BSc (Hons) in Information Technology
Year 1

Lab Sheet 05

IT1010 – Introduction to Programming Semester 1, 2021

Objectives:
At the end of the class the students should be able to:

- Write and debug while, do ..while and for loops

Exercise 1

1. This is a sample C program that displays numbers from 1 to 4 using *while* repetition control structure.

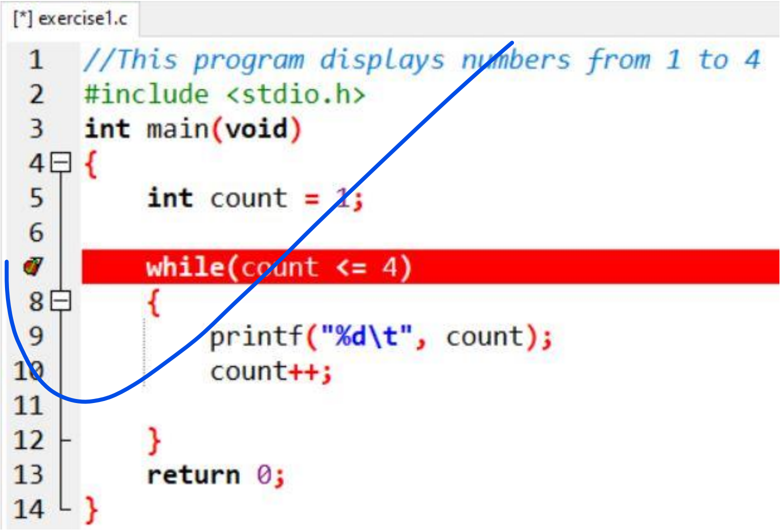
Using the debugging option, observe how the control value changes in given loop.

```
//This program displays numbers from 1 to 4
#include <stdio.h>
int main(void)
{
        int count = 1;

        while(count <= 4)
        {
                printf("%d\t", count);
                count++;

        }
        return 0;
}
```

Step 01
Type the above sample code and save the program as **exercise1.c in** folder **Lab05** in the desktop

Step 02
Compile and run the program

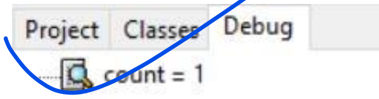
Step 03
Set a break point at the line no. 07.



```
1   //This program displays numbers from 1 to 4
2   #include <stdio.h>
3   int main(void)
4   {
5       int count = 1;
6   
7       while(count <= 4)
8       {
9           printf("%d\t", count);
10          count++;
11  
12      }
13      return 0;
14  }
```

Step 04

Using debugging option, add a watch to the variable called `count`.



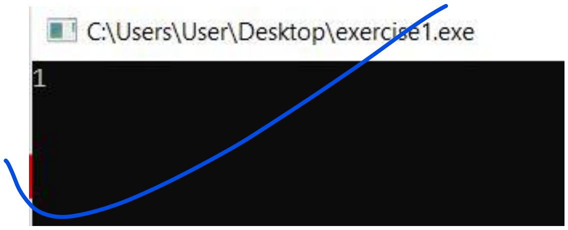
```
count = 1
```

Step 05

Using next line button, execute next statement. Then, the statement in line no. 7 will be executed. Here, the loop condition will be tested for the first time and `(1 <= 4)` will be true. So that, the statements in loop will be executed next.

Step 06

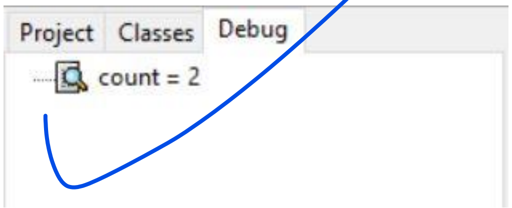
Use next line button again to execute next statement. Then, the statement in line no. 9 will be executed.
In output window, “1” will be printed.



```
1
```

Step 07

Using next line button, execute next statement. Then, the statement in line no. 10 will be executed. The value of `count` variable will be changed to 2.



```
count = 2
```

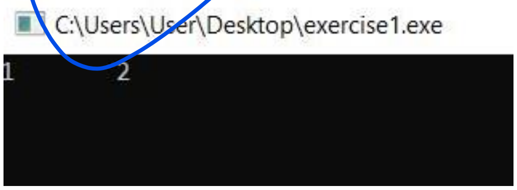
First repetition will be completed.

