

1 Introduction

User-defined functions in C allow programmers to modularize their code by creating functions tailored to specific tasks. These functions enhance code readability, reusability, and maintainability. A user-defined function is defined by the programmer, with a specified return type, function name, and parameters, making it a powerful tool for breaking down complex problems into simpler, manageable pieces. This lab focuses on understanding the structure, syntax, and application of user-defined functions in C.

2 C Programs, Algorithms and Outputs

2.1 Task: Check if a Number is Even or Odd using Function

Algorithm:

1. START
2. DECLARE `num`.
3. INPUT `num`.
4. IF `num % 2 == 0`, THEN
 - DISPLAY "num is an EVEN number".
5. ELSE
 - DISPLAY "num is an ODD number".
6. END.

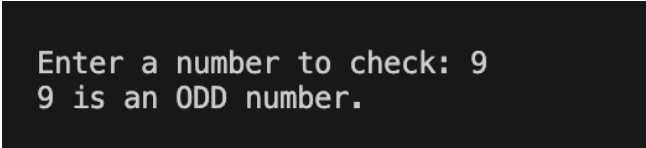
Code:

```
1 #include <stdio.h>
2
3 void evenOrOdd(int num);
4
5 int main()
6 {
7     int num;
8     printf("Enter a number to check: ");
9     scanf("%d", &num);
10    evenOrOdd(num);
11    return 0;
12 }
13
14 void evenOrOdd(int num)
```

```
15 {  
16     if (num % 2 == 0)  
17     {  
18         printf("%d is an EVEN number.\n", num);  
19     }  
20     else  
21     {  
22         printf("%d is an ODD number.\n", num);  
23     }  
24 }
```

Input: 9

Output:



```
Enter a number to check: 9  
9 is an ODD number.
```

2.2 Task: Check Whether a Given Number is a Perfect Number Using Function

Algorithm:

1. START
2. DECLARE num, sum, i.
3. SET sum = 0.
4. INPUT num.
5. FOR i = 1 TO i < num, DO
 - IF num % i == 0, THEN
 - ADD i to sum.
6. END FOR
7. IF sum == num, THEN
 - The number is a perfect number.
8. ELSE
 - The number is not a perfect number.

9. END IF

10. END.

Code:

```
1 #include <stdio.h>
2
3 int isPerfect(int num);
4
5 int main()
6 {
7     int start, end;
8     printf("Enter the STARTING and ENDING point: ");
9     scanf("%d %d", &start, &end);
10
11     printf("Perfect Numbers in given Interval are:\n");
12
13     while (start <= end)
14     {
15         if (isPerfect(start))
16         {
17             printf("%d\n", start);
18         }
19
20         start++;
21     }
22 }
23
24 int isPerfect(int num)
25 {
26     int sum = 0;
27     for (int i = 1; i < num; i++)
28     {
29         if (num % i == 0)
30         {
31             sum += i;
32         }
33     }
34     return sum == num;
35 }
```

Input: 100 1000

Output:

2.3 Task: Generate Fibonacci Series Using Function

Algorithm:

```
Enter the STARTING and ENDING point: 100 1000
Perfect Numbers in given Interval are:
496
```

1. START
2. DECLARE n, i, first, second, next.
3. INPUT n (number of terms).
4. SET first = 0, second = 1.
5. DISPLAY first, second.
6. FOR i = 3 TO n, DO
 - CALCULATE next = first + second.
 - DISPLAY next.
 - UPDATE first = second, second = next.
7. END FOR
8. END.

