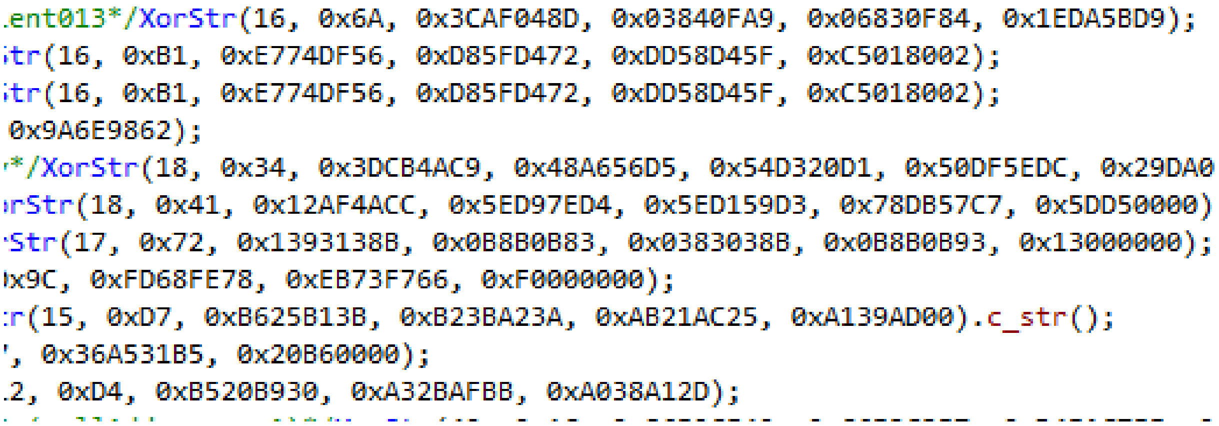
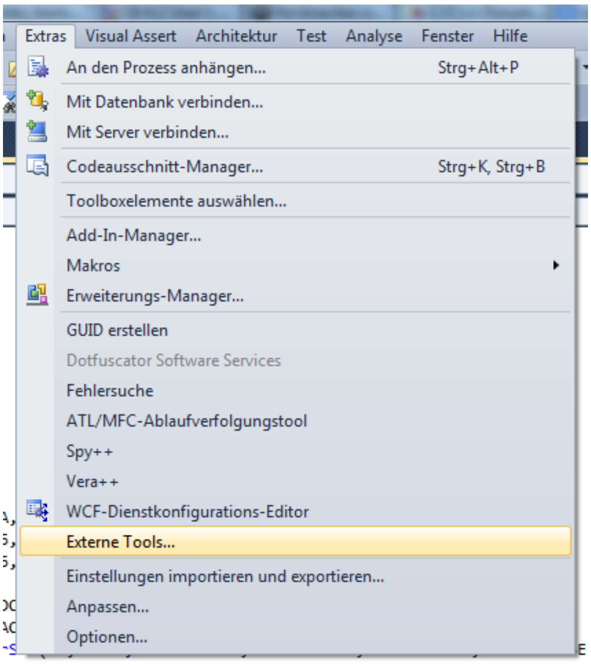
OldSchoolHack XorPreProcess

