Computing I – Programming Labs

Problem Set 5 - Using Bitwise Operators

Problem 1 (Shifting and Printing an Integer)

Write a program that right-shifts an integer variable 4 bits. The program should print the integer in bits before and after the shift operation. A function to display bits was presented in class. Does your system place zeros or ones in the vacated bits?

```
Enter an integer: 8

Before right shifting 4 bits is:
      8 = 00000000 00000000 00000000 00001000
After right shifting 4 bits is:
      0 = 00000000 00000000 00000000 00000000
```

```
Enter an integer: 16

Before right shifting 4 bits is:
     16 = 00000000 00000000 00000000 00010000
After right shifting 4 bits is:
      1 = 00000000 00000000 00000000 00000001
```

```
Enter an integer: 32

Before right shifting 4 bits is:
     32 = 00000000 00000000 00000000 00100000
After right shifting 4 bits is:
      2 = 00000000 00000000 00000000 00000010
```

Problem 2 (Multiplication Via Bit Shifting)

Left-shifting an unsigned integer by 1 bit is equivalent to multiplying the value by 2. Write function `power2` that takes two integer arguments, `number` and `pow`, and calculates

number * 2^{pow}

Use a shift operator to calculate the result. The program should print the values as integers and as bits.

```
Enter two integers: 1 1

number:
      1 = 00000000 00000000 00000000 00000001

power:
      1 = 00000000 00000000 00000000 00000001

1 * 2^1 = 2
      2 = 00000000 00000000 00000000 00000010
```

```
Enter two integers: 2 2

number:
      2 = 00000000 00000000 00000000 00000010

power:
      2 = 00000000 00000000 00000000 00000010

2 * 2^2 = 8
      8 = 00000000 00000000 00000000 00001000
```

```
Enter two integers: 3 3

number:
      3 = 00000000 00000000 00000000 00000011

power:
      3 = 00000000 00000000 00000000 00000011

3 * 2^3 = 24
     24 = 00000000 00000000 00000000 00011000
```

Problem 3 (Packing Characters into Unsigned Integers)

The left-shift operator can be used to pack four character values into a four-byte unsigned integer variable. Write a program that inputs four characters from the keyboard and passes them to function

```
unsigned packCharacters(char a, char b, char c, char d).
```

To pack four characters into an unsigned integer variable, called `pack` below inside this function, assign the first character, `a`, to the unsigned variable `pack` using a `static_cast`

```
unsigned pack{static_cast<unsigned>(a)};
```

Then shift the unsigned variable, `pack`, left by 8 bit positions and combine the shifted unsigned variable with the second character, `b`, using the bitwise inclusive-OR operator. Follow this strategy for the remaining two characters. Finally, the function should return the unsigned variable `pack`.
The program should output the characters in their bit format before and after they are packed into the unsigned integer to prove that they're in fact packed correctly in the unsigned variable.

```
Enter four characters: a b c d

'a' in bits as an unsigned integer is:
     97 = 00000000 00000000 00000000 01100001
'b' in bits as an unsigned integer is:
     98 = 00000000 00000000 00000000 01100010
'c' in bits as an unsigned integer is:
     99 = 00000000 00000000 00000000 01100011
'd' in bits as an unsigned integer is:
    100 = 00000000 00000000 00000000 01100100

The characters packed in an unsigned integer:
1633837924 = 01100001 01100010 01100011 01100100
```

