1 Introduction

Loops are vital in programming for repeating code based on a condition. In C, the while loop continues executing as long as the condition is true, checking the condition before each iteration. The do-while loop, however, ensures the code runs at least once before the condition is evaluated. This lab covers the syntax and use cases of both loops, highlighting their practical applications in programming.

2 C Programs and Flowcharts and Algorithms

2.1 Task: Print All Alphabets from a to z

Algorithm:

1. START.
2. INPUT char ch.
3. FOR $ch =' a'$ TO $'z'$ DO
   - DISPLAY "ch ".

   END FOR.
4. DISPLAY
5. END.

Flowchart:

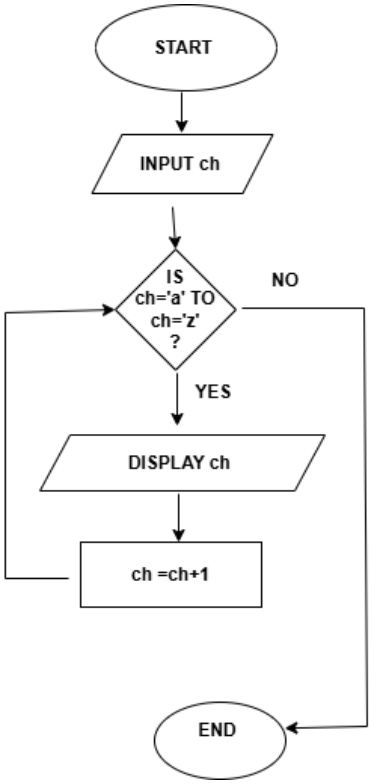


Figure 1: Flowchart for printing all alphabets from a to z

Code:

```c
#include <stdio.h>

int main()
{
    char ch;
    for (ch = 'a'; ch <= 'z'; ch++)
    {
        printf("%c ", ch);
    }
    printf("\n");
    return 0;
}
```

Output:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

2.2 Task: Print all even number 1 to 100

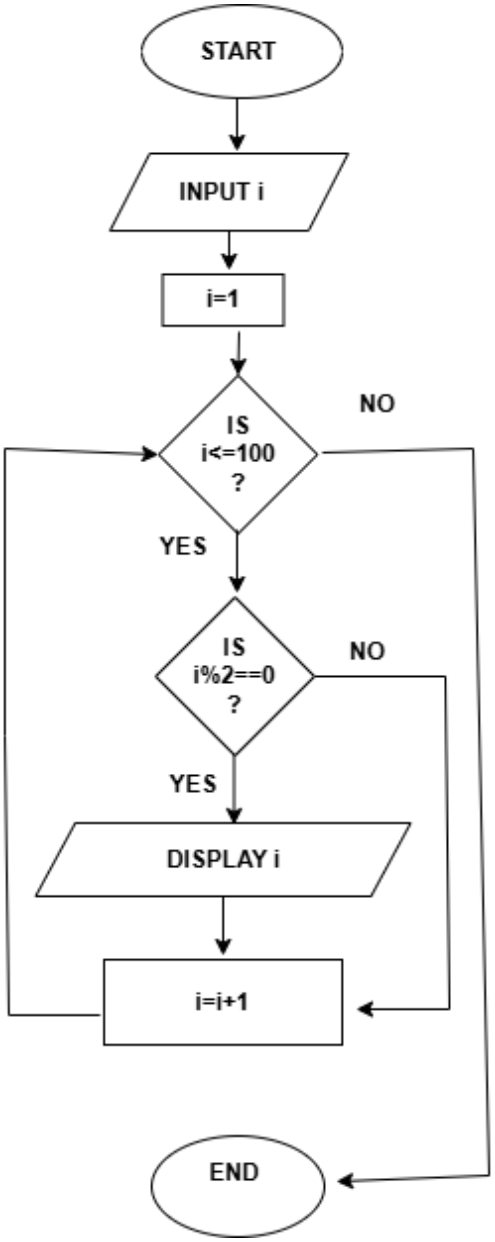
Algorithm:

1. START.
2. INPUT i.
3. INITIALIZE $i = 1$.
4. WHILE $i \leq 100$ DO
   - IF $i\%2 = 0$ THEN
     - DISPLAY "i ".

     END IF.
   - $i = i + 1$.

