Strings in C++

(CS 1002)

Dr. Mudassar Aslam

Cybersecurity Department,

National University of Computer & Emerging Sciences, Islamabad Campus

C- Strings

- In C programming, the collection of characters is stored in the form of arrays. This is also supported in C++ programming. Hence it's called C-strings.

- C-strings are arrays of type char terminated with null character, that is, ‘\0’ (ASCII value of null character is 0).

How to define a C-string?

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

- In the above code, str is a string and it holds 4 characters.
- Although, "C++" has 3 character, the null character \0 is added to the end of the string automatically.
- Need to insert <cstring> or <string.h> library to access the functions which we can use on char arrays.

How to define a C-string?

- In C, a string can be a specially terminated char array or char pointer
  - a char array, such as **char** str[ ]="high";
  - a char pointer, such as **char** *p = "high";
- If a char array, the last element of the array must be equal to **'\0',** signaling the end
- For example, the above str[] is really of length 5:
  str[0]='h' str[1]='i' str[2]='g' str[3]='h' str[4]='\0'
- The same array could've been declared as:
  - **char** str[5] = {'h','i', 'g','h','\0'};
- In **char** *p="high"; the system allocates memory of 5 characters long, stores "high" in the first 4, and '\0' in the 5th.

Passing a char array to a Function

- We **need to tell the compiler** what the **type of the array,** and give it a **variable name** (a **reference**)

```
char  a[  ]
```

- We **don't want to specify the size** so function can work with **different sized arrays**
- **Size** may be provided as **second parameter**
- **Arrays** are **automatically passed by reference**.
- **Do not use &** symbol

Example

```
void Display(char data[], int N) {

        cout << "Array contains" << endl;
        for (int k = 0; k < N; k++)
                cout << data[k] << " ";
        cout << endl;
}


int main()
{
        char arr[6] = "Hello";
        Display(arr, 6);
        return 0;
}
```

But we don’t need to send size of string to the function like we need for the arrays of other data types. Because, we know that strings are terminated by ‘\0’. So we can run loop on this string (or char array) until ‘\0’ is reached.

Example - Improved

```
void Display(char data[]) {

          cout << "Array contains" << endl;
          int k=0;
          while (data[k] != ‘\0’)
          {
               cout << data[k] << " ";
               k++;
          }
          cout << endl;
}

int main()
{
            char arr[6] = "Hello";
            Display(arr, 6);
            return 0;
}
```

Functions on Char Array

- The following slides will introduce library functions which can be used on char arrays (c-strings)
- You can use these functions in your programs
- However, you must be able to make all these functions yourself
- Example: String Length function shown in the next slides

Functions on Char Array

- **Read a line of text:**

cin.get(variable, size)

To read the text containing blank space, cin.get function can be used. This function takes two arguments.

First argument is the name of the string (address of first element of string) and second argument is the maximum size of the array.

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

int main()
{
   char str[100];
   cout << "Enter a string: "; //Programming is fun
   cin.get(str, 100);

   cout << "You entered: " << str << endl; //Programming is fun
   return 0;
}
```

Finding length of C-string (library)

strlen(char_array)

This function will return the total elements in an array (excluding \0 character)

Example:

```
#include <iostream>
using namespace std;
int main()
{
   char A[] = "ABCD";
   cout << strlen(A) << endl;   \\4
   return 0;
}
```

Finding length of C-string (library)

strlen(char_array)

This function will return the total elements in an array (excluding \0 character)

Example:

```
#include <iostream>
using namespace std;
int main()
{
   char A[] = "ABCD";
   cout << strlen(A) << endl;   \\4
   return 0;
}
```

Finding length of C-string (self made)

my_strlen(char_array)

This function will return the total elements in an array (excluding \0 character)

Example:

```
int my_strlen(char data[]) {

          int k=0;
          while (data[k] != '\0')
          {
                 k++;
          }
          return k;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    char A[] = "ABCD";
    cout << my_strlen(A) << endl;   \\4
    return 0;
}
```

Functions on Char Array

- **Copy C-string:**

strcpy (dest, source)

Copies the C string pointed by source into the array pointed by destination, including the terminating null character (and stopping at that point).

Example:

```
int main ()
{
  char str1[]="Sample string";
  char str2[40];
  char str3[40];
  strcpy (str2,str1);   // str2 = Sample string
  strcpy (str3,"copy successful");    // str3 = copy successful
   cout<< str1<< ‘  ‘ <<str2 << ‘  ‘ <<str3;
  return 0;
}
```

Functions on Char Array

- **Copy n elements of C-string:**
  strncpy (dest, source, n)

This function will copy source C-string to destination C-string till n.

Example:

