Chapter 4 - Loop Control Instruction

Why loops?

Sometimes we want our programs to execute a few sets of instructions over and over again, for eg. Printing 1 to 100, first 100 even numbers, etc.

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

Types of Loops: Primarily, there are three types of loop in c language:

1.While loop

2.do-while loop

3.for loop

We will look into this one by one

While Loop

```
While(condition is true) {


// Code                                        // The block keeps
executing as long as the condition is true


// Code


}
```

Copy
An example:

```
int i=0;
while (i<10){
printf(“The value of i is %d”,i); i++;
}
```

Copy
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 the initial loop counter i is initialized to 0.

The loop counter need not be int, it can be a 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--);
```

Copy
This first prints i and then decrements it

- +++ operators does not exists => 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)
```

Copy
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

For Loop

The syntax of for loop looks like this:

```
for( initialize; test; increment or decrement)


{


//code;


//code;


}
```

Copy

Initialize -> setting a lop 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");
}
```

Copy

Output:

0

1

2

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

A case of Decrementing for loop

```
for(i=5; i; i--)


   printf(“%d\n”,i);
```

Copy
This for loop will keep on running until i becomes 0.

The loop runs in the 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, and the code is executed if it's 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 controls are sent outside the loop.

Let us see with an example:

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

Copy
Output: 0, 1, 2, 3, 4, 5 and not 0 to 100

The continue statement in c

The continue statement in c is used to immediately move to the next 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);
}
```

Copy
Output: 5 and not 0...............9

Notes:

1. Sometimes, the name of the variable might not indicate the behavior of the program.

2. Break statement completely exits the loop

3. Continue statement skips the particular iteration of the loop

Chapter 4 - Practice Set

1. Write a program to print the multiplication table of a given number n.
2. Write a program to print a multiplication table of 10 in reversed order
3. A do-while loop is executed:

- At least once
- At least twice
- 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 the first ten natural numbers using a 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 occurring in the multiplication table of 8.(Consider 8x1 to 8x10)

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

9. Repeat 8 using a while loop.

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

11. Implement 10 using other types of loops.

KEEP LEARNING & KEEP PRACTICING😊