COMPILING MODULES

Workshop 1 (V0.91)
(V0.9 initial post)
(V0.91 renamed sict to sdds)

In your first workshop, you are to sub-divide a program into modules, compile each module separately and construct an executable from the results of each compilation.

LEARNING OUTCOMES

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

- organize source code into modules, using header and implementation files;
- compile and link modular programs;
- distinguish the contents of a header and an implementation file;
- describe to your instructor what you have learned in completing this workshop.

SUBMISSION POLICY

The workshop is divided into 3 sections;
in-lab - 30% of the total mark

To be completed before the end of the lab period and submitted from the lab.

at-home - 35% of the total mark

To be completed within 2 days after the day of your lab.
DIY (Do It yourself) – 35% of the total mark

To be completed within 3 days after the at-home due date.

The *in-lab* section is to be completed after the workshop is published, and before the end of the lab session. The *in-lab* is to be submitted during the workshop period form the lab.

If you attend the lab period and cannot complete the *in-lab* portion of the workshop during that period, ask your instructor for permission to complete the *in-lab* portion after the period. You must be present at the lab in order to get credit for the *in-lab* portion.

If you do not attend the workshop, you can submit the *in-lab* section along with your *at-home* section (see penalties below). The *at-home* portion of the lab is due on the day that is 2 days after your scheduled *in-lab* workshop (23:59) (even if that day is a holiday).

The DIY (Do It Yourself) section of the workshop is a task that utilizes the concepts you have done in the in-lab + at-home section. This section is completely open ended with no detailed instructions other than the required outcome. You must complete the DIY section up to 3 days after the at-home section.

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible to back up your work regularly.

Ask your professor if there are any additional requirements for your specific section.

CITATION AND SOURCES

When submitting the DIY part of the workshop, Project and assignment deliverables, a file called sources.txt must be present. This file will be submitted with your work automatically.

You are to write either of the following statements in the file "sources.txt":

I have done all the coding by myself and only copied the code that my professor provided to complete my workshops and assignments.

Then add your name and your student number as signature

OR:

Write exactly which part of the code of the workshops or the assignment are given to you as help and who gave it to you or which source you received it from.

You need to mention the workshop name or assignment name and also the file name and the parts in which you received the code for help.

Finally add your name and student number as signature.

By doing this you will only lose the mark for the parts you got help for, and the person helping you will be clear of any wrong doing.

LATE SUBMISSION PENALTIES:

- *In-lab* portion submitted late, with *at-home* portion:

    0 for *in-lab*. Maximum of **DIY+at-home**/10 for the workshop.

- at-home or DIY submitted late:

    1 to 2 days, -20%, 3 to 7 days -50% after that submission rejected.

- If any of *in-lab*, *at-home* or DIY portions is missing, the mark for the whole workshop will be **0**/10

ORIGINAL SOURCE CODE (THE SENEGRAPH PROGRAM)

`SeneGraph` is a program that receives several statistical sample values and compares those values using a horizontal Bar Chart.

> **NOTE**: For this part, if you have not setup your computer, it is better to do this lab at school and on a lab computer since it has "`Git`" and "`Tortoise Git`" installed. If you do have "`Git`" or "`Tortoise Git`" installed on your own computer, you can do this on your own personal computers.
>
> **NOTE**: `Tortoise Git` installation guidelines are in the *at-home* section of this lab.

RETRIEVE THE ORIGINAL PROGRAM:

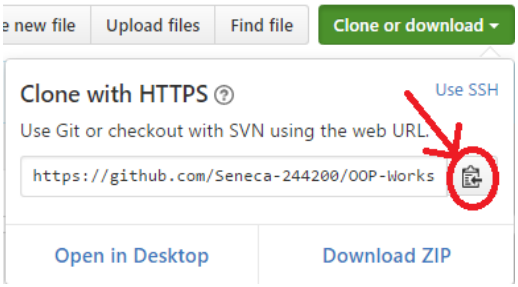
First go to MyApps on the lab computers and launch:
TortoiseGit
Putty
Visual Studio 2019

Workshop Steps:

1. Open https://github.com/Seneca-244200/OOP-Workshops and click on "Clone or download" Button; this will open "Clone with HTTPS" window.

> **NOTE**: If the window is titled "Clone with SSH" then click on "Use HTTPS":