```
int main ()
{
  char str1[]="Sample string";
  char str2[40];
  char str3[40];
  strcpy (str2,str1);   // str2 = Sample string
  strcpy (str3,"copy successful");    // str3 = copy successful
  strncpy (str1, str3, 5);
  cout << str1 << ' ';    // copy e string
  return 0;}
```

Functions on Char Array

- **Comparing two C-strings:**

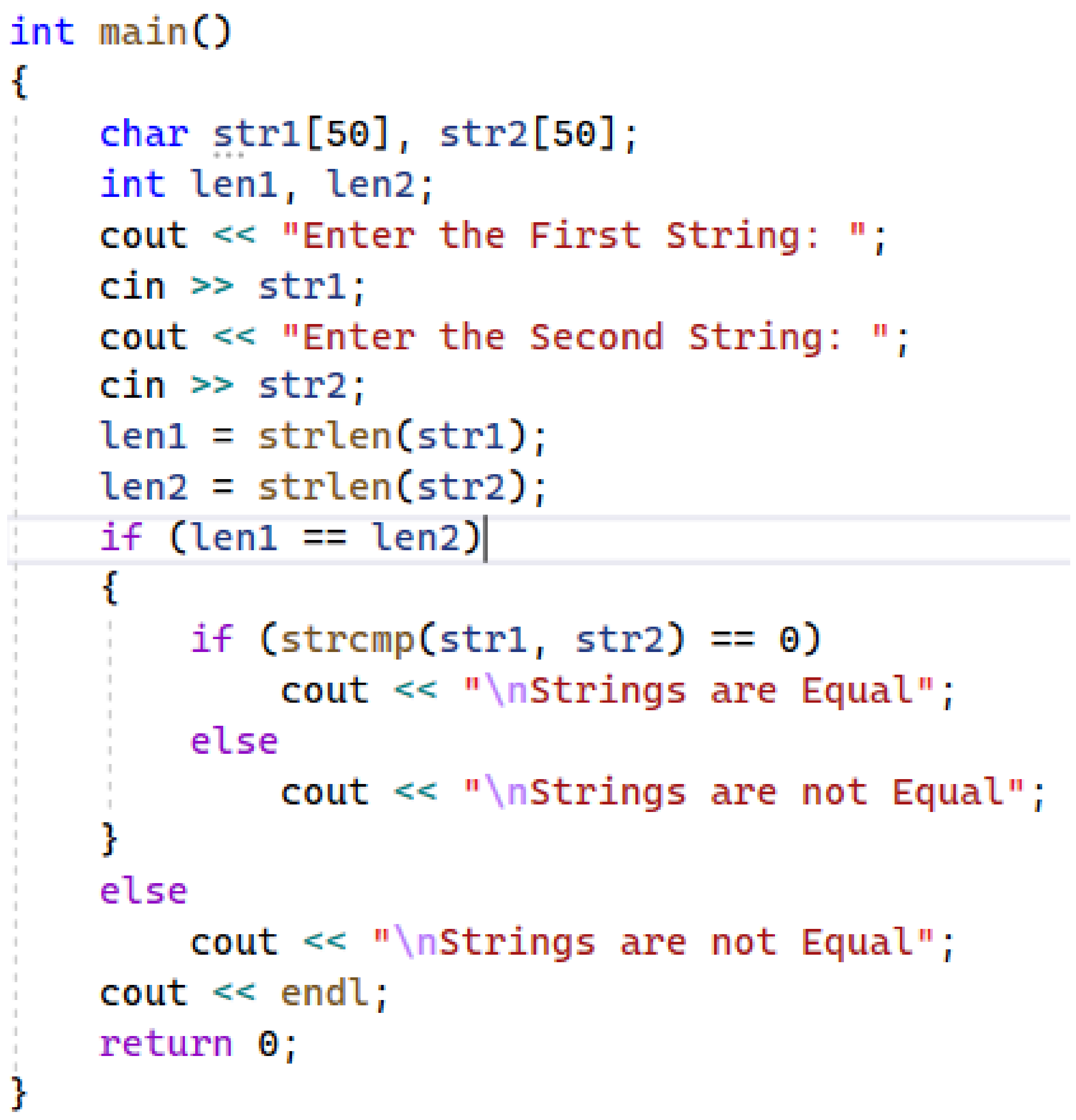
  strcmp (str1, str2)

Compares the C string str1 to the C string str2.

This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating null-character is reached.

It will return 0 if both strings are same.

Example:



