C language

Imad Kissami[1]
[1]Mohammed VI Polytechnic University, Benguerir, Morocco

Repetition & Looping

Example 1

```c
/* Read two integers and print sum */
int num1, num2, sum;

scanf("%d %d", &num1, &num2);
sum = num1 + num2;

printf("%d + %d = %d", num1, num2, sum);
```

- What if we want to process three different pairs of integers?

Repetition & Looping

Example 2

- One solution is to copy and paste the necessary lines of code. Consider the following modification:

```
1  scanf("%d %d", num1, num2);
2  sum = num1 + num2;
3  printf("%d + %d = %d\n", num1, num2, sum);
4
5  scanf("%d %d", &num1, &num2);
6  sum = num1 + num2;
7  printf("%d + %d = %d", num1, num2, sum);
8
9  scanf("%d %d", &num1, &num2);
10 sum = num1 + num2;
11 printf("%d + %d = %d", num1, num2, sum);
```

- What if you wanted to process four sets? Five? Six? $\cdots$

Repetition & Looping

Processing an arbitrary number of pairs

- We might be willing to copy and paste to process a small number of pairs of integers but
- How about 1,000,000 pairs of integers?
- The solution lies in mechanisms used to control the flow of execution
- In particular, the solution lies in the constructs that allow us to instruct the computer to perform a task repetitively

Repetition & Looping

Repetition (Looping)

- Use looping when you want to execute a block of code several times
  - Block of code = Body of loop
- C provides three types of loops
  - while statement
    * Most flexible
    * No 'restrictions'
  - for statement
    * Natural 'counting' loop
  - do-while statement
    * Always executes body at least once