   END WHILE.
5. END

Flowchart:



Code:

```
#include <stdio.h>

int main()

{
    int i = 1;
    while (i <= 100)
    {
        if (i % 2 == 0)
        {
            printf("%d ", i);
```

```
        }
        i++;
    }
    printf("\n");

    return 0;
}
```

Output:

```
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100
```

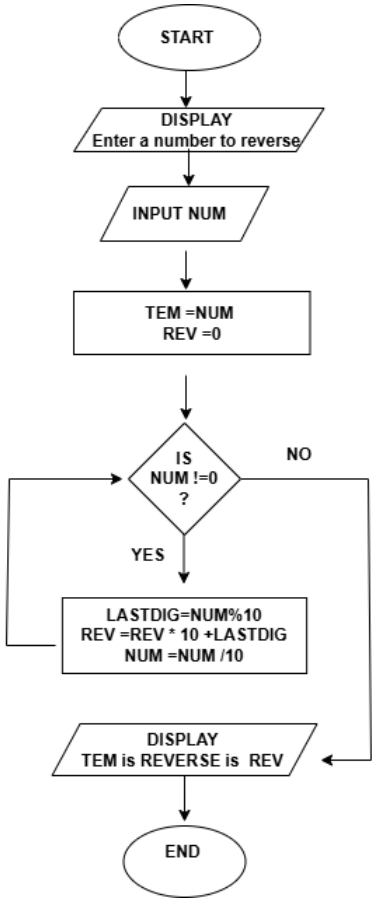
2.3 Task: Enter a Number and Print Its Digits in Reverse Order

Algorithm:

1. START.
2. DISPLAY "Enter a number to reverse: ".
3. INPUT NUM.
4. $TEM = NUM$.
5. $REV = 0$.
6. WHILE $NUM \neq 0$ DO
   - LASTDIG = NUM %10
   - REV = REV $\times 10 +$ LASTDIG$NUM = NUM/10$.

   END WHILE.
7. DISPLAY "$TEMP$ in REVERSE is = REV".
8. END.

Flowchart:



Code:

```c
#include <stdio.h>

int main()
{
    int num;
    printf("Enter a number to reverse: ");
    scanf("%d", &num);

    int temp = num;
    int rev = 0;

    while (num)
    {
        int lastDig = num % 10;
        rev = rev * 10 + lastDig;


        num = num / 10;
    }

```

```
21      printf("%d in REVERSE is = %d\n",temp, rev);
22
23      return 0;
24  }
```

