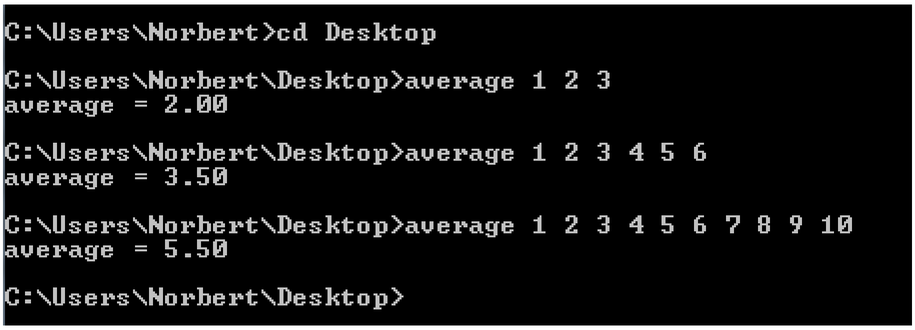
Computing I – Programming Labs

Problem Set 4 - Working with Vectors

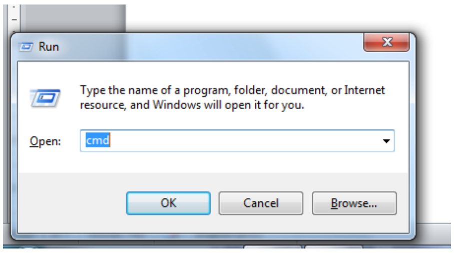
Problem 1 (Average Calculation with Command Line Input)

Write a program average.exe. Its purpose is to calculate the average of a series of integer numbers entered into a console window after the program name (see below). Use a one-dimensional array of type `vector` to store the numbers entered via the command line. Print out the average as a double with two digits after the decimal point.



```
C:\Users\Norbert>cd Desktop

C:\Users\Norbert\Desktop>average 1 2 3
average = 2.00

C:\Users\Norbert\Desktop>average 1 2 3 4 5 6
average = 3.50

C:\Users\Norbert\Desktop>average 1 2 3 4 5 6 7 8 9 10
average = 5.50

C:\Users\Norbert\Desktop>
```

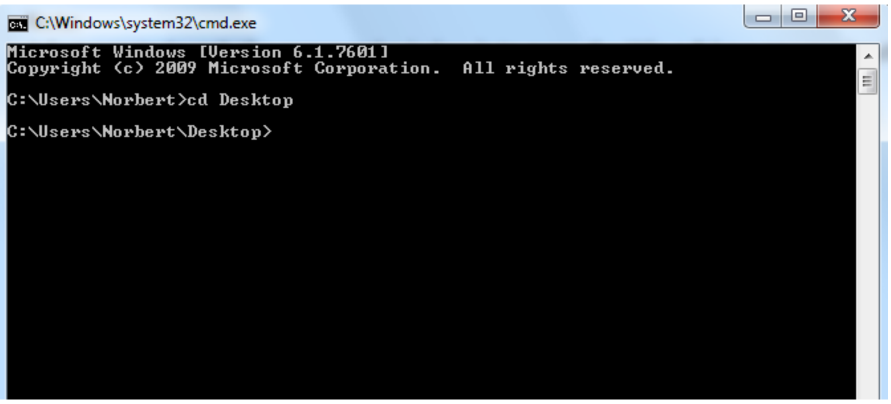
Note:
A fast way to launch the command window is to press the Win + R keys on your keyboard. Then, type cmd and press Enter or click/tap OK**.**



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: cmd

To run your executable program (*.exe), copy it to your Desktop. Then change the working directory to the Destop. Finally type the program name and its arguments (see above, for example).



