In this document I will describe the recommended method for creating a solution to an exercise like the one I have assigned you for assignment 1. I will use as an example for this document exercise 03\_14 that I assigned as one of the lab exercises in week 2 and for which I also provided a solution for you to download (both are in the "Week 2" D2L Content folder).

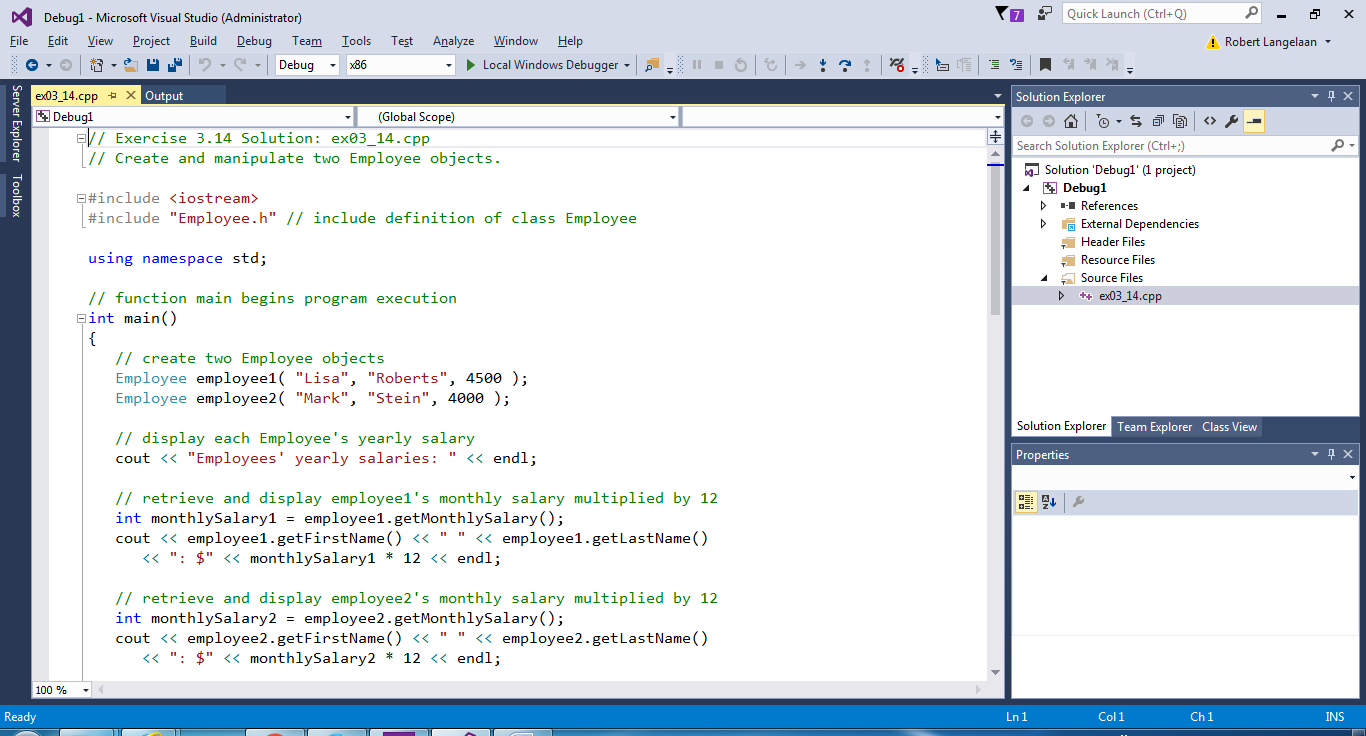
Step 1:

Create a new project. Study the doc in ShareOut titled "Start VS 2012" if you are unsure on how to create a new project, add new files to a project, compile a .cpp file, build the complete solution and execute the solution.

