Lecture 07

Repetition Structures

CSE115: Computing Concepts

An Example Problem

- Read in 10 integers and output their sum

An Example Problem

- Read in 10 integers and output their sum

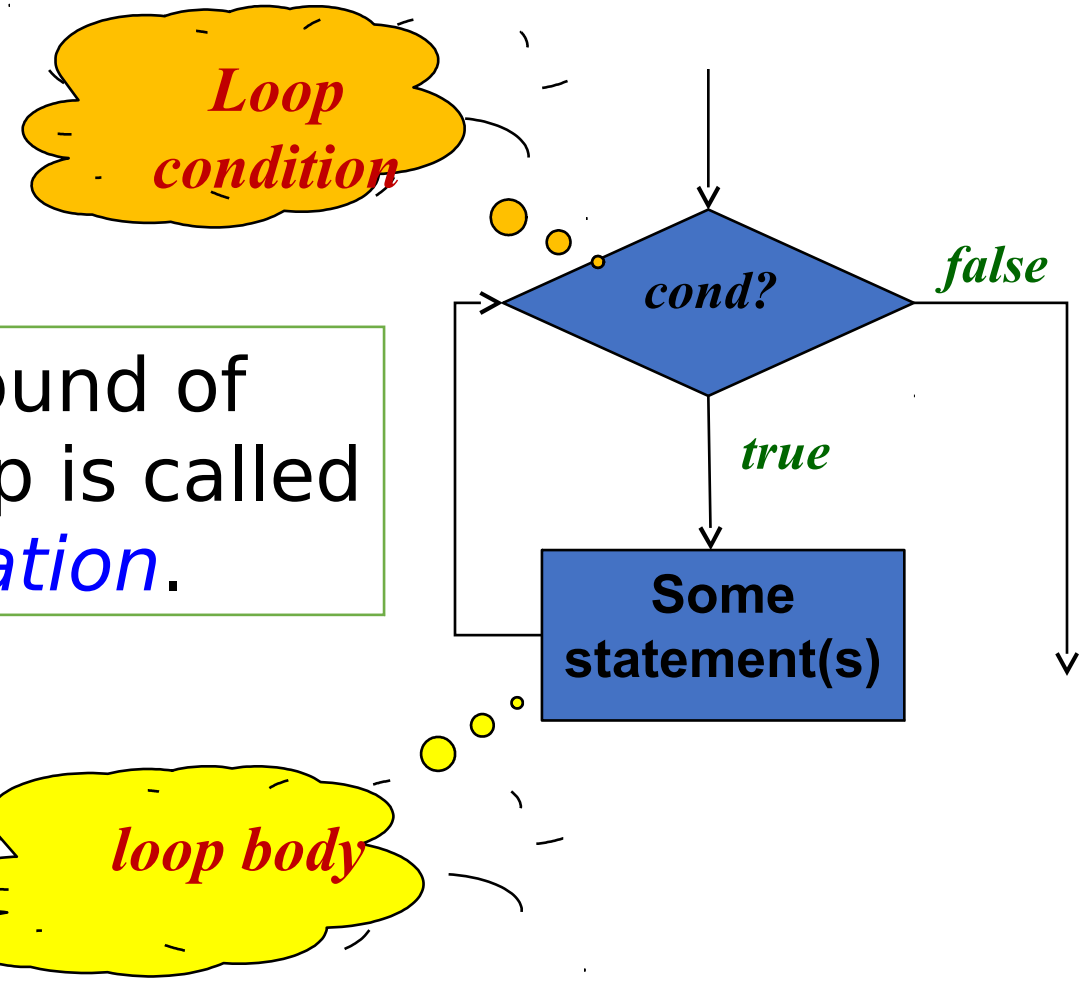
```
#include <stdio.h>
int main()
{
    int a, sum = 0;
    printf("Enter a number:
");
    scanf("%d",&a);
    sum += a;
    printf("Enter a number:
");
    scanf("%d",&a);
    sum += a;
    printf("Enter a number:
");
    scanf("%d",&a);
    sum += a;
    printf("Enter a number:
");
    scanf("%d",&a);
    printf("Enter a number: ");
    scanf("%d",&a);
    sum += a;
    printf("Enter a number: ");
    scanf("%d",&a);
    sum += a;
    printf("Enter a number: ");
    scanf("%d",&a);
    sum += a;
    printf("Enter a number: ");
    scanf("%d",&a);
    sum += a;
    printf("Enter a number: ");
    scanf("%d",&a);
    sum += a;
    printf("Enter a number: ");
    scanf("%d",&a);
    sum += a;
    return 0;
}
```

Repetition Structure (Loop)

- Executes a number of statements more than one time without having to write the statements multiple times.