Step 08

Using next line button, execute next statement. Then, the statement in line no. 7 will be executed again. Here, the loop condition will be tested for the second time and `(2 <= 4)` will be true. So that, the statements in loop will be executed next.

Step 09

Using next line button, execute next statement. Then, the statement in line no. 9 will be executed again. In output window, “2” will be printed.

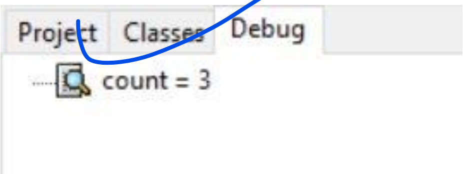


C:\Users\User\Desktop\exercise1.exe

```
1       2
```

Step 10

Using next line button, execute next statement. Then, the statement in line no. 10 will be executed again. The value of `count` variable will be changed to 3.



count = 3

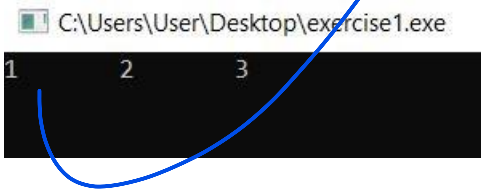
Second repetition will be completed.

Step 11

Using next line button, execute next statement. Then, the statement in line no. 7 will be executed again. Here, the loop condition will be tested for the third time and `(3 <= 4)` will be true. So that, the statements in loop will be executed next.

Step 12

Using next line button, execute next statement. Then, the statement in line no. 9 will be executed again. In output window, “3” will be printed.

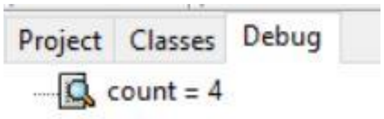


C:\Users\User\Desktop\exercise1.exe

```
1       2       3
```

Step 13

Using next line button, execute next statement. Then, the statement in line no. 10 will be executed again. The value of `count` variable will be changed to 4.



count = 4

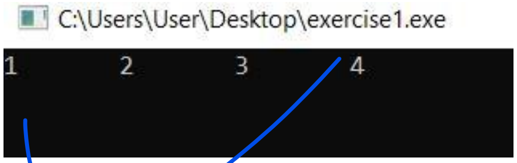
Third repetition will be completed.

Step 14

Using next line button, execute next statement. Then, the statement in line no. 7 will be executed again. Here, the loop condition will be tested for the fourth time and `(4 <= 4)` will be true. So that, the statements in loop will be executed next.

Step 15

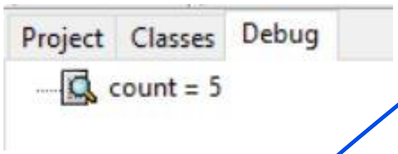
Using next line button, execute next statement. Then, the statement in line no. 9 will be executed again. In output window, "4" will be printed.



```
1         2         3         4
```

Step 16

Using next line button, execute next statement. Then, the statement in line no. 10 will be executed again. The value of `count` variable will be changed to 5.



count = 5

Fourth repetition will be completed.

Step 17

Using next line button, execute next statement. Then, the statement in line no. 7 will be executed again. Here, the loop condition will be tested for the fifth time. But `(5 <= 4)` will be false. So that, the repetition will be terminated.

Step 18

Using next line button, execute next statement. Then, the statement in line no. 13 will be executed. Then, your program execution will be terminated.

Here, you can observe how control variable values are changed within the loop and how loop condition tested. There will be four repetitions and the loop condition will be tested five times.

Step 19

Using **Stop Execution** button, stop debugging process.

2. Type the following program and save the program as **exercise1A.c in** folder **Lab05** in the desktop

```
#include <stdio.h>
int main(void)
{
      int count = 1;

      while(count <= 20)
      {
            printf("%d\t", count);
            count+=2;

      }
      return 0;
}
```

1,3,5,7,9,11,13,15,17,19,21

Debug the above program and aswer the following questions.

i. What are the values taken by count variabe during the execution of the progarm.
ii. How many times the while condition will be checked during the execution of program.

