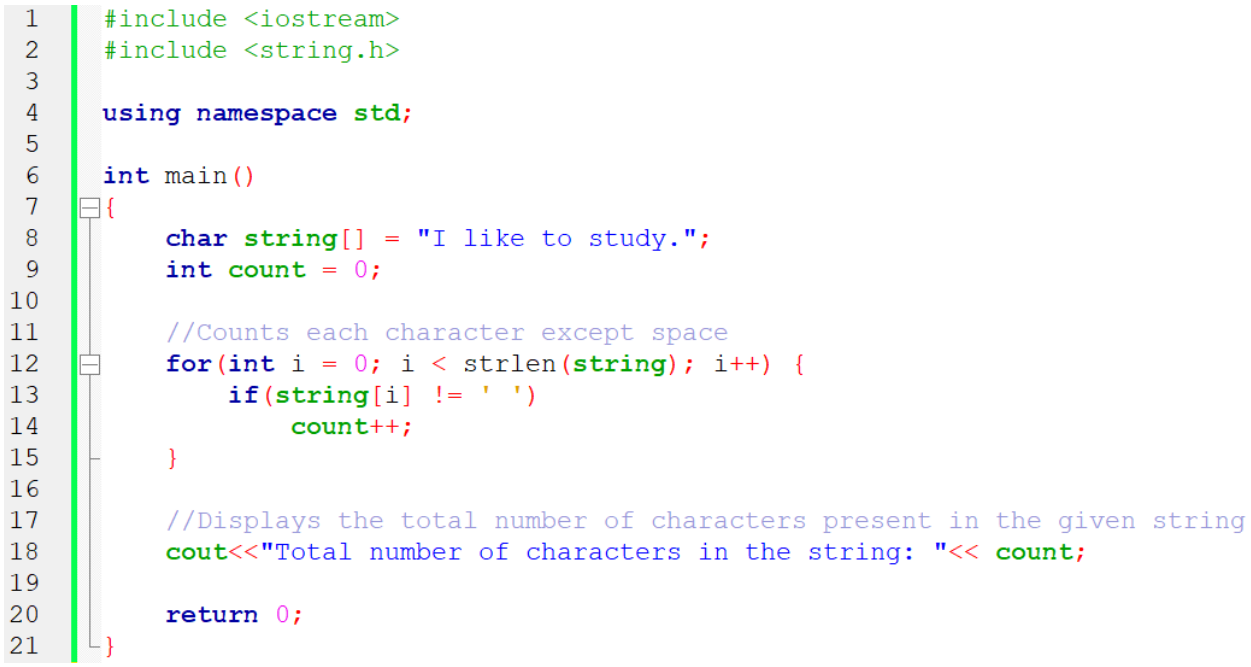




```cpp
#include <iostream>
using namespace std;
int main()  {
    int arr2D[10][10], * arr1D;
    int n, m, i, j;

    // Enter Matrix A
    cout<<"Enter Number of Rows: ";
    cin>>n;
    cout<<"Enter Number of columns: ";
    cin>>m;

    cout<<"Enter 2D Array:"<<endl;
    for (i = 0; i < n; ++i) {
        for (j = 0; j < m; ++j) {
            cin>>arr2D[i][j];
        }
    }
    cout<<endl<<"2D Array:"<<endl;
    for (i = 0; i < n; ++i) {
        for (j = 0; j < m; ++j) {
            cout<<arr2D[i][j]<<"\t";
        }
        cout<<endl;
    }
    // allocating memory to 1D dynamically, size of 1D array will be n*m
    arr1D = new int[n*m];

    for (i = 0; i < n; ++i) {
        for (j = 0; j < m; ++j) {
            // mapping 1D array to 2D array
            arr1D[j * n + i] = arr2D[i][j];
        }
    }
    cout<<endl<<"1D Array: ";

    for (i = 0; i < n * m; ++i) {
        cout<<arr1D[i]<<" ";
    }
}
```

