Title: Basic Introduction to C programming

Objective:

• To be familiar with basic syntax of c programming
• To learn problem solving techniques using C

History of C

C programming language was developed in 1972 by Dennis Ritchie at Bell Laboratories as a part of the UNIX operating system project. It evolved from earlier programming languages like B and BCPL, incorporating features that made it more efficient for system-level programming. Initially designed to reimplement the UNIX operating system, C quickly gained popularity due to its simplicity, portability, and flexibility. It became the foundation for many modern programming languages such as C++, Java, and Python, cementing its role as one of the most influential programming languages in history.

Header Files

Header files in C contain declarations for functions, macros, constants, and other constructs that can be shared across multiple program files. They typically have a .h extension and are included in source files using the #include directive. For example, #include <stdio.h> allows access to standard input/output functions like printf() and scanf(). Header files help modularize code, making it easier to manage and reuse.

Main Function

The main() function is the entry point of every C program. When a C program is executed, the runtime system starts by calling main(). It can have different forms, such as int main() or int main(int argc, char *argv[]), where argc and argv are used to handle command-line arguments. The main() function usually returns an integer, where return 0; indicates successful program termination.

Variables (Declaration and Initialization)

Variables in C are named memory locations used to store data. Before using a variable, it must be declared, specifying its type (e.g., int, float, char). Initialization is the process of assigning an initial value to a variable at the time of declaration, such as int num = 5;. Proper variable management is essential to ensure efficient memory usage and program logic.

Constants

Constants are fixed values that do not change during program execution. They can be defined using the const keyword (e.g., const int MAX = 100;) or the #define preprocessor directive (#define PI 3.14). Constants improve code readability and maintainability by providing meaningful names for values used repeatedly.

Keywords

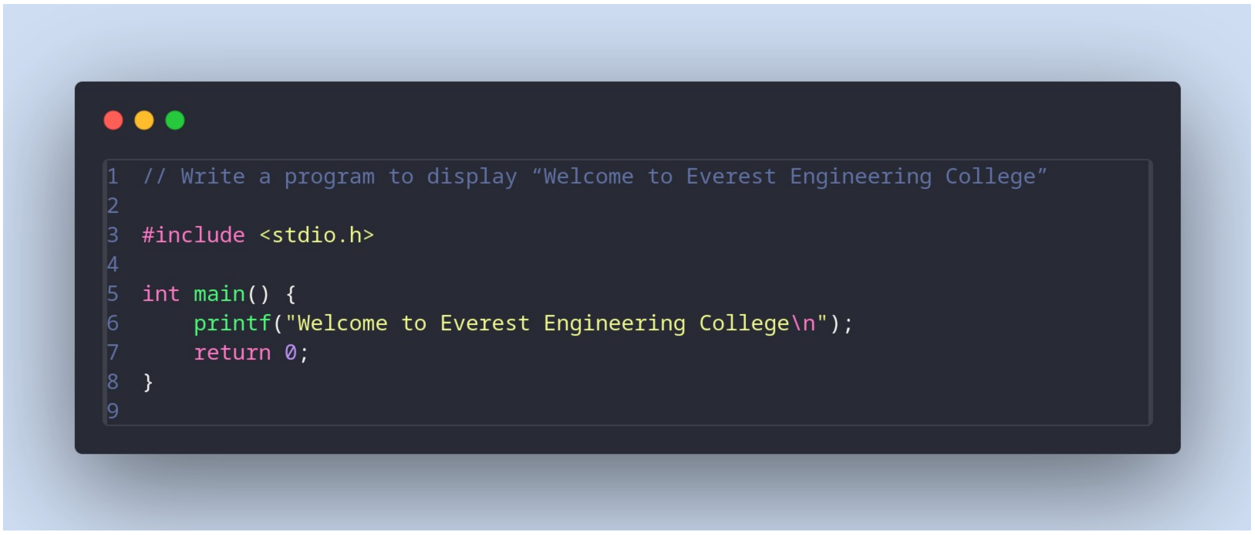
Keywords are reserved words in C that have a specific meaning and cannot be used as identifiers. Examples include int, return, if, else, while, and break. They form the building blocks of C programs, defining the syntax and structure.

