

## SWE20004 Technical Software Development

### Lab 4 (week 4)

You will need:  
Online C++ IDE / C++ Computer  
installed IDE  
A computer with internet access

#### In this lab you will investigate C++ Loop Structures

Before you start the lab exercise, download an appropriate C++ IDE to run your commands and programs.

1. Answer the following questions? Don't copy and paste – write down what it is - in your own words briefly.

##### 1.1 The while loop is an example of a(n)

The while loop is an example of a pretest loop. A pre-test loop is when prior to entering the loop, the condition is checked.

##### 1.2 A do-while loop is guaranteed to execute.....

A do-while loop is guaranteed to execute at least one time, because it evaluates the condition after the execution of the loop.

##### 1.3 The for loop has 3 expressions, write those in an order

1. Initialization: Executes at first and only once. Variables are usually declared and initialized at this stage.
2. Condition: This is a Boolean expression, either it's true or false.
3. Iterator: Changes the value of the initialized variables.

##### 1.4 A loop that is located inside another loop is called a(n).....

A loop that is located inside another loop is called a nested loop.

##### 1.5 In order to terminate the execution of a loop, we can use the ..... statement

In order to terminate the execution of a loop, we can use the break statement. It can also be used to terminate a case in a switch statement.

##### 1.6 If you know ahead of time how many times you need to loop, which loop would you use?

You should use a for loop.

Name: \_\_\_\_\_ Hong Hai Dang Nguyen \_\_\_\_\_ Student ID: \_\_\_\_\_ 103503191 \_\_\_\_\_

2. What is the value of `num` after the following code executes if the user enters `10` at the keyboard?

- ☐ `int num;`
- ☐ `cin >> num;`
- ☐ `if (num > 10)`
- ☐ `num -= 10;`
- ☐ `else`
- ☐ `num += 10;`

```
#include <iostream>
using namespace std;
int main()
{
    int num;
    cout<<"Initial num: ";
    cin>>num;
    if (num > 10)
        num -= 10;
    else
        num += 10;
    cout<<"Final num: "<<num;
    return 0;
}
```



```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int num;
6     cout<<"Initial num: ";
7     cin>>num;
8     if (num > 10)
9         num -= 10;
10    else
11        num += 10;
12    cout<<"Final num: "<<num;
13    return 0;
14 }
15
16
```

input

Initial num: 10  
Final num: 20

3. What does the following code snippet display if the user enters `70` at the keyboard?

- ☐ `int temperature;`
- ☐ `cout << "Enter a temperature: ";`
- ☐ `cin >> temperature;`
- ☐ `if (temperature < 50);`
- ☐ `cout << "It's cold!" << endl;`
- ☐ `if (temperature > 50)`
- ☐ `cout << "It's hot!" << endl;`
- ☐ `else`
- ☐ `cout << "Maybe it's raining?";`

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int temperature;
6     cout << "Enter a temperature: ";
7     cin >> temperature;
8     if (temperature < 50);
9     cout << "It's cold!" << endl;
10    if (temperature > 50)
11        cout << "It's hot!" << endl;
12    else
13        cout << "Maybe it's raining?";
14    return 0;
15 }
16
```

input

Enter a temperature: 70  
It's cold!  
It's hot!

This can be fixed by replacing the "If" In line 10 with "else If", as well as using "{ }" for the if else statement to get the desirable output.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int temperature;
6     cout << "Enter a temperature: ";
7     cin >> temperature;
8     if (temperature < 50){
9         cout << "It's cold!" << endl;
10    }
11    else if (temperature > 50){
12        cout << "It's hot!" << endl;
13    }
14    else{
15        cout << "Maybe it's raining?";
16    }
17    return 0;
18 }
```

input

Enter a temperature: 70  
It's hot!

