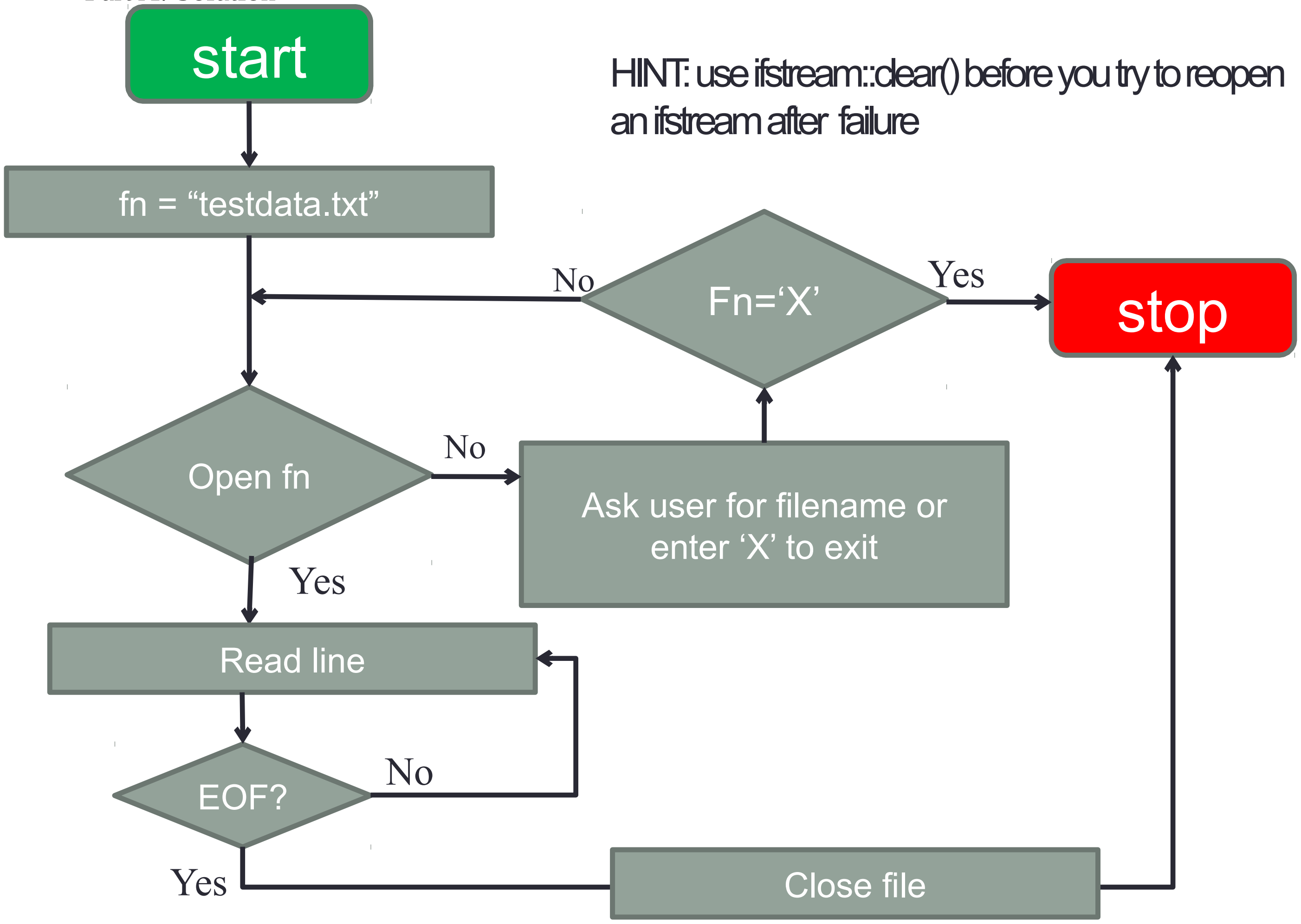
DEMO- Layout a project:

Part A: Open a file

Write a program that trys to open a default file for reading. If the default file isn’t there, it asks a user to enter a filename to open or an x to exit the program. If the filename entered isn’t there, program says so and ask for a different filename.

