

Re: Working with strings in c++

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2007-02/msg00170.html>

- *From:* "John Carson" <jcarson_n_o_sp_am_@xxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 9 Feb 2007 02:53:11 +1100
-

"Vladimir Grigoriev" <vlad.moscow@xxxxxxx> wrote in message
<news:%23K7IUc5SHHA.412@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

"John Carson" <jcarson_n_o_sp_am_@xxxxxxxxxxxxxxxxxx> wrote in message
<news:%23YjI6R4SHHA.2212@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

Is not the two declaration above the same?
For example what is the difference between
int i = 10;
and
int i(10);

No difference when you are talking about built in types. When talking about user-defined types, however, there is a difference.

```
const CString myStr = "val1;val2;val3";
```

is supposed to involve two steps:

1. The creation of a temporary CString object initialised by "val1;val2;val3"
2. The use of a copy constructor to initialise myStr from the temporary CString object.

By contrast,

```
const CString myStr("val1;val2;val3");
```

involves a single step: the calling of the CString constructor to initialise myStr.

You are wrong! In both cases the single CString constructor with parameters is called. Neither copy constructor is called. I think you can test this yourself with a simple test program.

Try reading the answer.

"as an optimisation, the compiler is allowed (but not obliged) to replace the two steps involved in

```
const CString myStr = "val1;val2;val3";
```

with a single call to the myStr constructor."

That is exactly what VC++ does. Nevertheless, your code should allow the compiler to do it the two-step way, or the compiler is allowed to refuse to compile the code --- even if it would optimise away the two steps were those two steps possible.

To illustrate Paul's point, try compiling this:

```
#include <cstring>
using namespace std;

class MyString
{
char *str;
public:
explicit MyString(const char* arg)
{
str = new char[strlen(arg)+1];
strcpy(str, arg);
}
MyString(const MyString& rhs)
{
delete[] str;
str = new char[strlen(rhs.str)+1];
strcpy(str, rhs.str);
}
};

int main()
{
MyString ms2 = "test";
return 0;
}
```

To illustrate my point, try compiling this using Comeau online

<http://www.comeaucomputing.com/tryitout/>

```
#include <cstring>
using namespace std;
```

```
class MyString
{
char *str;
public:
MyString(const char* arg)
{
str = new char[strlen(arg)+1];
strcpy(str, arg);
}
private:
MyString(const MyString& rhs)
{
delete[] str;
str = new char[strlen(rhs.str)+1];
strcpy(str, rhs.str);
}
};
```

```
int main()
{
MyString ms2 = "test";
return 0;
}
```

—
John Carson

.