4. What does the following code snippet display if the user enters 20 at the keyboard?

- ☐ `int favorite;`
- ☐ `cout << "Enter your favorite number: ";`
- ☐ `cin >> favorite;`
- ☐ `if (favorite == 13)`
- ☐ `cout << "That my favorite number too!" << endl;`
- ☐ `cout << "That's amazing!" << endl;`
- ☐ `cout << "Great minds think alike!" << endl;`

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int favorite;
6     cout << "Enter your favorite number: ";
7     cin >> favorite;
8     if (favorite == 13)
9         cout << "That my favorite number too!" << endl;
10    cout << "That's amazing!" << endl;
11    cout << "Great minds think alike!" << endl;
12    return 0;
13 }
14
```

input

Enter your favorite number: 20  
That's amazing!  
Great minds think alike!

This can be fixed by adding “{ }” for the if statement to be correct.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int favorite;
6     cout << "Enter your favorite number: ";
7     cin >> favorite;
8     if (favorite == 13){
9         cout << "That my favorite number too!" << endl;
10        cout << "That's amazing!" << endl;
11        cout << "Great minds think alike!" << endl;}
12    return 0;
13 }
14
```

input

Enter your favorite number: 20

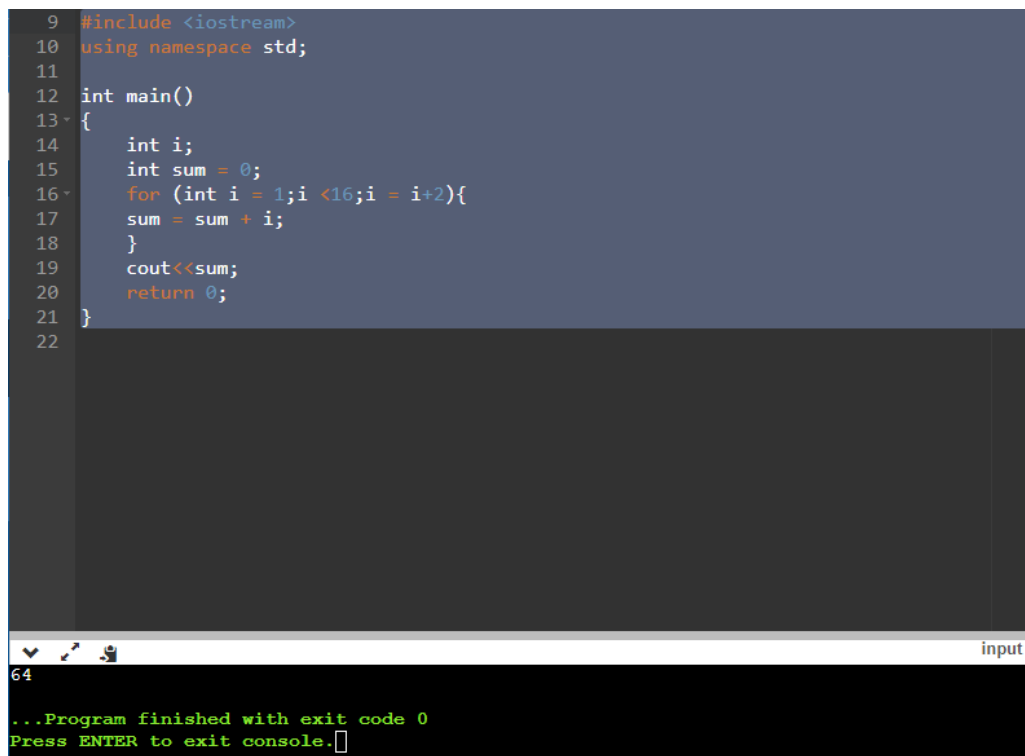
...Program finished with exit code 0  
Press ENTER to exit console.



6. Write a program: For loop – Sum of Odd integers

Write code that uses a for loop to calculate the sum of the odd integers from 1 to 15, inclusive. The final result should be stores in an integer variable named **sum**.

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    int sum = 0;
    for (int i = 1; i < 16; i = i+2){
        sum = sum + i;
    }
    cout<<sum;
    return 0;
}
```