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Norbert>cd Desktop

C:\Users\Norbert\Desktop>
```

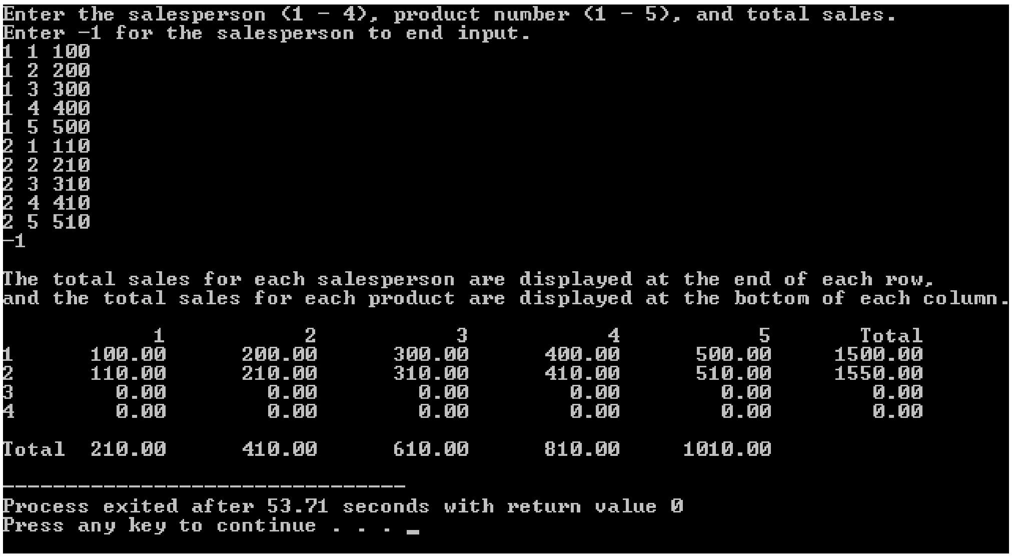
Problem 2 (Sales Summary)

Use a two-dimensional array of type `vector` to solve the following problem: A company has four salespeople (1 to 4) who sell five different products (1 to 5). At the end of each month, each salesperson passes in a slip for each different type of product sold. Each slip contains the following:

a) The salesperson number
b) The product number
c) The total value of that product sold that month (in Euro)

Write a program into which you can enter the total sales for each salesperson and each product for last month. The program should summarize the total sales by salesperson and by product. All totals should be stored in the two-dimensional `vector sales`.

Print the results in tabular format. Each of the rows should represent a particular salesperson and each of the columns should represent a particular product. Calculate the total across each row to get the total sales for each salesperson for last month. Calculate the total along each column to get the total sales for each product for last month. Your tabular printout should include these cross totals to the right of the totaled rows and to the bottom of the totaled columns. Follow the example dialog and output shown below.



```
Enter the salesperson (1 - 4), product number (1 - 5), and total sales.
Enter -1 for the salesperson to end input.
1 1 100
1 2 200
1 3 300
1 4 400
1 5 500
2 1 110
2 2 210
2 3 310
2 4 410
2 5 510
-1

The total sales for each salesperson are displayed at the end of each row,
and the total sales for each product are displayed at the bottom of each column.

            1           2           3           4           5       Total
1      100.00      200.00      300.00      400.00      500.00     1500.00
2      110.00      210.00      310.00      410.00      510.00     1550.00
3        0.00        0.00        0.00        0.00        0.00        0.00
4        0.00        0.00        0.00        0.00        0.00        0.00

Total  210.00      410.00      610.00      810.00     1010.00

--------------------------------
Process exited after 53.71 seconds with return value 0
Press any key to continue . . . _
```

You find the `main` function below. Implement the functions `getsalesdata` and `display`.

```
// Problem2.cpp (Sales Summary)
#include <iostream>
#include <iomanip>
#include <vector>

using namespace std;

void getsalesdata(vector<vector<double>>&);
void display(const vector<vector<double>>&);

int main()
{
   const size_t PEOPLE = 5;
   const size_t PRODUCTS = 6;

   vector<vector<double>> sales(PEOPLE, vector<double> (PRODUCTS));
   // This vector can be visualised as a two dimensional array, e.g., sales[i][j],
   // with 'PEOPLE' rows and 'PRODUCT' columns.
   getsalesdata(sales);
   display(sales);

   return 0;

