STRINGS IN C++

STRINGS IN C++

ARE JUST SUCH A JOY TO WORK WITH
WHEN CONTRASTED WITH STRINGS IN C

THERE IS A `string` CLASS THAT
BASICALLY DOES EVERYTHING YOU NEED

C++ STRINGS ARE DIFFERENT FROM C STRINGS -
NOT DELIMITED BY '\0' - BUT YOU CAN EASILY
CONVERT BETWEEN C `char*` AND C++ `string`

THERE IS A `string` CLASS THAT BASICALLY DOES EVERYTHING YOU NEED

C++ STRINGS ARE DIFFERENT FROM C STRINGS - NOT DELIMITED BY '\0'
- BUT YOU CAN EASILY CONVERT BETWEEN C `char*` AND C++ `string`

YOU CAN CREATE, CONCATENATE, MANIPULATE AND COMPARE STRINGS VERY EASILY INDEED

ASIDE: C++ ALSO HAS A 2 BYTE CHAR, CALLED `wchar_t` TO HOLD NON-ASCII CHARACTERS (NON-ENGLISH LANGUAGES, FOR INSTANCE)

EXAMPLE 14:

CREATE, PRINT, AND CONCATENATE C++ STRINGS

EXAMPLE 14: CREATE, PRINT, AND CONCATENATE C++ STRINGS

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

int main()
{
  string firstName("Vitthal");
  string lastName = "Srinivasan";
  string fullName = firstName + lastName;

  cout << "First name = " << firstName << endl;
  cout << "Last name = " << lastName << endl;
  cout << "Full name = " << fullName << endl;



}
```

CREATING STRINGS

EXAMPLE 14: CREATE, PRINT, AND CONCATENATE C++ STRINGS

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

int main()
{
  string firstName("Vitthal");
  string lastName = "Srinivasan";
  string fullName = firstName + lastName;

  cout << "First name = " << firstName << endl;
  cout << "Last name = " << lastName << endl;
  cout << "Full name = " << fullName << endl;



}
```

JUST THIS 1, OBVIOUSLY NAMED, # INCLUDE

CREATING STRINGS

EXAMPLE 14: CREATE, PRINT, AND CONCATENATE C++ STRINGS

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

int main()
{
  string firstName("Vitthal");
  string lastName = "Srinivasan";
  string fullName = firstName + lastName;

  cout << "First name = " << firstName << endl;
  cout << "Last name = " << lastName << endl;
  cout << "Full name = " << fullName << endl;



}
```

ITS THIS EASY TO INITIALISE A STRING

CREATING STRINGS

EXAMPLE 14: CREATE, PRINT, AND CONCATENATE C++ STRINGS

AND ITS THIS EASY TO CONCATENATE 2 STRINGS

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

int main()
{
  string firstName("Vitthal");
  string lastName = "Srinivasan";
  string fullName = firstName + lastName;

  cout << "First name = " << firstName << endl;
  cout << "Last name = " << lastName << endl;
  cout << "Full name = " << fullName << endl;



}
```

CREATING STRINGS

EXAMPLE 14: CREATE, PRINT, AND CONCATENATE C++ STRINGS

THIS IS OUR FIRST REAL ENCOUNTER WITH AN OVERLOADED OPERATOR

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

int main()
{
  string firstName("Vitthal");
  string lastName = "Srinivasan";
  string fullName = firstName + lastName;

  cout << "First name = " << firstName << endl;
  cout << "Last name = " << lastName << endl;
  cout << "Full name = " << fullName << endl;



}
```

CREATING STRINGS

EXAMPLE 14: CREATE, PRINT, AND CONCATENATE C++ STRINGS

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

int main()
{
  string firstName("Vitthal");
  string lastName = "Srinivasan";
  string fullName = firstName + lastName;

  cout << "First name = " << firstName << endl;
  cout << "Last name = " << lastName << endl;
  cout << "Full name = " << fullName << endl;

}
```

THIS IS OUR FIRST REAL ENCOUNTER WITH AN OVERLOADED OPERATOR

THE OPERATOR + HAS BEEN OVERLOADED, SO THAT IT CAN ALSO ADD 2 `string` OBJECTS!

THE OPERATOR + HAS BEEN OVERLOADED, SO THAT IT CAN ALSO ADD 2 `string` OBJECTS!