```cpp
int main()
{
    char str1[50], str2[50];
    int len1, len2;
    cout << "Enter the First String: ";
    cin >> str1;
    cout << "Enter the Second String: ";
    cin >> str2;
    len1 = strlen(str1);
    len2 = strlen(str2);
    if (len1 == len2)
    {
        if (strcmp(str1, str2) == 0)
            cout << "\nStrings are Equal";
        else
            cout << "\nStrings are not Equal";
    }
    else
        cout << "\nStrings are not Equal";
    cout << endl;
    return 0;
}
```

Functions on Char Array

- **Comparing n elements of two C-strings:**

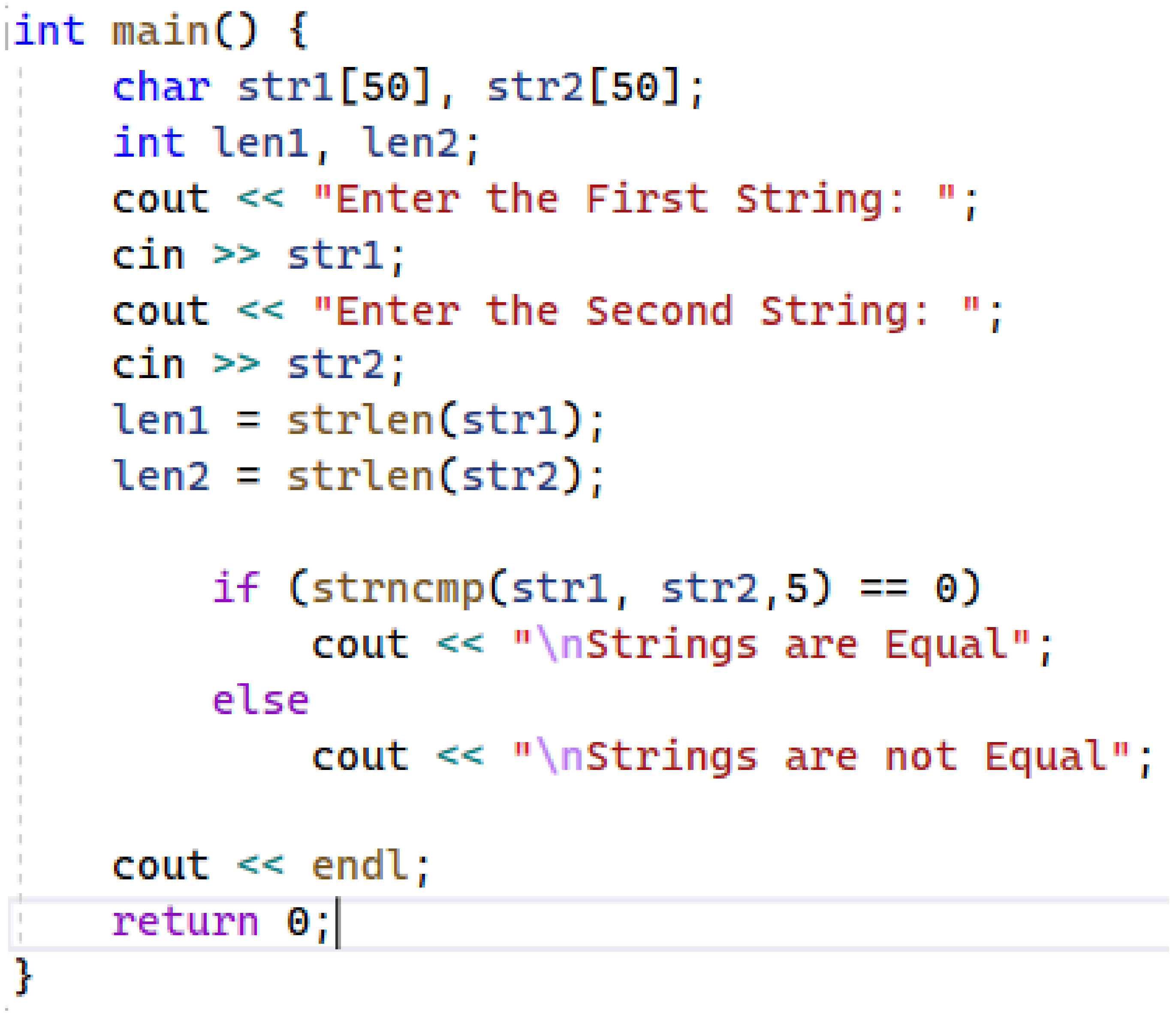
  strncmp (dest, source, n)

Compares the C string str1 to the C string str2 till n.

This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating null-character is reached or until the n is reached.

It will return 0 if both strings are same.

Example:



