Looping

Dr. Nachiket Tapas

Looping

- Executing a pair of statements again and again

- We have seen in the previous chapter that it is possible to execute a segment of a program repeatedly by introducing a counter and later testing it using the if statement.

Sum of series: 1^2 + 2^2 + 3^2 + … + 9^2 + 10^2

```
-----------
-----------
sum = 0;
n = 1;
loop:
sum = sum + n*n;
if (n == 10)
    goto print;
else
{
    n = n+1;
    goto loop;
}
print:
-----------
```

Loop

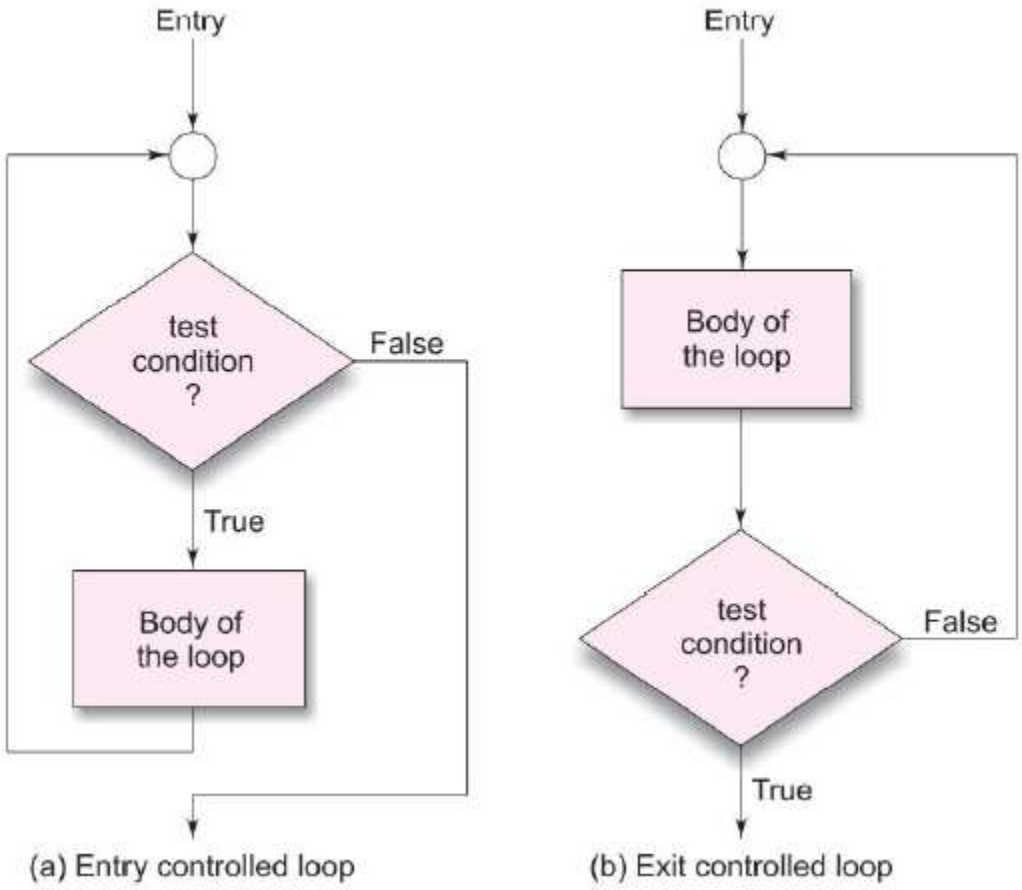
n = 10,
end of loop

Program Explanation

This program does the following things:

1. Initializes the variable n.
2. Computes the square of n and adds it to sum.
3. Tests the value of n to see whether it is equal to 10 or not. If it is equal to 10, then the program prints the results.
4. If n is less than 10, then it is incremented by one and the control goes back to compute the sum again.

Types



(a) Entry controlled loop

(b) Exit controlled loop

Looping process

1. Setting and initialization of a condition variable.
2. Execution of the statements in the loop.
3. Test for a specified value of the condition variable for execution of the loop.
4. Incrementing or updating the condition variable.

Types

- The **while** statement
- The **for** statement
- The **do** statement

Type based on termination condition

- Counter-controlled loops (finite loops)
  - Based on the value of the variable

- Sentinel-controlled loops (infinite loops)
  - Based on some special value like -1 or 999

While Loop/Statement

Also known as entry-control loop.

Syntax

```
while (test condition)
{
        body of the loop
}
statement
```

Sum of series: 1^2 + 2^2 + 3^2 + … + 9^2 + 10^2

```
sum = 0;
n = 1; /* Initialization */
while(n <= 10) /* Testing */
{
        sum = sum + n * n;
        n = n+1; /* Incrementing */
}
printf("sum = %d\n", sum);
```

Do While Loop/Statement

Also known as exit-control loop.