WE WILL HAVE AN ENTIRE SECTION ON OPERATOR OVERLOADING

BUT FOR NOW JUST REMEMBER WHAT IT IS - AND THAT IT IS POSSIBLE

EXAMPLE 14: CREATE, PRINT, AND CONCATENATE C++ STRINGS

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

int main()
{
  string firstName("Vitthal");
  string lastName = "Srinivasan";
  string fullName = firstName + lastName;

  cout << "First name = " << firstName << endl;
  cout << "Last name = " << lastName << endl;
  cout << "Full name = " << fullName << endl;

}
```

THIS IS OUR FIRST REAL ENCOUNTER WITH AN OVERLOADED OPERATOR

THE OPERATOR + HAS BEEN OVERLOADED, SO THAT IT CAN ALSO ADD 2 `string` OBJECTS!

EXAMPLE 14: CREATE, PRINT, AND CONCATENATE C++ STRINGS

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

int main()
{
  string firstName("Vitthal");
  string lastName = "Srinivasan";
  string fullName = firstName + lastName;

  cout << "First name = " << firstName << endl;
  cout << "Last name = " << lastName << endl;
  cout << "Full name = " << fullName << endl;


}
```

AND COUT IS SMART ENOUGH TO PRINT STRINGS TOO!

EXAMPLE 14: CREATE, PRINT, AND CONCATENATE C++ STRINGS

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

int main()
{
   string firstName("Vitthal");
   string lastName = "Srinivasan";
   string fullName = firstName + lastName;

   cout << "First name = " << firstName << endl;
   cout << "Last name = " << lastName << endl;
   cout << "Full name = " << fullName << endl;
}
```

AND COUT IS SMART ENOUGH TO PRINT STRINGS TOO!

THE OPERATOR << HAS BEEN OVERLOADED, SO THAT IT CAN DISPLAY A `string` OBJECT TO SCREEN!

EXAMPLE 15:

INPUT STRINGS, INCLUDING MULTILINE STRINGS

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

TO INPUT A SINGLE STRING, JUST INPUT USING CIN

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

TO INPUT MULTIPLE LINES, USE

getline

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

TO INPUT MULTIPLE LINES, USE

getline

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name: " << firstName << endl;
cout << "Last name: " << lastName << endl;
cout << "Full name: " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

TO INPUT MULTIPLE LINES, USE

getline

WHERE TO READ THE LINES FROM?

THE STANDARD INPUT STREAM **cin**

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

TO INPUT MULTIPLE LINES, USE

getline

WHICH STRING VARIABLE TO STORE THE INPUT IN? THE STRING VARIABLE CALLED

USERINPUT

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address = " << endl;
cout << userInput << endl;
```

TO INPUT MULTIPLE LINES, USE

getline

WHAT CHARACTER WILL SIGNIFY END OF THE INPUT? (SINCE IT CAN'T BE A NEWLINE!) .HERE WE SPECIFY A TILDE (~)

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please enter your address
(Press ~ when done)
A-1102, Mantri Espana
Bangalore
India
~
First name = Vitthal
Last name = Srinivasan
Full name = VitthalSrinivasan
Address:

A-1102, Mantri Espana
Bangalore
India
```

INPUTING STRINGS

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please enter your address
(Press ~ when done)
A-1102, Mantri Espana
Bangalore
India
~
First name = Vitthal
Last name = Srinivasan
Full name = VitthalSrinivasan
Address:

A-1102, Mantri Espana
Bangalore
India
```

INPUTING STRINGS

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

```
string firstName,lastName, fullName;
cout << "Please could you enter your FIRST name" << endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;

fullName = firstName + lastName;
string userInput;
cout << "Please enter your address" << endl;
cout << "(Press ~ when done)" << endl;
getline(cin,userInput, '~');    //reads multiple lines

cout << "First name = " << firstName << endl;
cout << "Last name = " << lastName << endl;
cout << "Full name = " << fullName << endl;
cout << "Address:" <<endl;
cout << userInput << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please enter your address
(Press ~ when done)
A-1102, Mantri Espana
Bangalore
India
~
First name = Vitthal
Last name = Srinivasan
Full name = VitthalSrinivasan
Address:

A-1102, Mantri Espana
Bangalore
India
```

INPUTING STRINGS

EXAMPLE 15: INPUT STRINGS, INCLUDING MULTILINE STRINGS

TO INPUT MULTIPLE LINES, USE

`getline`

YOU ALSO NEED `getline` TO INCLUDE STRINGS WITH SPACES. REMEMBER THIS!