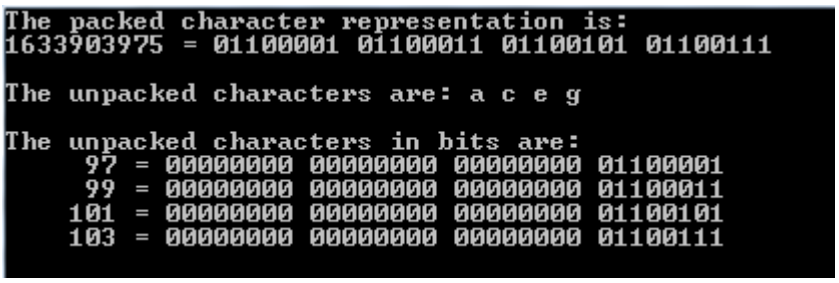
```
Enter four characters: a c e g

'a' in bits as an unsigned integer is:
     97 = 00000000 00000000 00000000 01100001
'c' in bits as an unsigned integer is:
     99 = 00000000 00000000 00000000 01100011
'e' in bits as an unsigned integer is:
    101 = 00000000 00000000 00000000 01100101
'g' in bits as an unsigned integer is:
    103 = 00000000 00000000 00000000 01100111

The characters packed in an unsigned integer:
1633903975 = 01100001 01100011 01100101 01100111
```

Problem 4 (Unpacking Characters from Unsigned Integers)

Using the right-shift operator, the bitwise AND operator and four masks, write function `unpackCharacters` that takes the second unsigned integer from Problem 3 ('a', 'c', 'e', 'g') and unpacks it into four characters. To unpack four characters from an unsigned four-byte integer, combine the unsigned integer with the masks

- 11111111 00000000 00000000 00000000 (4278190080)
- 00000000 11111111 00000000 00000000 (16711680)
- 00000000 00000000 11111111 00000000 (65280)
- 00000000 00000000 00000000 11111111 (255)

and right shift the results by 24, 16, and 8 bits, respectively. Assign the resulting value to a `char` variable using a `static_cast<char>(.)` operator. The program should print the unsigned integer in bits before it's unpacked, then print the characters in bits to confirm that they were unpacked correctly**.**

