C++: Practical session 8

1 LSC group class structure

Consider the LSC research group. It is made up of a number of people, who have varying rôles and attributes. The following roles exist:

- Head of group
- Post-docs
- Ph.D. students
- MPhil students
- Interns
- Visitors

These have one, more, or none of the following attributes:

- Income (called salary for some people, and grant for others)
- Number of students supervised
- Hours of lectures given per week
- Hours of lectures taken per week
- Submission deadline

Design a class structure that implements this hierarchy. You only need to write the class definitions, not the definitions of all the necessary functions. There should be a consistent base-class for all the classes, called `Researcher`, which contains a researcher's name and title.

In the course of designing the class structure, you should use virtual functions and pure virtual functions, protected and private member data, and also publically available data-access functions. You may wish to implement functions that modify the classes, such as setting a new income value or assigning more lectures to them.

You may also wish to have a separate abstract `Lecturer` class, leading to a `LecturerPostDoc` class. How could you then use `dynamic_cast` to detect whether a `PostDoc` is also a `Lecturer`?

1.1 Detailed example

If you wish a more directed task than the somewhat open-ended problem above, then try to implement a class hierarchy so that the following code compiles:

```
int main(void)
{
  std::array<Researcher*,5> group;
  group[0] = new MPhilStudent("Zebedee", 18);
  group[1] = new PhdStudent("Florence");
  group[2] = new PostDoc("Dougal", 6);
  group[3] = new HeadOfGroup("Ermintrude");
  group[4] = new Visitor("Rusty");

  for(unsigned int i=0 ; i < 5 ; i++)
  {
    std::cout << "Researcher " << i << " is called " <<
   group[i]->name() << " and is a " << group[i]->title() <<
   std::endl;

    Student* s = dynamic_cast<Student*>(group[i]);
    if(s)
    {
      std::cout << s->name() << " is a student and must attend
   " << s->lectureHoursTakenPerWeek() << " hours of lectures
   per week" << std::endl;
    }

    Staff* s2 = dynamic_cast<Staff*>(group[i]);
    if(s2)
    {
      std::cout << s2->name() << " is a member of staff and
   must give " << s2->lectureHoursGivenPerWeek() << " hours of
   lectures per week" << std::endl;
    }

    if(!s2 && !s)
    {
      std::cout << group[i]->name() << " is neither student
   nor staff" << std::endl;
    }
    std::cout << "__________________________" << std::endl;
  }

  for(unsigned int i=0 ; i < 5 ; i++)
  {
    delete group[i];
  }
  return 0;
}
```

and produces the output:

```
Researcher 0 is called Zebedee and is a MPhil Student
Zebedee is a student and must attend 18 hours of lectures per week
__________________________
Researcher 1 is called Florence and is a PhD Student
Florence is a student and must attend 0 hours of lectures per week
__________________________
Researcher 2 is called Dougal and is a Post Doc.
Dougal is a member of staff and must give 6 hours of lectures per week
__________________________
Researcher 3 is called Ermintrude and is a Head Of Group
Ermintrude is a member of staff and must give 0 hours of lectures per week
__________________________
Researcher 4 is called Rusty and is a Visitor
Rusty is neither student nor staff
```

```
-------------------------
```

Each constructor takes the person's name and (potentially) the number of hours for which they must either lecture or be present in a lecture. This will default to zero for PhD students and Heads of Group.

2 More class structures

It is a well-known fact that a circle is a special case of an ellipse. Construct a pair of classes `Circle` and `Ellipse`, such that one derives from the other.

The `Circle` class should have a `getRadius()` function, and the `Ellipse` class should have a `setAxes(a,b)` function.

If you have difficulty with this, try to track down the precise cause of your difficulty.

For a full discussion of this problem, see `https://isocpp.org/wiki/faq/proper-inheritance#circle-ellipse`