Data structure (2) in C++

C++ Strings

C++ Strings

Strings are used for storing text.

A `string` variable contains a collection of characters surrounded by double quotes:

Example

Create a variable of type `string` and assign it a value:

```
string greeting = "Hello";
```

To use strings, you must include an additional header file in the source code, the `<string>` library:

Example

```
// Include the string library
#include <string>

// Create a string variable
string greeting = "Hello";
```

C++ String Concatenation

String Concatenation

The `+` operator can be used between strings to add them together to make a new string. This is called **concatenation**:

Data structure (2) in C++

Example

```
string firstName = "John ";
string lastName = "Doe";
string fullName = firstName + lastName;
cout << fullName;
```

In the example above, we added a space after firstName to create a space between John and Doe on output. However, you could also add a space with quotes (`" "` or `' '`):

Example

```
string firstName = "John";
string lastName = "Doe";
string fullName = firstName + " " + lastName;
cout << fullName;
```

Append

A string in C++ is actually an object, which contain functions that can perform certain operations on strings. For example, you can also concatenate strings with the `append()` function:

Example

```
string firstName = "John ";
string lastName = "Doe";
string fullName = firstName.append(lastName);
cout << fullName;
```

Data structure (2) in C++

C++ Numbers and Strings

Adding Numbers and Strings

WARNING!

C++ uses the + operator for both **addition** and **concatenation**.

Numbers are added. Strings are concatenated

Example

```
int x = 10;
int y = 20;
int z = x + y;      // z will be 30 (an integer)
```

If you add two strings, the result will be a string concatenation:

Example

```
string x = "10";
string y = "20";
string z = x + y;   // z will be 1020 (a string)
```

If you try to add a number to a string, an error occurs:

Example

```
string x = "10";
int y = 20;
string z = x + y;
```

Data structure (2) in C++

C++ String Length

String Length

To get the length of a string, use the `length()` function:

Example

```
string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
cout << "The length of the txt string is: " << txt.length();
```

Tip: You might see some C++ programs that use the `size()` function to get the length of a string. This is just an alias of `length()`. It is completely up to you if you want to use `length()` or `size()`:

Example

```
string txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
cout << "The length of the txt string is: " << txt.size();
```

C++ Access Strings

Access Strings

You can access the characters in a string by referring to its index number inside square brackets `[]`.

This example prints the **first character** in **myString**:

Example

```
string myString = "Hello";
cout << myString[0];
// Outputs H
```

Data structure (2) in C++

Note: String indexes start with 0: [0] is the first character. [1] is the second character, etc

This example prints the **second character** in **myString**:

Example

```
string myString = "Hello";
cout << myString[1];
// Outputs e
```

Change String Characters

To change the value of a specific character in a string, refer to the index number, and use single quotes:

Example

```
string myString = "Hello";
myString[0] = 'J';
cout << myString;
// Outputs Jello instead of Hello
```

C++ Special Characters

Because strings must be written within quotes, C++ will misunderstand this string, and generate an error:

```
string txt = "We are the so-called "Vikings" from the north.
```

The solution to avoid this problem, is to use the **backslash escape character**.

Data structure (2) in C++

The backslash (\) escape character turns special characters into string characters:

Escape character	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

The sequence `\"` inserts a double quote in a string:

Example

```
string txt = "We are the so-called \"Vikings\" from the north.";
```

```
We are the so-called "Vikings" from the north.
```

The sequence `\'` inserts a single quote in a string:

Example

```
string txt = "It\'s alright.";
```

```
It's alright.
```

The sequence `\\` inserts a single backslash in a string:

Example

```
string txt = "The character \\ is called backslash.";
```

```
The character \ is called backslash.
```

Data structure (2) in C++

C++ User Input Strings

It is possible to use the extraction operator `>>` on `cin` to display a string entered by a user:

Example

```
string firstName;
cout << "Type your first name: ";
cin >> firstName; // get user input from the keyboard
cout << "Your name is: " << firstName;

// Type your first name: John
// Your name is: John
```

However, `cin` considers a space (whitespace, tabs, etc) as a terminating character, which means that it can only display a single word (even if you type many words):

Example

```
string fullName;
cout << "Type your full name: ";
cin >> fullName;
cout << "Your name is: " << fullName;

// Type your full name: John Doe
// Your name is: John
```

error

From the example above, you would expect the program to print "John Doe", but it only prints "John".

That's why, when working with strings, we often use the `getline()` function to read a line of text. It takes `cin` as the first parameter, and the string variable as second:

Data structure (2) in C++

Example

```
string fullName;
cout << "Type your full name: ";
getline (cin, fullName);
cout << "Your name is: " << fullName;

// Type your full name: John Doe
// Your name is: John Doe
```

C++ String Namespace

Omitting Namespace

You might see some C++ programs that runs without the standard namespace library. The `using namespace std` line can be omitted and replaced with the `std` keyword, followed by the `::` operator for `string` (and `cout`) objects:

Example

```
#include <iostream>
#include <string>

int main() {
  std::string greeting = "Hello";
  std::cout << greeting;
  return 0;
}
```

String is immutable

Data structure (2) in C++