

Lab 5 - Iterators

This tutorial is based on the `FileCharacterCounter` that was developed in Lab 4. It analysed a given text file and recorded the frequencies of the individual characters occurring in that file and saved the statistics in an output text file. We used an array and standard array access (within a for loop) to write the individual frequencies to an output stream. Moreover, in order to remove clutter (irrelevant information) we had to define a guard to filter zeros from the result.

In this tutorial, we wish to experiment with iterators. In particular, we shall create an iterator for `CharacterCounter`, called `CharacterCounterIterator`, which implements the standard interface for a C++ forward iterator. As an added feature, this iterator automatically filters all zero occurrences, hence the iterator dereference operator only returns non-zero elements.

First, when using an iterator to print the character frequencies we need to map a given frequency to its corresponding character. This was easy, when we used an array and a for-loop. The index variable could be converted into a `char` value (see solution for tutorials 2 and 4). To recover this feature, we would have to maintain an extra variable. The result would become rather clumsy and counterintuitive to the elegance offered by iterators. We need to find a better solution. We need a new data type `FrequencyMap`, which establishes the required association between a character and its number of occurrences. The following class specifications suggests a possible solution:

FrequencyMap.h:

```
#pragma once
class FrequencyMap
{
private:
    char fChar;
    int fFrequency;
public:
    FrequencyMap(); // default constructor needed when used as base type of arrays
    FrequencyMap(char aChar, int aFrequency); // initialize with concrete values

    // read-only getters
    char getCharacter() const; // retrieve character
    int getFrequency() const; // retrieve frequency
};
```

We need to add a new method to class `CharacterCounter` in order to populate the map. Please note that for Lab 5 we will be using `unsigned char` in place of `char*` for the variable received by the `count` method. Do adjust your code accordingly. :

CharacterCounter.h:

```
#pragma once
#include <iostream>
class CharacterCounter
{
private:
    int fTotalNumberOfCharacters;
    int fCharacterCounts[256];
public:
    CharacterCounter();
    void count(unsigned char aCharacter); // We count all 256 byte values
    friend std::ostream& operator<<(std::ostream& aOStream,
        const CharacterCounter& aCharacterCounter);

    // new lab 5: frequency indexer method
    int operator[](unsigned char aCharacter) const;
};
```

We can now define a `CharacterCounterIterator`:

CharacterCounterIterator.h:

```
#pragma once

#include "CharacterCounter.h"
#include "FrequencyMap.h"
class CharacterCounterIterator
{
private:
    FrequencyMap fMaps[256];
    int fIndex;
public:
    CharacterCounterIterator(CharacterCounter& aCounter);
    CharacterCounterIterator(const CharacterCounterIterator& aObj);
    CharacterCounterIterator(const CharacterCounterIterator& aObj, int aIndex);
    const FrequencyMap& operator*() const; // return current frequency map
    CharacterCounterIterator& operator++(); // prefix
    CharacterCounterIterator operator++(int); // postfix (extra unused argument)
    bool operator==(const CharacterCounterIterator& aOther) const;
    bool operator!=(const CharacterCounterIterator& aOther) const;
    bool operator<=(const CharacterCounterIterator& aOther) const;
    CharacterCounterIterator begin() const;
    CharacterCounterIterator end() const;
};
```

The specification captures a standard C++ **forward iterator**. It takes a `CharacterCounter` object and provides access to *non-zero frequency* `FrequencyMap` objects. The iterator preserves the original array order, that is, the iterator returns `FrequencyMap` objects in the sequence determined by the ASCII encoding of characters.

You may refer to [Lect4 - Basic Concepts Slides 37-43](#) on how to implement part of this.