Input: 134

Output:

```
Enter a number to reverse: 134
134 in REVERSE is = 431
```

2.4 Task: Find the Frequency of Each Digit in a Given Integer

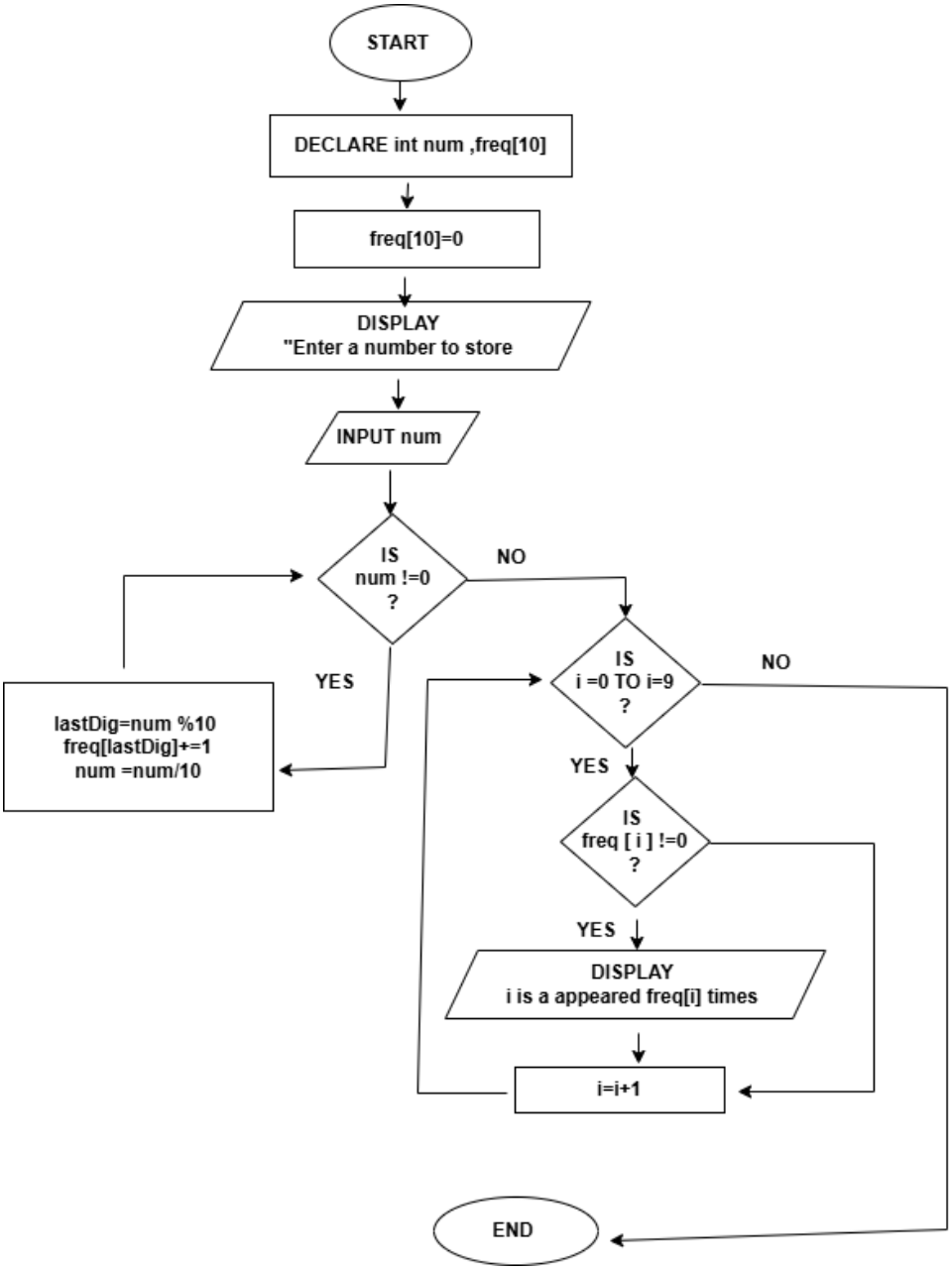
Algorithm:

1. START.
2. DECLARE int $num, freq[10]$.
3. INITIALIZE $freq[10] = 0$.
4. DISPLAY "Enter a number to start: ".
5. INPUT num.
6. WHILE $num \neq 0$ DO
   - SET $lastDig = num\%10$.
   - $freq[lastDig] += 1$.
   - SET $num = num/10$.

   END WHILE.
7. FOR $i = 0$ TO 9 DO
   - IF $freq[i] \neq 0$ THEN
     - DISPLAY "i is appeared $freq[i]$ times".
   - END IF.

   END FOR.
8. END.