Step 2:

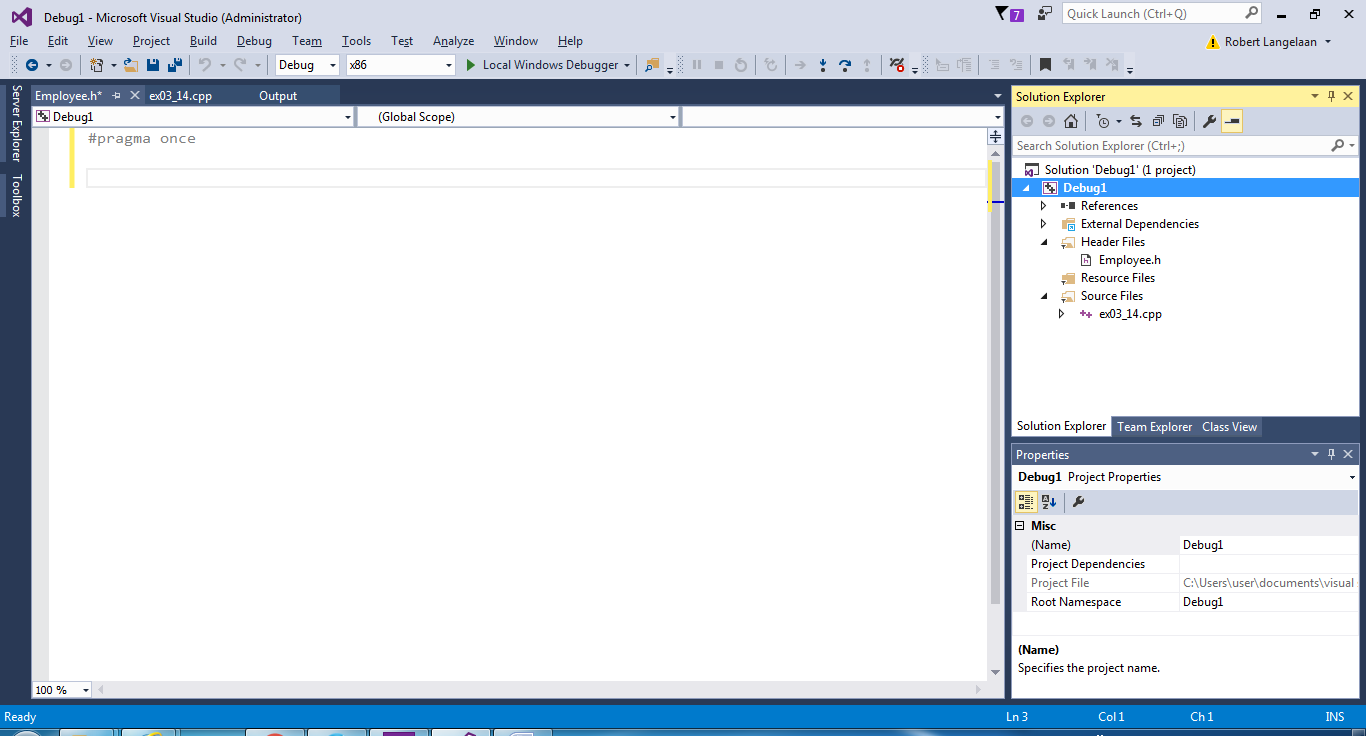
Create a main() client .cpp file that will be used to test the class that you will be creating. For both exercise 03\_14 and assignment 1 I have provided you with this file. You may want to add additional statements to the file I provided in order to test your class solution more completely.   
  
When I provide the client .cpp file, you should copy it into the project folder. If you put it in the correct folder, you should be able to see it in the default folder when you perform "Add -> Existing Item...".