Length of String

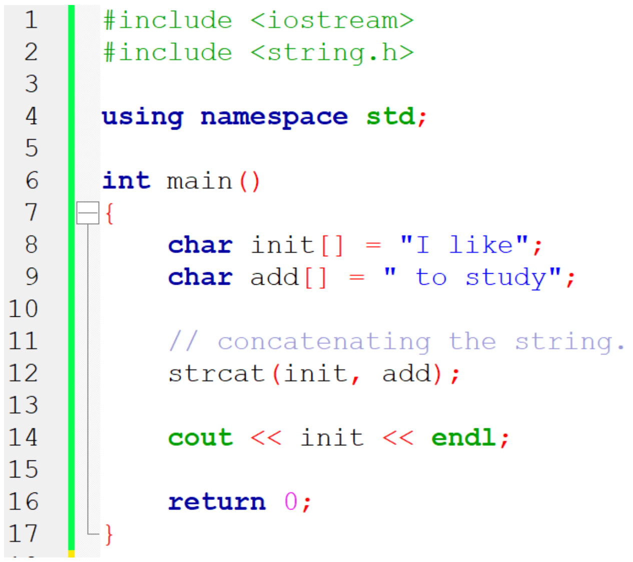
This operation is used to count number of characters in the string. In order to count number of characters.



```cpp
#include <iostream>
#include <string.h>

using namespace std;

int main()
{
    char string[] = "I like to study.";
    int count = 0;

    //Counts each character except space
    for(int i = 0; i < strlen(string); i++) {
        if(string[i] != ' ')
            count++;
    }

    //Displays the total number of characters present in the given string
    cout<<"Total number of characters in the string: "<< count;

    return 0;
}
```

Concatenation of two Strings

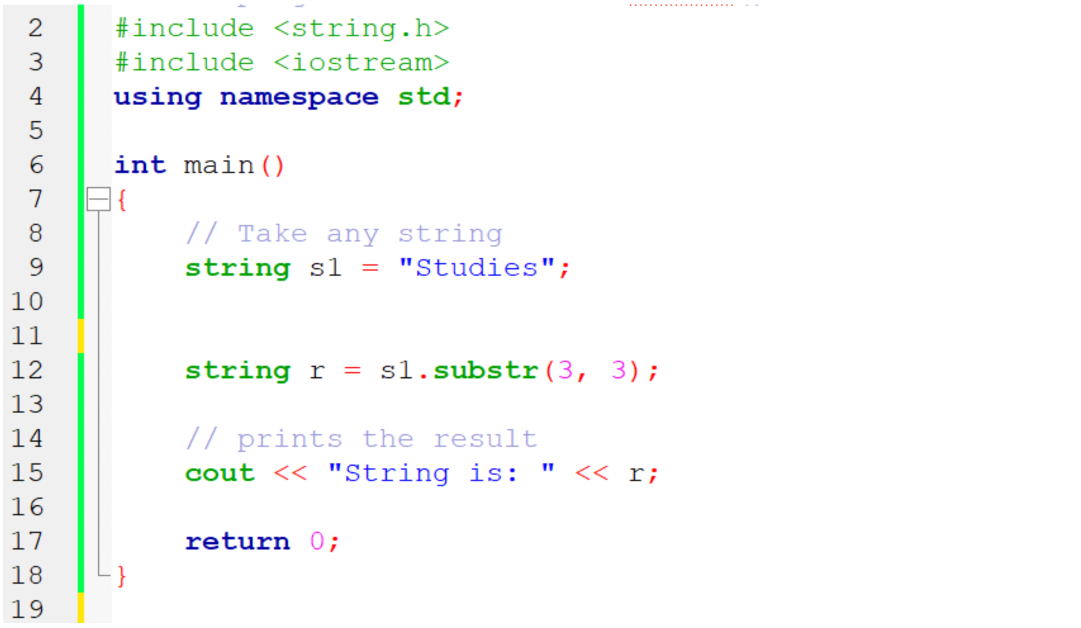
This operation is used to concatenate (combine) two strings to form a new string.



```cpp
#include <iostream>
#include <string.h>

using namespace std;

int main()
{
    char init[] = "I like";
    char add[] = " to study";

    // concatenating the string.
    strcat(init, add);

    cout << init << endl;

    return 0;
}
```

Substring:

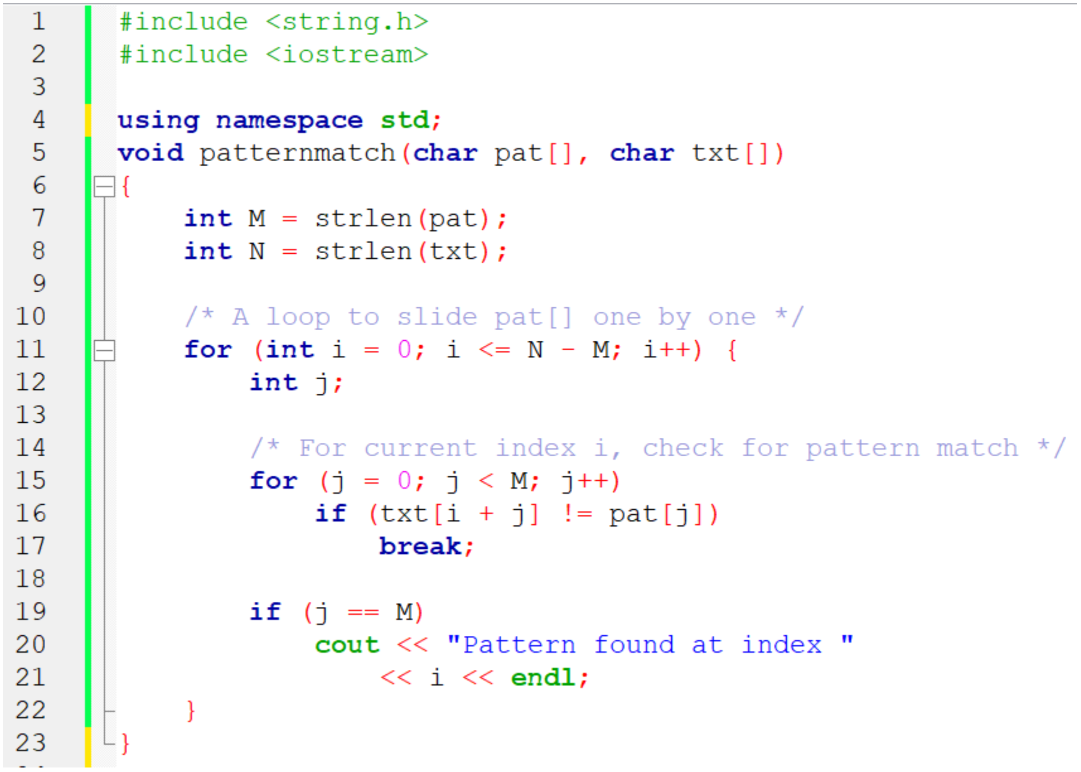
Extracting a sequence of Characters from the given string by providing the start index and length of substring. Substring of the given length is returned from the function.



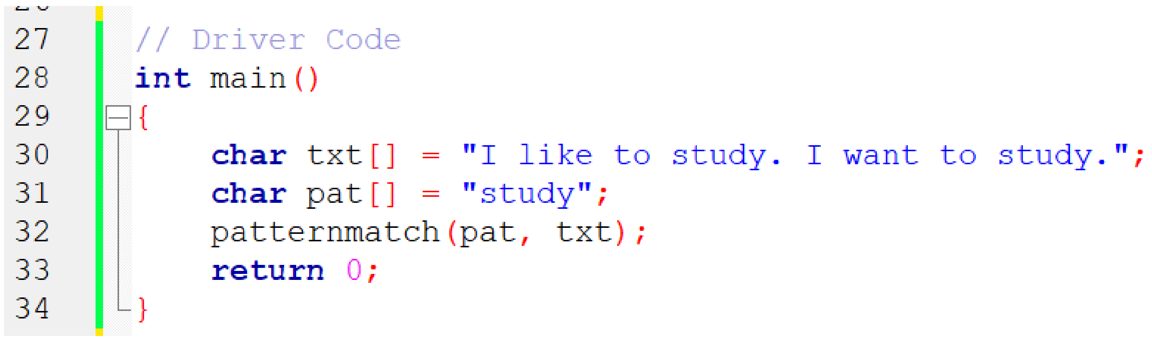
```
#include <string.h>
#include <iostream>
using namespace std;

int main()
{
    // Take any string
    string s1 = "Studies";


    string r = s1.substr(3, 3);

    // prints the result
    cout << "String is: " << r;

    return 0;
}

```

Pattern Matching:

Pattern matching compares the pattern to the text, one character at a time, until unmatching characters are found. The algorithm can be designed to stop on either the first occurrence of the pattern, or upon reaching the end of the text.



```
#include <string.h>
#include <iostream>

using namespace std;
void patternmatch(char pat[], char txt[])
{
    int M = strlen(pat);
    int N = strlen(txt);

    /* A loop to slide pat[] one by one */
    for (int i = 0; i <= N - M; i++) {
        int j;

        /* For current index i, check for pattern match */
        for (j = 0; j < M; j++)
            if (txt[i + j] != pat[j])
                break;

        if (j == M)
            cout << "Pattern found at index "
                 << i << endl;
    }
}
```



```
// Driver Code
int main()
{
	char txt[] = "I like to study. I want to study.";
	char pat[] = "study";
	patternmatch(pat, txt);
	return 0;
}
```

LAB ASSIGNMENT

1. Create a program to merge two unsorted arrays.

SUBMISSION GUIDELINES

- Take a screenshot of each task (code and its output), labeled properly.
- Place all the screenshots and the code files (properly labeled) in a single folder labeled with Roll No and Lab No. e.g. **‘cs191xxx_Lab01’**.
- Submit the folder at LMS
- **-100%** policies for plagiarism.