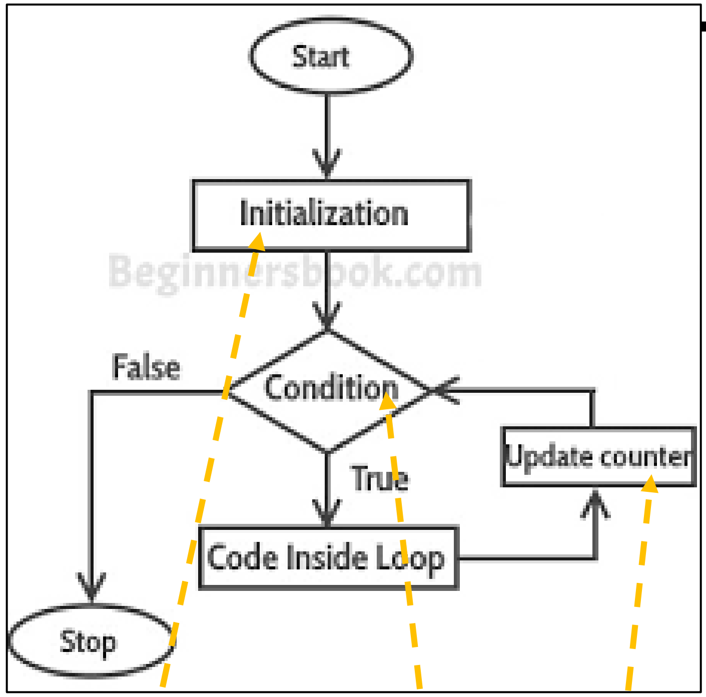
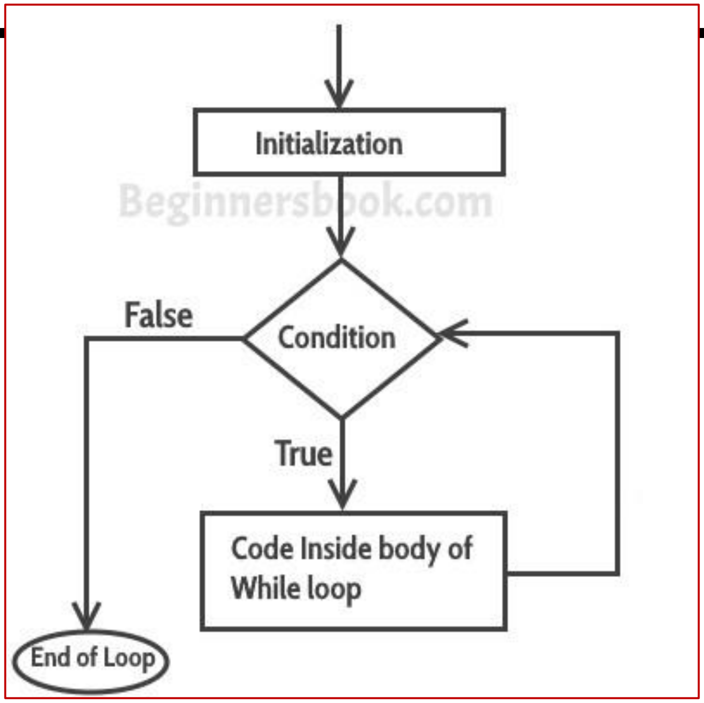
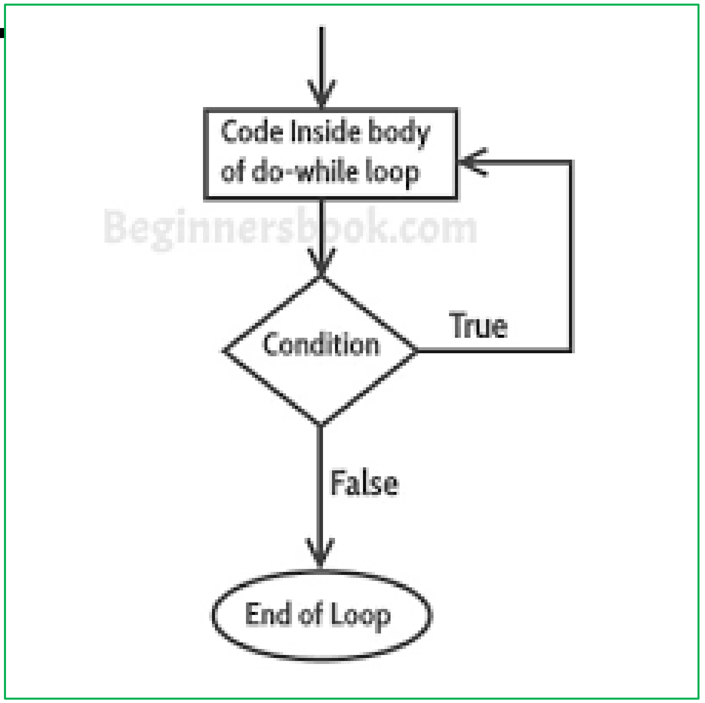
Data Structure & Programming I