Syntax

```
do
{
        body of the loop
}
while (test-condition);
```

Sum of series: 1^2 + 2^2 + 3^2 + … + 9^2

```
sum = 0;
n = 1; /* Initialization */
do
{
        sum = sum + n * n;
        n = n+1; /* Incrementing */
} while(n < 10) /* Testing */
printf("sum = %d\n", sum);
```

FOR loop

- Entry controlled loop.
- Initialization of the control variable (ex. i = 1)
- The value of the control variable is tested using the test-condition.
- After the execution of the body, the control goes back to execute the increment statement.

Syntax

```
for ( initialization(1); test-condition(2); increment(3) ) {
        body of the loop
}
```

Example

```
#include<stdio.h>
void main() {
        int x;
        for ( x = 0; x <= 9; x = x + 1 ) {
                printf("%d\n", x);
        }
        printf("\n");
}
```

0
1
2
3
…
9

Sum of series: 1^2 + 2^2 + 3^2 + … + 9^2

```
int sum = 0,n;
for ( n = 1; n < 10; n = n + 1)
{
        sum = sum + n * n;
}
printf("sum = %d\n", sum);
```

Comparison

for	*while*	*do*

for (n=1; n<=10; ++n)
{
_____
_____
{
	n = 1;
	while (n<=10)
	{
	_____
	_____
	n = n+1;
	}
		n = 1;
		do
		{
		_____
		_____
		n = n+1;
		}
		while(n<=10);

Question

- Using all three loops, write a C program to print all the prime numbers between 1 and n where ‘n’ is supplied by the user.
- Using all three loops, write a C program to print the Fibonacci sequence.
  - 0 1 1 2 3 5 8 13 21 …

Additional Features of For

- Multiple initializations possible.
- Multiple increment statements possible.
- Compound test statements (condition_1 && condition_2) possible.
- Elimination of one or all elements of for loop possible.

Nesting of Loops and its need

- Nesting of loops is possible.

```
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
```

Jumps in Loops

- Break
- GoTo
- Continue

- for(;;) {
  - for(;;) {
    - if(x>12) break;
    - ....
    - ...

  - }
  - break;

  }
  statement x

Break in Loop

```
while (---------)
{
    ---------
    ---------
    if(condition)
        break;
    ---------
    ---------
}
---------
```

Exit from loop

(a)

```
do
{
    ---------
    ---------
    if(condition)
        break;
    ---------
    ---------
} while (---------);
---------
```

Exit from loop

(b)

```
for (---------)
{
    ---------
    ---------
    if(error)
        break;
    ---------
    ---------
}
---------
```

Exit from loop

(c)

```
for (---------)
{
    ---------
    for (---------)
    {
        ---------
        if(condition)
            break;
        ---------
    }
    ---------
}
```

Exit from inner loop

(d)

GoTo in Loop

```
while (---------)
{
if(error)
goto stop;
---------
if(condition)
goto abc;

  ---------
  ---------
abc:
---------
}
stop:
---------
---------
```

Jump within loop

Exit from loop

(a)

```
for (---------)
{    ---------
     for (---------)
     {
     ---------
     if(error)
     goto error;
     ---------
     }
     ---------
}
error;
---------
---------
```

Exit from two loops

(b)

Continue in Loop

```
while (test-condition)
{
---------
if (---------)
    continue;
---------
---------
}
```

(a)

```
do
{
        ---------
        if (---------)
            continue;
        ---------
        ---------
} while (test-condition);
```

(b)

```
for (initialization; test condition; increment)
{
    ---------
    if (---------)
        continue;
    ---------
    ---------
    }
```

(c)

Jump out of program

- exit(0)
- we can jump out of a program by using the library function exit( ).
- The exit( ) function takes an integer value as its argument.
- Normally zero is used to indicate normal termination and a nonzero value to indicate termination due to some error or abnormal condition.
- The use of exit( ) function requires the inclusion <stdlib.h>.

- void main(){
  - ....
  - exit(0);
  - printf(“Hello World”);
  - }

Usage

```
........
........
if (test-condition) exit(0) ;
........
........
```

Question

- Write a C program to print the following pattern.

```
            0
          1 0 1
        2 1 0 1 2
      3 2 1 0 1 2 3
    4 3 2 1 0 1 2 3 4
  5 4 3 2 1 0 1 2 3 4 5
6 5 4 3 2 1 0 1 2 3 4 5 6
7 6 5 4 3 2 1 0 1 2 3 4 5 6 7
```