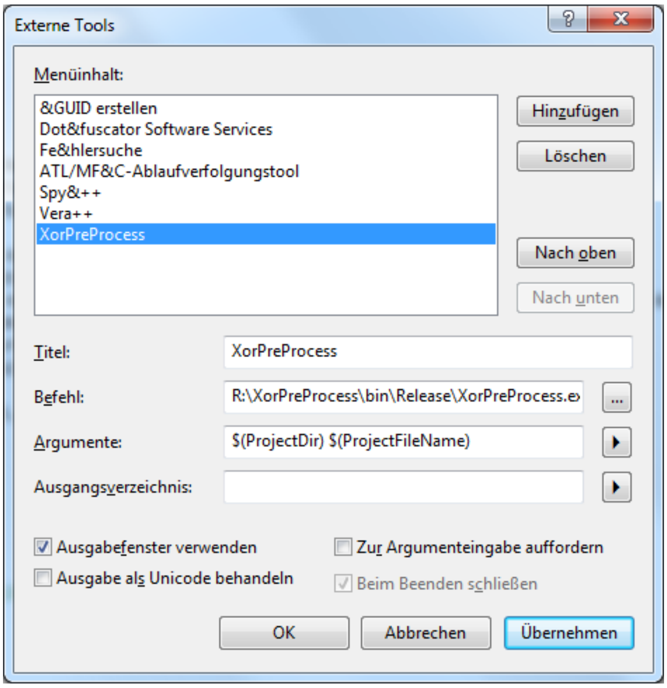


```
.ent013*/XorStr(16, 0x6A, 0x3CAF048D, 0x03840FA9, 0x06830F84, 0x1EDA5BD9);
itr(16, 0xB1, 0xE774DF56, 0xD85FD472, 0xDD58D45F, 0xC5018002);
itr(16, 0xB1, 0xE774DF56, 0xD85FD472, 0xDD58D45F, 0xC5018002);
 0x9A6E9862);
'*/XorStr(18, 0x34, 0x3DCB4AC9, 0x48A656D5, 0x54D320D1, 0x50DF5EDC, 0x29DA0
rStr(18, 0x41, 0x12AF4ACC, 0x5ED97ED4, 0x5ED159D3, 0x78DB57C7, 0x5DD50000)
Str(17, 0x72, 0x1393138B, 0x0B8B0B83, 0x0383038B, 0x0B8B0B93, 0x13000000);
x9C, 0xFD68FE78, 0xEB73F766, 0xF0000000);
r(15, 0xD7, 0xB625B13B, 0xB23BA23A, 0xAB21AC25, 0xA139AD00).c_str();
', 0x36A531B5, 0x20B60000);
.2, 0xD4, 0xB520B930, 0xA32BAFBB, 0xA038A12D);
```

1. Unpack XorPreProcess.rar.
2. Choose **External Tools…** in the Tools menu.


Extras
Visual Assert
Architektur
Test
Analyse
Fenster
Hilfe
An den Prozess anhängen...
Strg+Alt+P
Mit Datenbank verbinden...
Mit Server verbinden...
Codeausschnitt-Manager...
Strg+K, Strg+B
Toolboxelemente auswählen...
Add-In-Manager...
Makros
Erweiterungs-Manager...
GUID erstellen
Dotfuscator Software Services
Fehlersuche
ATL/MFC-Ablaufverfolgungstool
Spy++
Vera++
WCF-Dienstkonfigurations-Editor
Externe Tools...
Einstellungen importieren und exportieren...
Anpassen...
Optionen...

3. Type in a Title, the path to XorPreProcess.exe and select **Use Output Window**.



Menüinhalt:

&GUID erstellen
Dot&fuscator Software Services
Fe&hlersuche
ATL/MF&C-Ablaufverfolgungstool
Spy&++
Vera++
XorPreProcess

Titel: XorPreProcess

Befehl: R:\XorPreProcess\bin\Release\XorPreProcess.e

Argumente: $(ProjectDir) $(ProjectFileName)

Ausgangsverzeichnis:

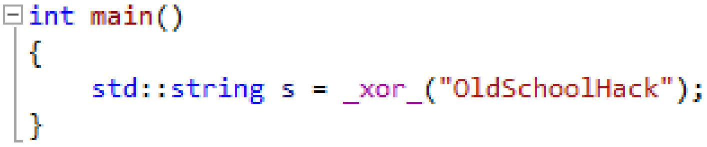
☑ Ausgabefenster verwenden ☐ Zur Argumenteingabe auffordern

☐ Ausgabe als Unicode behandeln ☑ Beim Beenden schließen

4. First parameter must be **$(ProjectDir)**. Second parameter must be **$(ProjectFileName)**. Third parameter is optional and can be **refresh**. With refresh all XorStrings will be newly generated.

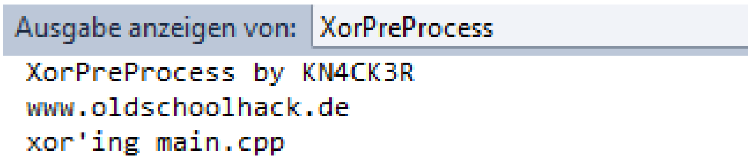
   copy-paste:
   $(ProjectDir) $(ProjectFileName)
   $(ProjectDir) $(ProjectFileName) refresh

5. In your project include the **XorStr.hpp** header.
6. You can use it in your code like this:



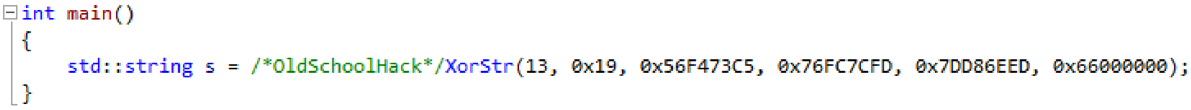
```
int main()
{
    std::string s = _xor_("OldSchoolHack");
}
```