EXAMPLE 16:

CARRY OUT COMMON STRING OPERATIONS

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO CALCULATE THE LENGTH OF A STRING, USE `string::size()`

```
cout << "The number of characters in " << fullName << " is " << fullName.size() << endl;
```

LENGTH OF A STRING

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO CALCULATE THE LENGTH OF A STRING, USE `string.size()`

YOU CAN ALSO USE

`string::length()`

```
cout << "The number of characters in " << fullName << " is " << fullName.size() << endl;

cout << "The number of characters in "
     << fullName << " can also be found as "
     << fullName.length()
     << endl;
```

LENGTH OF A STRING

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO FIND A SUBSTRING OF A STRING, USE `string::substr()`

```
string subString = firstName.substr(0,3);
```

SUBSTRINGS

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO FIND A SUBSTRING OF A STRING, USE `string::substr()`

```
string subString = firstName.substr(0,3);
```

STARTING INDEX OF SUBSTRING

SUBSTRINGS

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO FIND A SUBSTRING OF A STRING, USE `string::substr()`

```
string subString = firstName.substr(0,3);
```

LENGTH OF SUBSTRING

SUBSTRINGS

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO ERASE PART OF A STRING, USE string::erase()

```
string subString = firstName.erase(0,3);
```

STARTING INDEX OF SUBSTRING

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO ERASE PART OF A STRING, USE `string::erase()`

```
string subString = firstName.erase(0,3);
```

HOW MANY CHARACTERS TO ERASE?

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO INSERT ONE STRING INTO ANOTHER, USE

string::insert()

```
firstName.insert(0,subString);
```

WHAT POSITION TO INSERT?

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO INSERT ONE STRING INTO ANOTHER, USE

string::insert()

```
firstName.insert(0,subString);
```

WHAT STRING TO INSERT?

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO FIND ONE STRING INSIDE ANOTHER, USE `string::find()`

```
size_t positionOfFirstName = fullName.find(firstName);
```

C++ ALIAS FOR UNSIGNED INT

SEARCHING INSIDE STRINGS

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO FIND ONE STRING INSIDE ANOTHER, USE `string::find()`

```
size_t positionOfFirstName = fullName.find(firstName);
```

WHAT STRING TO FIND?

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO FIND ONE STRING INSIDE ANOTHER, USE `string::find()`

```
size_t positionOfFirstName = fullName.find(firstName);
```

INSIDE WHICH STRING?

SEARCHING INSIDE STRINGS

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO FIND ONE STRING INSIDE ANOTHER, USE `string::find()`

```
size_t positionOfFirstName = fullName.find(firstName);
```

STARTING INDEX, IF FOUND, ELSE `string::npos`

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

TO FIND ONE STRING INSIDE ANOTHER, USE `string::find()`

```
size_t positionOfFirstName = fullName.find(firstName);
```

STARTING INDEX, IF FOUND, ELSE

`string::npos`

THIS IS A SPECIAL VALUE, DEFINED INSIDE THE STRING CLASS. TO BE PRECISE, ITS A `const static` MEMBER (MORE ON THIS LATER!)

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
string firstName,lastName,nickName, fullName;
cout << "Please could you enter your FIRST name" <<endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;
cout << "Please could you enter your NICK name" <<endl;
cin >> nickName;
fullName = firstName + lastName;

cout << "The number of characters in " << fullName << " is "
     << fullName.size() << endl;
cout << "The number of characters in " << fullName << " can also be found as "
     << fullName.length() << endl;
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
string firstName,lastName,nickName, fullName;
cout << "Please could you enter your FIRST name" <<endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;
cout << "Please could you enter your NICK name" <<endl;
cin >> nickName;
fullName = firstName + lastName;