} // end main
```

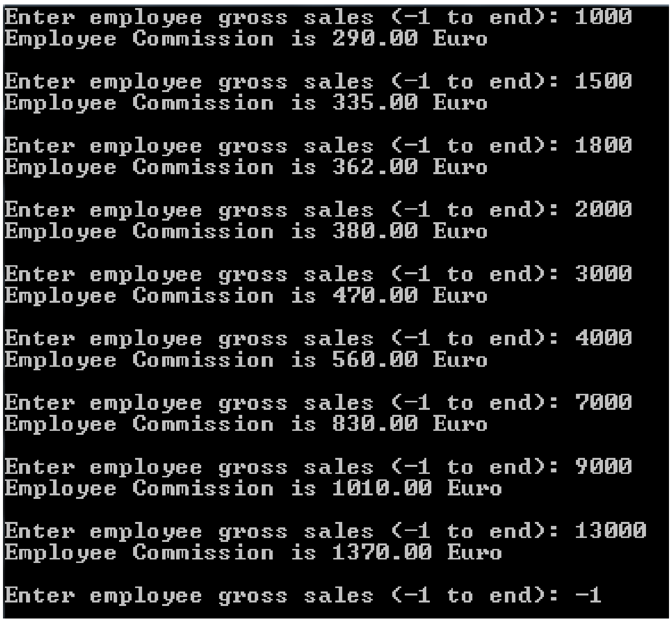
Problem 3 (Salesperson Salary Ranges)

Use one-dimensional arrays of data type vector to solve the following problem. A company pays its salespeople on a commission basis. The salespeople each receive 200 Euro base salaries per week plus 9 percent of their gross sales for that week. For example, a salesperson who grosses 5000 Euro in sales in a week receives 200 Euro plus 9 percent of 5000 Euro, or a total of 650 Euro.

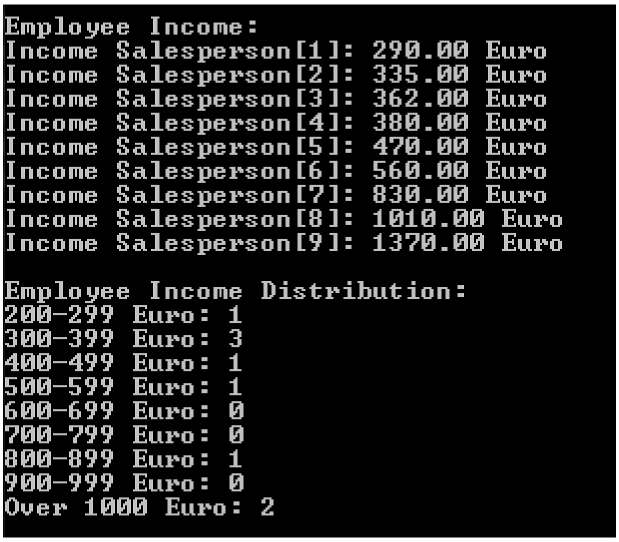
Write a program with which you can enter the sales of salespeople. Enter "-1" to signal end of input. Keep track of the entries using a vector. Then display how much each salesperson earned in a given month. Finally, determine how many of the salespeople earned salaries in each of the following ranges (assume that each salesperson's salary is truncated to an integer amount):

1. 200–299 Euro
2. 300–399 Euro
3. 400–499 Euro
4. 500–599 Euro
5. 600–699 Euro
6. 700–799 Euro
7. 800–899 Euro
8. 900–999 Euro
9. 1000 Euro and over

Your input/output should follow the example displayed below:



```
Enter employee gross sales (-1 to end): 1000
Employee Commission is 290.00 Euro

Enter employee gross sales (-1 to end): 1500
Employee Commission is 335.00 Euro

Enter employee gross sales (-1 to end): 1800
Employee Commission is 362.00 Euro

Enter employee gross sales (-1 to end): 2000
Employee Commission is 380.00 Euro

Enter employee gross sales (-1 to end): 3000
Employee Commission is 470.00 Euro

Enter employee gross sales (-1 to end): 4000
Employee Commission is 560.00 Euro

Enter employee gross sales (-1 to end): 7000
Employee Commission is 830.00 Euro

Enter employee gross sales (-1 to end): 9000
Employee Commission is 1010.00 Euro

Enter employee gross sales (-1 to end): 13000
Employee Commission is 1370.00 Euro

Enter employee gross sales (-1 to end): -1
```



```
Employee Income:
Income Salesperson[1]: 290.00 Euro
Income Salesperson[2]: 335.00 Euro
Income Salesperson[3]: 362.00 Euro
Income Salesperson[4]: 380.00 Euro
Income Salesperson[5]: 470.00 Euro
Income Salesperson[6]: 560.00 Euro
Income Salesperson[7]: 830.00 Euro
Income Salesperson[8]: 1010.00 Euro
Income Salesperson[9]: 1370.00 Euro

Employee Income Distribution:
200-299 Euro: 1
300-399 Euro: 3
400-499 Euro: 1
500-599 Euro: 1
600-699 Euro: 0
700-799 Euro: 0
800-899 Euro: 1
900-999 Euro: 0
Over 1000 Euro: 2
```

You find the `main` function below. Implement the functions `wages` and `display`.

