Programming in C and C++ — Course: CH-230-A
Jacobs University Bremen — November 19th, 2019
Dr. Kinga Lipskoch

Assignment 12 - Inheritence, `static` Members, Operator Overloading, Polymorphism

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules:
  https://grader.eecs.jacobs-university.de/courses/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 12.1 *`Hexagon` class* (1 point)
Presence assignment, due by 11:00 AM today **Graded manually**
Language: C++
Download the files:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Shapes.h`
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Shapes.cpp`
Write a class called `Hexagon` which is derived from the `RegularPolygon` class. A hexagon has a side and a color. Provide methods for computing the perimeter and the area of a hexagon. For each property appropriate setter and getter methods need to be provided (i.e., it should not be possible to manipulate data directly). The class should also a parametric constructor, a copy constructor and a destructor.

Note, that the area of a hexagon of side `t` can be computed by using the formula $\frac{3\sqrt{3}}{2}\mathtt{t}^2$.
Name the files `Shapes.h` and `Shapes.cpp`. Then write a testing program `testHexagon.cpp` that:

a) creates a blue hexagon that has the side of 9,

b) creates a green hexagon that has the side of 15,

c) creates a copy for the second hexagon using the copy constructor, and

d) computes the perimeter and area of all three hexagons and prints the results on the screen.

Problem 12.2 *`TournamentMember` class* (2 points)
Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++
Imagine that you are in the design stage of a software system for handling the data of the participants of a major soccer tournament. As different roles will be present (players, coaches, referees, etc.) you are required to develop a class handling the data of a generic league member. For each person the following data is at least needed

- first name as character array (36 characters characters including '`\0`')
- last name as character array (36 characters)
- date of birth as character array (11 characters, storage format is `yyyy-mm-dd`)

In addition the whole team is located somewhere, so `location` is an additional `static` property of the class.
Design and implement a class for holding these data. In addition add at least two other general properties to this class.
The class, which will be called `TournamentMember`, should provide constructors (empty and parametric), a destructor and also a copy constructor. The class should also provide `inline` setter and getter methods (either inside or outside of the class).

Moreover, in order to carry out the functionality of the application, the following methods are required:

- a method which prints the information of a tournament member on the screen,
- a method which changes the `location`.

Also all constructors and the destructor should print a short informational message on the screen, such that you can see which is being called when.
You should provide three files: a header file named `TournamentMember.h` with the declaration of the class, a file named `TournamentMember.cpp` with its definition, and an additional file called `testTournamentMember.cpp` with a `main()` function which tests the functionality of the class.
The needed data can be initialized in the code from the `main()` function.

Problem 12.3 *`Player` class* (1 point)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++
A `Player` class should be derived from the `TournamentMember` class. It holds additional properties such as number, position, number of goals scored, and whether the player is left-footed or right-footed. For each property appropriate setter (except the number of goals scored) and getter methods need to be provided as inline methods, and it should not be possible to manipulate data directly. An appropriate constructor to set all properties on creation should be provided as well as a copy constructor that creates a correct copy of a player, and a destructor. Also all constructors and the destructor should print a short informational message on the screen such that you can see which is being called when.
Also the following methods are required:

- a method which prints all the information of a player on the screen,
- a method which increments the number of goals scored by a player.

Add code to the files `TournamentMember.h`, `TournamentMember.cpp`, and write a testing program named `testPlayer.cpp` that tests the functionality of the `Player` class. Create three players with different properties. Then move all players to the location `"Hamburg"`.
The needed data can be initialized in the code from the `main()` function.

Bonus Problem 12.4 *`Referee` class* (3 points)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++
Write a class named `Referee` derived from the `TournamentMember` class. This class should have the following additional properties:

- `int yellowCardCount;`
- `Player *yellowCardList[40];`
- `int redCardCount;`
- `Player *redCardList[40];`

And the following methods should be implemented:

- `bool addToYellowCardList(Player *p);`
- `bool addToRedCardList(Player *p);`

If the player `p` is not yet on the yellow card list then it should be added to it, but if the player `p` is already on the yellow card list then the player should be removed from the yellow card list and should be added to the red card list. If the player `p` is not yet on the red card list then it should be added to it, but if the player `p` is already on the red card list then nothing should happen. Both methods should return `true` if the adding was successful and `false` if not.
Add code to the files `TournamentMember.h`, `TournamentMember.cpp`, and write a testing program named `testReferee.cpp` that tests the functionality of the `Referee` class. Create a referee and some players. Your code should illustrate the referee actions for adding players to the list of players with yellow cards and to the list of players with red cards.
You can assume that the input or the setting of the data will be valid.

Bonus Problem 12.5 *Choosing a random color* (1 point)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++
Write a function which randomly chooses one color from `RED`, `BLACK`, `VIOLET` and `BLUE`. Write a program which calls this function 15 times and prints the chosen color on the screen. You can adapt and modify the program below.
The following code snippet simulates a die throw 10 times. Use an array of strings for the colors and then randomly choose a color by its index.