cout << "The number of characters in " << fullName << " is "
     << fullName.size() << endl;
cout << "The number of characters in " << fullName << " can also be found as "
     << fullName.length() << endl;
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
string firstName,lastName,nickName, fullName;
cout << "Please could you enter your FIRST name" <<endl;
cin >> firstName;
cout << "Please could you enter your LAST name" <<endl;
cin >> lastName;
cout << "Please could you enter your NICK name" <<endl;
cin >> nickName;
fullName = firstName + lastName;

cout << "The number of characters in " << fullName << " is "
     << fullName.size() << endl;
cout << "The number of characters in " << fullName << " can also be found as "
     << fullName.length() << endl;
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
cout << "The number of characters in " << fullName << " is "
     << fullName.size() << endl;
cout << "The number of characters in " << fullName << " can also be found as "
     << fullName.length() << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
string subString = firstName.substr(0,3);


cout << "let's find the first 3 characters of the name "
     << firstName << " use substr "
     << subString
     << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
string subString = firstName.substr(0,3);


cout << "let's find the first 3 characters of the name "
     << firstName << " use substr "
     << subString
     << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
firstName.erase(0,3);
```

```
cout << "After erasing the first 3 characters it has becomeg "
     << firstName
     << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
firstName.erase(0,3);
```

```
cout << "After erasing the first 3 characters it has becomeg "
     << firstName
     << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
firstName.insert(0,subString);
```

```
cout << "To re-insert those ffirst 3 characters of the string "
     << firstName << " use insert : "
     << firstName
     << endl;
```

REMEMBER `subString = "Vit"`

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
firstName.insert(0,subString);
```

```
cout << "To re-insert those ffirst 3 characters of the string "
     << firstName << " use insert : "
     << firstName
     << endl;
```

REMEMBER `subString = "Vit"`

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
cout << "Your nickname is " << nickName << endl;

size_t positionOfFirstName = fullName.find(firstName);
cout << "Position of first name " << firstName
     << "Inside the full name is" << positionOfFirstName << endl;

fullName.replace(positionOfFirstName,firstName.length(),"");
fullName.insert(positionOfFirstName,nickName);
cout << "New Full name:" << fullName << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
cout << "Your nickname is " << nickName << endl;

size_t positionOfFirstName = fullName.find(firstName);
cout << "Position of first name " << firstName
     << "Inside the full name is" << positionOfFirstName << endl;

fullName.replace(positionOfFirstName,firstName.length(),"");
fullName.insert(positionOfFirstName,nickName);
cout << "New Full name:" << fullName << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
size_t positionOfFirstName = fullName.find(firstName);
cout << "Position of first name " << firstName
     << "Inside the full name is" << positionOfFirstName << endl;
```

REMEMBER fullName = "VitthalSrinivasan"

REMEMBER firstName = "Vitthal"

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

```
cout << "Your nickname is " << nickName << endl;

size_t positionOfFirstName = fullName.find(firstName);
cout << "Position of first name " << firstName
     << "Inside the full name is" << positionOfFirstName << endl;

fullName.replace(positionOfFirstName,firstName.length(),"");
fullName.insert(positionOfFirstName,nickName);
cout << "New Full name:" << fullName << endl;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

REPLACE EVERYTHING STARTING AT POSITIONOFFIRSTNAME AND WITH LENGTH FIRSTNAME.LENGTH() BY THE EMPTY STRING

```
fullName.replace(positionOfFirstName,firstName.length(),"");
fullName.insert(positionOfFirstName,nickName);
cout << "New Full name:" << fullName << endl;
```

VITTHALSRINIVASAN

I.E REPLACE "VITTHAL" WITH ""

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

INSERT THE NICKNAME STARTING AT POSITION POSITIONOFFIRSTNAME

```
fullName.replace(positionOfFirstName,firstName.length(),"");
fullName.insert(positionOfFirstName,nickName);
cout << "New Full name:" << fullName << endl;
```

HUMPTYDUMPTYSRINIVASAN

I.E INSERT "HUMPTYDUMPTY" AT 0

EXAMPLE 16: CARRY OUT COMMON STRING OPERATIONS

HUMPTYDUMPTYSRINIVASAN

```
fullName.replace(positionOfFirstName,firstName.length(),"");
fullName.insert(positionOfFirstName,nickName);
cout << "New Full name:" << fullName << endl:
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example16.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please could you enter your FIRST name
Vitthal
Please could you enter your LAST name
Srinivasan
Please could you enter your NICKT name
HumptyDumpty
The number of characters in VitthalSrinivasan is 17
The number of characters in VitthalSrinivasan can also be found as 17
let's find the first 3 characters of the name Vitthal use substr Vit
After erasing the first 3 characters it has becomeg thal
To re-insert those ffirst 3 characters of the string Vitthal use insert : Vitthal
Your nickname is HumptyDumpty
Position of first name VitthalInside the full name is0
New Full name:HumptyDumptySrinivasan
```

