VE 280 Lab 4

Out: June 9, 2020 **Due**: June 16, 2020

Ex1. Apple pies

Related Topics: *exceptions*, *program arguments*.

This exercise will help you better understand how `try`, `throw` and `catch` work for exception handling. Also, you will practise using `program arguments`.

```
"After finishing the last online class today, you are feeling really hungry. Some
kind of attractive sweet smell from your neighbour's kitchen is drifting into your
window, which reminds you of the tasty apple pie your grandma made for you in your
childhood. Hearing the growling of your stomach, you suddenly come up with the
idea of making some yourself. Since it's already 7 pm now, you are not sure
whether the markets are still open or whether they still have the ingredients you
need. You quickly grab your bag and head to the markets..."
```

For Ex1, you are going to simulate the process of buying ingredients for apple pies. The following are some basic rules:

- To make an apple pie, you need **250 grams of flour, 1 egg and 2 apples**.
- There are 2 markets that you can visit: `market1` and `market2`. `market1` sells flour and eggs, and `market2` sells apples.
- You should first visit `market1` and then `market2`. If `market1` is open, you should first buy flour and then buy eggs. During the procedure, if any "exception" happens (eg: `market1` is closed, eggs in `market1` are not enough, etc), the buying process will end (exception caught). See `ex1.cpp` for details.
- The input will be a sequence of numbers read from **program arguments** (6 numbers in total): number of apple pies (integer type), status of `market1` (1 for open and 0 for closed), status of `market2` (1 for open and 0 for closed), amount of flour remaining in `market1` (in grams, float type), number of eggs remaining in `market1` (integer type), number of apples remaining in `market2` (integer type). For example, if the program name is `ex1` and the arguments are "`2 1 1 355 10 5`", i.e.,

  ```
  ./ex1 2 1 1 355 10 5
  ```

  then you will make 2 pies, both markets are open, 355g of flour and 10 eggs remain in `market1`, and 5 apples remain in `market2`.
- Part of the program has already been implemented. Your work is to complete the program by filling the parts marked with `TODO`. Please **do not change** any code that is already written. See `ex1.cpp` for further instructions.

Testing

Compile `ex1.cpp` to an executable file `ex1`. Execute it with 6 arguments chosen by yourself. For example,

```
./ex1 3 1 1 1000 10 5
```

Then the output should be

```
You visit market1 first...
You've bought enough flour and eggs. Then you visit market2...
Apples in market2 are not enough. You still need 1 more.
It seems that today is not a good day for making apple pies.
```

Ex2. Too many apples

Related Topics: *IO*, *compound objects*, *program arguments*.

This exercise will help you get familiar with `file streams` and `structs`.

```
"After buying some flour and eggs, you hurried to market2 for the apples. Usually
the fruit stalls should be almost empty around 7 pm, but today, you find
surprisingly that one of the stalls is still full of different kinds of apples.
However, they are only sold in boxes. You don't mind buying more apples to share
with your family, but you want to get the lowest price on average. You quietly
pull out your phone and begin calculating the prices..."
```

In this exercise, you are going to read a file. The file contains all the price information for different kinds of apples. Here is an example:

```
variety1 56.81 5.12
variety2 59.50 5.52
variety3 79.28 7.25
```

The file is composed of 3 columns. The first column is the name of the apple variety (assume there is no space in the names), the second column is the price of the box of apples (in RMB, double type), and the third column is the total weight of the box of apples (in kg, double type). Note that **the number of lines of this file is not fixed**. You may get a price list with a much larger number of lines.

The **name of the file is not fixed** as well. In your lab materials, an example file named `price_list.txt` is provided. However, in some test cases, the name of the file could be different. You need to read the name of the file from **program argument**:

```
./ex2 <file_name>
```

For this exercise, please write a program that can read in all the information from the file, and then find out the apple variety with the lowest average price (RMB/kg). If two or more kinds of apples have the same minimum average price (RMB/kg), just choose the one that appears the earliest in the list. The following are the requirements:

- Name your file as `ex2.cpp`.
- Make sure to use `file streams` to read the file and `structs` to contain the information.
- Output: Print the **name, price and weight** of the apple variety with the lowest average price (RMB/kg) into stdout. Use 1 space to seperate them. Format:

  ```
  <name> <price> <weight>
  ```

  For consistency, you can assume that all the `price` and `weight` numbers in the file have 2 decimal places. In your output, the `price` and `weight` should have 2 decimal places as well. Hint: you may need the `<iomanip>` library.

Testing

Example file (suppose its name is `price_list.txt`):

```
variety1 56.81 5.12
variety2 59.50 5.52
variety3 79.28 7.25
```

Compile your `ex2.cpp` to an executable file named `ex2`. Execute it by

```
./ex2 price_list.txt
```

The output should be

```
variety2 59.50 5.52
```

Note that it is `59.50` instead of `59.5`.

Ex3. Guest list

Related Topics: *IO*.

This exercise will help you get familiar with `string streams`.

```
"After a great success of making the apple pies, you cannot wait to show off your
cooking skills in front of your friends. Tim, a close friend of you, volunteered
to contact and invite others to your apple pie party. He sent you a message of all
the people he has invited just now, but message seems a bit messy, maybe because
he was typing too fast...
```

Tim's message is something like

```
Ann ,Peter , Owen,Alice ,  Bob, Tim
```

He uses spaces and commas to separate the names, but sometimes there are too many spaces between a name and a comma and sometimes there's no space. Now, you want to make a name list using that message. Also, you want to know the exact number of people.

Please write your code in a file named `ex3.cpp`. Here are the requirements:

- Input: The input is Tim's message like the one shown above. You can assume that all the names only contain English letters, and there is no space inside the names (i.e, names like "Tim White" will not appear in the test cases). Also, you can assume that only 1 comma can appear between 2 names. However, **the number of spaces and the number of names are not fixed**. The name list could be much longer.
- Output: Output the list of guest names followed by the number of guests. Example:

  ```
  Ann
  Peter
  Owen
  Alice
  Bob
  Tim
  6
  ```

  The sequence of names should be **the same** as the input message. Each name or the number occupies one line.
- Hint: Please use `string stream` to solve this problem, which would greatly ease your work.

Testing

Compile `ex3.cpp` to an executable file `ex3`. Execute it and then input the message.

Example input:

```
Ann ,Peter , Owen,Alice ,  Bob, Tim
```

Example output:

```
Ann
Peter
Owen
Alice
Bob
Tim
6
```

Created by Jinglei Xie.
Last update: May 29, 2020
@UM-SJTU Joint Institute