CCNU-UOW
CSCI851 Advanced Programming
Autumn 2020

Prof. Zhifeng Wang

Laboratory Exercise 6 (Week 6)

1 Task One: Warm–up exercises

1. A makefile is provided, with associated files. Try using it. What is wrong? Fix it. Have a look at `Makefile_RACAH` too, it's an example of a makefile supporting compilation for multiple platforms.

2. Debug: `Debug-A.cpp`. The output should be

```
ID # 111
Name: Alice Anteater
Salary: $23.45
```

2 Task Two: Goldilocks & the 3 Software Engineers: Part One

You should work in groups of two or three for this task. Describe the data members and methods associated with classes and objects necessary to model the story `"Goldilocks and The Three Bears"`. A copy of the story is in the file `Goldilocks.txt`.

3 Task Three: A beginner class

Write code `Basic.cpp` with a class definition for the class `Basic`. The class should have a single private integer field and a single function, either a public constructor or, if you don't know how to implement a constructor, a public function. The constructor or function should simply display the message "Constructing". Include a `main()` function to demonstrate the instantiation of one `Basic` object.

4 Task Four: Not so private

1. What is this doing?

   Comment out the illegal lines and play around with the impact of changing variable types in places.

```
class thing {
public:
    int value1 = 5;
    void display(){cout << value2 << endl;}
private:
    int value2 = 77;
};
```

```
int main()
{
    thing A;
    cout << A.value1 << endl;
    cout << A.value2 << endl;
    cout << *(&A.value1 + 1) << endl;
    A.display();
    cin >> *(&(A.value1) + 1);
    A.display();
    return 0;
}
```

2. Here goes a similar restriction broken...

```
class thing {
public:
    int value1 = 5;
    const int value2 = 77;
};

int main()
{
    thing A;
    cout << A.value1 << "  " << A.value2 << endl;
    cin >> *(&(A.value1) + 1);
    cout << A.value2 << endl;
    return 0;
}
```

5 Task Five: A basic class

In the lab for Week 4 you were supposed to write a `Dog` struct. Modify this so you have a `Cat` class. Again you can use functions or a constructor. You can use them in `main()` to demonstrate the class operates correctly. Here goes some sample output.

```
Cat: Tigger is a Fluffy unit.
The cat's age is 3.
License fee: $10.
```

6 Task Six: A less basic class

Create a piece of code `Bonus.cpp`. This should contain a class `Staff` with the following:

- Data fields holding a staff number, last name, first name, base salary, sales made, staff class, and the bonus for which the person is eligible.
- Constant static fields holding the bonus rates per sale, as a percentage of salary, in accordance with the following array:

Sales	Class A	Class B	Class C
$0-20$	0.03	0.02	0.005
$21-50$	0.05	0.035	0.015
51+	0.075	0.055	0.04

- Include a static function that displays the bonus table.
- Include a function `setFields()` to set all the field values on the basis of the staff number, last and first names, base salary, sales made and staff class.
- Include a function `computeBonus()` to determine the bonus earnt.
- Include a function `display()` to display all the information for a staff member.
- In the `main()` function you should display the Bonus table, construct a couple of `Staff` objects, use `setFields()` for each, passing whatever fields you want to demonstrate the functionality and finally run the `display()` function for each. Note that after `setFields()` runs the bonus should have been determined.

In writing the code you should consider carefully whether data fields and functions should be public or private. You should call the fields by appropriate names.

There is an example of output in `Sample.txt`.

There is no need to make the output particular flash in terms of aligning columns exactly and so on.

7 * Task Seven: Analysis of `typeid().before();`

You are determine the purpose of the `before()` member function of `typeid`. The syntax is as follows:

```
typeid(float).before(typeid(int))
typeid(x).before(typeid(y))
```

You should determine what `before(...)` does, in particular in the context of the types

`int, float, double, string, char, long int, char[], bool.`

and some ADT's that you define yourself. Call them W, X, Y and Z or something similar.

What difference does forward referencing make?