Topic 7- Loop **do-while**

Using Loops







Loop: **for**

Loop: **while**

Loop: **do … while**

initialization Condition Update

```
for(int k=10; k>0; k=k-1){
      printf("%d", k);
}
```

```
int k=5;
while( k < 10 ){
      printf("%d", k);
}
```

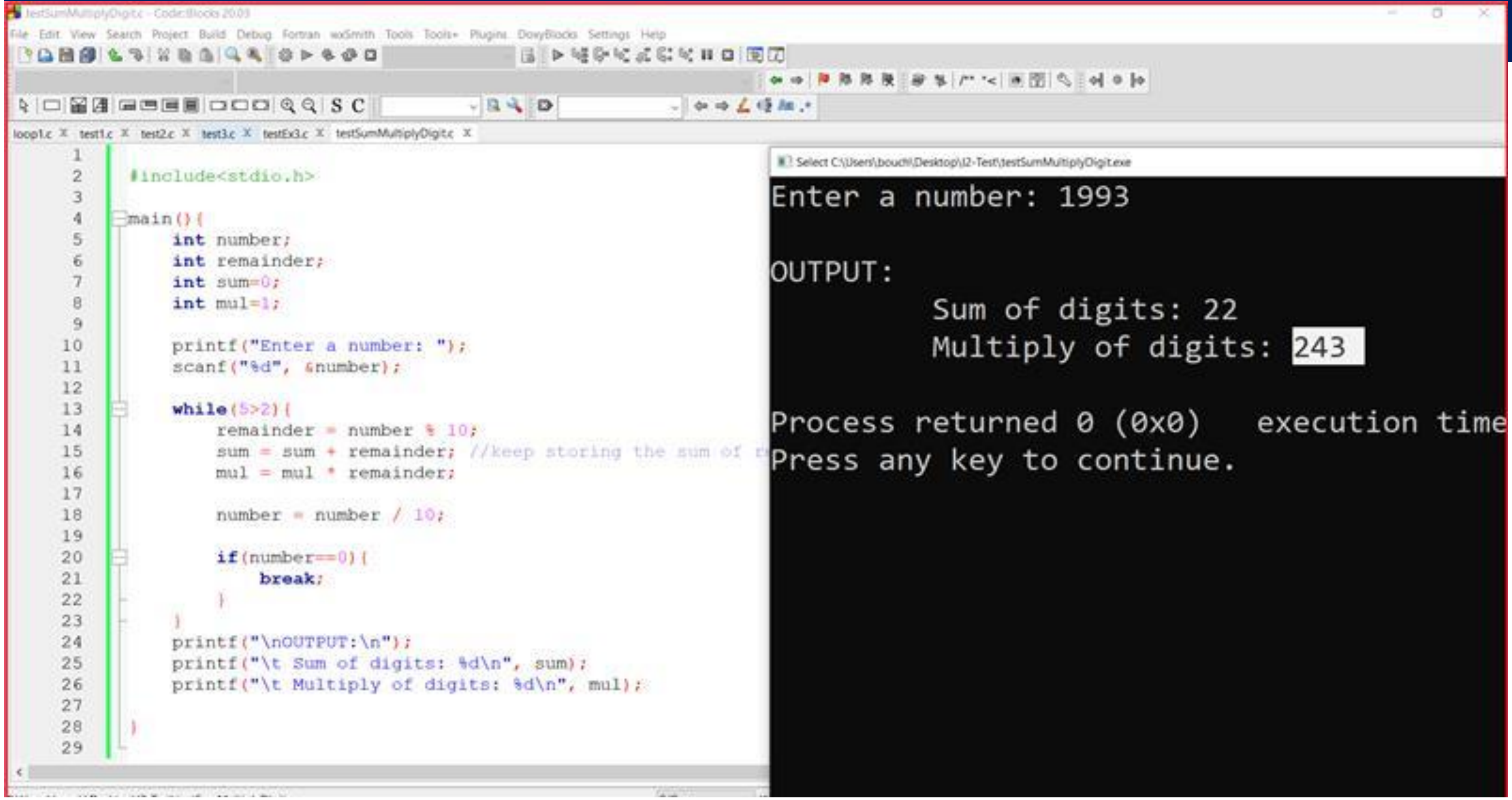
```
int k=20;
do{
      printf("%d", k);
}while( k < 10 );
```