Part A: Solution


start
fn = “testdata.txt”
HINT: use ifstream::clear() before you try to reopen an ifstream after failure
No
Fn=‘X’
Yes
stop
Open fn
No
Ask user for filename or enter ‘X’ to exit
Yes
Read line
EOF?
No
Yes
Close file

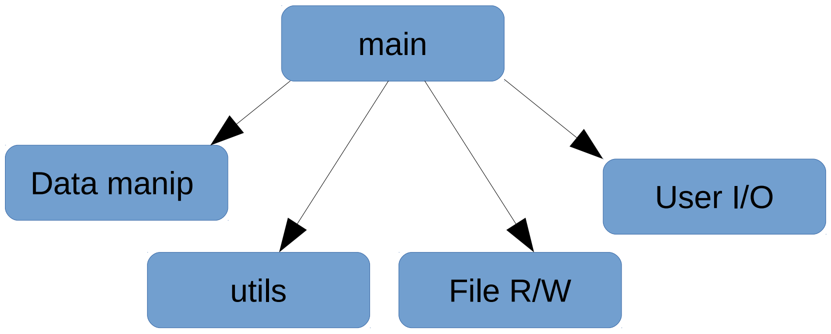
```
//============================================================================
// Name        : 3_File_IO.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
#include <fstream>
#include <string>

using namespace std;
std::string MYFILE= "MyTestFile.txt";        //global to this file
const int FAIL = -1;
const int USER_CHOSE_TO_EXIT= -2;

const int SUCCESS =0;

//***README***  use parts of the following code to implement the flowchart found in
//4- Headers, Streams, Namespaces lecture

int classExercise(string &filename){
        //code to read a file
        ifstream myInputfile;
        myInputfile.open(filename.c_str(), ios::in);

        while (!myInputfile.is_open()){
                cout << "Please enter a filename or \'x\' to exit"<<endl;
                cin >> filename;
                if (filename == "x" || filename == "X")
                        return USER_CHOSE_TO_EXIT;
                myInputfile.open(filename.c_str(), ios::in);
        }

        //read and count the data
        if (myInputfile.is_open()){
                std::string line;

                //read entire file
                while (!myInputfile.eof()) {                //exits when reach end of file
                        getline(myInputfile, line);         //gets a line up to '/n' char
                        cout<<line;
                }
                myInputfile.close();
        }
        return SUCCESS;
}

int main() {
    return classExercise(MYFILE);
}
```

Part B: Process contents

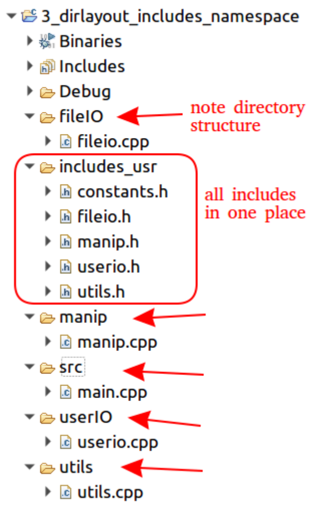
Expand on part A to do the following:

1. Open and read file
2. manipulate data from file
3. get destination filename to put manipulated data
4. save data to destination file

Module relationships (some may not be used in smaller projects)



Set up directory structure based on above relationships



```
3_dirlayout_includes_namespace
  Binaries
  Includes
  Debug
  fileIO
    fileio.cpp
  includes_usr
    constants.h
    fileio.h
    manip.h
    userio.h
    utils.h
  manip
    manip.cpp
  src
    main.cpp
  userIO
    userio.cpp
  utils
    utils.cpp
```

Now flesh out code in each module

1. create userio (cpp and h) files
2. move header to includes_usr (adjust header include in userio.cpp)
3. add two functions in userio.h (note: the '&' means pass by ref, we will discuss this next week, included early here to encourage good habits.)

```
std::string read(const std::string &phrase);
void write(const std::string &phrase);
```

    but these are common so put them in a namespace to avoid collisions

```
4. then main
   whole thing is a loop,
   also may need to ask for "Please enter a filename or X to exit"
```

so make a constant in the constants file

```
const std::string ASK_FOR_FILE_OR_EXIT_CHAR "Please enter a filename or X to exit";
```

5. build the constants.h file as we go

6. As we build the modules move the header files into includes_usr and adjust the paths in the cpp files (use relative includes)

7. declare minimal functionality in the headers, define minimal functions in cpp

8. get it to compile and run

9. fill infunction definitions one at a time

One last thing, put all functions in namespaces