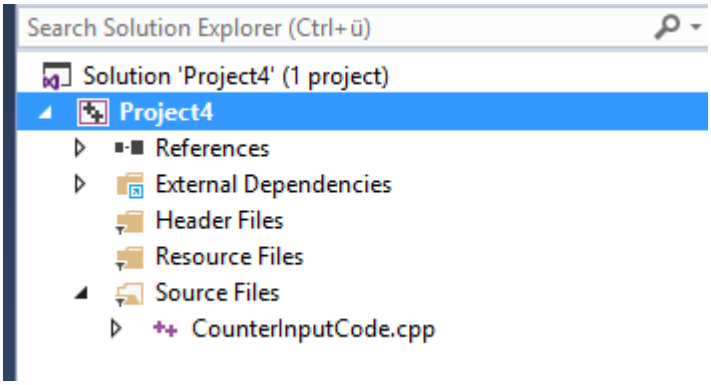


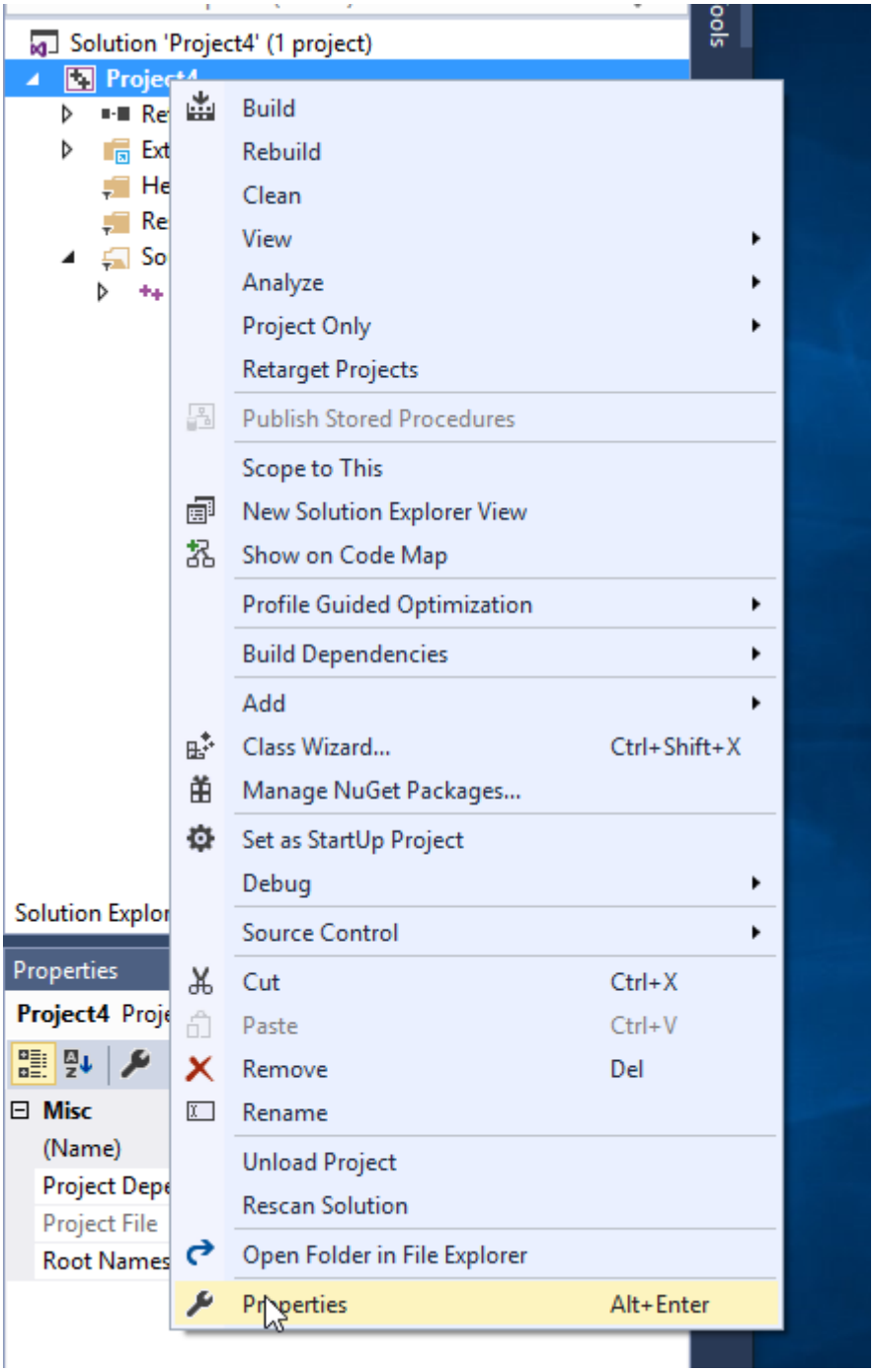
Note

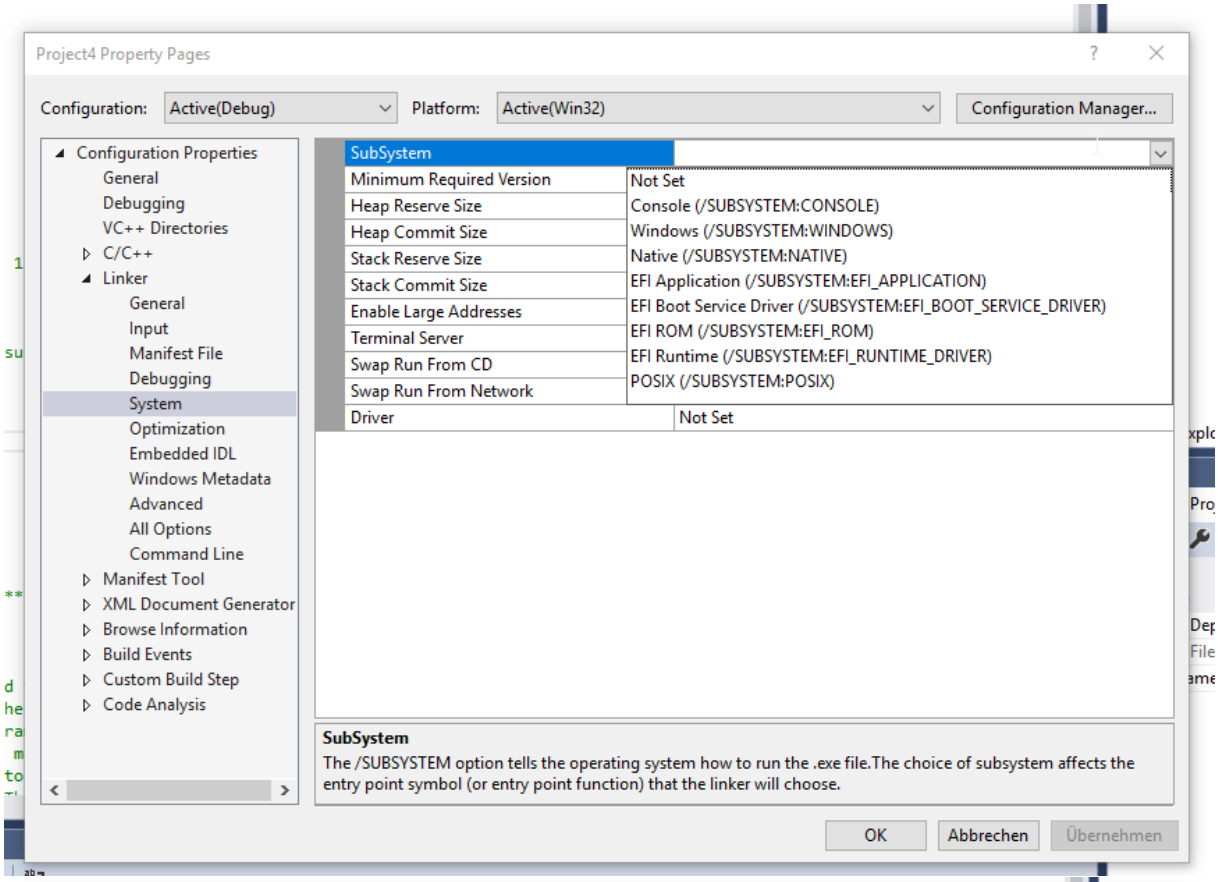
To prevent that Visual Studio closes the console immediately after a program has finished, you need to set up the command **Set