Part B - Foundations   
  
**Derived Classes**

Workshop 8 – V1.1

In this workshop, you are to code to code an inheritance relationship between two classes.

**LEARNING OUTCOMES**

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

Inherit a class from a base class

Shadow a member function of a base class with a member function of a derived class

Define a helper function in terms of a member function of the supported class

Access a shadowed member function in a base class

Convert a shadowed method to virtual to be able to call the latest version of the method using parent’s reference or pointer. (inclusion polymorphism)

Reflect on the concepts learned in this workshop

**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.
* <https://www.youtube.com/watch?v=oW4viEA72UI>

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

**part 1: HERO Class**

Design and code a class named **Hero** that holds information about a character in a game. Place your class definition in a header file named **Hero.h** and your function definitions in an implementation file named **Hero.cpp**.  Include in your design all of the statements necessary to compile and to run your code successfully under a standard C++ compiler.

Upon instantiation, a **Hero** object may receive **no** information or may receive **two** values:

* The address of a C-style null terminated string holding the name of the hero. This string is always of length 20 or less excluding the null terminator.
* a positive double for the strength of the hero,

If no arguments are provided, or validation fails, the object is set to a *safe empty state.* Your design includes three member functions and one ~~helper~~ (V1.1) member operator:

* **bool empty() const** - a query that returns true if the object is in a safe empty state; false otherwise.
* **double strength() const** - a query that returns the strength of the hero if the hero object is not empty. This query returns ~~nothing~~ zero if the object is empty.
* **std::ostream& display(std::ostream& os) const** - a query that receives a reference to an **ostream** object and inserts into that object the name of the hero and its strength as shown in the example below and then returns the **ostream** object.  If the current object is empty, this function does nothing.
* **bool** **operator<(const Hero &rightOperand)const** –compares the strengths of this Hero and the right-operand Hero and returns the result as a **bool**.

The following program uses your **Hero** class and produces the output shown below:

|  |  |
| --- | --- |
| // OOP244 Workshop 8: Derived Classes  // File w8\_part1.cpp  // Version 1.1  // Date 2015/05/21  // Author Franz Newland  // Modified by Fardad Soleimanloo  // Date 2015/20/7  // Description  // This file demonstrates the client  // module of w8 part1  //////////////////////////////////////////  #include <iostream>  using namespace std;  #include "Hero.h"  using namespace oop244;  int main(){  Hero stuart("Stu Dent", 400.00);  Hero trevor("T Cher", 650.00);  stuart.display(cout) << endl;  trevor.display(cout) << endl;  if (trevor<stuart)  cout << "Stewart is stronger.";  else  cout << "Trevor is stronger.";  return 0;  } | Stu Dent- 400  T Cher- 650  Trevor is stronger. |

**part 2: SuperHero Class**

Derive from the **Hero** class a class named **SuperHero** that holds information about a super hero. Place your class definition in a header file named **SuperHero.h** and your function definitions in an implementation file named **SuperHero.cpp**.  Include in your design all of the statements and keywords necessary to compile and to run your code successfully under a standard C++ compiler.

Upon instantiation, a **SuperHero** object may receive no information or may receive four values:

* an array of integers. These represent additional powers for the super hero. Note – negative values in this array should be set to 0.
* the size of the array of integers.
* the address of a C-style null terminated string holding the name of the super hero. This string is always of length 20 or less excluding the null terminator.
* a positive double for the strength of the super hero.

Your design includes the following member function:

* **std::ostream& display(std::ostream& os) const** - a query that receives a reference to an **ostream** object and inserts into that object the name of the super hero, its strength, and the names of the powers using the following mapping:  
  + **0** – failed power
  + **1-10** – firebolt
  + **11-20** – invisibility
  + **21-30** – double strength
  + **31-40** - shield
  + **>40** - speed

as shown in the example below.  If the current object is empty, this function does nothing. If the superhero has no powers, then it will display “No powers” and a newline.

At end, display will return the incoming ostream.

* Make sure you have a destructor to deallocate the memory.
* Make sure a Superhero cannot get copied or assigned to another Superhero. (Remove this restriction for bonus part).

The program on the following page uses your **Hero** and **SuperHero** classes and produces the output shown below:

|  |  |
| --- | --- |
| // OOP244 Workshop 8: Derived Classes  // File w8\_part2.cpp  // Version 1.1  // Date 2015/05/21  // Author Franz Newland  // Modified by Fardad Soleimanloo  // Date 2015/20/7  //  /////////////////////////////////////////////////////  #include <iostream>  using namespace std;  #include "SuperHero.h"  using namespace oop244;  int main()  {  int powers[] = { 2, 44, 20, -3 };  SuperHero hercules(powers, 4, "Hercules", 400);  SuperHero Nothing;  SuperHero Prof((int\*)0, 0, "Fardad", 200);  hercules.display(cout)  << "----------------------" << endl;  Prof.display(cout)  << "----------------------" << endl;  Nothing.display(cout)  << "----------------------" << endl;  \*((Hero\*)&Nothing) = Hero("Invisible Man", 300);  Nothing.display(cout)  << "----------------------" << endl;  // uncomment next line and you should get an error  // Nothing = Prof;  // uncomment next line and you should get an error  // SuperHero someone(Prof);  return 0; } | Hercules- 400  Superhero's powers:  2-firebolt  44-speed  20-invisibility  0-failed power  ----------------------  Fardad- 200  No Powers!  ----------------------  ----------------------  Invisible Man- 300  No Powers!  ---------------------- |

**part 3: applying virtuals for proper printing and preventing memory leak.**

Add the following to your  **Hero** class:

* **A virtual empty destructor**
* **Make the display function virutal**
* Overload **operator<<** to be able to print the **Hero** class with **cout**. Note that **operator<<** only calls the display method and returns it.

