Assignment: Understanding the While Loop in C Programming

**Introduction**

Loops are essential constructs in programming, allowing for repeated execution of a block of code until a specified condition is met. In C programming, the while loop is one of the fundamental looping structures. It repeatedly executes a set of statements as long as a given condition remains true, making it useful for situations where the number of iterations is unknown beforehand.

**Syntax of the While Loop**

The general syntax of the while loop in C is as follows:

while (condition) {

// statements to be executed

}

In the syntax above:

* **condition**: An expression that evaluates to either true (non-zero) or false (zero).
* **statements**: The code block that executes repeatedly as long as the condition is true

**Working of the While Loop**

1. **Condition Check**: The loop begins by evaluating the condition.
2. **Execution**: If the condition evaluates to true, the code within the braces {} executes.
3. **Repeat or Exit**: After executing the code block, the loop returns to the condition and re-evaluates it. If the condition is still true, the loop executes again. If it evaluates to false, the loop terminates, and control moves to the next statement outside the loop.

**Example of a While Loop**

1. **Write a C program that takes an integer as input and outputs the sum of its digits**

#include<stdio.h>

int main() {

    //

    int n;

    printf("Enter a number:: ");

    scanf("%d", &n);

    int sum = 0;

    while (n != 0) {

        int ele = n % 10;

        n = n / 10;

        sum = sum + ele;

    }

    printf("sum of digit:: %d\n", sum);

    return 0;

}

**Output:**

**Enter a number:: 1234**

**sum of digit:: 10**

1. **Write a C program that takes an integer as input and outputs the reverse of that integer.**

*#include*<stdio.h>

*#include*<math.h>

*int* *main*() {

*int* *n*;*//123*

*printf*("Enter n:: ");

*scanf*("%d", *&n*);

*int* *new* *=* 0;*//*

*int* *temp* *=* *n*;*//123*

*int* *count=*0;*//size = 300+20+1*

*while* (*temp* *!=* 0) {

*int* *ele* *=* *temp* *%* 10;

*temp* *=* *temp* */* 10;

*// printf("%d\n", temp);*

*count++*;

    }

*while* (*count!=*0) {

*int* *ele* *=* *n* *%* 10;*//1*

*n* *=* *n* */* 10;*//1*

*count--*;

*new* *=* *new* *+* *ele\*pow*(10, *count*);

    }

*printf*("reverse::  %d", *new*);

*return* 0;

}

**Output:**

**Enter n:: 1234**

**reverse:: 4321**

1. **Write a C program that checks if a given integer is a palindromic number.**

*#include*<stdio.h>

*#include*<math.h>

*int* *main*() {

*int* *n*;

*printf*("Enter n:: ");

*scanf*("%d", *&n*);

*int* *new* *=* 0;

*int* *temp* *=* *n*;

*int* *old* *=* *n*;

*int* *count* *=* 0;

*while* (*temp* *!=* 0) {

*int* *ele* *=* *temp* *%* 10;

*temp* *=* *temp* */* 10;

*count++*;

    }

*while* (*count* *!=* 0) {

*int* *ele* *=* *n* *%* 10;

*n* *=* *n* */* 10;

*count--*;

*new* *=* *new* *+* *ele* *\** *pow*(10, *count*);

    }

*if* (*new* *==* *old*) {

*printf*("%d is palindromic number. \n", *new*);

    }

*else* {

*printf*("%d is not a palindromic number. \n", *new*);

    }

*return* 0;

}

**Output:**

**Enter n:: 121**

**121 is a palindromic number.**

**Enter n:: 123**

**123 is not a palindromic number.**

**Conclusion**

The while loop in C programming is a versatile construct, particularly useful when the number of iterations is unknown. However, it requires careful handling to avoid infinite loops. Proper use of conditions and variables within the loop is key to making while loops efficient and reliable in programs.

* **Write a C program to print all prime numbers between 2 and 100**

*#include*<stdio.h>

*int* *main*() {

*int* *to*;

*int* *frm*;

*printf*("From:: ");

*scanf*("%d", *&frm*);

*printf*("to:: ");

*scanf*("%d", *&to*);

*for* (*int* *n* *=* *frm*;*n* *<=* *to*;*n++*) {

*if* (*n* *==* 0 *||* *n* *==* 1) *continue*;

*int* *isPrime* *=* 1;

*for* (*int* *i* *=* 2;*i* *<=* *n/*2;*i++*) {

*if* (*n* *%* *i* *==* 0) {

*isPrime* *=* 0;

            }

        }

*if* (*isPrime* *==* 1) {

*printf*("%d  ", *n*);

        }

    }

*return* 0;

}

**Output:**

**From:: 2**

**to:: 100**

**2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97**