Repetition & Looping

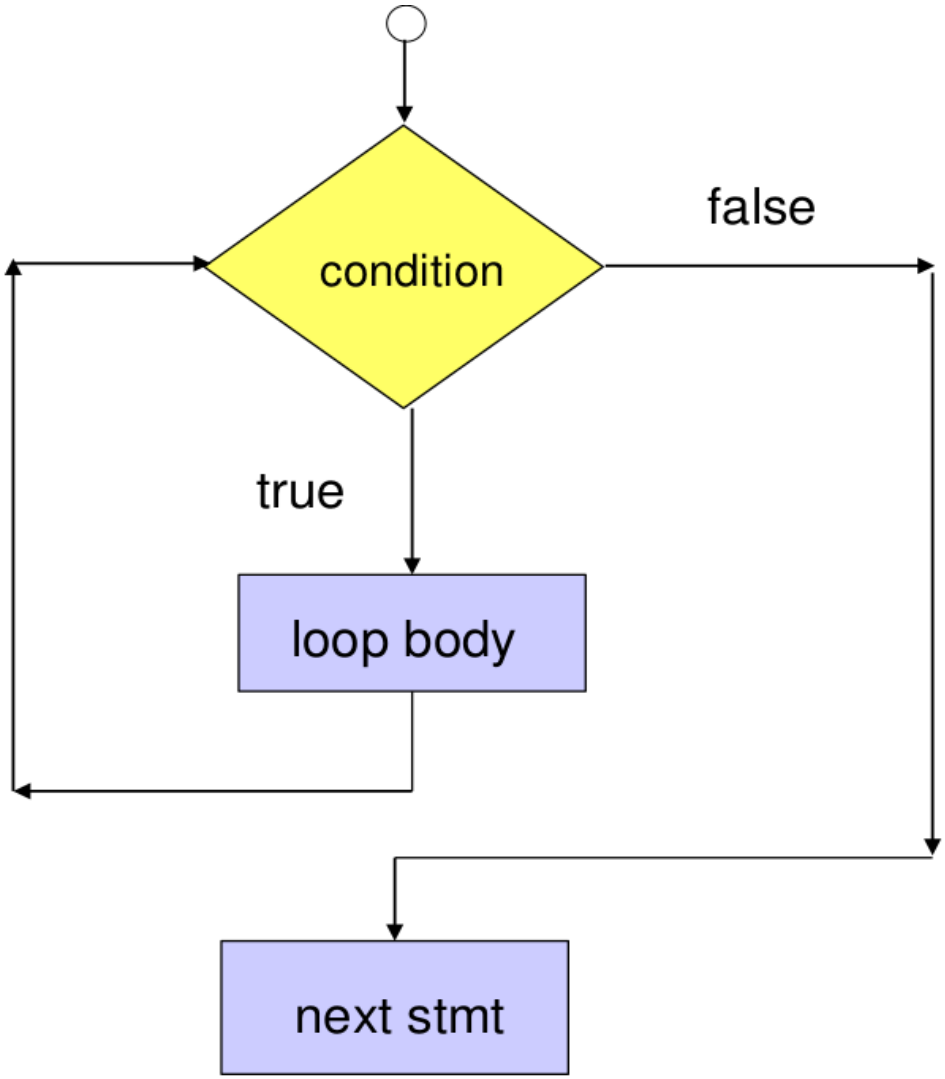
The while Repetition Structure

- Repetition structure
  - Programmer specifies
    - Condition under which actions will be executed
    - Actions to be repeated
  - Pseudocode
    - While there are more items on my shopping list Purchase next item and cross it off my list
- while loop repeated
  - As long as condition is true
  - Until condition becomes false

Repetition & Looping

The while Repetition Structure

- The condition is tested
- If the condition is true, the loop body is executed and the condition is retested.
- When the condition is false, the loop is exited.



Repetition & Looping

The while Repetition Structure

- Syntax:

```
1 while (expression)
2     basic block
```

- Expression = Condition to be tested
  - Resolves to true or false
- Basic Block = Loop Body
  - Reminder – Basic Block:
    * Single statement or
    * Multiple statements enclosed in braces

Repetition & Looping

Loop Control Variable (LCV)

- The loop control variable is the variable whose value controls loop repetition.
- For a while loop to execute properly, the loop control variable must be
  - declared
  - initialized
  - tested
  - updated in the body of the loop in such a way that the expression/condition will become false
    * If not we will have an endless or infinite loop

Repetition & Looping

Counter-Controlled Repetition

- Requires:
  1. Counter variable , LCV, initialized to beginning value
  2. Condition that tests for the final value of the counter (i.e., whether looping should continue)
  3. Constant increment (or decrement) by which the control variable is modified each time through the loop
- Definite repetition
  - Loop executes a specified number of times
  - Number of repetitions is known

Repetition & Looping

Example 3

```
1  int count;                  // LCV: Loop Control Variable
2  int num1, num2, sum;
3
4  count = 0;                  // 1. Initialize LCV
5
6  while (count < 5)          // 2. Test LCV
7
8      scanf("%d %d", &num1, &num2);
9      sum = num1 + num2;
10     printf("%d + %d = %d", num1, num2, sum);
11     count++;                // 3. Increment LCV
```

EXECUTION CHART

count	count<5	repetition
0	true	1
1	true	2
2	true	3
3	true	4
4	true	5
5	false	

Repetition & Looping

Loop Pitfalls

```
1  int count;                  // LCV: Loop Control Variable
2  int num1, num2, sum;
3
4  count = 0;                  // 1. Initialize LCV
5
6  while (count < 5)           // 2. Test LCV
7
8      scanf("%d %d", &num1, &num2);
9      sum = num1 + num2;
10     printf("%d + %d = %d", num1, num2, sum);
11     count++;                // 3. Increment LCV
```

EXECUTION CHART

count	count<5	repetition
0	true	1
1	true	2
2	true	3
3	true	4
4	true	5
5	false	

Repetition & Looping

Loop Pitfalls

```
1  // Echo numbers entered back to user
2  printf("Enter number or zero to end: ");
3  scanf("%d", &num);
4
5  while(num != 0);
6  {
7      printf("Numer is %d\n", num);
8      printf("Enter another number or zero to end: ");
9      scanf("%d", &num);
10 }
```

```
1 Enter value or zero to end: 2
```

- What is wrong with my program? It just sits there!!