EXAMPLE 17:

COMPARING STRINGS

EXAMPLE 17: COMPARING STRINGS

THE OPERATORS <,>,=ETC HAVE BEEN OVERLOADED, SO THAT THEY CAN COMPARE `string` OBJECTS!

EXAMPLE 17: COMPARING STRINGS

THE OPERATORS <,>,=ETC HAVE BEEN OVERLOADED, SO THAT THEY CAN COMPARE `string` OBJECTS!

THIS MEANS THAT WE CAN COMPARE STRINGS EXACTLY AS WE WOULD NUMBERS

EXAMPLE 17: COMPARING STRINGS

THE OPERATORS <,>,=ETC HAVE BEEN OVERLOADED, SO THAT THEY CAN COMPARE `string` OBJECTS!

THIS MEANS THAT WE CAN COMPARE STRINGS EXACTLY AS WE WOULD NUMBERS

WE WILL HAVE AN ENTIRE SECTION ON OPERATOR OVERLOADING

BUT FOR NOW JUST REMEMBER WHAT IT IS - AND THAT IT IS POSSIBLE

EXAMPLE 17: COMPARING STRINGS

```
string firstString;
string secondString;

while (true)
  {
    cout << "Please enter string 1 (empty string to exit)" << endl;
    getline(cin,firstString);
    if (firstString == "")
      {
       cout << "You entered an empty string - so exiting" <<  endl;
        break;
      }
    cout << "Please enter string 2" << endl;
    getline(cin,secondString);
    if(firstString < secondString)
    {
      cout << "First string " << firstString << " is lexicographically less than Second string " << secondString << endl;
    }
    else if(firstString > secondString)
    {
        cout << "First string " << firstString
             << " is lexicographically greater than than Second string " << secondString << endl;
    }
    else
      {
        cout <<  "Strings are equal!" << endl;
      }


  }
```

COMPARING STRINGS

EXAMPLE 17: COMPARING STRINGS

```
string firstString;
string secondString;

while (true)
  {
    cout << "Please enter string 1 (empty string to exit)" << endl;
    getline(cin,firstString);
    if (firstString == "")
      {
       cout << "You entered an empty string - so exiting" <<  endl;
        break;
      }
    cout << "Please enter string 2" << endl;
    getline(cin,secondString);
    if(firstString < secondString)
    {
      cout << "First string " << firstString << " is lexicographically less than Second string " << secondString << endl;
    }
    else if(firstString > secondString)
    {
        cout << "First string " << firstString
             << " is lexicographically greater than than Second string " << secondString << endl;
    }
    else
      {
       cout <<  "Strings are equal!" << endl;
      }


  }
```

COMPARING STRINGS

EXAMPLE 17: COMPARING STRINGS

```
string firstString;
string secondString;

while(true)
  {
    cout << "Please enter string 1 (empty string to exit)" << endl;
    getline(cin,firstString);
    if (firstString == "")
      {
       cout << "You entered an empty string - so exiting" <<  endl;
        break;
      }
    cout << "Please enter string 2" << endl;
    getline(cin,secondString);
    if(firstString < secondString)
    {
      cout << "First string " << firstString << " is lexicographically less than Second string " << secondString << endl;
    }
    else if(firstString > secondString)
    {
        cout << "First string " << firstString
             << " is lexicographically greater than than Second string " << secondString << endl;
    }
    else
      {
       cout <<  "Strings are equal!" << endl;
      }


  }
```

