STRINGS

The string class

- `string` is a powerful class for handling text in C++.
- It is **not** part of the STL but belongs to the C++ Standard Library.
- Defined in `<string>` header.
- Supports dynamic allocation and various operations.

```
#include <string>
using namespace std;
string s = "Hello, World!";
```

Basic string operations

- `length()` / `size()`: Get string length.
- `append()` / `+`: Concatenate strings.
- `substr(start, length)`: Extract substring.
- `find(substring)`: Locate substring.

```
std::string s1 = "Hello";
std::string s2 = " World";
std::string s3 = s1 + s2;
std::cout << s3;
// Output: Hello World
```

STRINGS MOR DETAIL

Strings are stored as character arrays with NULL character at the end:

```
string word  = "happy";
```

'h'	'a'	'p'	'p'	'y'	'\0'
0	1	2	3	4	5

Index

C- STRINGS (C PROGRAMMING LANGUAGE)

C-style strings are also stored as character arrays with NULL character at the end:

```
char word[6] = "happy";
```

'h'	'a'	'p'	'p'	'y'	'\0'
0	1	2	3	4	5

Index

You cannot use = or == with c-strings

```
#include <cstring>
//strcpy, strncpy - assignment
//strcmp, strncmp - compare strings
//strcat, strncat - add strings together (concatenate)
```

C++ STRING CLASS

	#include<cstring> **char s1[100], s2[100];**	#include<string> **string s1, s2;**
Assigning strings	strcpy(s1,"hello");	s1 = "hello";
Assigning strings	strcpy(s2, s1);	S2 = s1;
Adding strings	strcat(s1,s2);	s1 = s1 + s2;
String length	int len = strlen(s1);	int len = s1.length();
Comparing strings	if(strcmp(s1, s2) == 0)	if(s1 == s2)

THE STANDARD STRING CLASS IN C++

Converting string objects and c-style strings

```
char cString[] = "This is a string";
string sVariable = "This is also a string";

char cString2[40];
string sVariable2;

sVariable2 = cString; //this is OK
cString2 = sVariable; //this is NOT OK


strcpy(cString2, sVariable); //this is NOT OK
strcpy(cString2, sVariable.c_str());//this is OK
```

THE STANDARD STRING CLASS IN C++

Strings are stored as character arrays with NULL character at the end:

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

Allows the programmer to treat string values and expressions very much like a simple type

= assignment operator
+ adds strings together (concatenate) – overloaded operator

Default constructor

```
string word1; //automatically the empty string
string word2 = "everyday";//one way to initialize a string
string word3("every");//another way to initialize a string
```

THE STANDARD STRING CLASS IN C++

String processing

String has all the advantages of an array, you can access the character elements
Also, the string will automatically increase capacity

```
//makes an all uppercase copy of a string
//and returns it
string MakeUpper(const string& s1)
{
     string word(s1);//word is a copy of s1
     int len = s1.length();
     for (int i = 0; i < len; i++)
     {
          //word[i] = toupper(s1[i]);
          word.at(i) = toupper(s1.at(i));
     }
     return word;
}
```

```
//creates and returns new string out of a string
string CreateNewString(const string& s1)
{
     //string word = &s1[1];
     //start at the index 1 "veryday"
     string word = &s1.at(1);
     //this is the same as string word = &s1[1];
     word.erase(4, 3); //remove "day"
     word += " good"; //add good
     return word;//return "very good"
}
```

THE STANDARD STRING CLASS IN C++

= assignment operator
+ adds strings together (concatenate) – overloaded operator
== to compare

Member function length()

```
int len = s1.length();
```

Member function erase()

```
word.erase(4, 3); //remove a substring
```

Member function at()

```
//at() will check for illegal index
//these are the same
word[i] = toupper(s1[i]);
word.at(i) = toupper(s1.at(i));
```

THE STANDARD STRING CLASS IN C++

Converting string objects and c-style strings

```
char cString[] = "This is a string";
string sVariable = "This is also a string";

char cString2[40];
string sVariable2;

sVariable2 = cString; //this is OK
cString2 = sVariable; //this is NOT OK


strcpy(cString2, sVariable); //this is NOT OK
strcpy(cString2, sVariable.c_str());//this is OK
```