- Two designs of loop :
  - To execute a number of instructions from the program for a finite, **pre-determined number of time** (Counter-controlled loop)
  - To execute a number of instructions **indefinitely until the user tells it to stop or a special condition is met** (Sentinel-controlled loop)

Repetition Structure (Loop)



Each round of the loop is called an *iteration*.

Repetition Structure (Loop)

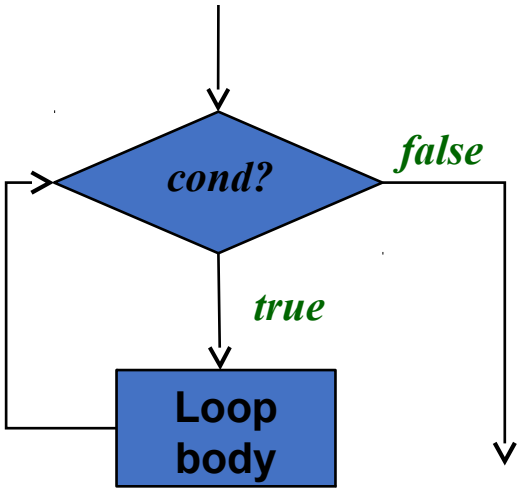
- There are 3 types of loops in C:
  - **`while`**
  - **`for`**
  - **`do while`**

Repetition : **`while`** Loop

- Syntax :

```
while (condition)
   statement;
```

Similar as in the *if* statement, the condition is an expression that can return *true* or *false*.



- As long as the condition is met (the *condition* expression is *true*), the statement inside the *while* loop (also called loop body) will get executed.
- When the *condition* is no longer met (the condition expression is false), the program will continue on with the next instruction (the one after the *while* loop).

Repetition : **`while`** Loop

- In this example :
  - **`(i < 5)`** is known as loop *repetition condition* (counter-controlled) .
  - **`i`** is the loop *counter variable.*
  - In this case, this loop will keep on looping until the *counter variable* is equal to 4. Once i = 5, the loop will terminate.
- The `printf()` statement will get executed as long as the variable **`i`** is less than 5. Since the variable **`i`** is incremented each time the loop is executed, the loop will stop after the 5th output.
- Output:

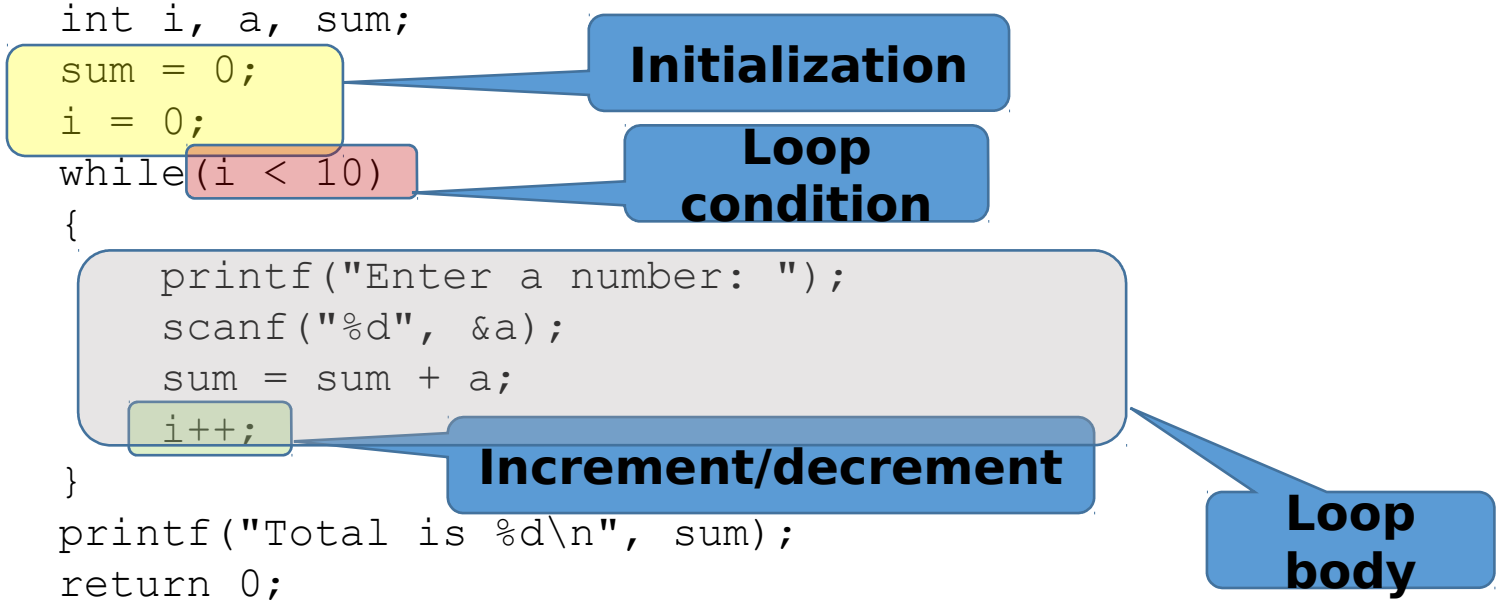
```
i = 0
i = 1
i = 2
i = 3
i = 4
Done
```