Console (/SUBSYSTEM:CONSOLE)** in **Linker Options**. (https://stackoverflow.com/questions/454681/how-to-keep-the-console-window-open-in-visual-c)

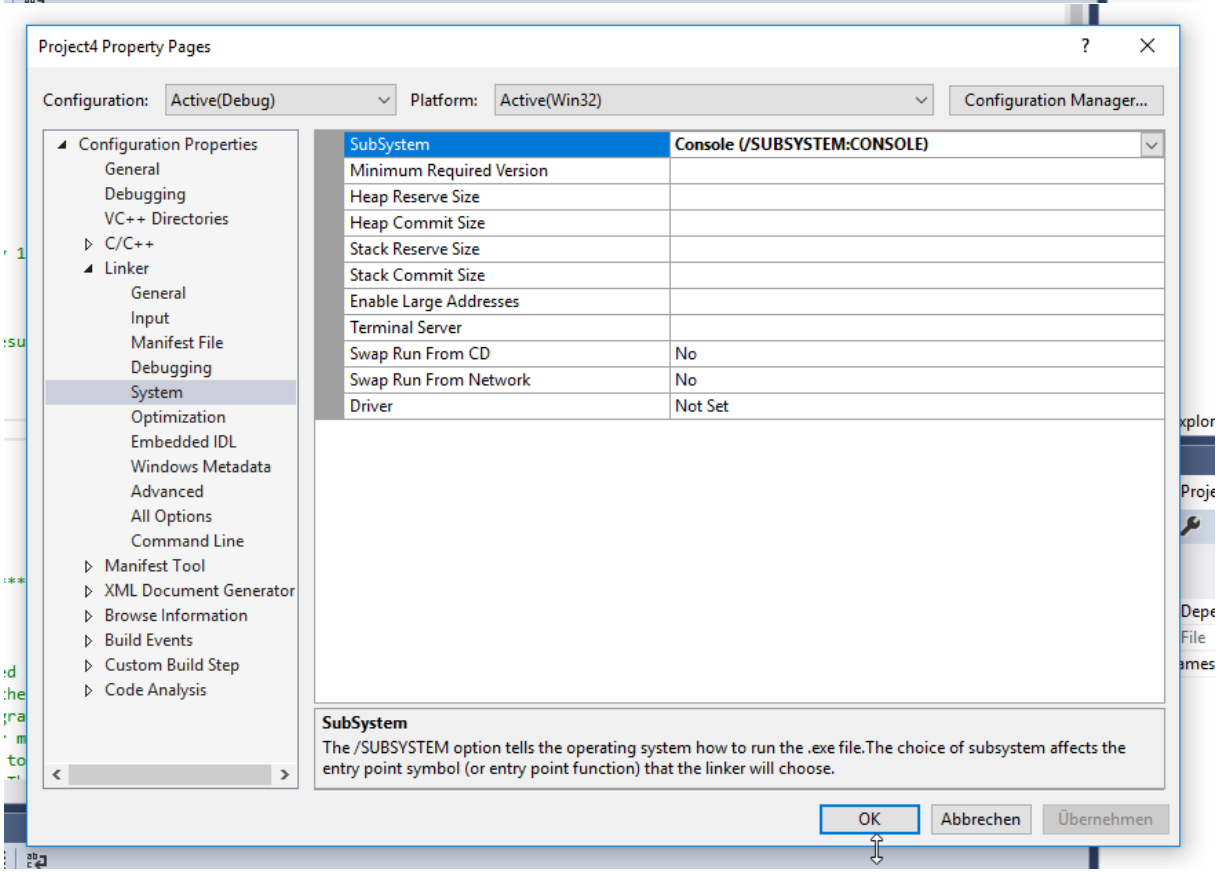
To accomplish this, proceed as follows (see also pictured displayed below):