11 times

Exercise 2

1. This is a sample C program to input integer numbers from the keyboard and display until user inputs -1.

Using debugging option, observe the variable value changes in the given loop.

```
/*This is a sample C program to input integer numbers
from the keyboard and display until user inputs -1*/
#include <stdio.h>
int main(void)
{
      int number;

      printf("Enter number : ");
      scanf("%d", &number);

      while(number != -1)
      {
            printf("%d\n", number);

            printf("Enter number : ");
            scanf("%d", &number);
      }
      return 0;
}
```

Step 01

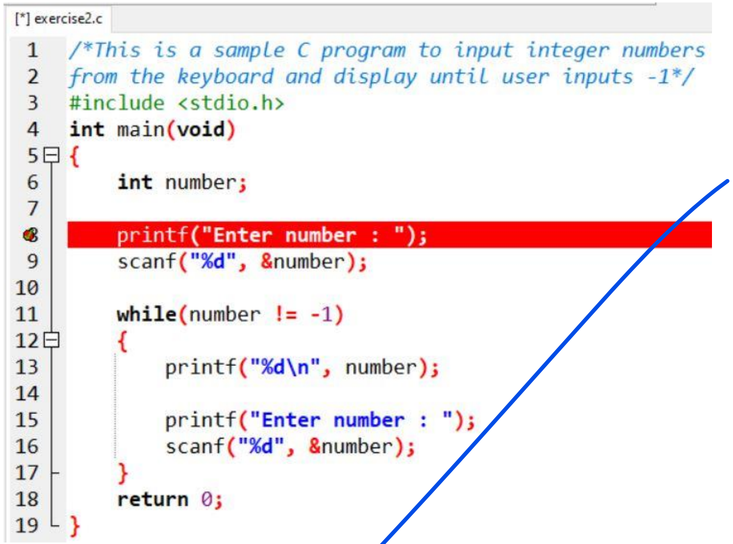
Type the above sample program and save the program as **exercise2.c**

Step 02

Compile and run the program

Step 03

Set a break point at the line no. 08

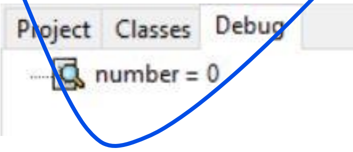


[*] exercise2.c

```
 1  /*This is a sample C program to input integer numbers
 2  from the keyboard and display until user inputs -1*/
 3  #include <stdio.h>
 4  int main(void)
 5  {
 6      int number;
 7  
 8      printf("Enter number : ");
 9      scanf("%d", &number);
10  
11      while(number != -1)
12      {
13          printf("%d\n", number);
14  
15          printf("Enter number : ");
16          scanf("%d", &number);
17      }
18      return 0;
19  }
```

Step 04

Using debugging option, add a watch to the variable called `number`.



number = 0

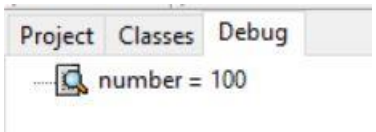
Step 05

Using next line button, execute next statement. Then, the statement in line no. 8 will be executed. In output window, “Enter number : ” will be printed.

Step 06

Using next line button, execute next statement. Then, the statement in line no. 9 will be executed. In output window, you can input 100 as the first user input.

Then, `number` variable value will be changed.



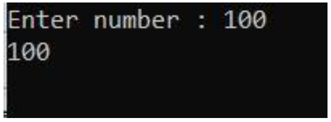
number = 100

Step 07

Using next line button, execute next statement. Then, the statement in line no. 11 will be executed. Here, the loop condition will be tested for the first time and `(100 != -1)` will be true. So that, the statements in loop will be executed next.

Step 08

Using next line button, execute next statement. Then, the statement in line no. 13 will be executed. In output window, "100" will be printed.



```
Enter number : 100
100
```

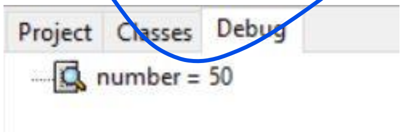
Step 09

Using next line button, execute next statement. Then, the statement in line no. 15 will be executed. In output window, "Enter number : " will be printed.

Step 10

Using next line button, execute next statement. Then, the statement in line no. 16 will be executed. In output window, you can input 50 as the second user input.

Then, `number` variable value will be changed.



number = 50

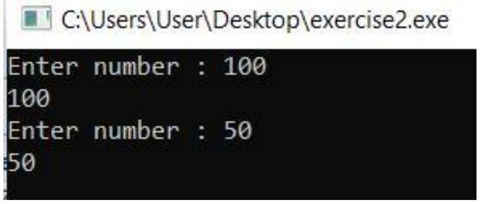
The first repetition will be completed.