```
// Problem 1 (Salesperson Salary Ranges)
#include <iostream>
#include <iomanip>
#include <vector>
using namespace std;

void wages( vector< double > & ); // function prototype
void display( const vector< double > & ); // function prototype

int main()
{
   vector< double > salaries; // vector to hold salaries

   wages( salaries ); // get wages
   display( salaries ); // display wages and their ranges

  return 0;

} // end main
```

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):

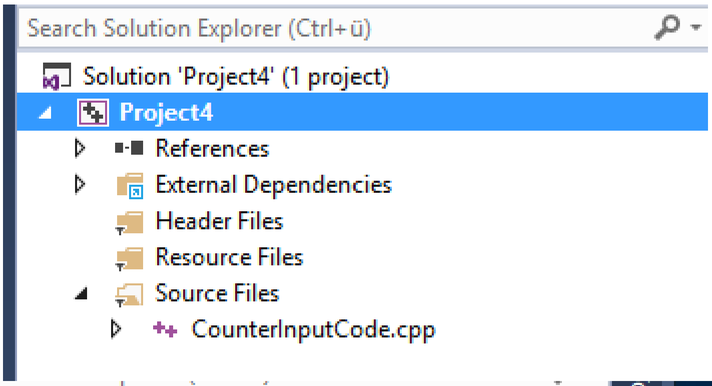
