SCIT-EIS-UOW
CSCI251 - Advanced Programming
Autumn 2022
Lab 4 for Week 8 & 9

General note

We will work with an IDE (such as Code::Blocks or similar), or to use Bitvise or similar to connect to `capa.its.uow.edu.au`. Provide a README file to confirm which compiler/IDE environment you use (in general, someone else needs to re-compile your source codes). In addition, you also need to record which file is for which task in README. Submission points will be set up on Moodle close about midnight Fridays (week 3, 5, 7, 9, 11).

For each task you might save the code file(s) like

- `t1.cpp`: only one file.
- `t2-1.cpp, t2-2-lib.h, t2-3-lib.cpp`: if multiple subtasks or files then put in folder T2

At the end, all the files (t1.cpp, t3.cpp...), sub-folders (T2, T4...), and README file will place in one big folder named studentname_studentID_lab-order (e.g. Ngoc_007_Lab4). Then zip this folder to the zip file and submit this zip file via Moodle.

1 Task One: Debugging

1. Buggy inheritance: Fix `Debug-A.cpp`. The run of this program, with input as shown (Blob etc.), should be:

```
Enter painting's title Blob
Enter artist Degas
Enter painting's title Blob
Enter artist Alice
Enter painting's title Blob
Enter artist Bob
Enter painting's title Blob
Enter artist Picasso

Blob by Degas value $25000
Blob by Alice value $400
Blob by Bob value $400
Blob by Picasso value $25000
```

2. Why will the following fail to compile?

```
class B
{
public:
virtual void X() = 0;
```

```
};

class D : B
{
public:
virtual void X() {cout << "D object" << endl;}
};

int main()
{
B objB;
}
```

2 Task Two: Relationships

1. A Box in a Box in a Box: Write code in `Box.cpp` to represent storing a collection of Boxes, i.e. `Box` objects, within a `Box` object. You will need functionality to add a `Box` to a `Box`. You can play around with this idea a fair bit, and potentially include limits on the number of Boxes or the size and capacity of Boxes to make sure that a `Box` cannot be added to itself.

2. Consider `A-Class.cpp`.

   (a) What concept does it illustrate?

   (b) How are the classes related?

   (c) What happens if either the worker or company is deleted in main, after the Contract is set up but before the output is displayed?

   (d) What happens if we add an additional company and add a contract between John and the new company? How would we manage this situation, or the case where a new worker wants to work for Bell?

3 Task Three: Inheritance, abstraction, and polymorphism

1. **An Animal hierarchy**:

   (a) Modify the provided `Cat` class so it is derived from a base class `Animal`.. Write the base class `Animal`. Make another subclass of `Animal` and make sure there is some functionality distinguishing them. Write a main function to illustrate creating instances of all three classes.

   (b) Make the base class `Animal` abstract, and modify the derived classes if necessary to make sure they are not–abstract classes. The Animal instance can be commented out in main for the version you submit.

   (c) Write driver code containing a vector of pointers to `Animal` in which you store multiple different types within the `Animal` hierarchy, and demonstrate adding different types of `Animal` to this vector.

2. **A Transportation hierarchy** : For each of the classes below you should provide sufficient functionality to test your understanding of the relationships between the classes. You should, in particular, see how you can use constructors across classes. The code should be able to compile but you don't need to store data or have other within each class at this point.

   (a) Define a base class `Transportation`.

   (b) Define three derived classes, `SeaTransport`, `LandTransport`, and `AirTransport`.

(c) Inheritance can occur across multiple levels. Define `Car` and `Canoe` as classes derived from appropriate classes.

(d) It's possible to have multiple inheritance in C++. Define `Hovercraft` as derived from two appropriate classes.

3. This involves extending the `Transportation` hierarchy set up above.

   (a) Now add some data elements and member functions to support the hierarchy.

   (b) Make `Transportation` an abstract class by adding an appropriate `display()` function.

   (c) Make a collection that could contain various different forms of transport. Set different elements of the collection to refer to a range of different forms of transport.

   (d) You can add in additional types of transport if you like.

   (e) Illustrate the polymorphism present in the use of the `display()` function.

   (f) What is the diamond problem. Explore how you deal with that problem in the context of the `HoverCraft`?

4 Task Four: Libraries

This section is quite methodical but it's important to know how to do this because you will need to use a provided library in A3.

1. The files `driver.cpp`, `mylibrary.cpp`, and `mylibrary.h` form a program.

   (a) Produce an executable from those files.

   ```
   g++ driver.cpp mylibrary.cpp -o M1
   ```

   (b) Convert the non–driver cpp file into a static library `libcode.a`.

   ```
   g++ -c mylibrary.cpp
   ar -crv libcode.a mylibrary.o
   ```

   (c) Generate an executable.

   ```
   g++ driver.cpp libcode.a -o M2
   ```

   (d) Convert the non—driver cpp file into a shared (dynamically linked) library.

   ```
   LD_LIBRARY_PATH=.
   export LD_LIBRARY_PATH
   g++ -fpic -shared -o libcode.so mylibrary.cpp
   ```

   (e) Generate an executable.

   ```
   g++ -I. -L. driver.cpp -lcode -o M3
   ```

2. How do the 3 executables differ in size?

3. Which of the 3 executables run without the corresponding library being present?

4. Using a prebuilt static libary: The file `mash.h` is the header file associated with the static library in `libmash.a`. Write a program `mainMash.cpp` that includes the header to access the function mash.

   (a) In your `main()` you should apply mash to each command line argument, including the first. Note that you can an convert `argv[i]` to a string by initialising a string with the C–string as an argument, so using

```
string temp(argv[i]);
```

(b) As per the instructions in the lecture notes you can compile to link the library in using the following:

```
g++ mainMash.cpp libmash.a -o Mash
```

(c) What does `mash` appear to do?

i. What happens if you feed the input back in? Consider feeding 83 in, then the output, then the output, ...

ii. How is the pigeon–hole principle relevant?

5 Task Five: The Three Little Software Engineers

Carry out these tasks for `The Three Little Pigs`, a story in `3Pigs.txt`. Ideally you should be discussing this with other people.

1. Write a timeline identifying the events in the story.
2. Identify appropriate objects/classes and attributes/behaviours for those classes.
3. Sketch a class diagram showing the classes and appropriate relationships. This may be a somewhat iterative process with multiple versions.
4. Write code to implement the identified structures.
5. Write `main()` function(s) which when run will approximate the events of the story, or similar stories. It can be interactive if you like, to cover variations on the stories.