Repetition & Looping

Loop Pitfalls

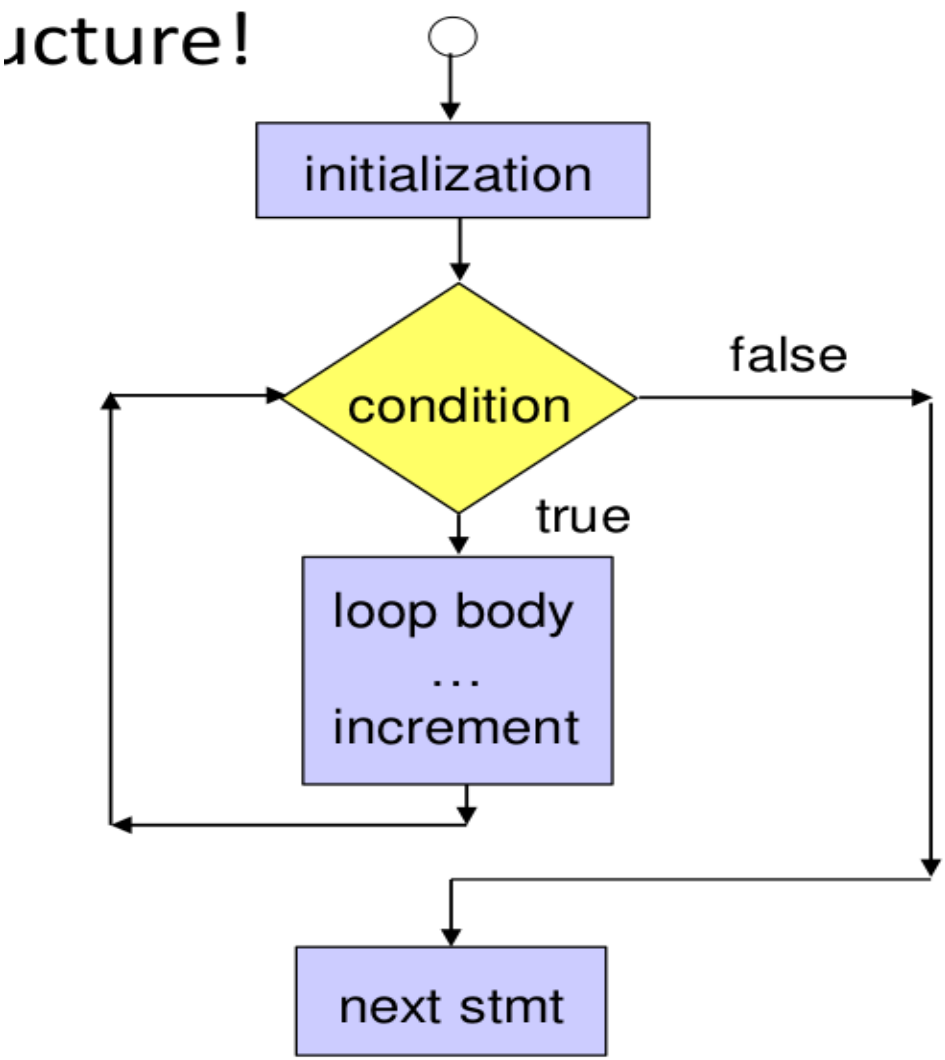
```
1  // Echo numbers entered back to user
2  printf("Enter number or zero to end: ");
3  scanf("%d", &num);
4
5  while(num != 0);      /*<-- Do not place semi-colon here !!*/
6  {
7      printf("Numer is %d\n", num);
8      printf("Enter another number or zero to end: ");
9      scanf("%d", &num);
10 }
```

- Notice the ';' after the while condition!
  - Body of loop is between) and ;
- Result here: INFINITE LOOP!
  - Ctrl-c = Kill foreground process

Repetition & Looping

The while Repetition Structure

- A natural 'counting' loop
- Steps are built into for structure!
  1. Initialization
  2. Loop condition test
  3. Increment or decrement



Repetition & Looping

The for Repetition Structure

- Syntax:

```
for (initialization; test; increment)
    basic block
```

- Example: Prints the integers from one to ten

```
int counter;
for(counter=1; counter <= 10; counter++)
{
    printf("%d\n", counter);
}
```

```
int counter;
counter = 1;
while (counter <= 10)
{
    printf("%d\n", counter);
    counter++;
}
```

Repetition & Looping

The for Repetition Structure

- How many times does loop body execute?

```
1 int counter;
2 for(counter=0; counter < 3; counter++)
3 {
4     printf("Bite %d -- ", counter);
5     printf(""Yum!\n");
6 }
```

Repetition & Looping

The for Repetition Structure

- How many times does loop body execute?