```
9 #include <iostream>
10 using namespace std;
11
12 int main()
13 {
14     int i;
15     int sum = 0;
16     for (int i = 1; i < 16; i = i+2){
17         sum = sum + i;
18     }
19     cout<<sum;
20     return 0;
21 }
22
```

64

...Program finished with exit code 0  
Press ENTER to exit console.

7. Write a program: While loop – Sum of even integers

Write code that uses a for loop to calculate the sum of the even integers from 0 to 20, inclusive. The final result should be stores in an integer variable named **sum**.

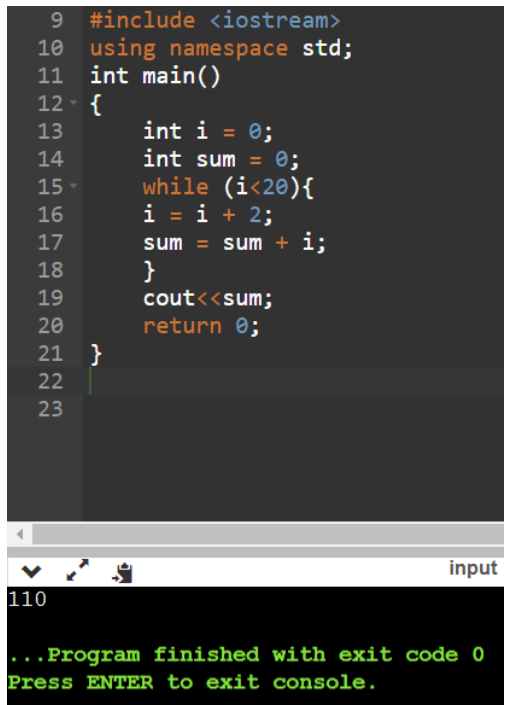
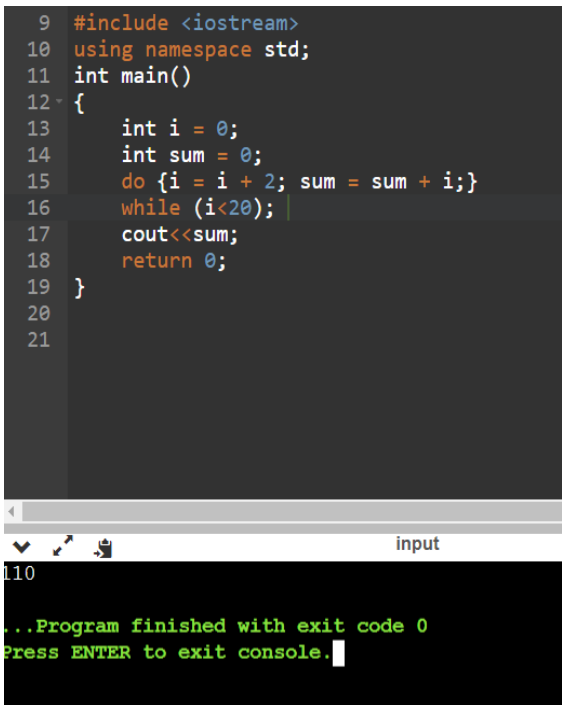
```
#include <iostream>
using namespace std;
int main()
{
    int i = 0;
    int sum = 0;
    while (i<20){
        i = i + 2;
        sum = sum + i;
    }
    cout<<sum;
    return 0;
}
```

```
9  #include <iostream>
10 using namespace std;
11
12 int main()
13 {
14     int i = 0;
15     int sum = 0;
16     while (i<20){
17         i = i + 2;
18         sum = sum + i;
19     }
20     cout<<sum;
21     return 0;
22 }
23
```

input

```
110
...Program finished with exit code 0
Press ENTER to exit console.
```

8. Write difference between while loop and do-while loop with an example

while loop	do-while loop
The condition is tested first, and then the statement(s) are executed.	The statement(s) are run at least once, and the condition is then checked.
If the condition is false, the statement(s) may be executed zero times.	At least once the statement(s) is executed.
There is no semicolon at the end of while. <i>while(condition)</i>	There is a semicolon at the end of while. <i>while(condition);</i>
Brackets are not necessary when there is only one statement.	Brackets are always needed.
Before the loop is executed, the variable in condition is initialised.	Variables can be set before or after the loop.
while loop is an entry-controlled loop.	do-while loop is an exit-controlled loop.
while(condition) { statement(s); }	do { statement(s); } while(condition);
<p>Example (this is based on exercise 7):</p> 	<p>Example (this is based on exercise 7):</p> 



9. Write a program to show the following output. User will enter two inputs number and range. You are allowed to use any loop.