COMPARING STRINGS

EXAMPLE 17: COMPARING STRINGS

```
string firstString;
string secondString;

while(true)
  {
    cout << "Please enter string 1 (empty string to exit)" << endl;
    getline(cin,firstString);
    if (firstString == "")
      {
       cout << "You entered an empty string - so exiting" <<  endl;
        break;
      }
    cout << "Please enter string 2" << endl;
    getline(cin,secondString);
    if(firstString < secondString)
    {
      cout << "First string " << firstString << " is lexicographically less than Second string " << secondString << endl;
    }
    else if(firstString > secondString)
    {
        cout << "First string " << firstString
             << " is lexicographically greater than than Second string " << secondString << endl;
    }
    else
      {
        cout <<  "Strings are equal!" << endl;
      }


  }
```

COMPARING STRINGS

EXAMPLE 17: COMPARING STRINGS

```
string firstString;
string secondString;

while(true)
  {
    cout << "Please enter string 1 (empty string to exit)" << endl;
    getline(cin,firstString);
    if (firstString == "")
      {
       cout << "You entered an empty string - so exiting" <<  endl;
        break;
      }
    cout << "Please enter string 2" << endl;
    getline(cin,secondString);
    if(firstString < secondString)
    {
      cout << "First string " << firstString << " is lexicographically less than Second string " << secondString << endl;
    }
    else if(firstString > secondString)
    {
        cout << "First string " << firstString
             << " is lexicographically greater than than Second string " << secondString << endl;
    }
    else
      {
        cout <<  "Strings are equal!" << endl;
      }


  }
```

COMPARING STRINGS

EXAMPLE 17: COMPARING STRINGS

```
if(firstString < secondString)
{
  cout << "First string " << firstString << " is lexicographically less than Second string " << secondString << endl;
}
else if(firstString > secondString)
{
    cout << "First string " << firstString
         << " is lexicographically greater than than Second string " << secondString << endl;
}
else
  {
    cout <<  "Strings are equal!" << endl;
  }
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please enter string 1 (empty string to exit)
A
Please enter string 2
B
First string A is lexicographically less than Second string B
Please enter string 1 (empty string to exit)
B
Please enter string 2
A
First string B is lexicographically greater thans than Second string A
Please enter string 1 (empty string to exit)
The
Please enter string 2
The
Strings are equal!
Please enter string 1 (empty string to exit)

You entered an empty string - so exiting
```

EXAMPLE 17: COMPARING STRINGS

```
if(firstString < secondString)
{
  cout << "First string " << firstString << " is lexicographically less than Second string " << secondString << endl;
}
else if(firstString > secondString)
{
    cout << "First string " << firstString
         << " is lexicographically greater than than Second string " << secondString << endl;
}
else
  {
    cout <<  "Strings are equal!" << endl;
  }
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please enter string 1 (empty string to exit)
A
Please enter string 2
B
First string A is lexicographically less than Second string B
Please enter string 1 (empty string to exit)
B
Please enter string 2
A
First string B is lexicographically greater thans than Second string A
Please enter string 1 (empty string to exit)
The
Please enter string 2
The
Strings are equal!
Please enter string 1 (empty string to exit)

You entered an empty string - so exiting
```

EXAMPLE 17: COMPARING STRINGS

```
if(firstString < secondString)
{
  cout << "First string " << firstString << " is lexicographically less than Second string " << secondString << endl;
}
else if(firstString > secondString)
{
    cout << "First string " << firstString
         << " is lexicographically greater than than Second string " << secondString << endl;
}
else
  {
    cout <<  "Strings are equal!" << endl;
  }
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Please enter string 1 (empty string to exit)
A
Please enter string 2
B
First string A is lexicographically less than Second string B
Please enter string 1 (empty string to exit)
B
Please enter string 2
A
First string B is lexicographically greater thans than Second string A
Please enter string 1 (empty string to exit)
The
Please enter string 2
The
Strings are equal!
Please enter string 1 (empty string to exit)

You entered an empty string - so exiting
```

EXAMPLE 18:

