C SC 397a, Spring 2010
Assignment 2
Due: Monday, February 1 at 22:00:00 MST

Problem 1. (55 points) Range

In this problem you are to implement a simple C++ class named Range whose instances represent a range of integers. Here's the interface (i.e., the public member functions):

Range(int lower, int upper, char name)
Constructs a Range with the given lower and upper bounds, which are inclusive. Ranges have a single-character name. Assume that lower <= upper.

Example: Range r(-2,3,'r');

int contains(int value)
Returns 1 if value is in the range, 0 otherwise.

int span()
Returns the number of integers in the range. For example, given the above range r, r.span() returns 6. (Yes, unsigned or perhaps long would be a better return type but we won't worry about this for now. Consider the behavior of span() to be undefined if the number of integers in the range is more than an int can represent.)

void print()
Prints a representation of the range. Given the above range r, r.print() yields this output:

```
Range 'r': [-2..3]
```

Here is a suitable printf() format string: "Range '%c': [%d..%d]\n"

char name()
Return the name of the range.

The interface of Range must be exactly as described above—no more and no less. In particular, your version must not have any additional public member functions. The set of data members is entirely up to you, but all data members must be private.

Additionally, you are to provide the following global functions (define them at file scope—not inside a class definition—just like in C):

int equal(Range* a, Range* b)
Return 1 if a and b have the same span, 0 otherwise.

void print(Range *ranges[ ])
Invoke the print() member function on each Range in ranges, a zero-terminated array of pointers to instances of Range.

void print_in_which(int value, Range *ranges[ ])

print_in_which consults in turn each Range in ranges, a zero-terminated array. Iff (if and only if) value is contained in the Range, the name of the Range is printed.

If only a single Range contains the value, a single character, the name of the Range, is output. If two or more Ranges contain the value, a comma and a space appear between names.

If no ranges contain the value, no output is produced.

Here are examples of possible outputs: “”, “`a`”, “`x, y`”, “`., ,, :`”.

Part of this problem is dealing with the choices that C++ allows for distributing source code between files. <u>You'll lose five points if your code doesn't meet these specifications</u>:

Put the class definition of Range in Range.h. Define the body of the contains(), span(), and name() methods in the class definition. Define the body of the constructor, the print() method, and the global functions in Range.cc. You'll need to put <u>declarations</u> (not definitions) of the global functions in Range.h. (Example: extern int equal(Range* a, Range* b);) I'll be happy to inspect your solutions before the deadline to confirm your code meets these specs on source code organization. Just turnin your files and send me mail asking for an inspection.

The directory $FILES/a2[1] contains *reference versions* of Range.h, and Range.o (my Range.cc compiled with g++ -c Range.cc). If you wish to see what my version does for some particular case, you might create a test program, say mytest.cc,

```
#include "/cs/www/classes/cs397a/spring10/files/a2/Range.h"
#include <cstdio>

int main()
{
   Range t(0,0,'t');

   t.print();
}
```

which includes my header file, then compile and link with my object file:

```
% g++ mytest.cc $FILES/a2/Range.o
```

Reference versions are intended to provide a working example of the specifications. Let me know

[1] In this assignment and others I'll be using the notation $FILES to represent the directory /cs/www/classes/cs397a/spring10/files. Suggestion: Add a shell variable for this directory to your shell initialization file on lectura.

if you notice a behavior in a reference version that seems to contradict or extend the specification.

*Note: If you look at $FILES/a2/Range.h you'll see that no member function bodies appear in the class definition and that the "real" data members are hidden (it uses the "Pimpl" idiom—private implementation). **DON'T** do that in your version—you'll be both making the problem too hard and violating the specifications for code organization!*

Grading for Range

Your implementation will be tested with only one main program: $FILES/a2/rtest1.cc. In that same directory is rtest1, an executable produced by compiling my Range implementation with rtest1.cc. **The executable produced by compiling your implementation of Range with rtest1.cc must produce exactly the same output that $FILES/a2/rtest1 produces. If it does not, your score on this problem will be zero.**

Use diff to ensure that your output matches my output! Here's an example of using diff:

```
% cp $FILES/a2/rtest1.cc .            (copy in test program source)
% g++ rtest1.cc Range.cc              (build with test source and your Range.cc)
% a.out >my.output                    (capture output of your version)
% $FILES/a2/rtest1 >whm.output        (capture output of whm's version)
% diff -c my.output whm.output        (no diff output indicates no differences)
%
```

As a first step for this problem, I recommend you study rtest1.cc to understand its operation. Also, consider whether each line is syntactically valid C.