In order to test the iterator, use the following approach:

```
CharacterCounter lCounter;
unsigned char lChar;

while ( lInput >> lChar ) {
    lCounter.count( lChar );
}
lOutput << lCounter;

// test iterator
cout << "The frequencies: " << endl;
for ( CharacterCounterIterator iter( lCounter ); iter <= iter.end(); iter++ )
{
    cout << (*iter).getCharacter() << ": " << (*iter).getFrequency()
    << endl;
}
```

Your **main.cpp** should look something like the following:

```
#include "CharacterCounter.h"
#include "CharacterCounterIterator.h"
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
    CharacterCounter lCounter;
    unsigned char lChar;

    ifstream lInput;
    string inputFilename = "CCinput.txt";

    lInput.open(inputFilename, ifstream::in);

    if (!lInput.good())
    {
        cerr << "Cannot open input file: " << inputFilename << endl;
        return 2;
    }
    else {
        while (lInput >> lChar)
        {
            lCounter.count(lChar);
        }
        lInput.close();
    }

    ofstream lOutput;
    string outputFilename = "CCoutput.txt";

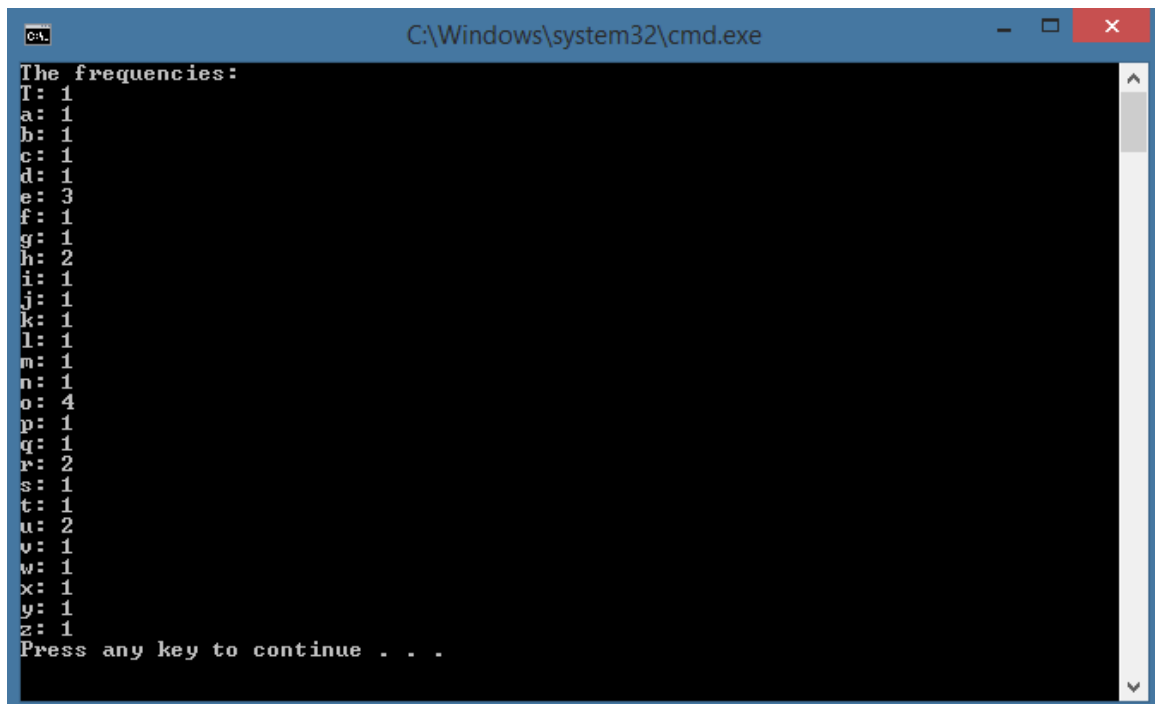
    lOutput.open(outputFilename, ofstream::out);

    if (!lOutput.good())
    {
        cerr << "Cannot open output file: " << outputFilename << endl;
        lOutput.close();
        return 3;
    }
    else {
        lOutput << lCounter;
        lOutput.close();

        // test iterator
        cout << "The frequencies: " << endl;
        for (CharacterCounterIterator iter(lCounter); iter <= iter.end(); iter++)
        {
            cout << (*iter).getCharacter() << ": " << (*iter).getFrequency() << endl;
        }
    }

    return 0;
}
```

Applied to the main.cpp file this code should produce to following console output:



```
C:\Windows\system32\cmd.exe

The frequencies:
T: 1
a: 1
b: 1
c: 1
d: 1
e: 3
f: 1
g: 1
h: 2
i: 1
j: 1
k: 1
l: 1
m: 1
n: 1
o: 4
p: 1
q: 1
r: 2
s: 1
t: 1
u: 2
v: 1
w: 1
x: 1
y: 1
z: 1
Press any key to continue . . .
```