1. Right-click on project name.

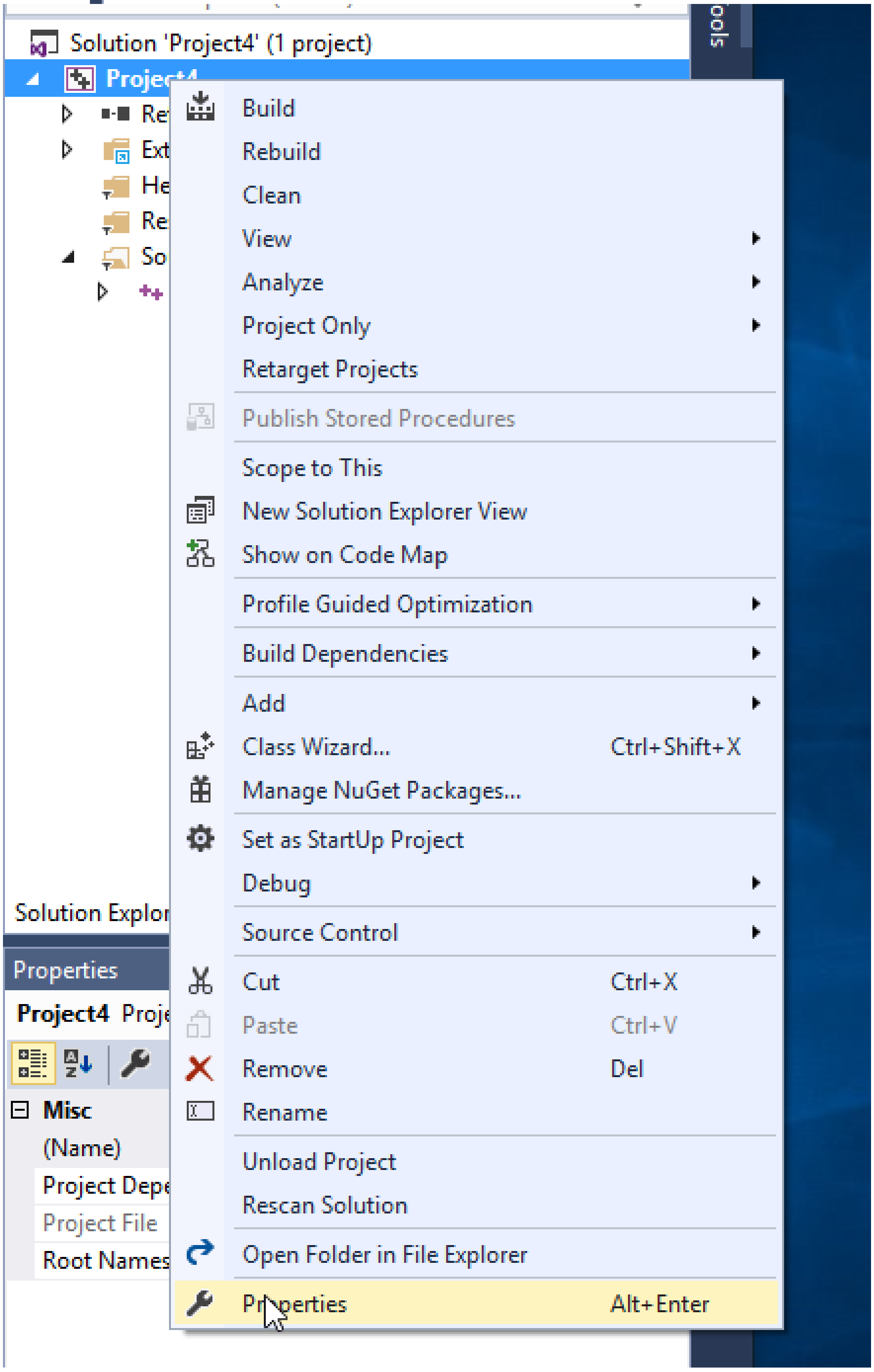
2. Select Properties from context menu.

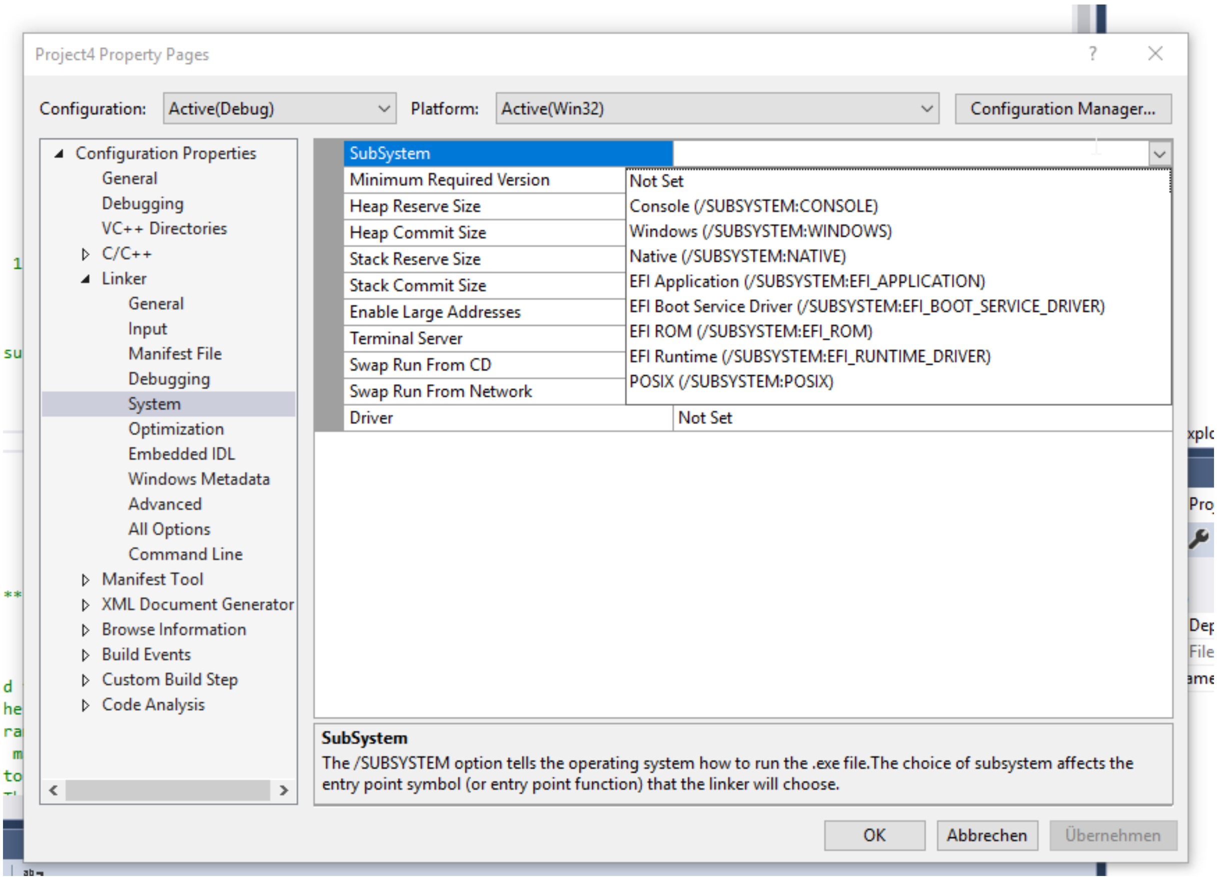
3. Select Configuration Properties>Linker>System.

4. Click into the drop-down-box to the right and the choose "Console (/SUBSYSTEM:CONSOLE)"

5. Select Apply and then OK.





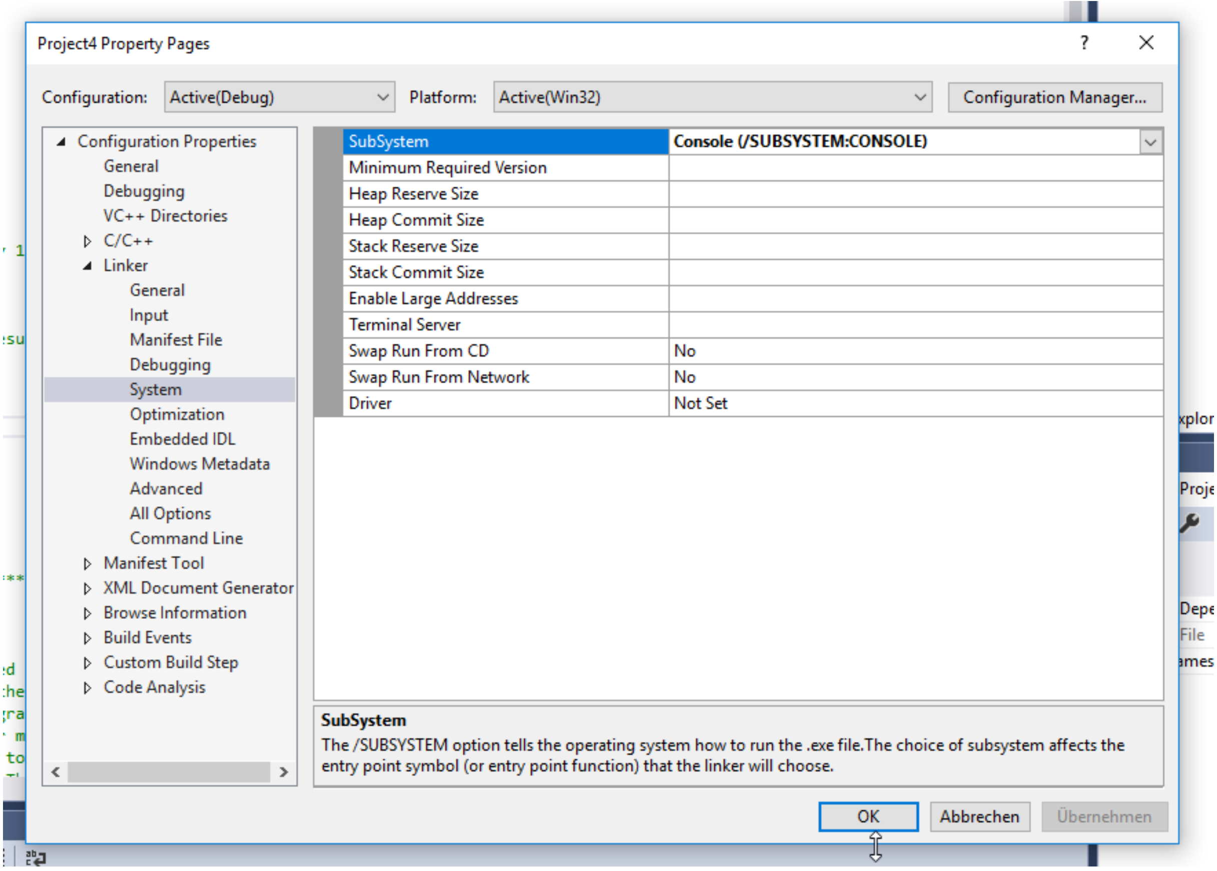


Configuration: Active(Debug)

Platform: Active(Win32)

SubSystem	
Minimum Required Version	Not Set
Heap Reserve Size	Console (/SUBSYSTEM:CONSOLE)
Heap Commit Size	Windows (/SUBSYSTEM:WINDOWS)
Stack Reserve Size	Native (/SUBSYSTEM:NATIVE)
Stack Commit Size	EFI Application (/SUBSYSTEM:EFI_APPLICATION)
Enable Large Addresses	EFI Boot Service Driver (/SUBSYSTEM:EFI_BOOT_SERVICE_DRIVER)
Terminal Server	EFI ROM (/SUBSYSTEM:EFI_ROM)
Swap Run From CD	EFI Runtime (/SUBSYSTEM:EFI_RUNTIME_DRIVER)
Swap Run From Network	POSIX (/SUBSYSTEM:POSIX)
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.



Configuration: Active(Debug)

Platform: Active(Win32)

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.