Console displays wrong characters

DavidIRE (16) Mar 25, 2014 at 9:32pm

Hi,

I am using Microsoft Visual C++ 2010 Express in Windows 7 Ultimate 64 bits.
I have created an empty project and written this program:

```
1  #include <iostream>               // Object cout, manipulator endl
2
3  using std::cout;
4  using std::endl;
5
6  int main()
7  {
8       cout << "ñ" << endl;
9       cout << "á" << endl;
10      cout << "é" << endl;
11      cout << "í" << endl;
12      cout << "ó" << endl;
13      cout << "ú" << endl;
14      return 0;
15 }
```

And I got the next output in the console:

```
±
ß
Ú
Ý
¾
·
```

Why does the console display this wrong characters and not the characters I wrote in the source file?

Thanks.

Little Bobby Tables (997) Mar 25, 2014 at 10:04pm

hmmm i think those are all ascii characters but i could be wrong. http://www.cplusplus.com/reference/locale/ could be worth looking into. you might also have to use utf chars

DavidIRE (16) Mar 27, 2014 at 7:18pm

Well, I already have solved it.

The problem was about character encodings. String literals and character literals (this is, the hard-coded text in the source file) are encoded using the character encoding utilized to encoding the source file. The console uses its own character encoding to display the characters that are sended to it. If the character encodings used by the source file and the console are not the same, maybe there will be wrong characters in the output printed by the console.

In my case, the source file is encoded using the Windows-1252 character encoding and the console uses the MS-DOS Latin 1 character encoding.
The codes (in decimal) of the involved characters in this two character encodings are the next:

Character	Windows-1252	MS-DOS Latin 1
ñ	241	164
á	225	160
é	233	130
í	237	161
ó	243	162
ú	250	163

As you can see, all the codes differ between this two character encodings, therefore there are wrong characters in the output.

What are happening?

For example, in

```
cout << "ñ" << endl;
```

The character 'ñ' is coded using Windows-1252 character encoding, therefore its value (in decimal) is 241. The value 241 is written in the standard output (the console) which uses the MS-DOS Latin 1 character encoding to display the character. Since in the MS-DOS Latin 1 character encoding the value 241 is mapped to the character '±', the console

displays the character '±'. And this is a wrong character (not the same hard-coded character in the source file, 'ñ').

With the rest of characters ocurrs something similar. The codes of the involved characters in Windows-1252 have the next characters mapped in MS-DOS Latin 1:

```
Code (in decimal)     Character (in Windows-1252)     Character (in MS-DOS Latin 1)
       241                      ñ                               ±
       225                      á                               ß
       233                      é                               Ú
       237                      í                               Ý
       243                      ó                               ¾
       250                      ú                               ·
```

As you can see, this characters (of the third column) are the characters displayed by the console in my application.

To solve this problem, it is necesary to change the character encoding used to encode the source files in order that it be the same character encoding used by the console, or to change the character encoding used by the console in order that it be the same character encoding used to encode the source files. The first method is laborious and slow and, in addition, the character encoding used by the console may change from one machine to another (being necesary to encode again all the source files and compile again all the aplication). The second method is the best solution.

In Windows, the character encoding is named code page and each code page have a number that identify it (the number is the id of the code page); in addition, the console have two character encodings: one is used to translate the bits written in the standard output (using functions like the insertion operator << over the object cout) into the corresponding characters displayed in the console window (this is the output code page) and the other is used to translate characters read from the standard input (using functions like the extraction operator >> over the object cin) into the corresponding bits (this is the input code page).

The id of the code page for the character encoding Windows-1252 is 1252.
The id of the code page for the character encoding MS-DOS Latin 1 is 850.

In Windows, there are several functions (declared in the header file windows.h) that permit to get and set the output and input code pages used by the console:

```
1  // Return the number of the output code page used currently by the console
2  UINT GetConsoleOutputCP();
3
4  // Return the number of the intput code page used currently by the console
5  UINT GetConsoleCP();
6
7  // Set the output code page used by the console.
8  // "codePageID" is the id of the code page to set
9  BOOL SetConsoleOutputCP(UINT codePageID);
10
11 // Set the input code page used by the console.
12 // "codePageID" is the id of the code page to set
13 BOOL SetConsoleCP(UINT codePageID);
```

The output code page and the input code page must be the same to avoid wrong characters be displayed in the console: if a character is read in one character encoding but it is written in other different character encoding, the character displayed may be wrong.

So the solution is this:

The source file (encoded using the Windows-1252 character encoding):

```
1  #include <iostream>               // Object cout, manipulator endl
2
3  using std::cout;
4  using std::endl;
5
6  int main()
7  {
8       SetConsoleOutputCP(1252);
9       SetConsoleCP(1252);
10      cout << "ñ" << endl;
11      cout << "á" << endl;
12      cout << "é" << endl;
13      cout << "í" << endl;
14      cout << "ó" << endl;
15      cout << "ú" << endl;
16      return 0;
17 }
```

And the output of the application (this time, correct):

```
ñ
á
é
í
ó
ú
```

See you.

Last edited on Mar 27, 2014 at 7:36pm

Topic archived. No new replies allowed.