Code:

```
1 #include <stdio.h>
2
3 void fibonacci(int n)
4 {
5     int first = 0, second = 1, next;
6     printf("%d %d ", first, second);
7
8     for (int i = 3; i <= n; i++)
9     {
10         next = first + second;
11         printf("%d ", next);
12         first = second;
13         second = next;
14     }
15 }
16
17 int main()
18 {
19     int n;
20     printf("Enter the number of terms: ");
21     scanf("%d", &n);
```

```
22     fibonacci(n);  
23     return 0;  
24 }
```

Input: 9

Output:

```
Enter a number to check: 9  
9 is an ODD number.
```

2.4 Task: Find Strong Numbers Between a Given Interval Using Function

Algorithm:

1. START
2. DECLARE start, end, num, sum, temp, lastDig
3. INPUT start, end (interval range)
4. FOR num = start TO end, DO
 - INITIALIZE sum = 0, temp = num
 - WHILE temp != 0, DO
 - INITIALIZE lastDig = temp % 10
 - IF lastDig == 0 OR lastDig == 1, THEN
 - * ADD 1 to sum
 - ELSE
 - * INITIALIZE fact = 1
 - * FOR i = 2 TO lastDig, DO
 - MULTIPLY fact by i
 - * END FOR
 - * ADD fact to sum
 - UPDATE temp = temp / 10
 - END WHILE
 - IF sum == num, THEN
 - PRINT num
5. END FOR

6. END

Code:

```
1  #include <stdio.h>
2
3  int factorial(int num);
4  int sumOfFact(int num);
5
6  int main()
7  {
8      int start, end;
9      printf("Enter the interval: ");
10     scanf("%d %d", &start, &end);
11
12     while (start <= end)
13     {
14         if (sumOfFact(start))
15         {
16             printf("%d\n", start);
17         }
18         start++;
19     }
20 }
21
22 int factorial(int num)
23 {
24     if (num == 0 || num == 1)
25     {
26         return 1;
27     }
28     return num * factorial(num - 1);
29 }
30
31 int sumOfFact(int num)
32 {
33     int temp = num, sum = 0;
34     while (temp != 0)
35     {
36         int lastDig = temp % 10;
37         sum += factorial(lastDig);
38         temp /= 10;
39     }
40     return num == sum;
41 }
```

Input: 1 1000

Output:

```
Enter the interval: 1 1000
1
2
145
```

2.5 Task: Sum of All Digits of a Number Using Recursion

Algorithm:

1. START
2. DECLARE num, sum.
3. INPUT num.
4. SET sum = sumOfDigits(num).
5. DEFINE FUNCTION sumOfDigits(num):
 - IF num == 0, THEN
 - RETURN 0.
 - ELSE
 - RETURN (num % 10) + sumOfDigits(num / 10).
6. END FUNCTION
7. END

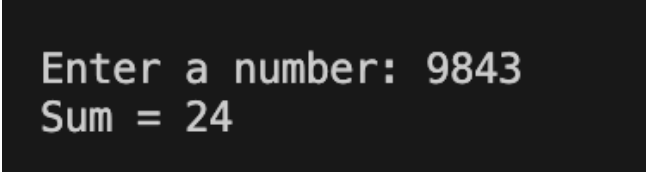
Code:

```
1 #include <stdio.h>
2 int sumOfDigits(int num);
3
4 int main()
5 {
6     int num;
7     printf("Enter a number: ");
8     scanf("%d", &num);
9
10    printf("Sum = %d\n", sumOfDigits(num));
11 }
12
13 int sumOfDigits(int num)
14 {
15     if (num == 0)
16     {
```

```
17     return 0;
18 }
19 return (num % 10) + sumOfDigits(num / 10);
20 }
```

Input: 9843

Output:

A terminal window with a dark background and light gray text. It displays the prompt "Enter a number: 9843" followed by the output "Sum = 24".

```
Enter a number: 9843
Sum = 24
```

3 Discussion

In this lab report, we worked on solving problems using user-defined functions. Writing these functions is similar to coding within the main function, but I initially found it confusing to decide whether to declare the function before the main function and define it afterward or to define it entirely before the main function. Both approaches work the same way, but the first method makes the code more structured and easier to understand. Since my code was relatively short, I chose to define the functions before the main function.

Another challenge I faced was understanding the concepts of strong numbers and perfect numbers. Initially, I was unfamiliar with them, but my instructor provided online resources that explained their definitions. After learning about them, I was able to implement the logic correctly in my code. This lab helped me improve my understanding of function usage and number classification in programming.