CPSC 427: Object-Oriented Programming

Michael J. Fischer

Lecture 9
September 28, 2016

Bar Graph Demo (continued)

Bar Graph Demo (continued)

Analysis of 08-BarGraph demo

- `main.cpp`
- `graph.hpp`
- `graph.cpp`
- `row.hpp`
- `row.cpp`
- `rowNest.hpp`
- `item.hpp`

main.cpp

Points to note:

- `run()` calls a static class method `Graph::instructions()` to print out usage information. It is called without an implicit parameter.
  By being static, the instructions can be printed before any `Graph` object is created.
- The file uses `cin.getline()` to safely read the file name into a `char` array `fname`.
  The simpler `cin >> fname` is unsafe. It should never be used.
  It would be okay if `fname` were a `string`.
- After the file has been opened, the work is done in two lines:

```
Graph curve( infile ); // Declare and construct a Graph object.
cout << curve;         // Print the graph.
```

Design issues for `main.cpp`

1. Should `instructions` be a static class method or a static constant?
2. Should `fname` be a `char[]` or a `string`? If the latter, how does one prevent buffer overrun?
3. Where should the file opening code go – in `run()` (where it is now), in `Graph`, or in a new **controller** class?

graph.hpp

Points to note:

- Class `Graph` *aggregates* 11 bars `Row`.
- The `Row` array is created by the constructor and deleted by the destructor.
- `insert()` is a private function. It creates an `Item` and inserts it into one of the `Row`s.
- `instructions()` is a `static` inline function. This shows how it is defined.
- `instructions()` could also be made out-of-line in the usual way, but the word `static` must `not` be given in the definition in the `.cpp` file; only in the declaration in the `.hpp` file.

graph.cpp

Points to note:

- The `for`-loop in the constructor does not properly handle error conditions and can get into an infinite loop.
  You should test yourself to be sure you know how to fix this problem.
- The constructor has an allocation loop. The destructor has a corresponding deallocation loop.
- `bar[index]->insert( initials, score );`
  shows the use of a subscript and a pointer dereferencing in the same statement.
- Why do we need the `*` in

  ```
  out << *bar[k] <<"\n";
  ```

Class discussion on bargraph design issues

Most of class time was spent discussing the design issues raised above in `main.cpp`, `graph.hpp`, and `graph.cpp`.

- Why is it useful for `Graph` to know the file name?
- If both `infile` and `fname` are passed as parameters to `Graph()`, the precondition that stream `infile` is opened on file `fname` cannot be checked. Why is this undesirable?
- What are the consequences of moving the file-opening code from `run()` to:
  - `main.cpp`, just after the call to `banner()`?
  - To the `Graph` constructor?
  - To a new controller class?
- Why is there a potential infinite loop in the `Graph` constructor? What should be done to fix it?