OUTPUT:

OUTPUT:

OUTPUT:

Sum and multiply of digits for a given number!


testSumMultiplyDigit.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
loop1.c test1.c test2.c test3.c testEx3.c testSumMultiplyDigit.c
Select C:\Users\bouch\Desktop\I2-Test\testSumMultiplyDigit.exe

```c
#include<stdio.h>

main(){
    int number;
    int remainder;
    int sum=0;
    int mul=1;

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

    while(5>2){
        remainder = number % 10;
        sum = sum + remainder; //keep storing the sum of
        mul = mul * remainder;

        number = number / 10;

        if(number==0){
            break;
        }
    }
    printf("\nOUTPUT:\n");
    printf("\t Sum of digits: %d\n", sum);
    printf("\t Multiply of digits: %d\n", mul);

}
```

```
Enter a number: 1993

OUTPUT:
         Sum of digits: 22
         Multiply of digits: 243

Process returned 0 (0x0)   execution time
Press any key to continue.
```

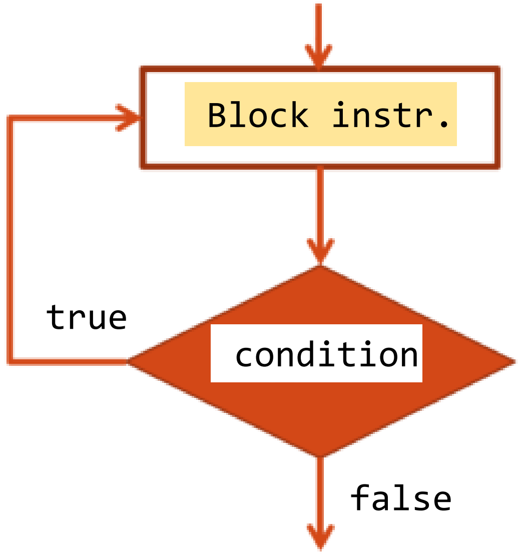
DO … WHILE

DO WHILE loop

- Syntax:

loopback condition for stopping loop when it turns `false`

```
do
   block of instructions
while(condition)
```



- Instruction from block of instructions can control the `condition`
- The block of instructions can be executed at least once

DO ... WHILE

DO WHILE loop

- Examples:

1

```
var n: integer
begin
    n ← 10
    do
        write(n)
        n ← n-2
    while(n>0)
end
```

2

```
var n: integer
begin
    n ← -10
    do
        write(n)
        n ← n+2
    while(n<=0)
end
```

3

```
var n: integer
begin
    read(n)
    do
        write(n)
        n ← n-1
    while(n>0)
end
```