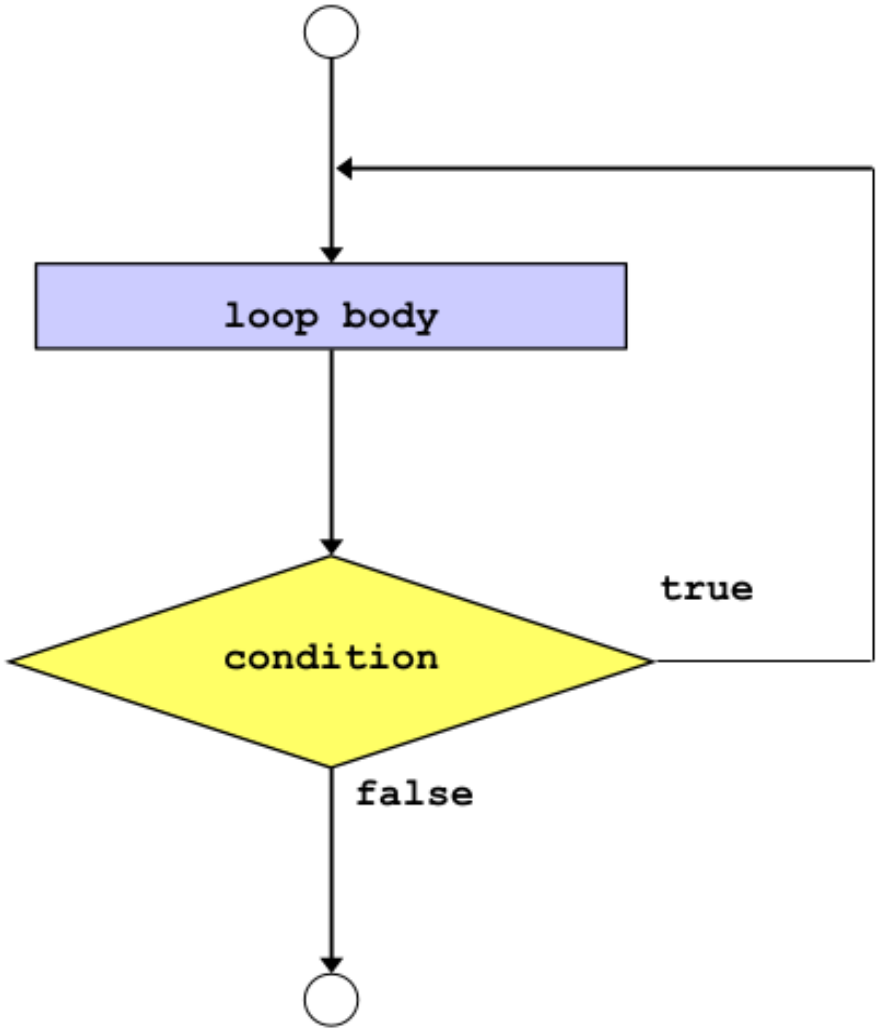
```
1 int counter;
2 for(counter=0; counter < 3; counter++)
3 {
4     printf("Bite %d  -- ", counter);
5     printf(""Yum!\n");
6 }
```

```
1 Bite 1 -- Yum!
2 Bite 2 -- Yum!
3 Bite 3 -- Yum!
```

Repetition & Looping

The do-while Repetition Structure

- The do-while repetition structure is similar to the while structure
  - Condition for repetition tested after the body of the loop is executed



Repetition & Looping

The do-while Repetition Structure

- Syntax:

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

- Example 1: Prints the integers from one to ten

```
int counter;
counter = 1;
do
{
    printf("%d\n", counter);
    counter++;
} while (counter <= 10);
```

- Example 2: Makes sure that the user enters a valid weight

```
do
{
    printf("Enter a positive weight! ");
     scanf("%d", &weight)
} while (weight <= 0);
```

Repetition & Looping

The break/continue Statements

- break
  - Causes immediate exit from a while, for, do/while or switch structure
  - We will use the break statement only to exit the switch structure!
- continue
  - Control passes to the next iteration
  - We will not use the continue statement!