Programming Design In-class Practices

C++ Strings, File I/O, and Header Files

Ling-Chieh Kung

Department of Information Management
National Taiwan University

Problem 1: Counting punctuation marks

- Given a sentence of English letters, punctuation marks, and spaces, count the number of punctuation marks.
  - The punctuation marks includes "`,.:;!?`".

Input:
```
Hi! Let us learn Programming together. What do you think?
```

Output:
```
3
```

Problem 2: Dictionary search

- Let x be an English word with English characters only. Let D be a collection of n English words with only lowercase letters. Words in D are sorted alphabetically. Determine whether x exists in D in a case-insensitive manner.
  - n is no larger than the largest possible value of an `int` variable.
  - Input: n in the first line, x in the second line, D in the third to $(n + 2)$th line.
  - Output: 1 if x is in D and 0 otherwise.
- Linear search? Binary search?
- This is the basis of spell checking.

Problem 2: Dictionary search

Input:
```
5
Watermelon
apple
banana
grapefruit
orange
watermelon
```

Output:
```
1
```

Input:
```
5
water
apple
banana
grapefruit
orange
watermelon
```

Output:
```
0
```

Problem 3: Dollar format

- When we have an integer value, we print it out directly. However, we should "decorate" it if it is representing an amount of money.
  - The beginning should be a dollar sign.
  - There should be a comma every three digit (starting from the least significant digit).
- Write a function to convert a simple integer into a string of a dollar format:

```
string toDollarNumber(int value);
```

  - E.g., given 1234, return $1,234.
  - E.g., given 123456789, return $123,456,789

Problem 3: Dollar format

- **Hint.** To convert an integer into a C++ string, there are many ways. Two ways are the following:

```
int main()
{
  int value = 0;
  cin >> value;
  char c[10] = {0};
  itoa(value, c, 10);
  string s = c;

  return 0;
}
```

```
int main()
{
  int value = 0;
  cin >> value;
  string s = to_string(value);

  return 0;
}
```

Problem 4A: Dictionary from a file

- Consider Problem 2 again. Now suppose that the dictionary is contained in a plain-text file.
  - Input: n in the first line, x in the second line, and a correct file name in the third line.
  - Output: 1 if x is in D and 0 otherwise.

Input:
```
5
Watermelon
test.txt
```

Output:
```
1
```

Input:
```
5
water
test.txt
```

Output:
```
0
```

Problem 4B: Dictionary from a file

- Now suppose that the number of words in the dictionary is not entered by the user. We must find it in the program.
  - Input: x in the first line and a correct file name in the second line.
  - Output: 1 if x is in D and 0 otherwise.

Input:
`Watermelon`
`test.txt`

Output:
`1`

Input:
`water`
`test.txt`

Output:
`0`

Problem 4C: Dictionary from a file

- If the file path may contain spaces, we may try to write as the program at the right.
- This does not work!
  - Why is that?
  - How to fix it?

```
int main()
{
  string target = "";
  cin >> target;
  makeLowercase(target);

  string filePath = "";
  getline(cin, filePath);

  //...

  return 0;
}
```

Problem 4D: Dictionary from a file

- Now we also want to record the history of comparisons like the following:

```
Comparing WaterMelon with apple
Comparing WaterMelon with banana
Comparing WaterMelon with grapefruit
Comparing WaterMelon with orange
Comparing WaterMelon with watermelon... Got it!
```

- Please record the history in a plain-text file created by your program.

Problem 5: streaming

- Recall our class **`MyVector`** again.
- Try to execute this main function:

- Why does it work?
  - How to express a file path?
  - How about operator overloading?

```
int main()
{
  double d = 1.23;
  MyVector v;

  ifstream fin("text\\MyVector.txt");
  ofstream fout("text\\out.txt");
  fin >> v;
  fout << v << endl;

  if(d == v)
    fout << "Equal!";
  else
    fout << "Unequal!";

  fin.close();
  fout.close();
  return 0;
}
```