The program on the following page uses your **Hero** and **SuperHero** classes and produces the output shown below:

|  |  |
| --- | --- |
| // OOP244 Workshop 8: Derived Classes  // File w8\_part3.cpp  // Version 1.1  // Date 2015/05/21  // Author Franz Newland  // Modified by Fardad Soleimanloo  // Date 2015/20/7  //  /////////////////////////////////////////////////////  #include <iostream>  using namespace std;  #include "SuperHero.h"  using namespace oop244;  int main()  {  int powers[] = { 2, 44, 20, -3 };  SuperHero hercules(powers, 4, "Hercules", 400);  SuperHero Nothing;  SuperHero Prof((int\*)0, 0, "Fardad", 200);  cout << hercules  << "----------------" << endl;  cout << Prof  << "----------------" << endl;  cout << Nothing  << "----------------" << endl;  \*((Hero\*)&Nothing) = Hero("Invisible Man", 300);  cout << Nothing  << "----------------" << endl;  // uncomment next line and you should get an error  // Nothing = Prof;  // uncomment next line and you should get an error  // SuperHero someone(Prof);  return 0;  } | Hercules- 400  Superhero's powers:  2-firebolt  44-speed  20-invisibility  0-failed power  ----------------  Fardad- 200  No Powers!  ----------------  ----------------  Invisible Man- 300  No Powers!  ---------------- |

**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 (Hero.cpp, Hero.h, SuperHero.cpp, SuperHero.h, w8\_part3.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 w8\_bonus.cpp.

Then run the following command:

> ~fardad.soleimanloo/submit\_w8<ENTER>

at your matrix console. (note that “w8\_part3.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 Thursday July 30, 2015, 23:59.

**bonus: SuperHero Class**

Add a copy constructor and an assignment operator to make sure SuperHero can safely get coped or assigned to another SuperHero. (Remove the deleted copy constructor an assignment operator implemented in part 2)

Add the following three operators.

* SuperHero operator+(int rightOperand)const;- an operator that takes an integer representing a new number to be added to the list of powers. This operator creates a copy of the owner and adds the number to the array of integers and then returns it. (Note that you should resize the allocated memory to and add one element to it).
* SuperHero operator+(int leftOperand, const SuperHero& rightOperand);- a helper operator overload that works exactly like the previous operator overload. The only difference is that the left operand in this operator is an int and the right operand is a SuperHero.
* SuperHero& operator+=(int rightOperand);- an operator that takes an integer representing a new number to be added to the list of powers. This operator adds the number to the array of integers.

The program on the following page uses your **Hero, SuperHero** bonus class and produces the output shown below:

|  |  |
| --- | --- |
| // OOP244 Workshop 8: Derived Classes  // File w8\_bonus.cpp  // Version 1.1  // Date 2015/05/21  // Author Franz Newland  // Modified by Fardad Soleimanloo  // Date 2015/20/7  /////////////////////////////////////////////////////  #include <iostream>  using namespace std;  #include "SuperHero.h"  using namespace oop244;  int main()  {  int powers[] = { 2, 44, 20, -3 };  SuperHero hercules(powers, 4, "Hercules", 400);  SuperHero Hcopy = hercules;  SuperHero Empty;  SuperHero Prof((int\*)0, 0, "Fardad", 200);  cout << "hercules-----------"  << endl << hercules << endl;  cout << "Hcopy--------------"  << endl << Hcopy << endl;  cout << "Prof---------------"  << endl << Prof << endl;  cout << "Empty--------------"  << endl << Empty << endl;  Hcopy = Empty;  cout << "Hcopy--------------"  << endl << Hcopy << endl;  Prof += 11;  cout << "Prof---------------"  << endl << Prof << endl;  Prof = hercules + 32;  cout << "Prof---------------"  << endl << Prof << endl;  hercules = 13 + hercules;  cout << "hercules-----------"  << endl << hercules << endl;  return 0;  } | hercules-----------  Hercules- 400  Superhero's powers:  2-firebolt  44-speed  20-invisibility  0-failed power  Hcopy--------------  Hercules- 400  Superhero's powers:  2-firebolt  44-speed  20-invisibility  0-failed power  Prof---------------  Fardad- 200  No Powers!  Empty--------------  Hcopy--------------  Prof---------------  Fardad- 200  Superhero's powers:  11-invisibility  Prof---------------  Hercules- 400  Superhero's powers:  2-firebolt  44-speed  20-invisibility  0-failed power  32-shield  hercules-----------  Hercules- 400  Superhero's powers:  2-firebolt  44-speed  20-invisibility  0-failed power  13-invisibility |

**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 (Hero.cpp, Hero.h, SuperHero.cpp, SuperHero.h, w8\_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 w8\_bonus.cpp.

Then run the following command:

> ~fardad.soleimanloo/submit\_w8b<ENTER>

at your matrix console. (note that “w8\_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 Thursday July 30, 2015, 23:59.