Example:

```
int i = 0;
while (i < 5)
{
      printf("i = %d\n", i);
      i++;
}
printf("Done");
```

Sum of 10 Integers

```
#include <stdio.h>
int main()
{
    int i, a, sum;
    sum = 0;
    i = 0;
    while(i < 10)
    {
        printf("Enter a number: ");
        scanf("%d", &a);
        sum = sum + a;
        i++;
    }
    printf("Total is %d\n", sum);
    return 0;
}
```



Repetition : `for` Loop

- Syntax :

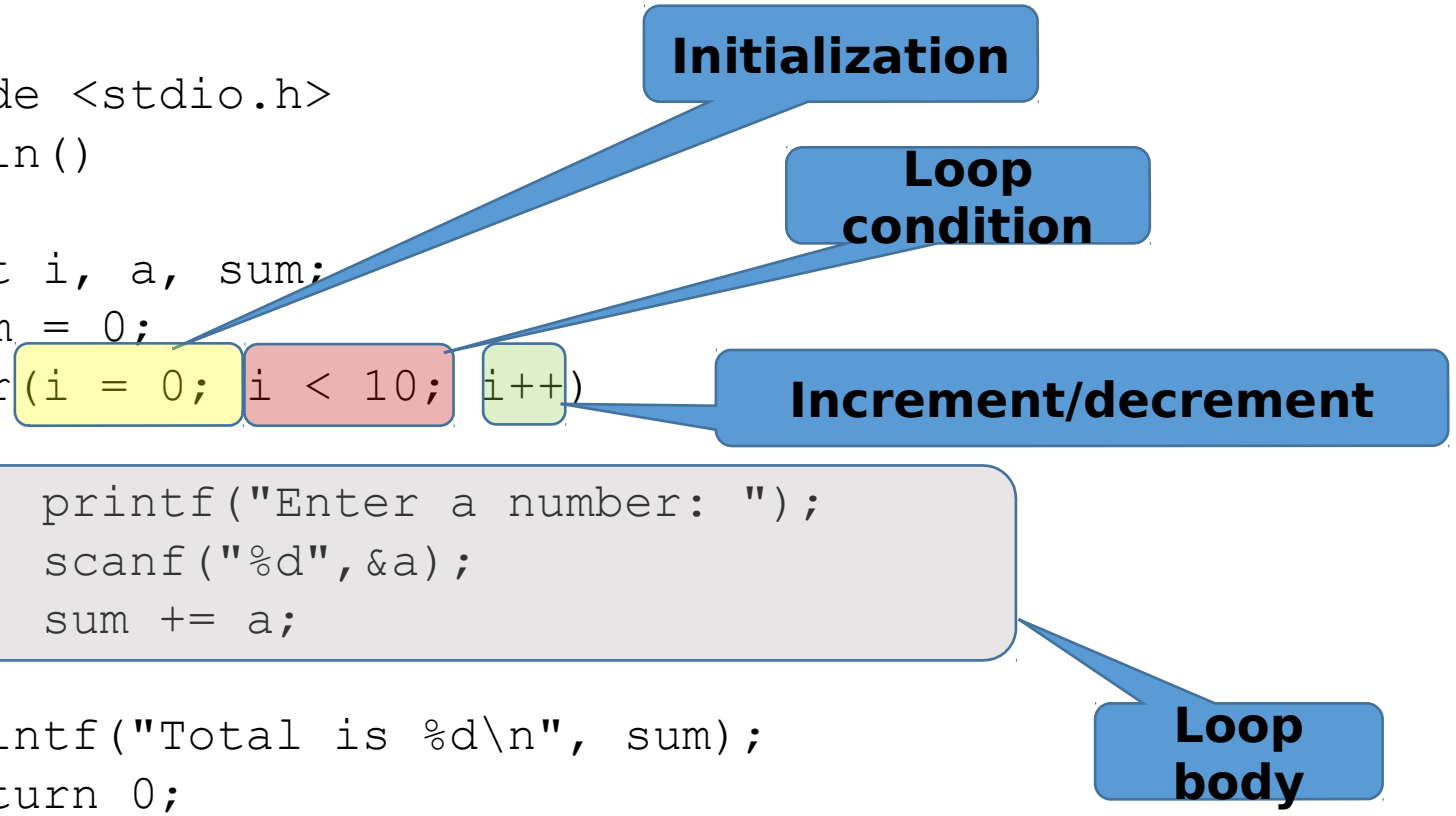
```
for (expression1; expression2; expression3)
     statement;
```