Step 11

Using next line button, execute next statement. Then, the statement in line no. 11 will be executed again. Here, the loop condition will be tested for the second time and `(50 != -1)` will be true. So that, the statements in loop will be executed next.

Step 12

Using next line button, execute next statement. Then, the statement in line no. 13 will be executed again. In output window, "50" will be printed.



C:\Users\User\Desktop\exercise2.exe

```
Enter number : 100
100
Enter number : 50
50
```

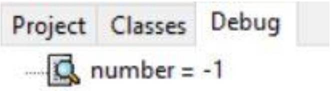
Step 13

Using next line button, execute next statement. Then, the statement in line no. 15 will be executed again. In output window, “Enter number : ” will be printed.

Step 14

Using next line button, execute next statement. Then, the statement in line no. 16 will be executed again. In output window, you can input -1 as the third user input.

Then, `number` variable value will be changed.



number = -1

The second repetition will be completed.

Step 15

Using next line button, execute next statement. Then, the statement in line no. 11 will be executed again. Here, the loop condition will be tested for the third time and `(-1 != -1)` will be false. So that, the repetition will be terminated.

Step 16

Using next line button, execute next statement. Then, the statement in line no. 18 will be executed. Then, your program execution will be terminated.

Here, you can observe how variable values are changed within the loop and how loop condition tested. There will be two repetitions and the loop condition will be tested three times.

Step 17

Using **Stop Execution** button, stop debugging process.

2. Start the debugging process again and set a break point at line no. 08. Using next line button, execute next statements. Here, enter -1 as the first user input. Then, observe how variable values are changed within the loop and how loop condition tested.

If user inputs -1 as the first user input, how many repetitions will be there ?

If user inputs -1 as the first user input, how many times the loop condition will be tested?

Exercise 3

1. This is a sample C program that displays numbers from 1 to 4 using *for* repetition control structure.

Using debugging option, observe the variable value changes in given loop.

```
//This program displays numbers from 1 to 4
#include <stdio.h>
```

```
int main(void)
{
      int count;

      for(count = 1; count <= 4; count++)
      {
            printf("%d\t", count);
      }
      return 0;
}
```

Step 01

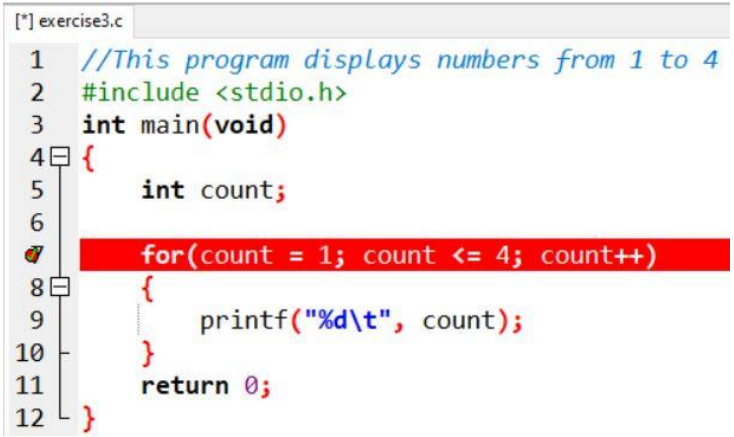
Type the above sample program and save the program as **exercise3.c**

Step 02

Compile and run the program

Step 03

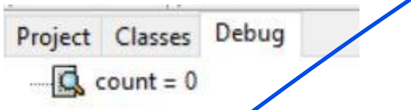
Set a break point at the line no. 07.



```
1  //This program displays numbers from 1 to 4
2  #include <stdio.h>
3  int main(void)
4  {
5      int count;
6
●      for(count = 1; count <= 4; count++)
8      {
9          printf("%d\t", count);
10     }
11     return 0;
12 }
```

Step 04

Using debugging option, add a watch to the variable called `count`.



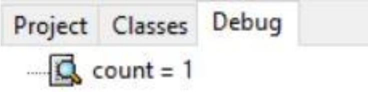
count = 0

Step 05

Using next line button, execute next statement.

Then, the first expression (`count = 1`) and the second expression (`count <= 4`) of *for* loop will be executed.

