C++ London University Session 4

Tristan Brindle

Any questions from last week’s material?

"Homework" questions from last week

- Write a new class `multiline`, which contains a vector of `points`. Which special members do you need to define for this class? Why?

- Add a member function `append(const point& p)` to your multiline class, which adds the given point to the end of the vector. Write member functions `front()` and `back()` which return the first and last points of the multiline respectively. Which of these member functions need to be declared `const`? Why?

- Which operator overloads (if any) should we add to our `multiline` class? Why? Implement those that you have chosen

- (Harder): write a `length()` member function for the `multiline` class which returns the total length of the line.

- Write a `test_multiline()` function which checks that your `multiline` class works as intended. Call this function from your `main()`.

- (Hint for testing `length()`: a multiline containing the points (0, 0), (3, 4), (15, 9) and (12, 5) in that order should have total length of 23)

“Homework” questions from last week

- *Write a new class* `multiline`*, which contains a vector of* `points`*. Which special members do you need to define for this class? Why?*

- My solution:

```
struct multiline {

    // Default constructor
    multiline() {}

    // Constructor taking two points
    multiline(const point& p1, const point& p2)
        : points{p1, p2}
    {}

    // All other special members are automatically generated

    std::vector<point> points;
};
```

“Homework” questions from last week

- *Add a member function* `append(const point& p)` *to your multiline class, which adds the given point to the end of the vector. Write member functions* `front()` *and* `back()` *which return the first and last points of the multiline respectively. Which of these member functions need to be declared* `const`*? Why?*

- My solution (member functions of class multiline):

```
void append(const point& p)
{
    points.push_back(p);
}

point front() const
{
    return points.front();
}

point back() const
{
    return points.back();
}
```

“Homework” questions from last week

- *Which operator overloads (if any) should we add to our* `multiline` *class? Why? Implement those that you have chosen*

- Remember, the golden rule is to avoid surprising behaviour from operator overloads

- For `multiline`, only == and `!=` have obvious meanings

- We may also want to provide an output stream operator overload for debugging

“Homework” questions from last week

- *Write a* `length()` *member function for the* `multiline` *class which returns the total length of the line.*
- The length of a `multiline` is the sum of the length of each line segment
- To calculate the length of each line segment, we need to use Pythagoras’s Theorem
- Implementation on Github

“Homework” questions from last week

- *Write a* `test_multiline()` *function which checks that your* `multiline` *class works as intended. Call this function from your* `main()`*.*

- For implementation, see

  https://github.com/CPPLondonUni/course_materials/tree/master/week3/homework

Any questions before we move on?

Today’s session

- “Homework” questions from last week
- Member access (public, protected, private)
- Inheritance
- Virtual functions and polymorphism
- Pointers and smart pointers

And now for something completely different

Group exercises!

https://github.com/CPPLondonUni/week4_group_exercises

“Homework”

- Finish the exercises we didn’t have time for
- (If you’re brave) upload your solutions to Github to share with the group

Online Resources

- https://isocpp.org/get-started
- cppreference.com — The bible, but aimed at experts
- cplusplus.com — Another reference site, also has a tutorial section
- learncpp.com — Free online tutorial, very up-to-date
- https://www.pluralsight.com/authors/kate-gregory - Comprehensive set of courses from an experienced C++ trainer (free trial)
- reddit.com/r/cpp_questions
- Cpplang Slack channel — https://cpplang.now.sh/ for an “invite”
- StackOverflow (but…)

Thanks for coming!

C++ London University:

- Website: cpplondonuni.com
- Github: github.com/CPPLondonUni

Where to find Tom Breza:

- On Slack: cpplang.slack.com #learn #cpplondon
- E-mail: tom@PCServiceGroup.co.uk
- Mobile: 07947451167

My stuff:

- Website: tristanbrindle.com
- Twitter: @tristanbrindle
- Github: github.com/tcbrindle

See you next time! 🙂