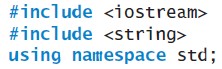
Strings

STRING

* The string is any sequence of characters • Every string is terminated by a ‘\0’ character.
* To use strings, you need to include the header **<string>**



* The string is one of the C++ built-in classes.
* C++ strings allow you to directly initialize, assign, compare, and reassign with the intuitive operators, as well as printing and reading (e.g., from the user).

# Declaring string Objects

**// declaring one object called str**

string

str;

**// str1, str2, and str3 would be string objects**

string str1, str2, str3;

**Initializing string Objects**

string str1 = "This is an example."; string str2 = str1;

# Reading a string input

//To read a single word:

cin >> str1;

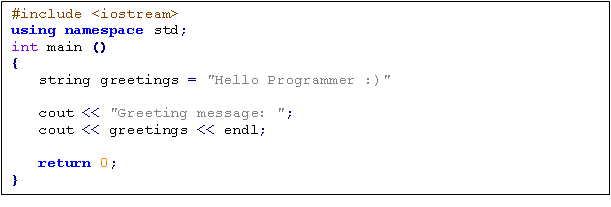
//To read a single line of input:

getline

(

cin, st1);

EXAMPLE:

# Output

Greeting message Hello Programmer :)

## EXAMPLE

• **string name; cout << "Enter your name: " << endl; cin >> name;**

**cout << “Welcome " << name <<“\n Have a nice day \n”;**

**Enter your name: Sidra**

**Welcome sidra**

**Have a nice day**

## Strings concatenation with the + operator

• C++ strings also provide many string manipulation facilities. The simplest string manipulation that we commonly use is concatenation, or addition of strings. In C++, we can use the **+** operator to concatenate (or “add”) two strings, as shown below:

**string result;**

**string s1 = "hello "; string s2 = "world"; result = s1 + s2;**

**// result now contains "hello world"**

## Access a character of string

The subscript operator, [ *int* ], can be used with strings to access and modify individual characters.

The strings have a first subscript of 0.

* **string x = “high”;**
* **char c = x[0]; // c is ‘h’**
* **c = x[1]; // c is ‘i’**
* **c = x[2]; // c is ‘g’**

Operators on string Objects

|  |  |  |
| --- | --- | --- |
| **Type** | **Operator** | **Action** |
| Assignment | = | Stores string |
|  | += | Concatenates and stores |
| Comparison | == | True if strings identical |
|  | != | True if strings not identical |
|  | > | True if first string greater than second |
|  | < | True if first string is less than second |
|  | >= | True if 1st string greater or equal than  2nd |
|  | <= | True if 1st string less or equal than |

2nd

CLICK TO EDIT MASTER TITLE STYLEInput/Output >> For input and string objects

<< For output and string objects

Character [ ] To access individual characters access

Concatenation + Connects two strings

String - Functions

|  |  |
| --- | --- |
| **Function** | **Description** |
| **size** | Return length of string |
| **length** | Return length of string |
| **max\_size** | Return maximum size of string |
| **resize** | Resize string |
| **capacity** | Return size of allocated storage |
| **reserve** | Request a change in capacity |
| **clear** | Clear string |
| **empty** | Test if string is empty |
| **assign** | Assign content to string |
| **insert** | Insert into string |
| **erase** | Erase characters from string |

String - Functions

|  |  |
| --- | --- |
| **replace** | Replace portion of string |
| **swap** | Swap string values |
| **copy** | Copy sequence of characters from string |
| **find** | Find content in string |
| **rfind** | Find last occurrence of content in string |
| **substr** | Generate substring |
| **compare** | Compare strings |
| **operator>>** | Extract string from stream |
| **operator<<** | Insert string into stream |
| **getline** | Get line from stream into string |

substr Function

* The required substring is specified by the starting position and the number of characters, taking into account that the position of the first character in the string is **0**. ob**.substr(** int**,** int**)**

position Number of character

**string text = "hello world, this is a test";**

CLICK TO EDIT MASTER TITLE STYLE**string fragment = text.substr(6, 5);**

**// start at 6, take 5 characters // fragment = “world”**

### String Functions Examples

**substr ()**

|  |  |
| --- | --- |
| #include <iostream> #include <string> **using** **namespace** std**;**    int main **()**  **{** string str**=**"We think in generalities, but we live in details."**;**    string str2 **=** str**.**substr **(**12**,**12**);** // "generalities"     |  | | --- | | **# Output generalities** |   cout **<<** str2**;**    **return** 0**;**  **}** |

#### erase ()

#include <iostream>

#include <string>

**using**

**namespace**

std

**;**

int

main

**()**

**{**

s

tring str

**(**

"This is an example sentence."

