

Fall 22, Xin Wang

Laboratory 03: Selection, Repetition and File Operation

This laboratory is to be performed the week starting Sep. 19th.

Prerequisite Reading

Lecture Slides for Chapter 3, Chapter 4 and Chapter 5

Lecture Content Review

I. Control Structures: Selection

- Selection structures allow users to provide alternative paths in a program.
- Syntax:
 - If statement

```
if (boolean_expression) {  
    statement block;  
} else {  
    statement block;  
}
```

- Switch statement

```
switch (control_expression) {  
    case constant:  
        statement block;  
        break;  
    ...  
    default:  
        statement block;  
}
```

II. Control Structure: Repetition

- Repetition structures allow users to repeat a set of steps as long as a condition is true.
- Syntax:

- while statement

```
while (boolean_expression) {  
    statement block;  
}
```

- do-while statement

```
do {  
    statement block;  
} while (boolean_expression);
```

- for statement

```
for (initialization_stmt; boolean_condition; modification_stmt) {  
    statement block;  
}
```

- Common repetition structures:
 - Counter controlled – used if the number of data values is known in advance
 - Sentinel controlled – used if a special data value exists indicating the end of data
 - End-of-data controlled
- break & continue
 - break – terminate the loop
 - continue – skip remaining statements of current iteration, force next iteration

III. File Operation

- need built-in library <fstream>
 - Reading / Generating data files
 - Declare an input / output file stream object
 - ifstream / ofstream objectName;
 - Open data file for input / output
 - objectName.open (fileName);
 - Read from / Write to data file
 - objectName >> variable;
 - objectName << variable << endl;
- Can be combined as:
ifstream / ofstream objectName (fileName);
- File Formats:
 - Specified number of lines in the first line
 - Trailer / sentinel signal in the last line
 - End-of-file

Purpose

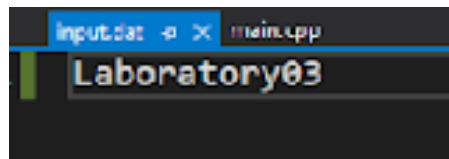
The purpose of this laboratory is to practice two control structures: Selection and Repetition. Meanwhile, instead of entering data from the keyboard and displaying data on the console, practice reading from and writing to data files by file operations.

Laboratory Tasks

P1. (15 points)

In the first problem, we mainly practice some basic file operations. Now we have a data file 'input.dat', what we need to do is just read the content from it and write this content to a new data file 'output.dat'.

Let's do this together to help you get familiar to the simple file operations in the program. First, take a look at the data file.



We can see that there's only one line of data in the file, which we can use a string variable to keep it. So, in the first few lines of our program, we need to include built-in libraries <fstream> and <string>.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
```

Now, in the main() function, we first create an object of ifstream class to associate with the input file. As a good habit, we should always check that whether we successfully open the files before we read or write the data.

```
ifstream fin("input.dat");
if (fin.fail())
{
    cerr << "Error opening input file!"
    exit(1);
}
```

Here we use cerr, which is defined in <iostream> to output the error message if the files cannot be opened correctly and exit the program with an error code 1.

Similarly, we do the same thing for the output file by define an ofstream object.

```
ofstream fout("output.dat");
if (fout.fail())
{
    cerr << "Error opening output file!"
    exit(1);
}
```

After making sure that we successfully open the two files, we can read and write data with a string class variable. From the code we can find that the usage of reading and writing operations are almost the same as cin and cout.

```
fin >> content;
fout << content;
```

Finally, don't forget to close the data files at the end by using the member function close().

```
fin.close();
fout.close();
```

This problem is not that hard, we give this to help you practice how to write a program with these file operations.

P2. (15 points)

Given a data file 'data1.dat', where each line contains a pair of values, respectively representing the width and the height of a rectangle. You need modify this 'data1.dat' if it is necessary, such that you can use counter-based controlling method when you read the data. Then, you're supposed to write a program to read data from this file and write the following information into a new data file named 'data1report.dat':

- The maximum area and the corresponding width and height
- The minimum area and the corresponding width and height
- The average area of all rectangles
- The average area of all squares

You're supposed to use **for** statements to solve this problem. Also, file opening states need to be checked before reading and writing.

P3. (15 points)

Use the data file 'data2.dat' as input file, and the corresponding output file becomes 'data2report.dat'. Similarly, you need modify this 'data2.dat' if it is necessary, such that you can use trailer / sentinel-based controlling method when you read the data. You need to write the information below to the output file:

- The maximum area and the corresponding width and height
- The minimum area and the corresponding width and height
- The average area of all rectangles

- **The average area of all squares**

You're required to use **while** statements to solve this problem. Also, file opening states need to be checked before reading and writing.

P4. (15 points)

Use the data file 'data3.dat' this time, the corresponding output file becomes 'data3report.dat'. Before you write the program, modify this 'data3.dat' if it is necessary, such that you can use end-of-file-based controlling method when you read the data. What you supposed to write to the output file include:

- **The maximum area and the corresponding width and height**
- **The minimum area and the corresponding width and height**
- **The average area of all rectangles**
- **The average of all squares**

You're required to use **do-while** statements to solve this problem. Also, file opening states need to be checked before reading and writing.

P5. (20 points)

We know the equation of Fibonacci sequence is $F(n) = F(n-1) + F(n-2)$.

Write a function that takes integer n ($n \leq 50$) as input and calculate $F(n)$. For this problem, you need to output the $F(10)$, $F(20)$, $F(30)$, $F(40)$, $F(50)$.

Example:

$F(0) = 0$

$F(1) = 1$

$F(2) = 1$

$F(3) = 2$

Hint: you may need to use **long long int instead of int** as the return type of the function.

P6. (Optional, 10 points)

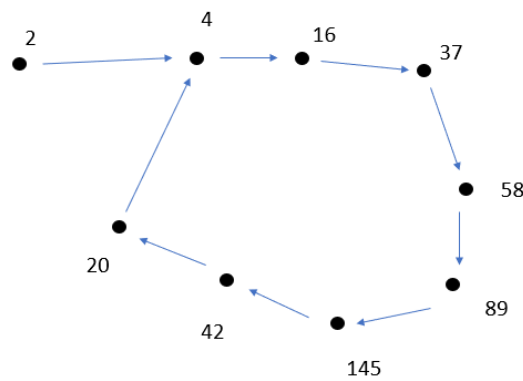
Write an algorithm to determine if a number n is happy.

A happy number is a number defined by the following process:

- Starting with any positive integer, replace the number by the sum of the squares of its digits.
- Repeat the process until the number equals 1 (where it will stay), or it **loops endlessly in a cycle** which does not include 1.
- Those numbers for which this process **ends in 1** are happy.

Return true if n is a happy number, and false if not.

Examples:

<p>Input: $n = 19$ Output: true Explanation:</p> <div>$1^2 + 9^2 = 82$$8^2 + 2^2 = 68$$6^2 + 8^2 = 100$$1^2 + 0^2 + 0^2 = 1$</div>	<p>Input: $n = 2$ Output: false Explanation:</p> 
---	--

Requirement: you are not allowed to use data structures not mentioned till now, such as array, vector, list, map, set and other high-level data structures.

Hint: recall the scene that you are running around the playground, you and someone run from the start point at the same time, even the other one runs faster than you, you will meet him/her at some point in the playground. For this problem, the meeting number decides whether n is happy or not.