```
int main() {
    char str1[50], str2[50];
    int len1, len2;
    cout << "Enter the First String: ";
    cin >> str1;
    cout << "Enter the Second String: ";
    cin >> str2;
    len1 = strlen(str1);
    len2 = strlen(str2);

        if (strncmp(str1, str2,5) == 0)
            cout << "\nStrings are Equal";
        else
            cout << "\nStrings are not Equal";

    cout << endl;
    return 0;
}
```

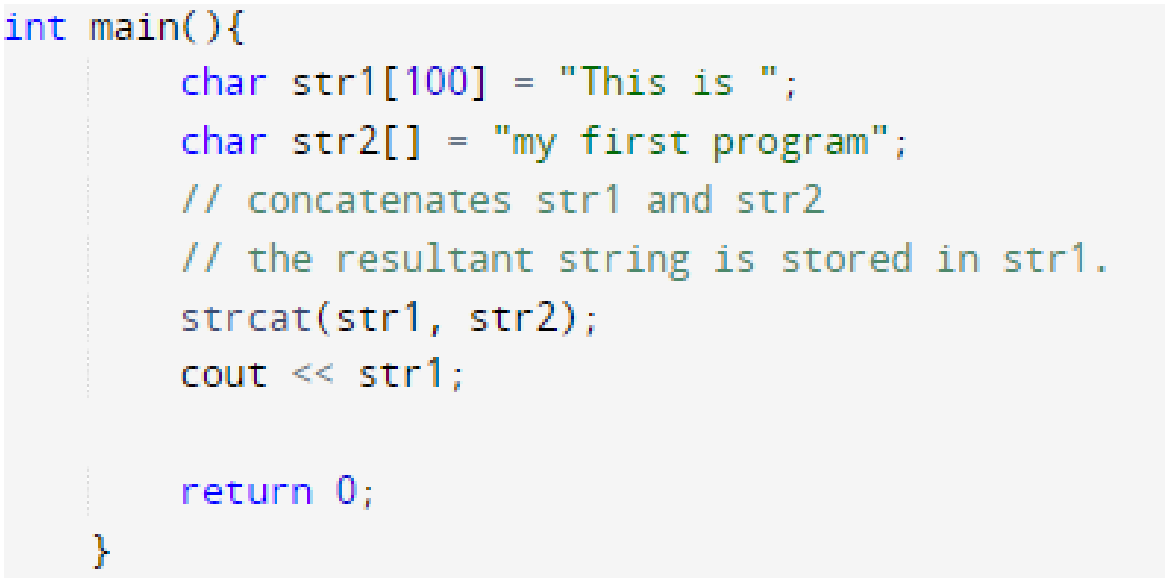
Functions on Char Array

- **Concatenating two C-strings:**

  strcat (dest, source)

Concatenates the C string str1 to the C string str2. Resultant string is stored in destination string.

Example:



```
int main(){
        char str1[100] = "This is ";
        char str2[] = "my first program";
        // concatenates str1 and str2
        // the resultant string is stored in str1.
        strcat(str1, str2);
        cout << str1;

        return 0;
    }
```

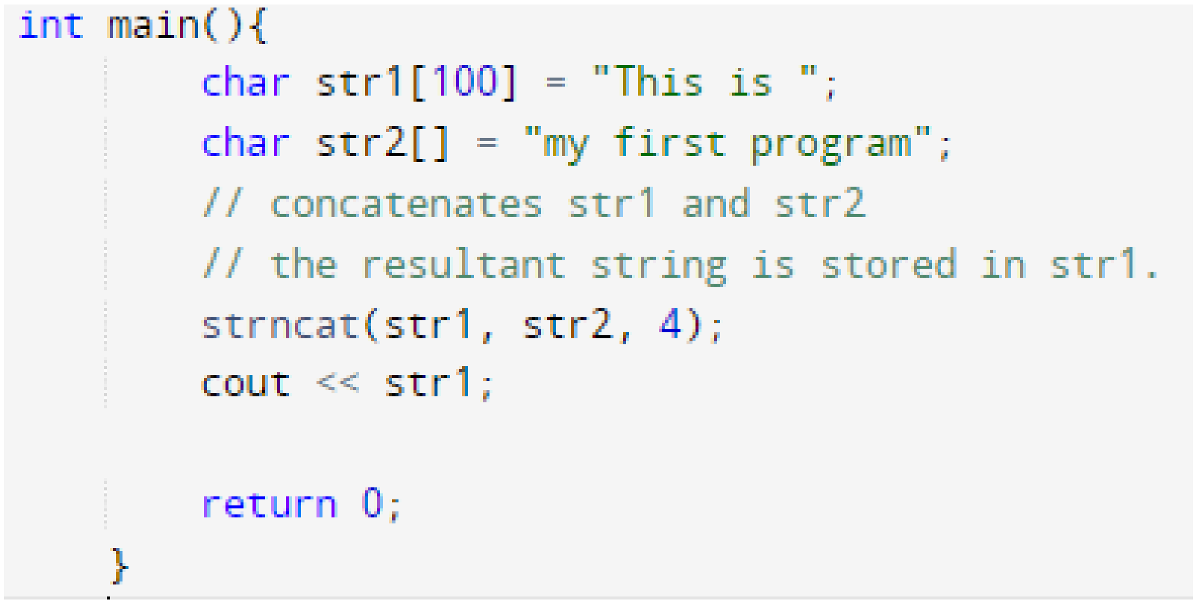
Functions on Char Array

- **Concatenating two C-strings till n:**

  strncat (dest, source, n)

Concatenates the C string str1 to the C string str2 till n. Resultant string is stored in destination string.

Example: Output: This is my f



