Part B - Foundations   
  
**Compositions**

**Sections SAA and SABB**

Workshop 7

In this workshop, you are to code a composition relationship between two classes.

**LEARNING OUTCOMES**

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

* code a compositional relationship across multiple classes
* manage dynamic memory within a class
* overload an operator as a member of a class
* overload an operator as a helper function of a class
* reflect upon what you have learned and/or send comments to the professor

**Standards to follow**

* Add inclusion safeguards to your header file so they can not be compiled more than once. See: <https://www.youtube.com/watch?v=EGak2R7QdHo>
* All work (modules) should be done in **oop244** namespace
* “using namespace” can only be used in cpp files and not header files.
* Each file must have a header comment similar to the ones in this workshop
* Tab character should not be used in your source code; set your environment to replace the tabs with spaces. (recommended for this workshop, mandatory for the rest of the deliverables through the semester):   
  <https://www.youtube.com/watch?v=oW4viEA72UI>

<https://www.youtube.com/watch?v=iOAVAzp2B-o>

**part 1: Engine Class**

Design and code a class named Engine that holds information about a type of simulated vehicle’s engine. Place your class definition in a header file named **Engine.h** and your function definitions in an implementation file named **Engine.cpp**.  Include in your solution all of the statements necessary for your code to compile under a standard C++ compiler.

Upon instantiation, an Engine object may receive no parameters and goes to a safe empty state or the following two parameters:

* A double value, representing the engine size.
* A null-terminated C-style string holding the type of engine. You may assume that the string is no larger than 30 characters, excluding the null terminator.

If the size provided is positive, and the type is not empty, the values are accepted. Otherwise the object adopts ***a safe empty state***. (i.e. **size** will be zero and type will be **an empty string**)

Your design includes the following member functions:

* double get() const- a query that returns the size of the engine.
* std::ostream& display(std::ostream& os) const - a query that displays the size as well as the type of the engine and then returns the **os**.  
  The format displayed is as follows:   
    
  *the size, displayed with 2 digits after decimal point.   
  space   
  the word “litre”  
  space, dash, space “ - “  
  the type.*

**CLIENT MODULE**

Here is the tester file for part one of w7 main module.

|  |  |
| --- | --- |
| // OOP244 Workshop 7: compositions  // File w7\_part1.cpp  // Version 1.0  // Date 7/13/2015  // Author Franz Newland  // Description  // This file demonstrates the client  // module of Engine.cpp  /////////////////////////////////////////////  #include "Engine.h"  #include <iostream>  using namespace std;  using namespace oop244;  int main(){  Engine sixLitreV8(6.0, "V8");  cout << "----------" << endl;  sixLitreV8.display(std::cout)<<endl;  cout << "----------" << endl;  return 0;  } | ---------------  6.00 litre - V8  --------------- |

**part 2: SHIP Class**

Design and code a class named Ship that holds information about a ship used in a simulator. Place your class definition in a header file named **Ship.h** and your function definitions in an implementation file named **Ship.cpp**.  Include in your solution all of the statements necessary for your code to compile under a standard C++ compiler. The Ship class contains an array of Engine objects of fixed size 10, a character array for the ship type of fixed size 7 and an integer storing the number of engines (<=10) currently defined for the ship.

Upon instantiation, a Ship object may receive no parameters **or** the following three parameters.

* A null-terminated C-style string holding the type of ship. This string always has a length of six, not including the null terminator.
* An array of engines, representing the ship engines used by the given ship (note large ships typically have many engines). Please note that this array is of type Engine, mentioned in Question one.
* The length of the Engine array passed in.

If the object receives a non-null string for the ship type of 6 characters or less, and an array of minimum length one for the engines, the object accepts the string as the ship’s type, and sets up the array.  Otherwise, the object assumes a ***safe empty state***.

Your design also includes the following member functions and helper operators:

* bool empty() const - a query that returns true if the object is in a safe empty state; false otherwise.
* double calculatePower() const - a query that returns the ship output power as a function of the number of engines and capacity of each engine. The output power for each engine is (engine size X 5). The ship output power is the sum of output power of each engine on the ship.
* std::ostream& display(std::ostream& os) const - a query that displays the ship type as well as the total ship power to two (2) decimal places in a field of six (6), as shown in the example below, as well as displaying the individual engine types and capacities used in the evaluation.  If the current object is empty, this function does nothing.
* bool operator<(double)-a n operator that compares the total output power this ship against the right hand operand. This function returns true if the total power of this ship is below the value provided, false otherwise.
* std::ostream& operator<<(std::ostream&, const Ship&) - A non-friend helper insertion operator that outputs the contents of the right-hand operator to the stream provided. This operator allows for cascading.