THE STANDARD STRING CLASS IN C++

Member Functions of the Standard Class `string`

Example	Remarks
Constructors	
`string str;`	Default constructor creates empty `string` object `str`.
`string str("sample");`	Creates a `string` object with data `"sample"`.
`string str(a_string);`	Creates a `string` object `str` that is a copy of `a_string`; `a_string` is an object of the class `string`.
Element access	
`str[i]`	Returns read/write reference to character in `str` at index `i`. Does not check for illegal index.
`str.at(i)`	Returns read/write reference to character in `str` at index `i`. Same as `str[i]`, but this version checks for illegal index.
`str.substr(position, length)`	Returns the substring of the calling object starting at `position` and having `length` characters.
Assignment/modifiers	
`str1 = str2;`	Initializes `str1` to `str2`'s data,
`str1 += str2;`	Character data of `str2` is concatenated to the end of `str1`.
`str.empty( )`	Returns *`true`* if `str` is an empty string; *`false`* otherwise.
`str1 + str2`	Returns a string that has `str2`'s data concatenated to the end of `str1`'s data.
`str.insert(pos, str2);`	Inserts `str2` into `str` beginning at position `pos`.
`str.remove(pos, length);`	Removes substring of size `length`, starting at position `pos`.
Comparison	
`str1 == str2  str1 != str2`	Compare for equality or inequality; returns a Boolean value.
`str1 < str2  str1 > str2` `str1 <= str2  str1 >= str2`	Four comparisons. All are lexicographical comparisons.
Finds	
`str.find(str1)`	Returns index of the first occurrence of `str1` in `str`.
`str.find(str1, pos)`	Returns index of the first occurrence of string `str1` in `str`; the search starts at position `pos`.
`str.find_first_of(str1, pos)`	Returns the index of the first instance in `str` of any character in `str1`, starting the search at position `pos`.
`str.find_first_not_of (str1, pos)`	Returns the index of the first instance in `str` of any character not in `str1`, starting the search at position `pos`.

String modification and Iteration

- `insert(position, str)`: Insert string
- `erase(position, length)`: Remove substring
- `replace(position, length, str)`: Replace substring

- Iterate using loops:

```
for (char letter : s)
    std::cout << letter << " ";
```

C++ STRING CLASS

	#include<cstring> **char s1[100], s2[100];**	#include<string> **string s1, s2;**
Assigning strings	strcpy(s1,"hello");	s1 = "hello";
Assigning strings	strcpy(s2, s1);	S2 = s1;
Adding strings	strcat(s1,s2);	s1 = s1 + s2;
String length	int len = strlen(s1);	int len = s1.length();
Comparing strings	if(strcmp(s1, s2) == 0)	if(s1 == s2)

C- STRINGS (CHARACTER ARRAYS)

Strings

```
#include<cstring>
```

A string in C is implemented as an array of characters:

```
char my_string[25];
```

- This string will only hold up to 24 characters.
- The last one is always assigned to a special notation character.

Initializing a string is very simple:

```
char your_string[10] = "BadDog";
```

- The array would look like this

B	a	d	D	o	g	\0	?	?

Notice the \0 after the last letter in the string.

- This is called the null character and indicates the end of the string.

C++ STRING CLASS

Strings

#include<cstring>

A string in C is implemented as an array of characters:

```
char my_string[25];
```

- This string will only hold up to 24 characters.
- The last one is always assigned to a special notation character.

Initializing a string is very simple:

```
char your_string[10] = "BadDog";
```

- The array would look like this

B	a	d	D	o	g	\0	?	?

Notice the \0 after the last letter in the string.