**;**

**)**

cout

**<<**

str

**<<**

'

\

n'

**;**

// "This is an example sentence."

str

**.**

erase

**(**

10

**,**

8

**;**

**)**

// ^^^^^^^^

cout

**<<**

str

**<<**

'

\

n'

**;**

// "This is an sentence."

str

**.**

erase

**(**

str

**.**

begin

**()+**

9

**)**

**;**

// ^

cout

**<<**

str

**<<**

'

\

n'

**;**

// "This is a sentence."

str

**.**

erase

**(**

str

**.**

begin

**()+**

5

**,**

str

**.**

end

**()**

**-**

9

**;**

**)**

// ^^^^^

c

out

**<<**

str

**<<**

'

\

n'

**;**

// "This sentence."

**return**

0

**;**

**}**

**Output**

**This is an example sentence.**

**This is an sentence.**

**This is a sentence.**

**This sentence.**

**replace ()**

// replacing in a string

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

**{**

string base**=**"this is a test string."**;** string str2**=**"n example"**;** string str3**=**"sample phrase"**;** string str4**=**"useful."**;**

// Using positions: 0123456789\*123456789\*12345 string str**=**base**;** // "this is a test string." str**.**replace**(**9**,**5**,**str2**);** // "this is an example string." (1) str**.**replace**(**19**,**6**,**str3**,**7**,**6**);** // "this is an example phrase." (2) str**.**replace**(**8**,**10**,**"just a"**);** // "this is just a phrase." (3) str**.**replace**(**8**,**6**,**"a shorty"**,**7**);** // "this is a short phrase." (4) str**.**replace**(**22**,**1**,**3**,**'!'**);** // "this is a short phrase!!!" (5)

cout **<<** str **<<** '\n'**;** **return** 0**;** **}**

**# Output this is a short phrase!!!**

empty Function

* the empty function determines whether a string object is empty or not. ob.**empty**();
* The function empty returns true if the string is empty; otherwise, it returns false.

**Str = “ Hi “;**

CLICK TO EDIT MASTER TITLE STYLE**bool flag = Str.empty(); // flag = False cout<< flag <<end; // output: 0**

### String Functions Examples

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

{

string str = "We think out of the box please "; cout << str.length() << endl; size\_t str1 = str.find("think"); cout << str1 << endl;

}

insert Function •Adds characters (string) to a string object str1.**insert**(pos1, str2);

* index is beginning position
* ob2 represents what is to be inserted

**s1 = "This is an example.";**

**s1.insert (8,"just ");**

CLICK TO EDIT MASTER TITLE STYLE**\\s1 = "This is just an example.“**

compare Function

Compare() Compare two strings

string s1 = “abd”, s2 = “abc”; int d = s1.compare(s2); if(d==0)

cout<<“both strings are identical”;

CLICK TO EDIT MASTER TITLE STYLEelse if (d>0) cout<<“s1 is greater then s2”; else

cout<<“s2 is greater then s1”;

Accessing Elements of String: at()

|  |
| --- |
| int main ()  { string s1 ="abcdefghijkl"; for(int i=0 ; i< s1.length();i++)  { cout<<s1.at(i)<<endl;  //using at()  } for(int i=0 ; i< s1.length();i++)  { cout<<s1[i]<<" ";//using []  operator }  } |

|  |
| --- |
| **Parameters**  *position*  Value with the position of a character within the string. |

|  |
| --- |
| **Return value**  The character at the specified position in the string. |

#### CLICK TO EDIT MASTER TITLE STYLE