Output:

```
10 8 6 4 2
```

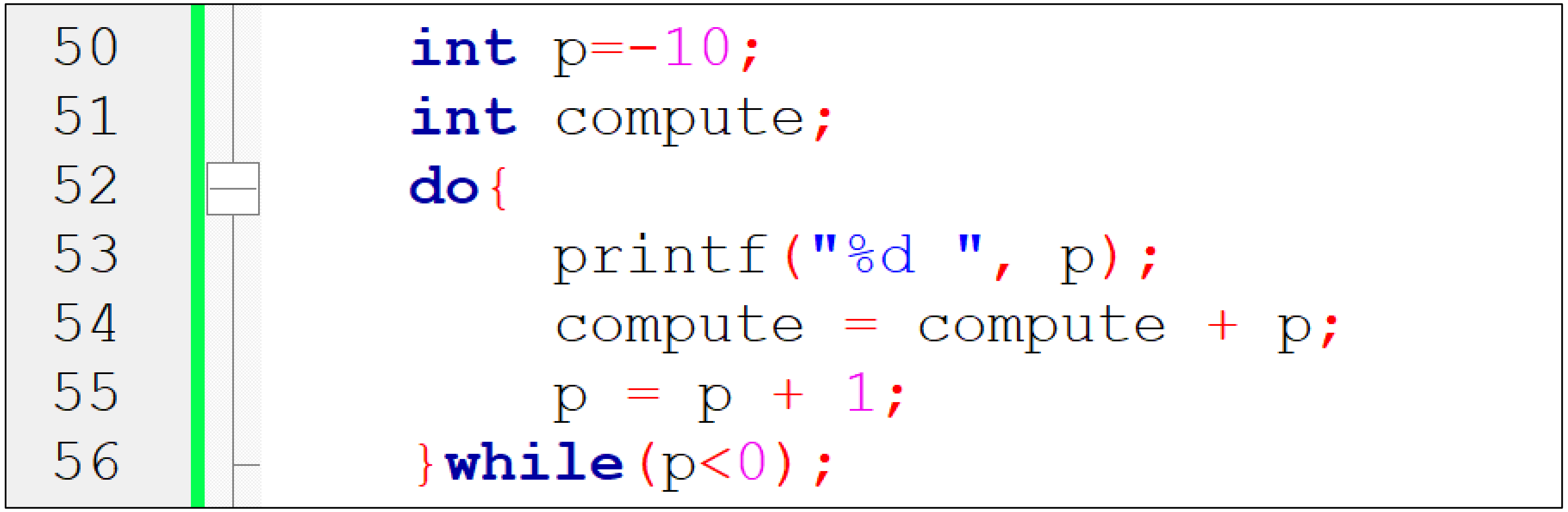
Output:

```
-10 -8 -6 -4 -2 0
```

Output:

```
?
```

Example: Using **do-while** loop in C programming



```
int p=-10;
int compute;
do{
    printf("%d ", p);
    compute = compute + p;
    p = p + 1;
}while(p<0);
```

What is the value of the variable ***compute*** after the loop finishes?

Infinite loop

Notation on infinite loop

- Example:

```
var n : integer
begin
    n ← 10
    while(n>0) do
        write(n, " ")
    end while
end
```

```
var n : integer
begin
    n ← 10
    do
        write(n, " ")
        n ← n+3
    while(n>0)
end
```

Output:

```
10 10 …
```

Output:

```
10 13 16 …
```

Break Vs. Continue keyword

`break` statement breaks the loop/switch whereas

`continue` skip the execution of current iteration and continue to the next iteration (it does not break the loop/switch)

Loops: **while** Vs. **do while**

What are the main differences?

```
while(condition){
      //body codes
}
```

Check condition first. Then run codes only if the condition is true.

```
do{
      //body codes
}while(condition);
```

Run codes first then check condition. Then run codes again if the condition is true.

Assignment

Deadline: **1 week**

`Write a C program using` **`do-while`** `loop to …`

1. Display all numbers from 99 to 1.
2. Display all numbers from 1 to 100 except the number 50.
3. Display odd numbers between 8 to 1000 except the numbers 11, 17 and 21.
4. Show all integer divisible by 3 between 1 to 100 except 30, 60, and 90.
5. Sum all numbers from 1 to 100 then display the result.
6. Multiply all numbers from 1 to 100 then display the result.

<u>Assignment</u>

Deadline: **1 week**

`Write a C program using` **`do-while`** `loop to …`

7. Compute and display the summation of the suit cube number starting from n up to 1, where n is the input number entered by a user, n is greater than 1.

   <u>Ex</u>: Suppose the input is 3, then display $3\text{^}3 + 2\text{^}3 + 1\text{^}3 = 36$

8. Check whether an input number is a primary number or not. The program runs indefinitely so that we can always check another input number.

9. Display all primary numbers in between 2 to 500.

10. Read 20 input numbers from a user and then find the maximum number and display it on screen.

Assignment

Deadline: **1 week**

`Write a C program using` **`do-while`** `loop to …`

11. Ask a user for a number, say n. Display all primary numbers from 2 to n. Keep the program running by asking the user to input another number again and again. The program stops only when the user input n as a negative number.