```
int main(){
        char str1[100] = "This is ";
        char str2[] = "my first program";
        // concatenates str1 and str2
        // the resultant string is stored in str1.
        strncat(str1, str2, 4);
        cout << str1;

        return 0;
    }
```

String Datatype

- C++ has a **<string>** library
- Include it in your programs when you wish to use strings: #include <string>
- In this library, a class string is defined and implemented
- It is very convenient and makes string processing easier than in C

Declaration of Strings

- The following instructions are all equivalent. They declare x to be an object of type string, and assign the string “high school” to it:

- string x(“high school”);
- string x= “high school”;

- string x;
  x=“high school”;

Difference b/w string and c-string

Comparison	Char Arrays	String
Basic	Character array is collection of variables, of character data type.	String is class and variables of string are the object of class "string".
Syntax	char array_name [size];	string string_name;
Indexing	An individual character in a character array can be accessed by its index in array.	In string the particular character can be accessed by the function "string_name.at(index)“ or string_name[index]
Data Type	A character array does not define a datatype.	A string defines a datatype in C++.
Boundary	Array boundaries are easily overrun.	Boundaries will not overrun.
Access	Fast accessing.	Slow accessing.

Functions on String

- **String_name[index]**

This function will return the char that occur at that index.

Example:

```
#include <string>
int main(){
                string str = "ABC";
                cout << str[2];   //C
                cout << str[6]; //Error; program will crash
                return 0;
        }
```

Functions on String

- **String_name.at(index)**

This function will return the char that occur at that index.

Example:

```
#include <string>
int main(){
            string str = "ABC";
            cout << str.at(0);   //A
            cout << str.at(100);  //Error; program will crash
            return 0;
      }
```

Functions on String

- **Extracting the substring.**

string_name.substr(starting_pos, length)

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

    // Copy three characters of s1 (starting
    // from position 1)
    string r = s1.substr(1, 3);

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

    return 0;
}
```

Functions on String

- **Find the substring position**

  string_name.find(string_name)

```
int main()
{
    string str= "java is the best programming language";
    cout <<  str<<'\n';
    cout <<" Position of the programming word is :";
    cout<< str.find("programming");    \\17
    return 0;
}
```

Functions on String

- **Find the substring position**

  string_name.find(string_name, pos)

pos defines the position of the character at which to start the search.

```
int main()
{
        string str = "Mango fruit is my favorite fruit";
        cout << str << '\n';
        cout << " position of fruit is :";
        cout << str.find("fruit",7);  \\27
        return 0;
}
```

Functions on String

- **Finding a char in a string**

string::npos is usually used to indicate no matches.

```
int main()
{
        string str = "Mango fruit is my favorite fruit";
        if (str.find(“fruit”)!= string::npos)
                cout << “String is found in the given sentence” <<endl;
        else
                cout << “String is not found in the given sentence” <<endl;
        return 0;
}
```

Functions on String

- **Find the frequency of a specific character**

```
int main()
{
   char c[] = "C++ programming is very easy.“;
   char check = 'm';
   int count = 0;

   for(int i = 0; c[i] != '\0'; ++i)
   {
       if(check == c[i])
           ++count;
   }
   cout << "Frequency of " << check <<  " = " << count;
   return 0;
}
```

Functions on String

- **Concatenating two strings:**

Let x and y be two strings. To concatenate two strings we can use ‘+’ operator.

```
int main()
{
        string x = "high";
        string y = "school";
        string z;
        z = x + y;
        cout << "z= " << z;    //highschool
}
```

Any Questions!