2. Copy the https URL by clicking on the button on the right side of the URL:

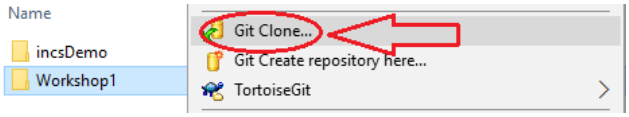


3. Open File Explorer on your computer and select or create a directory for your workshops.

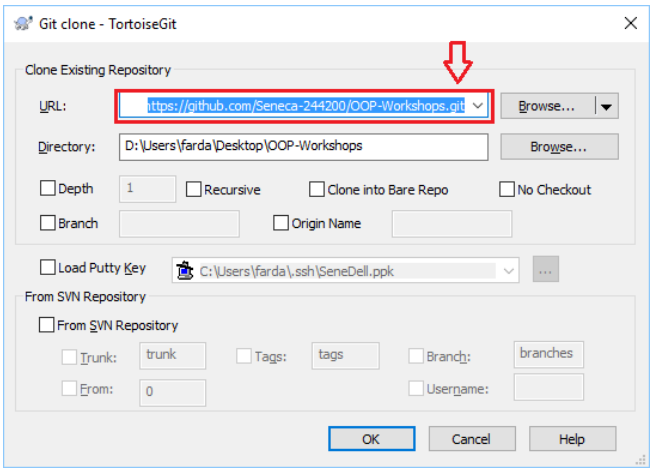
Now Clone (download) the original source code of `SeneGraph` (Workshop1) from GitHub in one of the following three ways: (methods 1 and 2 are preferred)

1. Using `TortoiseGit`:

   a. Right click on the selected directory and select "Git Clone":



   This will open the "Git Clone" window with the URL already pasted in the "URL" text box; if not, paste the URL manually.



   b. Click on OK.

   This will create on your computer a clone (identical directory structure) of the directory on `Github`. Once you have cloned the directory, you can open the directory `OOP-Workshops/WS01` and start doing your workshop. Note that you will repeat this process for all workshops and milestones of your project in this subject.

IMPORTANT: If your professor makes any changes to the workshop, you can right click on the cloned repository directory and select `TortoiseGit/pull` to update and sync your local workshop to the one on `Github` without having to download it again. Note that this will only apply the changes made and will not affect any work that you have done on your workshop.

2. Using the command line:

   a. Open the `git` command line on your computer.
   b. Change the directory to your workshops directory.
   c. Issue the following command at the command prompt in your workshops directory:

```
git clone https://github.com/Seneca-244200/OOP-Workshops.git<ENTER>
```

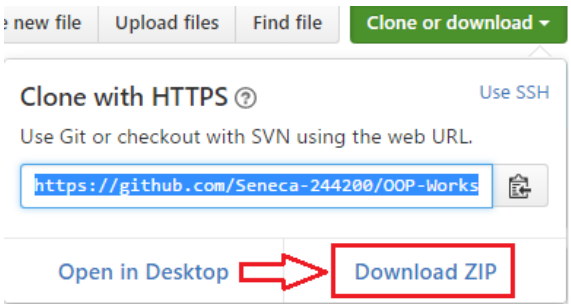
IMPORTANT: The URL for all the workshops are the same throughout the semester. The only thing that changes, is the workshop number.

This will create on your computer a clone (identical directory structure and content) of the `OOP-Workshops` directory on `Github`. Once you have cloned the directory, you can open subdirectory `OOP-Workshops/WS01` and start doing your workshop. Note that you will repeat this process for all workshops and milestones of your project in this subject.

IMPORTANT: If your professor makes any changes to the workshop, you can issue the command `git pull<ENTER>` in the cloned repository directory to update and sync your local workshop to the one on `Github` without having to download it again. Note that this will only apply the changes made and will not affect any work that you have done on your workshop.

3. Using the "Download ZIP" option:

   a. Open https://github.com/Seneca-244200/OOP-Workshops and click on "Clone or download" button and click on "Download ZIP".

   b. This will download to your computer a zipped file copy of the workshop repository in `Github`. You can extract this file to where you want to do your workshop.

IMPORTANT: Note that, if your professor makes any changes to the workshop, to get them you have to separately download another copy of the workshop and manually apply the changes to your working directory to **make sure nothing of your work is overwritten by mistake**.

IN-LAB (30%)

STEP 1: TEST THE PROGRAM

1. On Windows, using Visual Studio (VS)

a. Open the `OOP-Workshops/WS01/in_lab` directory (that you just cloned or downloaded) and click on `in_lab.vcxproj`. This will open a Visual Studio (VS) project in the same directory.

b. In VS, (if not open already) open Solution Explorer *(click on View/Solution Explorer)* and then add `w1_in_lab.cpp` file to your project.
   - Right click on "Source Files"
   - Select "Add/Existing Item"
   - Select `w1_in_lab.cpp` from the file browser
   - Click on "Ok"

c. Compile and run the program by selecting "Debug/Start without Debugging" or pressing "Ctrl-F5".

2. On Linux, in your `matrix` account.

d. Upload `w1_in_lab.cpp` to your matrix account (Ideally to a designated directory for your oop244 workshop solutions). Then, enter the following command to compile the source file and create an executable called `w1`:

```
g++ w1_in_lab.cpp -Wall -std=c++11 -o w1<ENTER>
```

   - `-Wall`: display all warnings
   - `-std=c++11`: compile using C++11 standards
   - `-o w1`: name the executable `w1`

e. Type the following to run and test the execution:

```
w1<ENTER>
```

STEP 2: CREATE THE MODULES

1. On Windows, using Visual Studio (VS)

In solution explorer, add three new modules to your project:

- `SeneGraph`
- `Graph`
- `Tools`

The `SeneGraph` module has an implementation (`.cpp`) file but no header file. The graph and tools modules have both implementation (`.cpp`) and header (`.h`) files:

- Add `seneGraph.cpp`, `graph.cpp` and `tools.cpp` to the "Source Files" directory (right click on "Source Files" and select "Add/New Item" and add a C++ file)
- Add `graph.h`, `tools.h` to the "Header Files" directory (right click on "Header Files" and select "Add/New Item" and add a header file)

Separate the functions in `w1_in_lab.cpp` and copy them into the modules as follows. Make sure that you include in the implementation file (`.cpp`) for each module `<iostream>` and insert the statement `using namespace std;` at the beginning of the file but after all include directives.