   Supported escaped characters are:
   \r, \n, \t, \x00 - \xFF
7. Save all files and run XorPreProcess from the Tools Menu.



```
XorPreProcess by KN4CK3R
www.oldschoolhack.de
xor'ing main.cpp
```

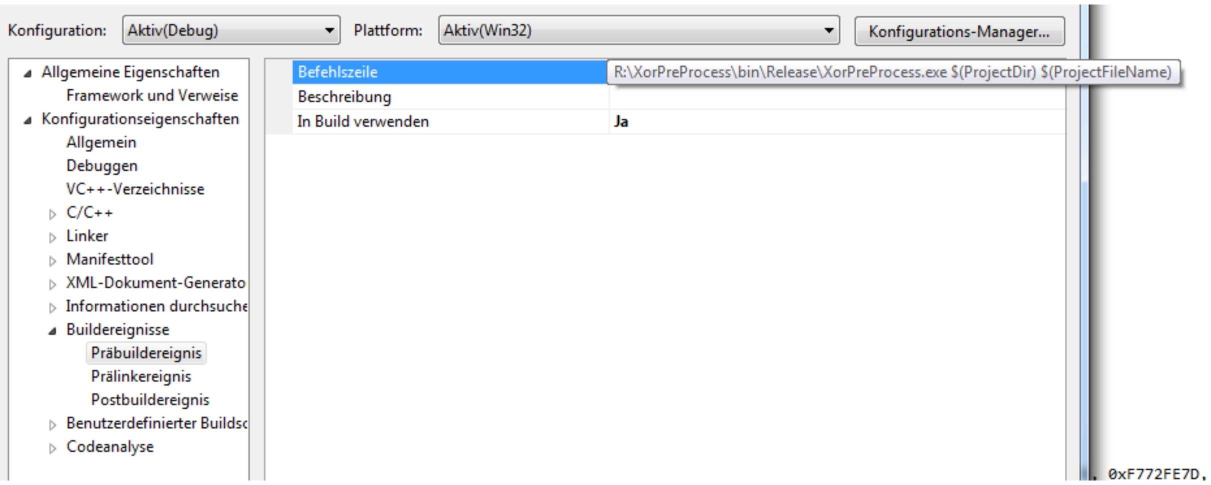
8. VisualStudio asks if it should reload changed files. Click Yes.
9. Your code has been changed and the string is now xor’ed.



```
int main()
{
    std::string s = /*OldSchoolHack*/XorStr(13, 0x19, 0x56F473C5, 0x76FC7CFD, 0x7DD86EED, 0x66000000);
}
```

VisualStudio can run XorPreProcess for you with every Build you make if you create a PreBuildEvent.

1. Open the **Project Properties** and select **Build Events Pre-Build Event.**



Konfiguration: Aktiv(Debug)
Plattform: Aktiv(Win32)

Befehlszeile	R:\XorPreProcess\bin\Release\XorPreProcess.exe $(ProjectDir) $(ProjectFileName)
Beschreibung	
In Build verwenden	**Ja**

2. Insert the path and the parameter as in the External Tools form. (see screenshot)

Have fun xor'ing.

KN4CK3R

http://www.oldschoolhack.de

```
//create a dummy XorStr function for IntelliSense
class XorStr;
XorStr _xor_(const char *x);

class XorStr
{
public:
    XorStr(int length, int key, ...)
    {
        data.reserve(length);

        int xor = key;
        int index = 0;
        int loops = length % 4 == 0 ? length / 4 : length / 4 + 1;

        va_list vl;
        va_start(vl, key);
        for (int i = 0; i < loops; ++i)
        {
            int hash = va_arg(vl, int);

            for (int j = 0; j < 4; j++)
            {
                WORD word = HIWORD(hash);

                if (j >= 2)
                {
                    word = LOWORD(hash);
                }

                switch (j)
                {
                case 0:
                case 2:
                    data += HIBYTE(word) ^ xor;
                    break;
                case 1:
                case 3:
                    data += LOBYTE(word) ^ xor;
                    break;
                }
                xor += 127;
                xor %= 256;

                ++index;

                if (index >= length)
                {
                    break;
                }
            }
        }
        va_end(vl);
    }

    operator const std::string&() const
    {
        return data;
    }

    const char* c_str() const
    {
        return data.c_str();
    }
private:
	std::string data;
};
```