Programming challenges in C++

5. Strings and output format

Nacho Iborra

IES San Vicente

Table of Contents

Programming challenges in C++

1.Strings manipulation....3
1.1.Try yourself: Quinto milenio....4
2.Formatting output....5
2.1.Some format examples....5
2.2.Try yourself: ¿Cuál es la siguiente matrícula?....6

1. Strings manipulation

If we want to work with strings in C++, we have the *string* data type, from the standard namespace. Besides, there are some useful functions that we can use (convert to upper or lower case, get a substring, find a text...). For some of them, we may need to include the *string* library.

We have already seen that we can get a particular word from the input with *cin*, or the whole line with *getline*.

- We can **concatenate** strings with the '+' operator, or in some cases with '+=' operator (if we want to add a string at the end of another).
- We can also treat the string as a **char array** (as in C#), and get to each character with the corresponding index. If we want to get the size or length of the string, we have the ***length*** method:

```
string text = "Hello world";

for (int i = 0; i < text.length(); i++)
    cout << text[i];
```

- We can compare two strings either by using == or != operators (which will tell us if strings are the same or not), or by using the ***compare*** method. This method returns a negative number if first string is lower, 0 if both strings are the same, and 1 if first string is greater.

```
if (text1.compare(text2) < 0)
  cout << "Second text is greater";
```

  We can also use the >, <, >=, <= comparators to see which is greater or lower.

- We can convert any char of the string to upper or lower case with ***toupper*** and ***tolower*** functions:

```
string text = "Hello world");
text[6] = toupper(text[6]);
```

- We can find a text in a string with the ***find*** method. It can get either a string or a single char as a parameter, and it returns the position where the first occurence of the text has been found, or *string::npos* if text has not been found in the string. We can also specify a second parameter to determine the string index from which we must start searching for the given text.

```
int pos = text.find("lo");
if (pos != string::npos)
  cout << "Found at " << pos;
```

- We can get a substring of a given string with the ***substr*** method. It has two parameters: the index from which we must start getting the substring (starting by 0), and how many characters we must take. If this second parameter is omitted, it returns the resulting string from the initial index to the end of the string.

```
string subtext = text.substr(3, 4);
```

- We can erase a substring from a given string with the ***erase*** method. It takes the same two parameters than *substr* method (initial position and number of characters to be erased)

- We can replace a substring with another one with the ***replace*** method. It has three parameters: the initial position to replace, the number of characters to replace, and the new text to put in this position. Lengths of old and new texts can be different (the whole string would shrink or expand in this case).

```
text.replace(6, 4, "everybody");
```

- There is no straightforward way to do some operations, such as splitting a string given a delimiter, or automatically join some substrings with a given delimiter between them.

1.1. Try yourself: Quinto milenio

Let's face this challenge to process strings and characters from a string.

2. Formatting output

Apart from traditional `cout` instruction to print data, we can use some other options if we want this data to have a given output format. To do this, we can use `printf` instruction from C programming language (we can use any C instruction within a C++ program). It has a variable number of parameters, and the first of all is the string to be printed out. Then, this string can have some special characters inside it, which determine the data types that must replace these characters. For instance, if we use this instruction:

```
printf("The number is %d", number);
```

then the symbol *%d* will be replaced by the variable `number`, and this variable must be an integer (this is what *%d* means).

There are some other symbols to represent different data types. Here are some of them:

- *%d* for integer types (long, int)
- *%f* for real types (float and double)
- *%s* for strings
- *%c* for chars

We can place as many symbols as we want inside the output string, and then we will need to add the corresponding number of parameters at the end of the `printf` instruction. For instance:

```
printf("The average of %d and %d is %f", number1, number2, average);
```

2.1. Some format examples

Besides the primary simbols %d and %f, we can add some other information between the '%' and the letter, that specify some format information.

2.1.1. Specifying integer digits

For instance, if we want to output an integer with a given number of digits, we can do it this way:

```
printf("The number is %05d", number);
```

where the `05` means that the integer is going to have, at least, 5 digits, and if there are not enough digits in the number, then it will be filled with zeros. The output of this instruction if `number` is 33 would be `The number is 00033`. If you don't put the 0, then the number will be filled with whitespaces. So if we have this instruction:

```
printf("The number is %10d", number);
```

and if `number` is 33, it would produce the following output:

```
The number is         33.
```

2.1.2. Specifying decimal digits

In the same way that we format integer numbers, we can format real numbers. We can use the same pattern seen before to specify the total number of integer digits:

```
printf("The number is %3f", number);
```

But, besides, we can specify the total number of decimal digits by adding a point and the total number desired, this way:

```
printf("The number is %3.3f", number);
```

Then, if `number` is 3.14159, the output would be `The number is  3.142`.

2.2. Try yourself: ¿Cuál es la siguiente matrícula?

To practise with strings and formats, you can try this challenge from Acepta el Reto.