While Loop

- The mechanism for repeating a set of statements allows execution to continue for as long as a specified logical expression evaluates to true.

  While this condition is True   or   while you are hungry
  Keep on doing this   eat sandwiches

- The general syntax for the while loop is as follows (one statement in body):

```
while( expression )          or          while (expression)
statement1;                              {
                                                  statement1;
                                                  statement2;
                                         }
```

- The condition for continuation of the while loop is tested at the start (top of the loop)
  - ➔ Pre-test loop
- If expression starts out false, none of the loop statements will be executed.
  - ➔ If you answer the first question “No, i’m not hungry, “ then you don’t get to eat any sandwiches all , and you move straight to the coffee.
- If the loop condition starts out as true, the loop body must contain a mechanism for changing this if the loop is to end.

Counter controlled while loop Example:
//Program to introduce the while statement.

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

while (count >= 5)
{
printf(“%i\n”, count );
++count;
}
return (0);
}
```

Logic Controlled While loop example

```
    int num = 0;
    scanf("%d\n",&num);

    While (num != -1)
    {
    //loop actions
    scanf("%d", &num);
    }
```

Do-while loop

- In the while loop, the body is executed while the condition is true.
- The do-while loop is a loop where the body is executed for the first time unconditionally.
    - ➔ Always guaranteed to execute at least once
    - ➔ Condition is at the bottom (post-test loop)
- After initial execution, the body is only executed while the condition is true

```
do                    do
statement             {
while ( expression);        prompt for password
                            Read user input
                      }
                      while (input not equal to password);
```

Do- While loop example

```
do
    scanf("%d", & number);
While (number != 20);
```

Or counter controlled

```
int number = 4;
do
{
printf("\nNumber = % d", number );
number++;
}
while (number < 4 );
```

- To make a while like a for, preface it with an initialization and include update statements.

    Initialize;

```
    while (test)
    {
```

```
Body;
Update;
}
```

Is the same as

```
for (initialized; test; update)
body;
```

- A for loop is appropriate when the loop involves initializing and updating a variable.
- A while loop is better when the conditions ARE otherwise.
- I prefer to use while loop for logic controlled loops and the for loop for counter controlled loops.
  while (scanf("%I" &num) == 1)
  For (count = 1 ; count <= 100; count ++)