1. Right-click on project name.
2. Select Properties from context menu.
3. Select Configuration Properties>Linker>System.
4. Click into the drop-down-box to the right and the choose "Console (/SUBSYSTEM:CONSOLE)"
5. Select Apply and then OK.




Solution 'Project4' (1 project)
Build
Rebuild
Clean
View
Analyze
Project Only
Retarget Projects
Publish Stored Procedures
Scope to This
New Solution Explorer View
Show on Code Map
Profile Guided Optimization
Build Dependencies
Add
Class Wizard... Ctrl+Shift+X
Manage NuGet Packages...
Set as StartUp Project
Debug
Source Control
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename
Unload Project
Rescan Solution
Open Folder in File Explorer
Properties Alt+Enter
Properties
Misc
(Name)
Project File
Root Names


Project4 Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
SubSystem
Minimum Required Version
Heap Reserve Size
Heap Commit Size
Stack Reserve Size
Stack Commit Size
Enable Large Addresses
Terminal Server
Swap Run From CD
Swap Run From Network
Driver
Not Set
Console (/SUBSYSTEM:CONSOLE)
Windows (/SUBSYSTEM:WINDOWS)
Native (/SUBSYSTEM:NATIVE)
EFI Application (/SUBSYSTEM:EFI_APPLICATION)
EFI Boot Service Driver (/SUBSYSTEM:EFI_BOOT_SERVICE_DRIVER)
EFI ROM (/SUBSYSTEM:EFI_ROM)
EFI Runtime (/SUBSYSTEM:EFI_RUNTIME_DRIVER)
POSIX (/SUBSYSTEM:POSIX)
SubSystem
The /SUBSYSTEM option tells the operating system how to run the .exe file.The choice of subsystem affects the entry point symbol (or entry point function) that the linker will choose.
OK
Abbrechen
Übernehmen


Project4 Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
SubSystem
Console (/SUBSYSTEM:CONSOLE)
Minimum Required Version
Heap Reserve Size
Heap Commit Size
Stack Reserve Size
Stack Commit Size
Enable Large Addresses
Terminal Server
Swap Run From CD
No
Swap Run From Network
No
Driver
Not Set
SubSystem
The /SUBSYSTEM option tells the operating system how to run the .exe file.The choice of subsystem affects the entry point symbol (or entry point function) that the linker will choose.
OK
Abbrechen
Übernehmen