If you have completed step 2, your IDE view should look something like:

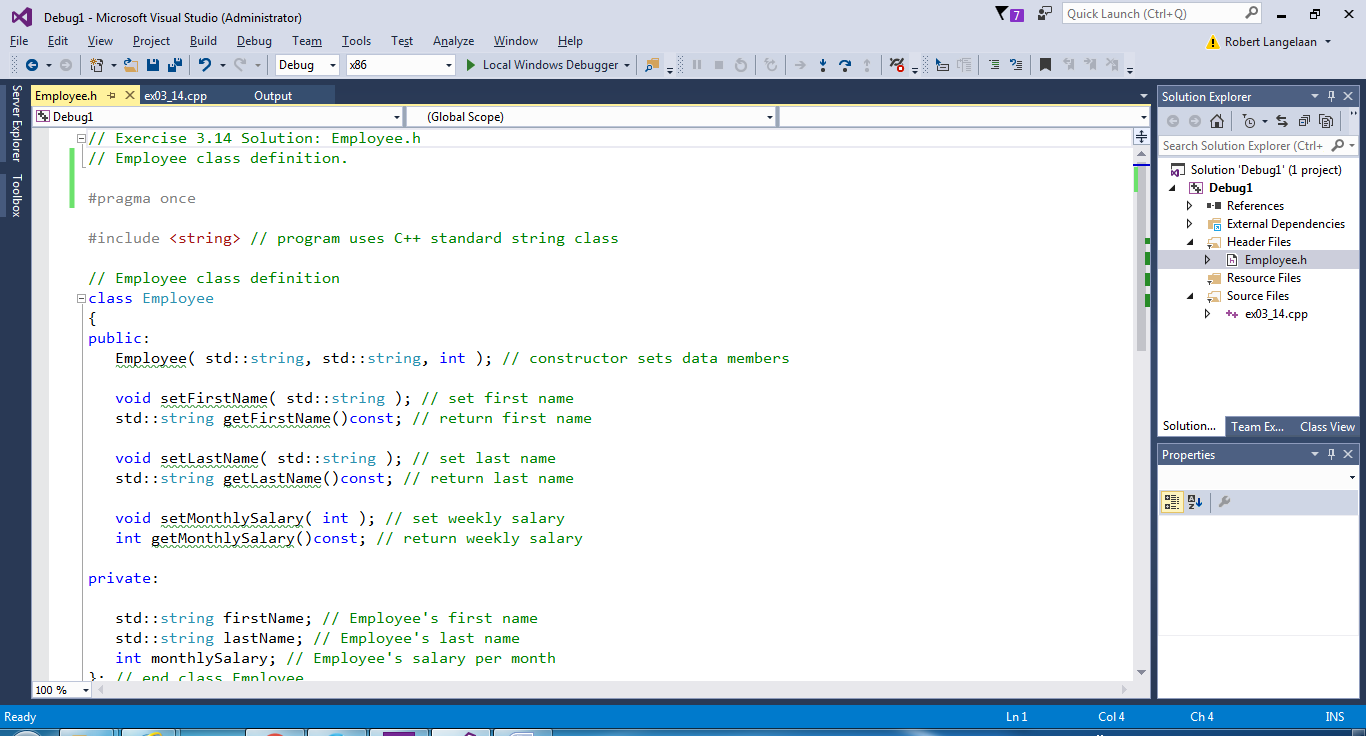


Step 3:

Create the class interface file based on the exercise requirements and the client .cpp file created to test your class.  
  
First add an empty .h file to your project ( Add -> New Item... ) with the appropriate name. For exercise 03\_14 this should look like:



Then complete the interface file so that it looks like:

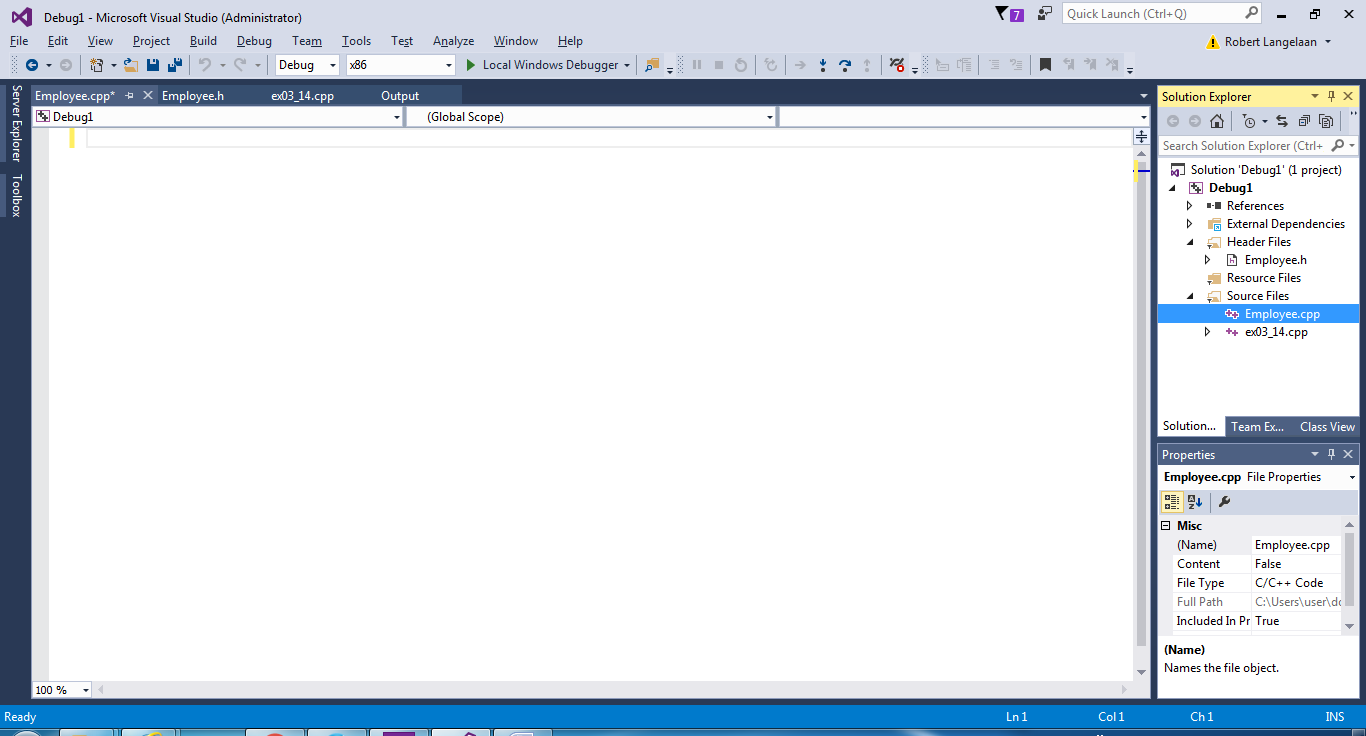


If you have completed the interface file correctly you should be able to compile the client .cpp file without errors and usually without warnings as well. To do so, right mouse button click on the client .cpp file in the Solution Explorer window (ex03\_14.cpp in the screenshot above) and select "Compile" . For exercise 03\_14 you will generate a couple of warnings that you can ignore. Later we will learn how to perform a cast to avoid these warnings.

Step 4:

Create a .cpp file for the implementation of the class member functions.

First add an empty .cpp file to your project ( Add -> New Item... ) with the appropriate name. For exercise 03\_14 this should look like:



Step 5:

Create stubs for all of the member functions and then compile without any errors. This will require the 5 mini steps listed below:

1) First we copy and paste the member function prototypes from the class interface file into the class .cpp file. For exercise 03\_14 that would be:

Employee( std::string, std::string, int ); // constructor sets data members

void setFirstName( std::string ); // set first name

std::string getFirstName()const; // return first name

void setLastName( std::string ); // set last name

std::string getLastName()const; // return last name

void setMonthlySalary( int ); // set weekly salary

int getMonthlySalary()const; // return weekly salary

2) Next we add comments, and include and using statements required at beginning of the file. Note that this mini step requires including the class interface file we created in Step 3. After doing so we have:  
  
 // Exercise 3.14 Solution: Employee.cpp

// Employee class member-function definitions.

