drill12.cpp – the first two drill exercises from Chapter 16

drill3.cpp – the third drill exercise from Chapter 15

Drill

Function graphing drill:

1. Make an empty 600-by-600 **Window** labeled "Function graphs."
2. Note that you'll need to make a project with the properties specified in the "installation of FLTK" note from the course website.
3. You'll need to move **Graph.cpp** and **Window.cpp** into your project.
4. Add an *x* axis and a *y* axis each of length 400, labeled "1 == 20 pixels" and with a notch every 20 pixels. The axes should cross at (300,300).
5. Make both axes red.

In the following, use a separate **Shape** for each function to be graphed:

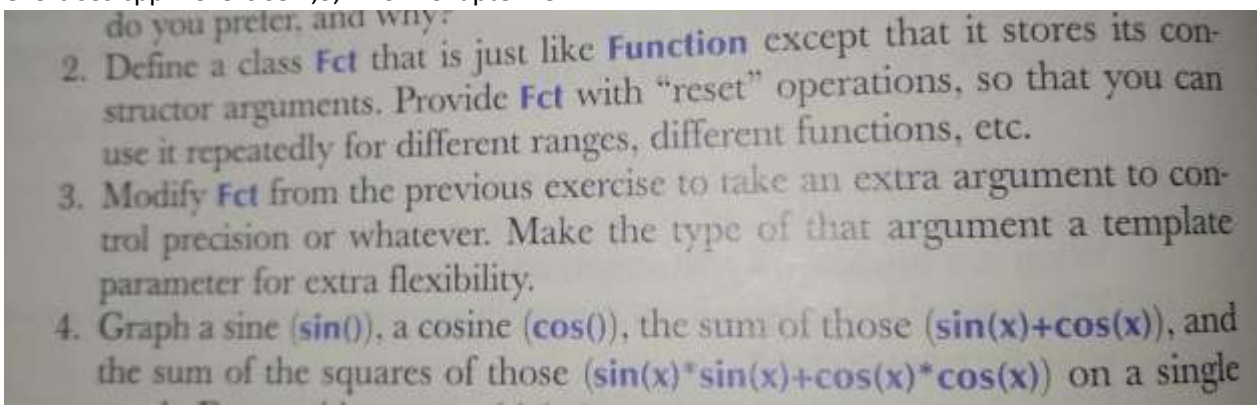
1. Graph the function **double one(double x) { return 1; }** in the range [–10,11] with (0,0) at (300,300) using 400 points and no scaling (in the window).
2. Change it to use *x* scale 20 and *y* scale 20.
3. From now on use that range, scale, etc. for all graphs.
4. Add **double slope(double x) { return x/2; }** to the window.
5. Label the slope with a **Text "x/2"** at a point just above its bottom left end point.

6. Add **double square(double x) { return x*x; }** to the window.
7. Add a cosine to the window (don't write a new function).
8. Make the cosine blue.
9. Write a function **sloping_cos()** that adds a cosine to **slope()** (as defined above) and add it to the window.

Class definition drill:

1. Define a **struct Person** containing a **string** name and an **int** age.
2. Define a variable of type **Person**, initialize it with "Goofy" and 63, and write it to the screen (**cout**).
3. Define an input (**>>**) and an output (**<<**) operator for **Person**; read in a **Person** from the keyboard (**cin**) and write it out to the screen (**cout**).
4. Give **Person** a constructor initializing **name** and **age**.
5. Make the representation of **Person** private, and provide **const** member functions **name()** and **age()** to read the name and age.
6. Modify **>>** and **<<** to work with the redefined **Person**.
7. Modify the constructor to check that **age** is [0:150) and that **name** doesn't contain any of the characters **; : " ' [ ] * & ^ % $ # @ !**. Use **error()** in case of error. Test.
8. Read a sequence of **Persons** from input (**cin**) into a **vector<Person>**; write them out again to the screen (**cout**). Test with correct and erroneous input.
9. Change the representation of **Person** to have **first_name** and **second_name** instead of **name**. Make it an error not to supply both a first and a second name. Be sure to fix **>>** and **<<** also. Test.

exercises.cpp – exercise 2,3,4 from Chapter 15



do you prefer, and why?

2. Define a class **Fct** that is just like **Function** except that it stores its constructor arguments. Provide **Fct** with "reset" operations, so that you can use it repeatedly for different ranges, different functions, etc.
3. Modify **Fct** from the previous exercise to take an extra argument to control precision or whatever. Make the type of that argument a template parameter for extra flexibility.
4. Graph a sine (**sin()**), a cosine (**cos()**), the sum of those (**sin(x)+cos(x)**), and the sum of the squares of those (**sin(x)*sin(x)+cos(x)*cos(x)**) on a single