Since first expression (`count = 1`) is executed, the value of `count` variable is changed to 1.

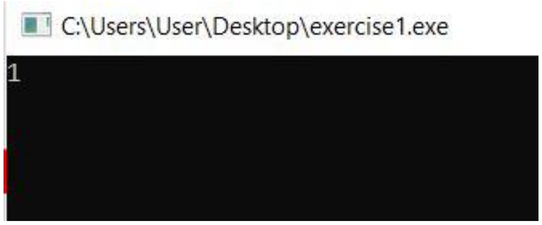


count = 1

Since the second expression is executed, the loop condition will be tested for the first time and `(1 <= 4)` will be true. So that, the statements in loop will be executed next.

Step 06

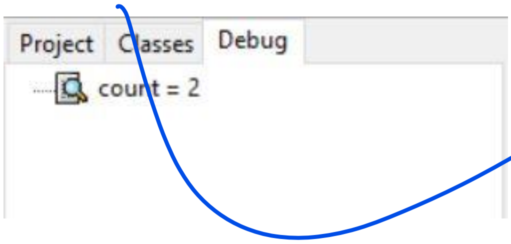
Using next line button, execute next statement. Then, the statement in line no. 9 will be executed. In output window, “1” will be printed.



```
1
```

Step 07

Using next line button, execute next statement. Then, the third expression (`count++`) of *for* loop will be executed. The value of `count` variable will be changed to 2.



count = 2

First repetition will be completed.

Step 08

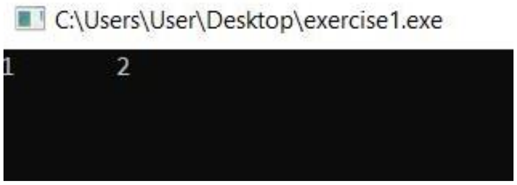
Using next line button, execute next statement.

Then, the second expression (`count <= 4`) of *for* loop will be executed.

Here, the loop condition will be tested for the second time and `(2 <= 4)` will be true. So that, the statements in loop will be executed next.

Step 09

Using next line button, execute next statement. Then, the statement in line no. 9 will be executed again. In output window, “2” will be printed.

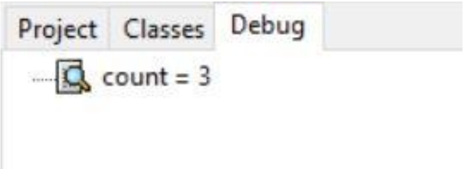


C:\Users\User\Desktop\exercise1.exe

```
1        2
```

Step 10

Using next line button, execute next statement. Then, the third expression (`count++`) of *for* loop will be executed. The value of `count` variable will be changed to 3.



count = 3

Second repetition will be completed.

Step 11

Using next line button, execute next statement.

Then, the second expression (`count <= 4`) of *for* loop will be executed.

Here, the loop condition will be tested for the third time and `(3 <= 4)` will be true. So that, the statements in loop will be executed next.

Step 12

Using next line button, execute next statement. Then, the statement in line no. 9 will be executed again. In output window, “3” will be printed.

Step 13

Using next line button, execute next statement. Then, the third expression (`count++`) of *for* loop will be executed. The value of `count` variable will be changed to 4.

Third repetition will be completed.

Step 14

Using next line button, execute next statement.

Then, the second expression (`count <= 4`) of *for* loop will be executed.

Here, the loop condition will be tested for the fourth time and `(4 <= 4)` will be true. So that, the statements in loop will be executed next.

Step 15

Using next line button, execute next statement. Then, the statement in line no. 9 will be executed again. In output window, “4” will be printed.

Step 16

Using next line button, execute next statement. Then, the third expression (`count++`) of *for* loop will be executed. The value of `count` variable will be changed to 5.

Fourth repetition will be completed.

Step 17

Using next line button, execute next statement.

Then, the second expression (`count <= 4`) of *for* loop will be executed.

Here, the loop condition will be tested for the fifth time and `(5 <= 4)` will be false. So that, the repetition will be terminated.

Step 18

Using next line button, execute next statement. Then, the statement in line no. 11 will be executed. Then, your program execution will be terminated.

Here, you can observe how variable values are changed within the loop and how loop condition tested. There will be four repetitions and the loop condition will be tested five times.

Step 19

Using **Stop Execution** button, stop debugging process.