Flowchart:



Code:

```c
#include <stdio.h>

int main()

{
    int num, freq[10] = {0};
    printf("Enter a number to start: ");
    scanf("%d", &num);

```

```
    while (num != 0)
    {
        int lastDig = num % 10;
        freq[lastDig]++;
        num /= 10;
    }

    for (int i = 0; i < 10; i++)
    {
        if (freq[i] != 0)
        {
            printf("%d is appeared %d times\n", i, freq[i]);
        }
    }

    return 0;
}
```

Input: 7643

Output:

```
Enter a number to start: 7643
3 is appeared 1 times
4 is appeared 1 times
6 is appeared 1 times
7 is appeared 1 times
```

2.5 Task: Find Sum of First and Last Digit of Any Number

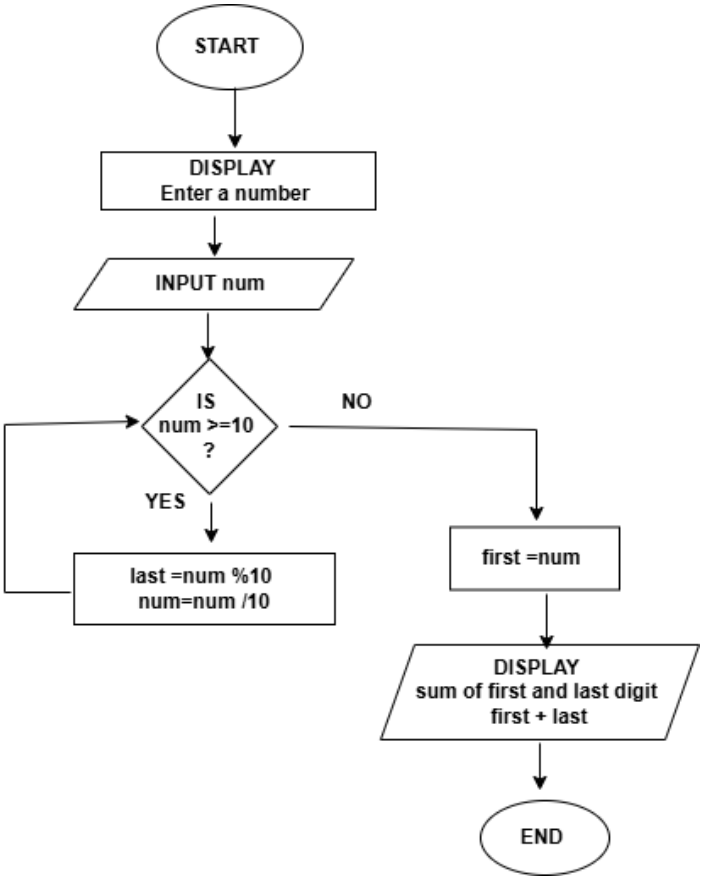
Algorithm:

1. START.
2. DECLARE int $num, first, last$.
3. DISPLAY "Enter a number: ".
4. INPUT num.
5. SET $last = num\%10$.
6. WHILE $num \geq 10$ DO
    - SET $num = num/10$.

END WHILE.

7. SET $first = num$.
8. DISPLAY "Sum of first and last digit: $first + last$".
9. END.

Flowchart:



Code:

```c
#include <stdio.h>

int main()
{
    int num, first, last;

    printf("Enter a number: ");
    scanf("%d", &num);

    last = num % 10;

```

```
    while (num >= 10)
    {
        num /= 10;
    }
    first = num;

    printf("Sum of first and last digit: %d\n", first + last);

    return 0;
}
```

Input: 8493
Output:

```
Enter a number: 8493
Sum of first and last digit: 11
```

3 Discussion

In this report, we have solved several problems related to the do-while loop in C. Loops are fun to work with, but they need to be handled carefully to avoid logical errors. During the implementation of the code, I encountered a few challenges. For instance, when working on the reverse number task, I initially faced an issue where the original number was lost, resulting in a NULL output. Later I realized that I needed to preserve the original value for future use. Another difficulty arose while printing alphabets. I struggled to increment the letter correctly by adding a number. To resolve this, I refer to online resources to understand the correct syntax. Overcoming these challenges improved my understanding of do-while loops and their implementation in C.