C Programming

Trainer : Nisha Dingare

Email : nisha.dingare@sunbeaminfo.com

Loops - Introduction :

- Control statements used for repeating a set of instructions number of times is called as “LOOP”.
- Loops help us to execute a statement /block of statements a certain number of times.
- Loops can be categorized as :
  1) Entry Controlled Loops
     1) While Loop
     2) For Loop

  2) Exit Controlled Loops
     1) Do – While Loop
- Every loop has
  - Initialization statement
  - Terminating condition
  - Modification statement(Increment/Decrement)
  - Body of loop
- The variable that is used for terminating condition is referred as “loop/control variable“.

Entry Controlled Loops :

- Checks the condition at an entry level(at the beginning).
- Execution Flow :

1) The control Variable is initialized first.
2) The variable is evaluated with the condition.
3) If the condition is false, the loop is terminated, else
4) If true, the loop body is executed and the variable is updated(incremented/decremented).
5) Steps 2 and 4 are repeated until condition is false.

- Types of Entry controlled Loops:

1) While Loop
2) For Loop

While Loop :

Syntax :

- Control variable initialization
- While (condition) // (control statement)

```
{
        // Loop Body
        control variable increment/decrement
  }
```

- Execution control enters the block ONLY if CONDITION IS TRUE.

For Loop :

Syntax :

- For(variable initialization; condition; modification)

{

loop Body;

}

- Execution control enters the block ONLY if CONDITION IS TRUE.

Exit Controlled Loop :

- Checks the condition at an exit level(at the end).
- Loop body executes at least once irrespective of the condition being true or false.
- Execution Flow :

1) The control variable is initialized first.
2) The loop body is executed. The variable gets updated(incremented/decremented).
3) The variable is evaluated with the condition.
4) If condition is false loop gets terminated else
5) Steps 2 and 3 are repeated till the condition is false.

- Type of Exit Controlled Loop

1)Do – while Loop

Do – While Loop

Syntax :

- Variable initialization

```
Do
{
        loop Body;
        variable modification
} while (condition);
```

- Loop executes at least once irrespective of the condition.

Example :

while

```
int num = 5;
  while (num <= 3)
  {
      printf("Inside Loop ");
  }
```

In this case, the control DOES NOT enter the loop as the condition is false at the entry level

Do-while

```
int num = 5;
do
   {
       printf("inside Loop");
   }while(num<=3);
```

In this case, the control goes inside the loop at least once as the condition is checked at the exit.

Infinite loop :

- If loop condition is always true, program never terminates.
- while(1) {
  …
  }

- for(; ;) {
  …
  }

- do {
  …
  } while(1);

Jump statements : break , continue :

- Break :
  - Used to early exit from loop, or to exit an infinite loop
  - Takes control out of current loop and continues execution of statements after the loop.
  - Statements after break are skipped.
  - break is used with loop/switch case.
  - In case of nested loops, break affects current loop only (not outer).
- Continue :
  - Used to continue next iteration of the loop.
  - Statements after continue are skipped (for current iteration).
  - Continue is used only with loops.
  - In case of nested loops, continue affects current loop only (not outer).

Jump statements : return , goto :

- Return :
  - Can be used inside function.
  - Helps to move execution control forcefully back to calling function.
- goto :
  - Jumps to statement label, must be within same function as the goto.
  - Statement label is an identifier followed by a colon (:)
  - Unstructured control statement
  - Used rarely (less readable)
  - Advised to use only for forward jump
  - Best use is to exit from deeply nested loops.
  - Syntax:
    - goto label_name;

      ……

      ……..

      label_name: C-statements

Thank You