Creative Software Programming Assignment (week-2)

Every assignment will be announced on **Thursday** and should be sumitted by next **Tuesday**.

In this week **Handed out will be Sep 10, 2020, Due Sep 15, 2020**

Submit your assignment only through the hconnect GitLab

1. Your assignment should be in `week-2` directory.
2. Commit & Push to the hconnect GitLab when you done.

Topics

1. Statements and flow control
   - Selection statements
   - Iteration statements
   - Jump statements
2. Handling input/output in assignment

1. Statemetns and flow control

A simple C++ statement is each of the individual instructions of a program, like the variable declarations and expressions seen in previous sections. They always end with a semicolon (;), and are executed in the same order in which they appear in a program.

But programs are not limited to a linear sequence of statements. During its process, a program may repeat segments of code, or take decisions and bifurcate. For that purpose, C++ provides flow control statements that serve to specify what has to be done by our program, when, and under which circumstances.

Many of the flow control statements explained in this section require a generic (sub)statement as part of its syntax. This statement may either be a simple C++ statement, -such as a single instruction, terminated with a semicolon (;) - or a compound statement. A compound statement is a group of statements (each of them terminated by its own semicolon), but all grouped together in a block, enclosed in curly braces: {}:

```
{ statement1; statement2; statement3; }
```

The entire block is considered a single statement (composed itself of multiple substatements). Whenever a generic statement is part of the syntax of a flow control statement, this can either be a simple statement or a compound statement.

a. Selection statements

The `if` keyword is used to execute a statement or block, if, and only if, a condition is fulfilled. Its syntax as follow:

```
if (condition)
{
    statements;
}
```

So, In case of below code, if `x` is exactly 42, program just print out message and if not, print out value of x.

```
if (x == 42) {
    std::cout << "X is the answer to life the universe and everything" <<
std::end;
} else {
    std::cout << "X is " << x << std::endl;
}
```

```
// file: statement.cc
#include <iostream>

int main() {
    if (true) {
        std::cout << "This statement is always executed." << std::endl;
    } else {
        std::cout << "This statement is never executed." << std::endl;
    }

    // Most programming languages assume nonzero numbers are true.
    if (0) {
        // so, this statement will never be executed
        std::cout << "0 is same as false" << std::endl;
    }

    if (1) {
        std::cout << "1 is same as true" << std::endl;
    } else if (2) {
        // This is true because 2 is also a non-zero number.
        // However, in the else if statement,
        // if the condition that is true from the top condition is met first,
        // the statement after it is not executed.
        std::cout << "2 is same as true" << std::endl;
    }

    if ("some string") {
        std::cout << "string is true! too" << std::endl;
    }

    if (true) {
        if (true) {
            if (true) {
                std::cout << "nested statement" << std::endl;
            }
        }
    }

    return 0;
}
```

b. Iteration statements (loops)

Loops repeat a statement a certain number of times, or while a condition is fulfilled, with keyword `while`, `do-while` and `for`.

```
while (condition) {
    statement;
}

do {
    statement;
} while (condition);

for (intialize; condition; increate) {
    statement;
}
```

The do-while syntax is very similar to the while syntax. The difference is that the condition is checked later, so even if it is a false condition, it is executed at least once.

```
// file: iteration.cc
#include <iostream>

int main() {
    // this is while loop
    int n = 0;
    while (n < 10) {
        std::cout << n << ", ";
        n = n + 1;
    }
    std::cout << std::endl;
    // expect, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

    // this is do-while loop
    int m = 0;
    do {
        std::cout << m << ", ";
        m = m + 1;
    } while (m < 10);
    std::cout << std::endl;
    // expect, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

    // this is for loop
    for (int i = 0; i < 10; i++) {
        std::cout << i << ", ";
    }
    std::cout << std::endl;
    // expect, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

    return 0;
}
```

c. Jump statements

Jump statements allow altering the flow of a program by performing jumps to specific locations

- break
- continue
- goto (not recommended)

The **break** statement leaves a loop, even if the condition for its end is not fulfilled.

```
int i = 0;
while (true) {
    i = i + 1;
    std::cout << i << ", ";
    if (i > 10) {
        break;
    }
}
```

Expected result `1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,`

The **continue** statement causes the program to skip the rest of the loop in the current iteration, as if the end of the statement block had been reached, causing it to jump to the start of the following iteration.

```
for (int i = 0; i < 10; i++) {
    if (i == 4) {
        continue;
    }
    std::cout << i << ", ";
}
```

Expected result: `0, 1, 2, 3, 5, 6, 7, 8, 9,`

The **goto** statement allows to make an absolute jump to another point in the program. **But not recommended**.

```
    int i = 0;
label:
    i = i + 1;
    std::cout << i << ", ";

    if (i < 10):
        goto label:
```

Expected result:
`1, 2, 3, 4, 5, 6, 7, 8, 9, 10,`

2. Handling input/output in assignment

Inputs and outputs are very important for assignments.

Keep in mind that if you simply miss the space or order, you may be submitted with the wrong answer.

Assignment example

Write a program that prints a multiplication table that fits the output format below when a natural number input N ($1<N<=9$) is given.

example

- INPUT:

```
N: 3
```

- OUTPUT:

```
1x3=1
2x3=6
3x3=9
4x3=12
5x3=15
6x3=18
7x3=21
8x3=24
9x3=27
```

```cpp
// file: assignment-example.cc
#include <iostream>

int main() {
    int N;
    std::cin << N;

    for (int i = 1; i < 10; i++) {
        std::cout << i << "x" << N << "=" << i * N << std::endl;
    }
    return 0;
}
```

reference

- http://www.cplusplus.com/

Notice

1. Week-2 assignment will be announce in Thursday (Sep 10, 2020)
2. The code we practiced today is not an assignment, it is a process necessary to solve the assignment.
3. In order to facilitate non-face-to-face classes, questions are frequently asked through chat rooms. (https://open.kakao.com/o/glEhaRtc, or mailto: maytryark@gmail.com)

```
```