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main()
{
    int die;
    int count = 0;
    int randomNumber;
    // init random number generator
    srand(static_cast<unsigned int>(time(0)));
    while (count < 10) {
        randomNumber = rand();
        die = (randomNumber % 6) + 1;
        cout << count << ": " << die << endl;
        count++;
    }
    return 0;
}
```

Problem 12.6 *Fractions I* (2 points)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++

- As a starting point, use the files `fraction.h, fraction.cpp, testfraction.cpp` (which you can download from:
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/fraction.h,`
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/fraction.cpp,`
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/testfraction.cpp`).
- Replace the method `print()` by an overloaded operator << such that you can use `cout <<` for printing a fraction on the screen.
- Overload the operator >> such that you can enter from the keyboard a fraction using `cin >>`. Check the validity of the input (you can assume that the numerator and denominator will be numbers).
- Overload the operators * and / for computing the multiplication and division of two fractions.
- In your testing program you should then be able to enter two fractional numbers (using `cin >>`), then the product and quotient are printed on the screen (one per line using the overloaded operator and `cout <<`).

Use the suggestions from slide 10 (Lecture 12) for choosing to write member methods or friend functions.
You can assume that the input will be valid.

Problem 12.7 *Fractions II* (2 points)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++
Continue with your program for **Problem 12.6** in the following manner. Remember to check the mathematical validity of the parameters (you can assume that the nominator and the denominator are always numbers). Use the suggestions from slide 10 (Lecture 12) for choosing to write member methods or friend functions.

Also consider the suggestions regarding the return types and parameter types.

- Overload the operators +, − for computing the sum and difference of two fractions.
- Overload the operator = for assigning.
- Overload the operators < and > to compare two fractions.
- In your testing program you should be able to enter two fractions from the keyboard using `cin >>`. Determine the greater fraction and print it on the screen using `cout <<`. Also compute the sum and the difference of the two fractions (storing the result in other objects) and print them on the screen (one per line using the overloaded operator and `cout <<`).

In order to implement the addition and subtraction of two fractions you will have to calculate the lowest common multiple (LCM) of the denominators of the two fractions. The LCM can be computed according to the formula: $\texttt{LCM(a,b)} = \frac{\texttt{a} \cdot \texttt{b}}{\texttt{GCD(a,b)}}$, where GCD is the greatest common divisor.

Operations with fractions

Addition:

$$\frac{a}{b} + \frac{c}{d} = \frac{a \cdot \mathrm{LCM}(b,d)/b + c \cdot \mathrm{LCM}(b,d)/d}{\mathrm{LCM}(b,d)}$$

Subtraction:

$$\frac{a}{b} - \frac{c}{d} = \frac{a \cdot \mathrm{LCM}(b,d)/b - c \cdot \mathrm{LCM}(b,d)/d}{\mathrm{LCM}(b,d)}$$

Multiplication:

$$\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$$

Division:

$$\frac{a}{b} \Big/ \frac{c}{d} = \frac{a \cdot d}{b \cdot c}$$

Problem 12.8 *Polymorphism I* (2 points)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++

- Download and unzip the archive:
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/a12.zip`
  For compiling the files you can use: 1) compile the files as part of a project or 2) from the terminal using `g++`.
- Draw (using ASCII characters) a diagram how these classes relate to each other and put this into `testvirtual.cpp` as part of your comments.
- For each numbered point in the file `testvirtual.cpp` add a detailed comment about what is happening in the program.
- Like in `Circle.cpp`, output a message on the screen when the method `calcArea()` is being called in any of the classes.
- Add a method to calculate the perimeter for each class definition.
- Change the test program to additionally print the total perimeter of all objects.
- Also print a message on the screen when the method `calcPerimeter()` is being called.
- Add a `Square` class (consisting of a header file and a cpp file), and add a square object to your test program. Consider the relation of a square to the other classes.

Submit a **zip** file containing all your `.h` and `.cpp` files related to this problem.

Problem 12.9 *Polymorphism II* (1 point)

Due by Monday, November 25th, 23:00 **Graded manually**
Language: C++
Change `testvirtual.cpp` such that 25 objects (circles, rings, rectangles, squares) are randomly created at runtime. Their colors (`RED`, `BLACK`, `VIOLET` and `BLUE`) and sizes (between 5 and 100) should also be randomly chosen.

Compute the area and the perimeter for each object and print them on the screen.
Submit a **zip** file containing all your `.h` and `.cpp` files related to this problem.

How to submit your solutions

- Your source code should be properly indented and compile with `gcc` or `g++` depending on the problem without any errors or warnings (You can use `gcc -Wall -o program program.c` or `g++ -Wall -o program program.cpp`). Insert suitable comments (not on every line ...) to explain what your program does.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files. Each program **must** include a comment on the top like the following:

```
/*
   CH-230-A
   a12_p1.[c or cpp or h]
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://cantaloupe.eecs.jacobs-university.de`.
  If there are problems (but **only** then) you can submit the programs by sending mail to `k.lipskoch@jacobs-university.de` **with a subject line that begins with CH-230-A**.
  It is important that you do begin your subject with the coursenumber, otherwise I might have problems to identify your submission.
- Please note, that after the deadline it will not be possible to submit any solutions. It is useless to send late solutions by mail, because they will not be accepted.

This assignment is due by Monday, November 25th, 23:00.