CONVERT C++ STRINGS TO C STRINGS

EXAMPLE 18: CONVERT C++ STRINGS TO C STRINGS

`char*` TO `string`

THE STRING CLASS HAS A CONSTRUCTOR THAT TAKES IN A CHAR*

EXAMPLE 18: CONVERT C++ STRINGS TO C STRINGS

`string` TO `char*`

THE STRING CLASS HAS A METHOD CALLED `string::c_str()`

EXAMPLE 18: CONVERT C++ STRINGS TO C STRINGS

```
string someString ("Vitthal Srinivasan");

char * oldSchoolString = new char [someString.length()+1];
strcpy (oldSchoolString, someString.c_str());

string recreatedString(oldSchoolString);


cout << " Modern (C++) string " << someString << endl
     << " converted to C-style (char*) string " << oldSchoolString <<endl
     << " reconverted to Modern C++ string " << recreatedString << endl;

// oldSchoolString now contains a c-string copy of str
delete[] oldSchoolString;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
 Modern (C++) string Vitthal Srinivasan
 converted to C-style (char*) string Vitthal Srinivasan
 reconverted to Modern C++ string Vitthal Srinivasan
```

EXAMPLE 18: CONVERT C++ STRINGS TO C STRINGS

```
string someString ("Vitthal Srinivasan");

char * oldSchoolString = new char [someString.length()+1];
strcpy (oldSchoolString, someString.c_str());

string recreatedString(oldSchoolString);


cout << " Modern (C++) string " << someString << endl
     << " converted to C-style (char*) string " << oldSchoolString << endl
     << " reconverted to Modern C++ string " << recreatedString << endl;

// oldSchoolString now contains a c-string copy of str
delete[] oldSchoolString;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
 Modern (C++) string Vitthal Srinivasan
 converted to C-style (char*) string Vitthal Srinivasan
 reconverted to Modern C++ string Vitthal Srinivasan
```

EXAMPLE 18: CONVERT C++ STRINGS TO C STRINGS

```
string someString ("Vitthal Srinivasan");

char * oldSchoolString = new char [someString.length()+1];
strcpy (oldSchoolString, someString.c_str());

string recreatedString(oldSchoolString);


cout << " Modern (C++) string " << someString << endl
     << " converted to C-style (char*) string " << oldSchoolString << endl
     << " reconverted to Modern C++ string " << recreatedString << endl;

// oldSchoolString now contains a c-string copy of str
delete[] oldSchoolString;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
 Modern (C++) string Vitthal Srinivasan
 converted to C-style (char*) string Vitthal Srinivasan
 reconverted to Modern C++ string Vitthal Srinivasan
```

EXAMPLE 18: CONVERT C++ STRINGS TO C STRINGS

```
string someString ("Vitthal Srinivasan");

char * oldSchoolString = new char [someString.length()+1];
strcpy (oldSchoolString, someString.c_str());

string recreatedString(oldSchoolString);


cout << " Modern (C++) string " << someString << endl
     << " converted to C-style (char*) string " << oldSchoolString << endl
     << " reconverted to Modern C++ string " << recreatedString << endl;

// oldSchoolString now contains a c-string copy of str
delete[] oldSchoolString;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
 Modern (C++) string Vitthal Srinivasan
 converted to C-style (char*) string Vitthal Srinivasan
 reconverted to Modern C++ string Vitthal Srinivasan
```

EXAMPLE 18: CONVERT C++ STRINGS TO C STRINGS

```
string someString ("Vitthal Srinivasan");

char * oldSchoolString = new char [someString.length()+1];
strcpy (oldSchoolString, someString.c_str());

string recreatedString(oldSchoolString);

cout << " Modern (C++) string " << someString << endl
     << " converted to C-style (char*) string " << oldSchoolString << endl
     << " reconverted to Modern C++ string " << recreatedString << endl;

// oldSchoolString now contains a c-string copy of str
delete[] oldSchoolString;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
 Modern (C++) string Vitthal Srinivasan
 converted to C-style (char*) string Vitthal Srinivasan
 reconverted to Modern C++ string Vitthal Srinivasan
```

EXAMPLE 18: CONVERT C++ STRINGS TO C STRINGS

```
string someString ("Vitthal Srinivasan");

char * oldSchoolString = new char [someString.length()+1];
strcpy (oldSchoolString, someString.c_str());

string recreatedString(oldSchoolString);

cout << " Modern (C++) string " << someString << endl
     << " converted to C-style (char*) string " << oldSchoolString << endl
     << " reconverted to Modern C++ string " << recreatedString << endl;

// oldSchoolString now contains a c-string copy of str
delete[] oldSchoolString;
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
 Modern (C++) string Vitthal Srinivasan
 converted to C-style (char*) string Vitthal Srinivasan
 reconverted to Modern C++ string Vitthal Srinivasan
```