URBAN EDGE

Chapter 4 - Loop Control Instruction

Why Loops

Sometimes we want our programs to execute few set of instructions over and over again. for ex: printing 1 to 100, first 100 even numbers etc.

Hence Loops make it easy for a programmer to tell computer that a given set of instructions must be executed repeatedly.

Types of Loops

Primarily, there are three types of loops in C Language:

1. While loop
2. do - while loop
3. for loop

We will look into these one by one

While loop

```
While (condition is true) {

    // Code
    // Code

    }
```

⇒ The block keeps executing as long as the condition is true.

Scanned with CamScanner

An example:

```
int i = 0

While (i < 10) {
    printf (" The value of i is %d", i); i++;
}
```

Note: If the condition never becomes false, the while loop keeps getting executed. Such a loop is known as an infinite loop.

Quick Quiz: Write a program to print natural numbers from 10 to 20 when initial loop counter i is initialized to 0.

The loop counter need not be int, it can be float as well.

Increment and decrement operators

i++ → i is increased by 1
i-- → i is decreased by 1

```
printf ("--i = %d", --i);
```

This first decrements i and then prints it

```
printf (" i-- = %d", i--);
```

This first prints i and then decrements it

Scanned with CamScanner

* +++ operator does not exist ⇒ Important
* += is compound assignment operator just like -=, *=, /= & %= ⇒ Also Important

do-While Loop.
The syntax of do-while loop looks like this:

```
do {
  // Code;
  // Code;
} while (condition)
```

do-while loop works very similar to while loop.
While → checks the condition & then executes the code
do-while → Executes the code & then checks the condition

do-while loop = While loop which executes at least once.

↳ Quick Quiz: Write a program to print first n natural numbers using do-while loop.

Input : 4

Output : 1
2
3
4

Scanned with CamScanner

URBAN EDGE

for Loop

The syntax of for loop looks like this:

```
for( initialize ; test ; increment
                         or decrement)
  {
   // Code;
   // Code;
   // Code;
  }
```

Initialize → Setting a loop Counter to an initial value
Test → Checking a condition
Increment → Updating the loop Counter

An example:

```
for( i=0 ; i<3 ; i++) {
    printf( "%d", i);
    printf( "\n");
  }
```

Output:

```
0
1
2
```

Quick Quiz: Write a program to print first n natural numbers using for loop

Scanned with CamScanner

URBAN EDGE

A Case of Decrementing for loop

```
for (i=5; i; i--)
    printf ("%d\n", i);
```

This for loop will keep on running until i becomes 0.
The loop runs in following steps:

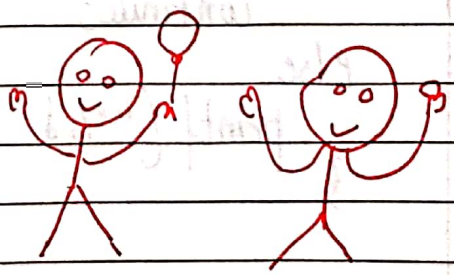
1. i is initialized to 5
2. The condition "i" (0 or none) is tested
3. The code is executed
4. i is decremented
5. Condition i is checked & code is executed if its not 0.
6. & so on until i is non 0

Quick Quiz: Write a program to print n natural numbers in reverse order.

The break Statement in C

The break statement is used to exit the loop irrespective of whether the condition is true or false.
Whenever a "break" is encountered inside the loop, the control is sent outside the loop

Let us see this with the help of an Example



Scanned with CamScanner

```
for (i=0; i<1000; i++) {
  printf ("%d\n", i);
  if (i==5) {
     break;
    }
}
```

output ⇒ 0
1
2
3
4
5
and not 0 to 100 ☺

The Continue statement in C
The Continue statement is used to immideately move to the next iteration of the loop.

The control is taken to the next iteration thus skipping everything below "Continue" inside the loop for that iteration

Let us look at an example

```
int skip = 5;
int i=0;

While (i<10) {
  if (i != skip)
     Continue;
  else
     printf ("%d", i);
  }
```

output ⇒ 5
and not 0 ... 9

Scanned with CamScanner

URBAN EDGE

Notes:

1. Sometimes, the name of the variable might not indicate the behaviour of the program.
2. break statement completely exits the loop.
3. Continue statement skips the particular iteration of the loop.

Scanned with CamScanner

URBAN EDGE

Chapter 4 - Practice Set

1. Write a program to print multiplication table of a given number n.

2. Write a program to print multiplication table of 10 in reversed order

3. A do while loop is executed:
   1. at least once
   2. at least twice
   3. at most once

4. What can be done using one type of loop can also be done using the other two types of loops - True or False?

5. Write a program to sum fist ten natural numbers using while loop.

6. Write a program to implement program 5 using for and do-while loop.

7. Write a program to calculate the sum of the numbers occuring in the multiplication table of 8. (Consider 8×1 to 8×10).

8. Write a program to calculate the factorial of a given number using a for loop.

Scanned with CamScanner

URBAN EDGE

9 Repeat 8 using while loops

10 Write a program to check whether a given number is prime or not using loops.

11 Implement 10 using other types of loops.

Scanned with CamScanner

URBAN EDGE

Project 1: Number guessing Game

Problem: This is going to be fun! We will write a program that generates a random number and asks the player to guess it. If the player's guess is higher than the actual number, the program displays "Lower number please". Similarly if the user's guess is too low, the program prints "Higher number please"
When the user guesses the correct number, the program displays the number of guesses the player used to arrive at the number.

Hint: Use loops
Use a random number generator

Scanned with CamScanner