- The **`Tools`** module contains the `menu()`,`goBack()`, `line()`, and `getInt()` functions. Copy their definitions to the module's `.cpp` file and their prototypes to the module's `.h` file. Make sure that you include `tools.h` in `tools.cpp`.

  To test that you have done this correctly, you can compile this module separately, by right clicking on `tools.cpp` and select `compile` from the menu. If the compilation is successful, most likely you have done it correctly.

  > **NOTE**: The equivalent of this on matrix is to add **`-c`** to the compile command:
  >
  > **`g++ tools.cpp –Wall -std=c++11 -c`**<ENTER>
  >
  > This will only compile `tools.cpp` and will not create an executable.

- The **`Graph`** module contains the `getSamples()`, `findMax()`, `printBar()` and `printGraph()` functions. Copy their definitions to the module's `.cpp` file and their prototypes to the module's `.h` file. Add the define statements for `MAX_NO_OF_SAMPLES` and `GRAPH_WIDTH` to `graph.h`. Make sure that you include `tools.h` and `graph.h` in `graph.cpp`.

  To test that you have done this correctly, you can compile this module separately, by right clicking on `graph.cpp` and select `compile` from the menu. If the compilation is successful, most likely you have done it correctly.

> **NOTE**: The equivalent of this on matrix is to add `-c` to the compile command:
>
> **`g++ graph.cpp –Wall -std=c++11 -c`**<ENTER>
>
> This will only compile `graph.cpp` and will not create an executable.

- The **SeneGraph** module contains the `main()` and `samplesFirst()` functions. Make sure that you include `tools.h` and `graph.h` in `seneGraph.cpp`.

Now remove `w1_in_lab.cpp` from the project. You can do this by right clicking on the filename in solution explorer and selecting remove in the menu (make sure you do not delete this file but only remove it).

Compile and run the project (as you did before) and make sure everything works.

2. On Linux, in your `matrix` account.

Upload the five files to your matrix account and compile the source code using the following command.

`g++ tools.cpp graph.cpp seneGraph.cpp -Wall -std=c++11 -o w1`<ENTER>

Run the program and make sure that everything works properly.

Output Sample (**red** values are entered by the user):

```
To Be completed
```

IN-LAB SUBMISSION (30%)

To Be Announced

> **IMPORTANT**: Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

At-Home Section (30%)

Step 1: Prepare your computer for OOP244 Subject

Please watch the videos under this playlist on YouTube:
Seneca-SDDS "how to" videos for C/C++ Core Subjects
and follow the instructions to have your computer ready for the subject.

Your work is to be done on a PC with Windows 10 Operating System under Visual Studio Integrated Development Environment. If you choose to do your work on other platforms, there will be limited help available, for trouble shooting your setup. If you don't have a Windows PC, it is strongly recommended to install a Virtual Machine that runs Windows 10 and the above setup is applied to it.

Step 2: Complete the Workshop

1. Open the *in-lab* section of your workshop solution and add compilation safeguards to your modules. Surround your prototypes in your header files with the following code to prevent multiple includes:

```
#ifndef NAMESPACE_HEADERFILENAME_H
#define NAMESPACE_HEADERFILENAME_H

// Your header file content goes here

#endif
```

   Here is the instructional video showing how the compiler works and why you need these safeguards in all of your header files:
   https://www.youtube.com/watch?v=EGak2R7QdHo
   Please note that the school name is changed. The name of our school is School of Software Design and Data Science and therefore we write our code under the name space `sdds`

2. Enclose the prototypes and function definitions in your `Graph` and `Tools` modules within the `sdds` namespace.
3. Include in your `SeneGraph` module a directive to use the ~~`sict`~~ `sdds` namespace.

AT-HOME SUBMISSION

To Be Announced

IMPORTANT: Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

DIY (DO IT YOURSELF 35%)

Open the DIY project (in WS01 directory) using Visual Studio. Run and test the program and check and review the source code in w1_diy.cpp. Make sure you copy the execution result somewhere for future review.

Divide the code in w1_diy.cpp into five modules:

"PhoneApp": The main function
"PhoneDir": Containing the user interface and its related functions.

"Contact": Containing all Contact related functions.
"File": Containing all the functions dealing with data file management.
"Tools": Containing all the helper functions who have no relation with a phonebook and are used as utilities only.

Remember that all the code written in OOP244 should be in "sdds" namespace and the program containing the main function uses the "sdds" namespace.

Compile the five modules separately to make sure they are set properly and finally run the whole modular program and make sure it runs exactly like it did before and then submit your work.

DIY SUBMISSION

To Be Announced

IMPORTANT: Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.