  - expression1: initialize variables including the loop counter variable
  - expression2: the loop condition
  - expression3: changes the value of loop counter variable after each iteration (one cycle of the loop)

- Note that each expression is separated by a semicolon (;)

Sum of 10 Integers

```
#include <stdio.h>
int main()
{
    int i, a, sum;
    sum = 0;
    for(i = 0; i < 10; i++)
    {
        printf("Enter a number: ");
        scanf("%d",&a);
        sum += a;
    }
    printf("Total is %d\n", sum);
    return 0;
}
```



Repetition : `for` Loop

- Notice that the output is the same as the one for the *while* loop example. In fact, the two examples are exactly equivalent. Using a *for* loop is just another way of writing a *while* loop that uses a controlling variable.

Repetition : `for` Loop

- It is also possible to omit one or more of the *for* loop expressions. In such a case, we just put the semicolon without the expression.

```
#include <stdio.h>
int main()
{
    int i, a, sum;
    sum = 0;
    i = 0;
    for(; i < 10; i++)
    {
        printf("Enter a number: ");
        scanf("%d",&a);
        sum += a;
    }
    printf("Total is %d\n", sum);
    return 0;
}
```

Repetition : `for` Loop

- It is also possible to have multiple initializations/increment/decrement statements separated by comma.

```
#include <stdio.h>
int main()
{
    int i, a, sum;
    for(i = 0, sum = 0; i < 10; i++)
    {
        printf("Enter a number: ");
        scanf("%d",&a);
        sum += a;
    }
    printf("Total is %d\n", sum);
    return 0;
}
```

Loop Until User Inputs 100

```
#include <stdio.h>
int main()
{
    int a;
    printf("Enter a number: ");
    scanf("%d",&a);
    for(; a != 100; )
    {
        printf("Enter a number: ");
        scanf("%d",&a);
    }
    return 0;
}
```

Repetition : **`do while`** Loop

- Syntax

```
do
{
   statement;
} while(condition);
```

- A ***do while*** loop is pretty much the same as the ***while*** loop except that the condition is checked after the iteration is complete.
- When there is a ***do while*** loop, the statement(s) inside it will be executed once no matter what. Only after that, the condition will be checked to decide whether the loop should be executed again or just continue with the rest of the program.

Repetition : **do while** Loop

```
#include <stdio.h>
int main()
{
    int a;
    do
    {
        printf("Enter a number: ");
        scanf("%d",&a);
    }
    while(a != 100);
    return 0;
}
```

Repetition : `do while` Loop

- Print the digits of an integer number in reverse order. For example if the input is 45718, your program should output 81754.

Repetition : **do while** Loop

- Print the digits of an integer number in reverse order. For example if the input is 45718, your program should output 81754.

```
#include <stdio.h>
int main()
{
    int number, digit;
    printf(“Enter a number: ");
    scanf("%d", &number);
    while(number > 0)
    {
        digit = number % 10;
        printf("%d", digit);
        number = number / 10;
    }
    return 0;
}
```

Repetition : `do while` Loop

- Sample run 1:
  - Enter a number: 2145
  - 5412
- Sample run 2:
  - Enter a number: 9
  - 9
- Sample run 3:
  - Enter a number: 0
- Sample run 4:
  - Enter a number: 8711541
  - 1451178

Infinite Loop

```
#include <stdio.h>
int main()
{
    int i;
    for(i = 0; i != 9; i+=2)
        printf("i = %d\n", i);
    return 0;
}
```

How many times will the loop iterate?

Infinite Loop

- If somehow the program never goes out of the loop, the program is said to be stuck in an **infinite loop**.
- The infinite loop bug happens because the condition expression of the while loop always return a *true*.
- If an infinite loop occurs, the program would never terminate and the user would have to terminate the program by force.