$FILES/a2/{rtest1.cc,rtest} will "freeze" at 17:00 on January 26 and won't change unless a serious problem is discovered.

Problem 2. (10 points) violate [Puzzle Problem—see section below]

Consider this class, in X.h:

```
class X {
   public:
      X(int v);
   private:
      int itsValue;
   };
```

In this problem you are to devise and describe a way to violate the encapsulation provided by C++ and, without changing the definition of X, print the value actually contained in an instance of X. Note that the body of the constructor is not shown—you can't assume that itsValue holds the value supplied to the constructor.

Here's one way to think about this problem: Given this main program,

```
#include <cstdio>
#include <cstdlib> // for atoi()
#include "X.h"

int main(int argc, char **argv)
{
   X x1(atoi(argv[1]));

   // ... print value in x1 ...
}
```

replace the comment with one or more lines of code that will print the value held in x1.

Submit your answer as a plain text file named violate.txt. It should show the code to print the value contained in x1 and also provide a concise explanation of how the code works.

Problem 3. (10 points) stackproof [Puzzle Problem—see section below]

Describe a way that, without citing any reference material on C++, one could "prove" to a skeptic that in this function,

```
void f()
{
   X x1(10);
   ...
}
```

the instance of X named x1 really does reside on the stack, not in the heap.

Submit your answer as a plain text file named stackproof.txt.

Problem 4. (Extra Credit) extra.txt

Submit a plain text file named extra.txt with the following.

(a) (1 point extra credit) Estimate how long it took you to complete this assignment. Other comments about the assignment are welcome, too. I appreciate all feedback, favorable or not.

(b) (1-3 points extra credit) Cite an interesting course-related observation (or observations) that you made while working on the assignment. The observation should have at least a little bit of depth.

Think of me saying "Good!" as one point, "Interesting!" as two points, and "Wow!" as three points. I'm looking for quality, not quantity.

Puzzle Problems

The assignments this semester will have some number of "puzzle problems". Puzzle Problems typically require a single key insight or observation to solve. A relaxed approach to puzzle problems often works well: Get an understanding of the problem as soon as possible but then spend only a few minutes at a time on it—ponder it during some spare minutes you have during the week. You may find that an answer comes to you when you think you're thinking about something else. Discussing Puzzle Problems with classmates is fine but there's a **Puzzle Problem Rule: If you suspect you've got a solution, you must drop out of the discussion immediately**, allowing your colleague(s) to have the pleasure of finding a solution on their own (regardless of whether they consider that to be "pleasure"!) If your brainstorm turns out to be a dead-end, you can rejoin the discussion, and share your dud with the others. You can discuss Puzzle Problems with persons outside the class, too, as long as you first ask them to follow the Puzzle Problem Rule.

If you pile up more than a half-hour of "CPU" time thinking about a puzzle problem and are still stuck, then it's definitely time to ask me for a hint.

Miscellaneous

Feel free to use comments to document your C++ source code as you see fit but note that no comments are required.

When writing slides and assignments I assume that you know C, but I'm still happy to answer questions about C. For example, if you don't understand the mechanics of a declaration like `Range *ranges[ ]`, I'd be happy to explain that to you.

Keep in mind that because C++ is a superset of C you already know a lot of C++. For example, as mentioned on slide 8, just about everything you know about C data types and control structures is applicable in C++.

Along with your knowledge of C, you should be able to do this assignment using only the material on slides 1-37. If you feel that you need to bring in additional elements of C++, you're probably making the problem too hard.

Deliverables

Use `turnin` on `lectura` with the tag `397a_2` to submit your solutions for grading. The deliverables for this assignment are `Range.h`, `Range.cc`, `violate.txt`, `stackproof.txt`, and if you wish, `extra.txt`.

If you develop your solutions on a machine other than `lectura`, be sure to allow time to ensure that your solutions properly compile and run on `lectura`.