- This is called the null character and indicates the end of the string.

```
#include<string>

string my_string = "BadDog";
```

C- STRINGS (CHARACTER ARRAYS)

```
#include<cstring>
```

strlen – string length

strcmp, strncmp – string comparison

strcat, strncat – string concatenation

strcpy, strncpy – string copy

C- STRINGS (CHARACTER ARRAYS)

This function will output the index of the null character which terminates a string

```
int length;
char word[25] = “happy”;
length=strlen(word); //the length would be 5
```

C++ STRING CLASS

```
#include<cstring>
```

This function will output the index of the null character which terminates a string

int length;
char word[25] = “happy”;
length=strlen(word); //the length would be 5

```
#include<string>

string my_string = “happy”;
int len = my_string.length();
```

C- STRINGS (CHARACTER ARRAYS)

strcat and strncat

These two functions are used for concatenating strings by taking two string objects and concatenating the second string to the first and leaving the second alone.

```
char word4[20]="skate";
char word5[20]="board";
strcat(word4,word5);
//after this word4 would be "skateboard"
// word4 = word4 + word5; //NOT allowed in C
```

```
char word6[20]="skate";
char word7[20]="board";
strncat(word6,word7, 2);
//after this word6 would be "skatebo"
```

C++ STRING CLASS

strcat and strncat

These two functions are used for concatenating strings by taking two string objects and concatenating the second string to the first and leaving the second alone.

```
char word4[20]="skate";
char word5[20]="board";
strcat(word4,word5);
//after this word4 would be "skateboard"
// word4 = word4 + word5; //NOT allowed in C
```

```
char word6[20]="skate";
char word7[20]="board";
strncat(word6,word7, 2);
//after this word6 would be "skatebo"
```

```
#include<cstring>
```

```
#include<string>

string s1= "skate";
string s2= "board";

s1 = s1 + s2; //skateboard

string s3= "skate";
string s4= "board";

s3.append(s4,0,2);//skatebo
```

C- STRINGS (CHARACTER ARRAYS)

```
//add one string onto the end of another (concatenate)
strcat(char string1[], char string2[])
```

To add one string to the end of another can be done with the `strcat` function.

Example:

```
char first[25] = "skate";
char second[35] = "board";

strcat(first, second);
strcat(second, "room");
```

C- STRINGS (CHARACTER ARRAYS)

```
//add only certain number of characters to the end
strncat(char string1[], char string2[], int numLetters)
```

strncat WILL ADD THE NULL CHARACTER

Example:

```
char first[25] = "skate";
char second[35] = "board";

strncat(first, second,3);
strncat(second, "room",2);
```

C- STRINGS (CHARACTER ARRAYS)

strcpy and strncpy

These two functions are used to copy string and takes two string arguments and replaces the first string with the second string and leaving the second string alone

```
char word8[20];
strcpy(word8, "hello");
//word8  = "hello" //NOT allowed in C
```

Like strncmp and strncat, strncpy adds a length variable to its declaration

```
strncpy(word8,"hello", 2 );
```

Note that with strncpy you need to update the '\0' character to the end of the string.
Ex: in the command above use word8[2]='\0';

C++ STRING CLASS

strcpy and strncpy

These two functions are used to copy string and takes two string arguments and replaces the first string with the second string and leaving the second string alone

```
char word8[20];
strcpy(word8, "hello");
//word8 = "hello" //NOT allowed in C
```

Like strncmp and strncat, strncpy adds a length variable to its declaration

```
strncpy(word8,"hello", 2 );
```

Note that with strncpy you need to update the '\0' character to the end of the string.
Ex: in the command above use word8[2]='\0';

```
#include<cstring>
```

```
#include<string>

string s1, s2, s3;
s1= "hello";

s2 = s1;

s3 = s1.substr(0,2);
```

C- STRINGS (CHARACTER ARRAYS)

strcpy

```
//copy one string to another
strcpy(char string1[], char string2[])
```

```
char receiving_string[20] = "hello";
char donating_string[20] = "goodbye";
//Be sure to check that the receiving string is large enough
strcpy(receiving_string, donating_string);
//after this both strings will be "goodbye"
```

C- STRINGS (CHARACTER ARRAYS)

strncpy

```
//copy only a certain number of characters
strncpy(char string1[], char string2[], int numLetters)
```

```
//The only problem with this function is that it does not automatically add the \0 character to
//the end of the new string. It must be done manually.
```

Example:

```
char word[10] = "gotcha";
char piece[6];
strncpy(piece, word, 3);
piece[3] = '\0';
```

OUTPUT:

The piece is got