```
Enter the number:
13
Enter the range:
12
13x1=13
13x2=26
13x3=39
13x4=52
13x5=65
13x6=78
13x7=91
13x8=104
13x9=117
13x10=130
13x11=143
13x12=156
```

```
#include <iostream>
using namespace std;
int main()
{
    int number;
    int range;
    int i;
    cout<<"Enter the number: \n";
    cin>>number;
    cout<<"Enter the range: \n";
    cin>>range;
    for (int i = 1; i <= range; i = i + 1){
        cout<<number<<" * "<<i<<" = "<<(number * i);
        cout<<("\n");
    }
    return 0;
}
```

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int number;
6     int range;
7     int i;
8     cout<<"Enter the number: \n";
9     cin>>number;
10    cout<<"Enter the range: \n";
11    cin>>range;
12    for (int i = 1; i <= range; i = i + 1){
13        cout<<number<<" * "<<i<<" = "<<(number * i);
14        cout<<("\n");
15    }
16    return 0;
17 }
18
```

input

```
Enter the number:
13
Enter the range:
12
13 * 1 = 13
13 * 2 = 26
13 * 3 = 39
13 * 4 = 52
13 * 5 = 65
13 * 6 = 78
13 * 7 = 91
13 * 8 = 104
13 * 9 = 117
13 * 10 = 130
13 * 11 = 143
13 * 12 = 156
```

## Report (SWE20004)

Write a one-page report on this lab covering the following:

1. Summarize the topics you explored and the activities you did during this lab.
2. Classify (group) these topics and actions under appropriate headings. Do not just copy the headings used in the instructions. For example, explain the following, what are the following commands do?
3. Discuss the relevance of these topics and actions in terms of C++ programming. i.e. How do the things in this lab work contribute to your understanding of C++ programming overall?
4. Why do you need to understand (and use) C++ **Loop Structures**?

This report is worth 5% towards your unit assessment. Use the below page as a template. Either you can type it or write it in your words.

## Introduction

This week's report will illustrate the topics covered and the activities of the lab. More specifically, we were taught about loop structures, or specifically the three types of loop in C++, including while loop, do-while loop and for loop. In addition, we went over some of the exercises in this week's lab in order to strengthen our knowledge and understanding of the topics.

## Loop structures

In programming, loops are used to continuously repeat a block of code until a certain condition is reached. The execution of loops can be terminated by using the **break** statement. As mentioned above, there are 3 types of loops used in C++:

The first one is **for** loop, with the following syntax:

```
for (initialization; condition; update) {  
    // body of-loop  
}
```

Here,

- Initialisation: initialises variables and is executed only once.
- Condition: if true, the body of the loop is executed; otherwise, the loop will be terminated.
- Update: changes the value of initialised variables and verifies the condition once more time.

The second one is **while** loop, with the following syntax:

```
while (condition) {  
    // body of the loop  
}
```

Here,

- The condition is evaluated via a while loop.
- If the condition is true, the loop's code is executed.
- The condition is assessed once more.
- This procedure is repeated until the condition is no longer true.
- The loop ends when the condition is false.

The third and last one is **do-while** loop, with the following syntax:

```
do (condition) {  
    // body of the loop  
}  
while (condition)
```

Here,

- The loop's body is executed initially. The `condition` is then examined.
- If the `condition` evaluates to true, the `do` statement executes the body of the loop again.
- The `condition` is assessed once more.
- If the `condition` is true, the body of the loop is executed again.
- This process continues until the evaluation of the `condition` is false. Then the loop is terminated.

### **Lab activities**

This week's lab exercises were rather more confusing than challenging, as the three loops could be used to deal with all exercises yet their syntaxes are not similar and could be mistakenly used with one another. Attention to small details is needed for these tasks as missing “{ }” or using the wrong syntax for the wrong loop could make the program produces undesirable results or even become unexecutable.

### **The importance of this week's topic**

Having a good understanding of the types of loops and knowing how to distinguish their syntax will help us to reduce the lines of code when we want to execute an action repeatedly and therefore we could save our time.

### **Conclusion**

Overall, this week's lab helped us to understand the loop structures, or different types of loops used in programs. I could now utilise the loops to reduce the lines of code if I want to execute an action continuously.

Name:       Hong Hai Dang Nguyen       Student ID:   103503191