Compiling Modular Source Code

Workshop 1 (out of 10 marks - 3% of your final grade)

OOP244SAB

In your first workshop, you are to sub-divide a simple program into two modules and compile the modules separately, on different platforms.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities

- to organize source code into modules, with header and implementation files
- to compile and run modular programs on different platforms
- to accurately describe the work you have done

ORIGINAL SOURCE CODE - ALL GROUPS

Save the following program as **`w1step1.cpp`** on your local computer:

```
// OOP244 Workshop 1: Compiling modular source code
// File       w1step1.cpp
// Version 1.0
// Date       2015/05/07
// Author     Franz Newland
// Description
// This provides some source code in a single file to break
// into modules and compile together
//
// Revision History
// -----------------------------------------------------------
// Name                    Date          Reason
// F.Newland  2015/01/09           Preliminary release
/////////////////////////////////////////////////////////////////

#include <iostream>
#include <iomanip>
using namespace std;

const char* printMsg = "Finished... Exiting\n"; //End message
const char* errMsg = "Try again\n";       //Error message
const char* cancelMsg = "Order cancelled - Start again\n";    //Cancel message
```

```
//Functions
int checkout();                    // Display menu and return selection

int main(){
	int iCost = 0;	// Selected item cost
	int iTotal = 0;
	cout << "SeneKEA Order Tool\n"
		<< "======================\n";

	// process user input
	while (iCost != 10){
		iCost = checkout();
		if (iCost == 0)
			cout << errMsg;
		else if (iCost == -1) {
			iTotal = 0;
			cout << cancelMsg;
		}
		else if (iCost == 10) {
			cout << "Total is $" << iTotal << endl;
			cout << printMsg;
		}
		else {
			cout << "Total increases by $" << iCost << endl;
			iTotal += iCost;
		}
	}
	return 0;
}

// int checkout()
// Description: prompts for and accepts an option selection from
// standard input and returns the integer price for the selected option
// Outputs: returns the price of the selected option, -1 to cancel, 10 to pay
// or 0 otherwise
////////////////////////////////////////////////////////////////////////
int checkout(){
	int iSelection = 0;		//storage of the user response
	int iPrice = 0;
	cout << '\n';
	cout << "Please select from the following options :\n";
	cout << " 1 Shelf unit $2\n";
	cout << " 2 Desk $4\n";
	cout << " 3 Bed $4\n";
	cout << " 4 Chair $2\n";
	cout << " 5 Pay\n";
	cout << " 6 Cancel\n";

	cin >> iSelection;
	if ((iSelection > 0) && (iSelection < 7)){//if user response is valid
		if (iSelection == 1 || iSelection == 4)
			iPrice = 2;
		else if (iSelection < 5)
			iPrice = 4;
		else if (iSelection == 5)
			iPrice = 10;
		else
```

```
                iPrice = -1;
        cout << "Thank you\n";
    }
    else
        cout << "Incorrect entry\n";

    return iPrice;
}
```

Compile **w1step1.cpp** and run the executable on

- Linux

```
g++ -o w1step1 w1step1.cpp -Wall
w1step1
```

- Windows

Download **w1step1.cpp** to your local windows PC.
Create a simple empty console project in visual studio and copy the files into your project directory, then compile and run your workshop;

Creating a simple empty console project:
- open visual studio
- click File/New/Project
- in New Project screen make sure the two checkboxes above the OK button are unchecked (uncheck “Create directory for solution” and “Add to source control”
- in New Project screen, in Name section type “workshop1”
- in New Project screen, in Location section select a proper directory for your workshop1
- in Template section select “Visual C++/Win32/Win32 Console Application”
- click on OK
- In “Win32 Application Wizard – workshop1” screen click on “Next” button
- make sure all the checkboxes are “UNCHECKED”, and then check “Empty project”
- Make sure that in this screen there are only two selected items; “Console Application” and “Empty project”
- click on Finish button

The empty project is created!

Copying and adding the files to your project.

- In solution explorer Right click on workshop1 and select “Open Folder in File Explorer”

- copy w1step1.cpp to the opened folder.
- in Solution Explorer/workshop1 right click on “Source Files” and select “Add/Existing Item”
- in “Add Existing Item – workshop1” screen select **w1step1.cpp** click on Add button select

The file is added to workshop1 project.

To compile and execute the project you can either press Ctrl+F5 or select “DEBUG/Start without debugging”

MODULAR SOURCE CODE – MINIMUM TASK (TOTAL 10 MARKS)

PART 1 - (DUE AT THE END OF THE LAB SESSION – 7 MARKS)

Once **w1step1.cpp** runs successfully on each platform, sub-divide **w1step1.cpp** into

1. a main module named **w1**
   - a header file named **w1.h**
   - an implementation file named **w1.cpp**
2. a **checkout** module
   - a header file named **checkout.h**
   - an implementation file named **checkout.cpp**

Your **w1.h** header file should only contain the message definitions.

Your **checkout.h** header file should only contain the **checkout()** prototype. Add header comments to each file.

Recompile your modularized source code and run the executable on

- Linux

```
g++ -o w1 w1.cpp checkout.cpp
w1
```

- Windows
  - If they are not already added to the project, add the files to the project you already created.
  - Remove w1setp.cpp from the project; right click on **w1step1.cpp** in Solution Explorer and click on “Remove”.
  - Compile and execute the project.

Part 1; In Lab Submission

1. You will be submitting your files directly from your **matrix** account.
2. Upload all your modules (**`w1.h, w1.cpp, checkout.h, checkout.cpp`**, **`w1.txt)`** your **matrix** account (if they are not there already)
3. From the directory of your workshop 1 on matrix run the following command:
   `> ~fardad.soleimanloo/submit_w1_in_lab <ENTER>`
4. Follow the instructions.

If everything is done properly, your workshop reflection will be submitted. If there is any problem a message will be shown explaining what the problem is.

Please not that if you do not submit this during the lab, you will have to submit Part 2 only. In this case, the maximum mark you can gain is 7.

PART 2 – REFLECTION

(DUE ON SEPTEMBER 20, 2015 AT 8PM – 3 MARKS)

Based on the work you have done for this workshop, please answer following questions and place them in a file name named **`reflect.txt`**.

1. What is the benefit of dividing this solution into 2 modules?
2. When compiling using **`g++`**, state the meaning of the following command line switches.
   `a. -o`
   `b. -Wall`

SUBMISSION

Matrix

1. You will be submitting your files directly from your **matrix** account.
2. Upload all your modules (**`w1.h, w1.cpp, checkout.h, checkout.cpp`**, **`w1.txt, reflect.txt)`** your **matrix** account.
3. From the directory of your workshop 1 on matrix run the following command:
   `> ~fardad.soleimanloo/submit_w1_reflect <ENTER>`
4. Follow the instructions

If everything is done properly, your workshop reflection will be submitted. If there is any problem a message will be shown explaining what the problem is.