cis15-fall2007-sklar, assignment III, part 1

instructions

- This is the first part of the assignment for Unit III. It is worth 5 points.
- The second part will distributed on or before October 22. The second part will also be worth 5 points.
- **Both parts are due on Monday October 29, but try to finish part 1 before the lab on Monday Oct 22, so you can try and decode each other's ciphers (see explanation below).**
- **Both parts of the assignment must be submitted by email. Follow these emailing instructions:**
  1. Create a mail message addressed to **sklar@sci.brooklyn.cuny.edu** with the subject line **cis15 hw3**.
  2. Write your name, that is the name under which you registered for the course, in the email. When I get an email from `deathmetal@aol.com` or `pinkprincess@yahoo.com`, I can usually guess whose program it is, but that is not as good as *knowing* whose program it is.
  3. Attach ONLY the **.cpp** source code file created below.
  4. Failure to follow these instructions will result in points being taken away from your grade. The number of points will be in proportion to the extent to which you did not follow instructions . . . (which can make it a lot harder for me to grade your work)

program description

For this assignment, you will develop a program that performs some simple cryptography. The program will allow the user to enter a string and a cryptographic key, and the program will then encrypt the string using the key and send the output of the encryption to a file.
To write this program, you will create four classes, each with multiple data and function members, and a `main()`. Each class is described in detail below, with step-by-step instructions for developing the various components of each class and testing the components individually. In the end, you'll write the `main()` and put all the pieces together—and this final product is what you'll submit.
The intermediary test programs are for your benefit as a developer and should not be submitted. Incidentally, these are called *unit tests* and are created to let you test and debug the individual units of a complex program. By the end of the semester, you should have gained the skills and experience to devise and construct your own unit tests, without me giving you step-by-step instructions.

A. the "message" class

Your final program will need to accept input as a string, but, because of the way that we will encrypt the message, it needs to be able to handle the string as a sequence of characters. The `message` class provides this functionality.

1. Create a `message` class. The class should have a `string` data member called $content$ and an `int` data member called $location$.

   The class should have two constructors:

   - One constructor should take no arguments. It should initialise $content$ to the null string `""` and $location$ to 0.
   - The other constructor should take a `string` as an argument. It should initialise $content$ to the string that is its argument and $location$ to 0.

   The class should have four function members:

   - `void print() const;`
     This function prints out the values of the data member $content$.

- `void addToContent( string next );`
  This function adds the string $next$ to the data member $content$.
- `char getNextChar()`
  This function is used to return a character from $content$. The first time that `getNextChar()` is called, it returns the first character from $content$, the second time `getNextChar()` is called, it returns the second character from $content$ and so on.
  Use the data member $location$ to keep track of which character to return next, and remember that a string is just an array of characters.
- `bool empty()`
  This function returns true if you have read to the end of the string $content$. If you are at the end of the string, then the next character to read will be the null character `'\0'`.

2. Create a `main()` for testing that creates an object `myMessage` of the `message` class, prompts for a string, reads one in, sets $content$ of `myMessage` to the string that was entered, and then uses the `print` method of `myMessage` to print the string out.
   This `main` is for unit testing, you don't hand it in.

B. the "caesar" class

1. Create a class called `caesar` that has one `int` data member called $shift$.
   The class should have two constructors:
   - One constructor should take no arguments. It should do nothing.
   - The other constructor should take an `int` as an argument, and set $shift$ to the value of that `int`.

   The class should have two function members:
   - One function member $setShift$ has an `int` as an argument, and sets $shift$ to the value of that argument.
   - The other function member $encrypt$ takes a character c as an argument and returns a character.
     $encrypt$ applies the Caesar cipher to its argument. It performs a static cast on c tomake it into an integer, adds the shift to the integer (remembering to apply modulus arithmetic so that applying a shift of 2 to the letter `z` will give `b`), and then casts the result back to a character.
2. Modify the `main()` that you created in part A to prompt for and read in a character, create an instance of the `caesar` class, and use the `encrypt` method to encrypt that character.
   Again, this `main()` is for unit testing only. Don't hand it in.

C. the "fileHandleChar" class

1. Create a `fileHandleChar` class that will be used as an interface between your program and an output file. It should have the following data and function members:
   - A data member of type `ofstream`
   - A constructor that opens the file `output.txt` for output using `ios::app` so that each time the file is used, new data gets added to the end of the file.
   - A destructor that closes the file `output.txt`.
   - A function member `passToFile` that takes a character as an argument and writes that character to the file `output.txt`.
2. Now modify the `main()` from the previous step so that it creates an object of type `fileHandleChar` and uses it to write the encrypted character to a file.

D. the "encryption" class

1. Create an `encryption` class with the following :

   - A data member of type `caesar`
   - A data member of type `message`
   - A data member of type `fileHandleChar`.
   - A function member called `encryptThisString`
     This function takes a string and an integer as its input, and uses these, along with the methods from `message` and `caesar` to encrypt the string by shifting the characters the amount of the integer

2. Now modify the `main()` from the previous step so that it creates an object of type `encryption`, passes it the string (called the "plain text") and the integer (called the "encryption key") that the user enters, and then uses `encryptThisString` to encrypt the string and send the encrypted string (the "cipher text") to the file `output.txt`.

   You should hand this `main` in.

E. marking rubric

This part of the assignment is worth 5 points. The breakdown is as follows:

- `message` class with its two data members, its two constructors and its four function members

  (1 point)

- `caesar` class with its one data member ($shift$), two constructors and two function members ($setShift()$ and $encrypt()$)

  (1 point)

- `fileHandleChar` class with one data member, constructor, destructor and function member.

  (1 point)

- `encryption` class, with 3 data members and one function member.

  (1 point)

- `main()` which reads in the encryption key and the plain text and outputs the cipher text to a file called `output.txt`:

  (1 point)