#include <iostream>

#include "Employee.h" // include Employee class interface file

using namespace std;

Employee( std::string, std::string, int ); // constructor sets data members

void setFirstName( std::string ); // set first name

std::string getFirstName()const; // return first name

void setLastName( std::string ); // set last name

std::string getLastName()const; // return last name

void setMonthlySalary( int ); // set weekly salary

int getMonthlySalary()const; // return weekly salary

3) Next we add class name to beginning of each member function name. This means in this case we add "Employee::" to the beginning of each member function name. We will then have:  
  
 // Exercise 3.14 Solution: Employee.cpp

// Employee class member-function definitions.

#include <iostream>

#include "Employee.h" // include Employee class interface file

using namespace std;

Employee::Employee( std::string, std::string, int ); // constructor sets data members

void Employee::setFirstName( std::string ); // set first name

std::string Employee::getFirstName()const; // return first name

void Employee::setLastName( std::string ); // set last name

std::string Employee::getLastName()const; // return last name

void Employee::setMonthlySalary( int ); // set weekly salary

int Employee::getMonthlySalary()const; // return weekly salary

4) Next we replace the ';' at the end of each member function prototype with opening and closing braces. Note that I have also moved the comment from the end of the member function prototype to just above the member function. We will then have:

// Exercise 3.14 Solution: Employee.cpp

// Employee class member-function definitions.

#include <iostream>

#include "Employee.h" // include Employee class interface file

using namespace std;  
  
 // constructor sets data members  
 Employee::Employee( std::string, std::string, int )   
 {  
 }

// set first name  
 void Employee::setFirstName( std::string )  
 {  
 }  
  
 // return first name  
 std::string Employee::getFirstName()const  
 {  
 }

// set last name

void Employee::setLastName( std::string )  
 {  
 }  
  
 // return last name

std::string Employee::getLastName()const  
 {  
 }

// set weekly salary

void Employee::setMonthlySalary( int )

{

}

// return weekly salary

int Employee::getMonthlySalary()const  
 {  
 }

5) Lastly we return a dummy value of the correct type from those member functions that return a value. In this example that is only the 3 get functions. Look below and you will see that the 3 get functions return a dummy value of the type the compiler requires/expects:

// Exercise 3.14 Solution: Employee.cpp

// Employee class member-function definitions.

#include <iostream>

#include "Employee.h" // include Employee class interface file

using namespace std;  
  
 // constructor sets data members  
 Employee::Employee( std::string, std::string, int )   
 {  
 }

// set first name  
 void Employee::setFirstName( std::string )  
 {  
 }  
  
 // return first name  
 string Employee::getFirstName()const  
 {  
 string x;  
 return x; // return dummy string value  
 }

// set last name

void Employee::setLastName( std::string )  
 {  
 }  
  
 // return last name

string Employee::getLastName()const  
 {  
 string x;  
 return x; // return dummy string value  
 }

// set weekly salary

void Employee::setMonthlySalary( int )

{

}

// return weekly salary

int Employee::getMonthlySalary()const  
 {  
 return 0; // return dummy int value

}

Note that I also replaced "std::string" with "string" above in the last mini step. We don't need "std::" to bring the string class into scope because the "using namespace std;" statement near the beginning of the file now does that for us.   
  
If we have done the 5 mini steps above correctly, we should now be able to compile the class .cpp file (Employee.cpp in this example) in the same way that we compiled the client .cpp file at the end of step 3 without generating any errors.   
  
In fact we should be able to build the complete solution without errors (compile both .cpp files and link them together into a .exe file). We can do this with Build -> Rebuild Solution (Cntrl - Alt - F7).

Step 6:

Complete the implementation of the member functions in the class .cpp file. This will include of course replacing the dummy return values with the correct return values. I suggest only implementing one or two member functions at a time and then compiling the class .cpp file to make sure you have not introduced any errors.

Step 7:

Execute and test your solution, making changes to your solution code as required, until you get the output you are looking for.

As was stated at the beginning of this document, you can view the final solution for exercise 03\_14 in the solutions folder I provide for the Week 2 in class labs in the Week 2 D2L Content folder.

Please let me know if you have any questions.