Control Statements for C++ (10th Lab)

- **Pseudo code**
  - Artificial, informal language used to develop algorithms
    - Used to “think out” a program before coding it
      - Easy to convert into a C++ program
  - Similar to everyday English
    - Only executable statements
      - No need to declare variables
  - Not executed on computers

- **Only three control structures are needed**
  - No goto statements
  - Three control structures
    - Sequence structure
      - Programs executed sequentially by default
    - Selection structures
      - if, if...else, switch
    - Repetition structures
      - while, do...while, for

---

**C++ Keywords**

Keywords common to the C and C++ programming languages

<table>
<thead>
<tr>
<th>auto</th>
<th>break</th>
<th>case</th>
<th>char</th>
<th>Const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continue</td>
<td>default</td>
<td>do</td>
<td>double</td>
<td>Else</td>
</tr>
<tr>
<td>enum</td>
<td>extern</td>
<td>float</td>
<td>for</td>
<td>Goto</td>
</tr>
<tr>
<td>if</td>
<td>int</td>
<td>long</td>
<td>register</td>
<td>Return</td>
</tr>
<tr>
<td>short</td>
<td>signed</td>
<td>sizeof</td>
<td>static</td>
<td>Struct</td>
</tr>
<tr>
<td>switch</td>
<td>typedef</td>
<td>union</td>
<td>unsigned</td>
<td>Void</td>
</tr>
<tr>
<td>volatile</td>
<td>while</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C++-only keywords

<table>
<thead>
<tr>
<th>and</th>
<th>and_eq</th>
<th>asm</th>
<th>bitand</th>
<th>Bitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>catch</td>
<td>class</td>
<td>compl</td>
<td>const_cast</td>
</tr>
<tr>
<td>delete</td>
<td>dynamic_cast</td>
<td>explicit</td>
<td>export</td>
<td>False</td>
</tr>
<tr>
<td>friend</td>
<td>inline</td>
<td>mutable</td>
<td>namespace</td>
<td>New</td>
</tr>
<tr>
<td>not</td>
<td>not_eq</td>
<td>operator</td>
<td>or</td>
<td>or_eq</td>
</tr>
<tr>
<td>private</td>
<td>protected</td>
<td>public</td>
<td>reinterpret_cast</td>
<td>static_cast</td>
</tr>
<tr>
<td>template</td>
<td>this</td>
<td>throw</td>
<td>true</td>
<td>Try</td>
</tr>
<tr>
<td>typeid</td>
<td>typename</td>
<td>using</td>
<td>virtual</td>
<td>wchar_t</td>
</tr>
<tr>
<td>xor</td>
<td>xor_eq</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
if...else double-selection statement

Dangling-else problem

Compiler associates else with the immediately preceding if

Example

```cpp
if ( x > 5 )
    if ( y > 5 )
        cout << "x and y are > 5";
    else
        cout << "x is <= 5";
else
    cout << "x is <= 5";
```

Compiler interprets as

```cpp
if ( x > 5 )
    if ( y > 5 )
        cout << "x and y are > 5";
    else
        cout << "x is <= 5";
else
    cout << "x is <= 5";
```

Rewrite with braces ({})

```cpp
if ( x > 5 )
{
    if ( y > 5 )
        cout << "x and y are > 5";
}
else
    cout << "x is <= 5";
```

while repetition statement

Counter-controlled repetition

Loop repeated until counter reaches certain value
Number of repetitions is known beforehand

```cpp
//GradeBook.h

#include <string>
using std::string;

//GradeBook class definition
class GradeBook
```
public:
    GradeBook(string);
    void setCourseName(string);
    string getCourseName();
    void displayMessage();
    void determineClassAverage();
private:
    string courseName;
};

// GradeBook.cpp
#include <iostream>
using std::cout;
using std::cin;
using std::endl;
using std::fixed;

#include <iomanip>
using std::setprecision;

#include "GradeBook.h"

// constructor initializes courseName with string supplied as argument
GradeBook::GradeBook(string name)
{
    setCourseName(name);
}

// function to set the course name
void GradeBook::setCourseName(string name)
{
    if(name.length() <= 25)
        courseName = name;
else
{
    courseName = name.substr(0, 25);
    cout << "Name " << name << " exceeds maximum length (25). Limiting courseName to first 25 characters."
    << endl;
}

//function to retrieve the course name
string GradeBook::getCourseName()
{
    return courseName;
}

//display a welcome message to the user
void GradeBook::displayMessage()
{
    cout << "Welcome to the grade book for " << getCourseName() << endl;
}

//determine class average based on 10 grades entered by users
void GradeBook::determineClassAverage()
{
    int total;
    int gradeCounter;
    int grade;
    double average;

    //initialization phase
    total = 0;
    gradeCounter = 0;

    cout << "Enter grade or -1 to quit: ";
    cin >> grade;
//processing phase
while (grade != -1)
{
    total = total + grade;
    gradeCounter = gradeCounter + 1;
    
    cout << "Enter grade or -1 to quit: ";
    cin >> grade;
}

if (gradeCounter != 0)
{
    average = static_cast<double>(total) / gradeCounter;
    
    // display total and average of grades
    cout << "Total of all " << gradeCounter << " grades is " << total << endl;
    cout << "Class average is " << setprecision(2) << fixed << average << endl;
}
else
    cout << "No grades were entered" << endl;

//main.cpp
#include "GradeBook.h"

//start main
int main()
{
    GradeBook myGradeBook("CS101 C++ Programming");
myGradeBook.displayMessage();
myGradeBook.determineClassAverage();
return 0;
}

//end main

Result:

Welcome to the grade book for
CS101 C++ Programming
Enter grade or -1 to quit: 97
Enter grade or -1 to quit: 88
Enter grade or -1 to quit: 72
Enter grade or -1 to quit: -1
Total of all 3 grades entered is 257
Class average is 85.67

■ Assignment Operators
  ■ Assignment expression abbreviations
    ◆ Addition assignment operator
      ● Example
        c = c + 3; abbreviates to c += 3;
  ■ Other assignment operators
    d -= 4     (d = d - 4)
    e *= 5     (e = e * 5)
    f /= 3     (f = f / 3)
    g %= 9     (g = g % 9)

■ Increment and Decrement Operators
  ■ Increment operator ++
    ◆ Increments variable by one
      ● Example
        c++
  ■ Decrement operator --
    ◆ Decrements variable by one
      ● Example
        c--
```cpp
#include <iostream>
using std::cout;
using std::endl;

int main()
{
    int c;
    c=5;
    cout<< c <<endl;
    cout<< c++ <<endl;
    cout<< c <<endl;
    cout<<endl;
    c=5;
    cout << c <<endl;
    cout << ++c <<endl;
    cout << c <<endl;
    return 0;
}
```

### 실습
다음의 `header` 파일의 클래스를 구현하는 프로그램을 작성하시오.
(각각의 operator들이 제대로 동작하는지 출력)

```cpp
//GradeBook.h

#include <string>
using std::string;

//GradeBook class definition
class GradeBook
{
public:
    GradeBook();
    void setGrades(); //grade 입력 및 average 계산
    void printResult(); //각 grade 및 average 출력(average는 소수점 2째 자리까지 출력)
    GradeBook operator+(GradeBook book); //GradeBook + GradeBook
    GradeBook operator++(); //각 grade 들과 average를 1점씩 증가시킴(전위 증가)
};
```
GradeBook operator++(int); // 각 grade들과 average를 1점씩 증가시킴 (후위 증가)

private:
    int grade[5];
    double average;
};

// end class Gradebook