**CLIENT MODULE**

Here is the tester file for part two of w7 main module.

|  |  |
| --- | --- |
| // OOP244 Workshop 7: compositions  // File w7\_part2.cpp  // Version 1.0  // Date 13/7/2015  // Author Franz Hewland  // Modified by Fardad Soleimanloo  // Description  // This file demonstrates the client module of w7  /////////////////////////////////////////////////////  #include <iostream>  using namespace std;  #include "Engine.h"  #include "Ship.h"  using namespace oop244;  int const MIN = 90;  int main()  {  Engine shipEngines[3] =  {  Engine(6.0, "V6"),  Engine(8.0,"V8"),  Engine(4.2, "Inline")  };  Ship titanic("Liner", shipEngines, 3);  cout << titanic << endl;    //Comparing with the standards:  if (titanic<MIN)  cout << "Below average!" << endl;  else  cout << "Above average!" << endl;  return 0;  } | Liner- 91.00  6.00 litre - V6  8.00 litre - V8  4.20 litre - Inline  Above average! |

**SUBMISSION**

If you are planning to do the bonus, do not submit this. Submitting the bonus in this workshop is enough. But I strongly suggest keeping this version separately in case you can not submit

Create a file called feedback.txt and in it, briefly reflect upon what you have learned. Also if you have comments or concerns about this workshop please mention it here.

Upload all your modules (Ship.cpp, Ship.h, Engine.cpp, Engine.h, w7\_part2.cpp and feedback.txt) in text mode to your matrix account. Compile test and run and make sure output is an exact match to the output of w7\_part2.cpp.

Then run the following command:

> ~fardad.soleimanloo/submit\_w7<ENTER>

at your matrix console. (note that “w7\_part2.cpp will be overwritten by the original one written here)

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

**dynamic memory in class Ship – Bonus task**

In class Ship, assume there is no limit for the array of engines, representing the engines in the ship.

If the object receives a non-null string for the ship type of 6 characters or less and a positive value for the size of the array, the object accepts the string as the engine type, and sets up the array.  Otherwise, the object assumes a ***safe empty state***.

*Remember to include all of the functions necessary to ensure the proper copying and assignment of the data from one object to another and to avoid memory leaks.*

**CLIENT MODULE- Bonus task**

Here is a sample of implementation file for part two of w7 bonus module.

|  |  |
| --- | --- |
| // OOP244 Workshop 7: compositions  // File w7\_part2\_bonus.cpp  // Version 1.0  // Date 13/7/2015  // Author Franz Hewland  // Modified by Fardad Soleimanloo  // Description  // This file demonstrates the client module of w7  /////////////////////////////////////////////////////  #include <iostream>  using namespace std;  #include "Engine.h"  #include "Ship.h"  using namespace oop244;  int const MIN = 90;  int main()  {  Engine shipEngines[3] =  {  Engine(6.0, "V6"),  Engine(8.0, "V8"),  Engine(4.2, "Inline")  };  cout << "titanic -->" << endl;  Ship titanic("liner", shipEngines, 3);  cout << titanic << endl;    Ship backup = titanic;  cout << "backup -->" << endl;  cout << backup << endl;    Ship blank;  titanic = blank;  blank = backup;  cout << "blank -->" << endl;  cout << blank << endl;  cout << "titanic -->" << endl;  cout << titanic << endl;  cout << "operator< -->" << endl;  //Comparing with the standards:  if (blank<MIN)  std::cout << "Below average!" << std::endl;  else  std::cout << "Above average!" << std::endl;  return 0;  } | titanic -->  liner- 91.00  6.00 litre - V6  8.00 litre - V8  4.20 litre - Inline  backup -->  liner- 91.00  6.00 litre - V6  8.00 litre - V8  4.20 litre - Inline  blank -->  liner- 91.00  6.00 litre - V6  8.00 litre - V8  4.20 litre - Inline  titanic -->  operator< -->  Above average! |

**SUBMISSION**

Create a file called feedback.txt and in it, briefly reflect upon what you have learned. Also if you have comments or concerns about this workshop please mention it here.

Upload all your modules (Ship.cpp, Ship.h, Engine.cpp, Engine.h, w7\_part2\_bonus.cpp and feedback.txt) in text mode to your matrix account. Compile test and run and make sure output is an exact match to the output of w7\_part2\_bonus.cpp.

Then run the following command:

> ~fardad.soleimanloo/submit\_w7b<ENTER>

at your matrix console. (note that “w7\_part2\_bonus.cpp will be overwritten by the original one written here)

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

**Due Date**

Due date for this workshop is Monday July 20, 2015, 23:59.