Loops in C

Loops in C is used to execute the block of code several times according to the condition given in the loop. It means it executes the same code multiple times so it saves code and also helps to traverse the elements of an array.

There are 3 types of loops in C:-

1.While loop in C

2.do – while loop in C

3.for loop in C

1. While Loop in C

While loop executes the code until the condition is false.

Syntax:

while(condition){

//code

}

while loop

2. do – while loop in C

It also executes the code until condition is false. In this at least once, code is executed whether condition is true or false but this is not the case with while. While loop is executed only when the condition is true.

Syntax:

do{

//code

}while(condition);

Example:

#include <stdio.h>

int main() {

int num;

int sum = 0;

do {

printf("Enter a number (enter 0 if done): ");

scanf("%d", &num);

sum += num;

} while (num != 0);

printf("The sum is: %d\n", sum);

return 0;

}

3. for Loop in C

It also executes the code until condition is false. In this three parameters are given that is

1.Initialization

2.Condition

3.Increment/Decrement

Syntax:

for (initialization; condition; increment/decrement) {

// Code statements to be executed

}

It is used when number of iterations are known where while is used when number of iterations are not known.

Example:

#include<stdio.h>

void main()

{

int i;

for( i = 20; i < 25; i++) {

printf ("%d " , i);

}

}

Output:

Loops in C Control Statements:

These control statements change the execution of the loop from its normal execution. The loop control structures are as follows:

1. Break Statement:

The break statement is used to terminate the execution of a loop or switch statement. When encountered, it immediately exits the loop or switch and transfers control to the statement following the loop or switch. This statement is often used to exit a loop prematurely if a certain condition is met.

#include <stdio.h>

int main() {

int i;

for (i = 1; i <= 10; i++) {

if (i == 5) {

break; // Exit the loop when i equals 5

}

printf("%d ", i);

}

return 0;

}

In this example, the “for” loop iterates from 1 to 10. However, when the value of “i” becomes 5, the “break” statement is encountered, and the loop is terminated prematurely. As a result, only the numbers 1, 2, 3, and 4 will be printed.

1. Continue Statement:

The continue statement is used to skip the remaining statements inside a loop and move to the next iteration of the loop. When encountered, it jumps to the loop’s condition evaluation and proceeds with the next iteration, skipping any code statements that follow the continue statement within the loop.

#include <stdio.h>

int main() {

int i;

for (i = 1; i <= 10; i++) {

if (i == 5) {

continue; // Skip the remaining statements in this iteration

}

printf("%d ", i);

}

return 0;

}

In this example, when the value of “i” is 5, the “continue” statement is encountered. This causes the program to skip the remaining statements within the current iteration of the loop and proceed to the next iteration. As a result, the number 5 is skipped, and the loop continues with the numbers 1, 2, 3, 4, 6, 7, 8, 9, and 10 being printed.

1. Go-to Statement

The go-to statement is used to transfer control to a labeled statement within the same function. It allows the program execution to jump to a different section of code based on a specified label. However, the use of go-to is generally discouraged in modern programming practices due to its potential to create complex and hard-to-maintain code.

#include <stdio.h>

int main() {

int i;

for (i = 1; i <= 10; i++) {

if (i == 5) {

goto skip; // Transfer control to the "skip" label

}

printf("%d ", i);

}

skip:

printf("\n skipped number 5.");

return 0;

}

In this example, when the value of “i” is 5, the “go to” statement is encountered. It transfers the control of the program to the label named “skip”. As a result, the remaining statements within the current iteration of the loop are skipped, and the program directly proceeds to the “skip” label. The label is defined using the syntax label\_name.

After skipping the loop, the program executes the statements following the label. In this case, it prints the message “Skipped number 5.” to indicate that the number 5 was skipped.