Comments

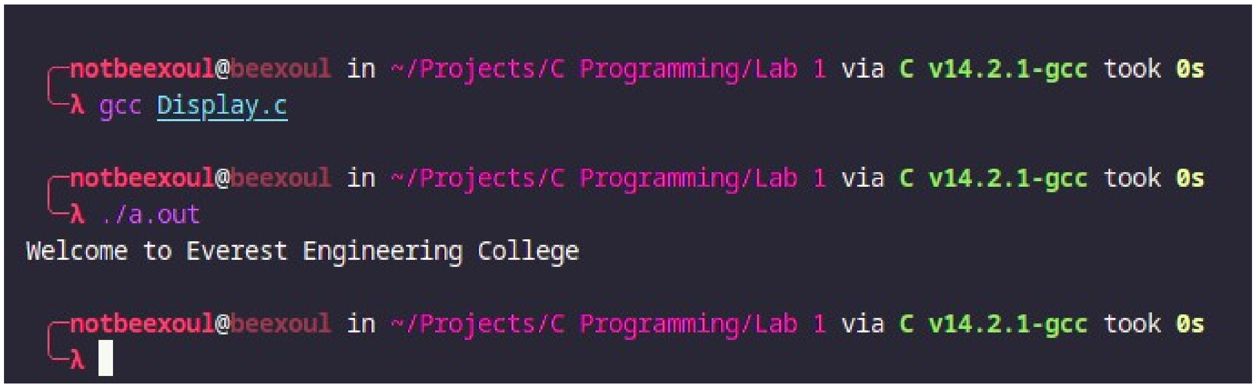
Comments are non-executable statements used to explain code and improve readability. In C, single-line comments begin with //, while multi-line comments are enclosed between /* and */. Comments are especially useful for documenting complex logic, making code easier to understand and maintain.

Some Programs and Output

1) Write a program to display “Welcome to Everest Engineering College”.



```c
// Write a program to display “Welcome to Everest Engineering College”

#include <stdio.h>

int main() {
    printf("Welcome to Everest Engineering College\n");
    return 0;
}

```



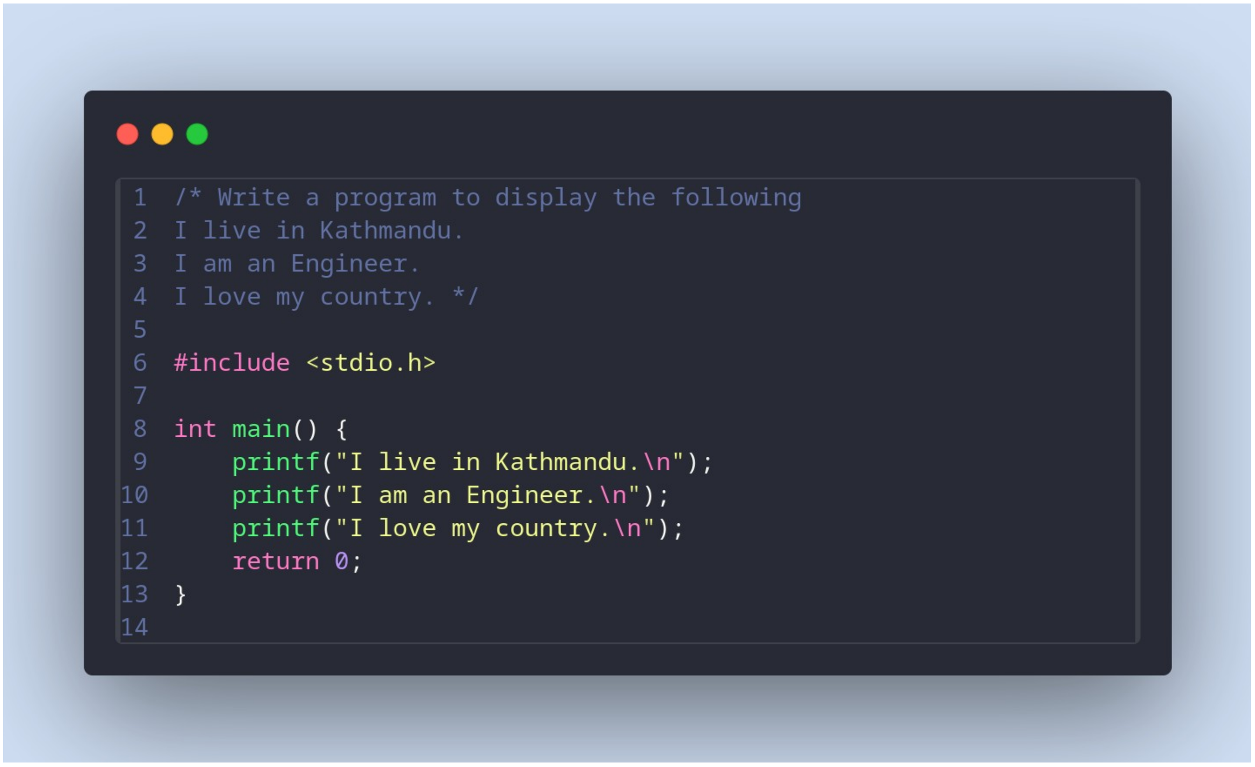
```
╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ gcc Display.c

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ ./a.out
Welcome to Everest Engineering College

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ
```

2) Write a program to display the following .

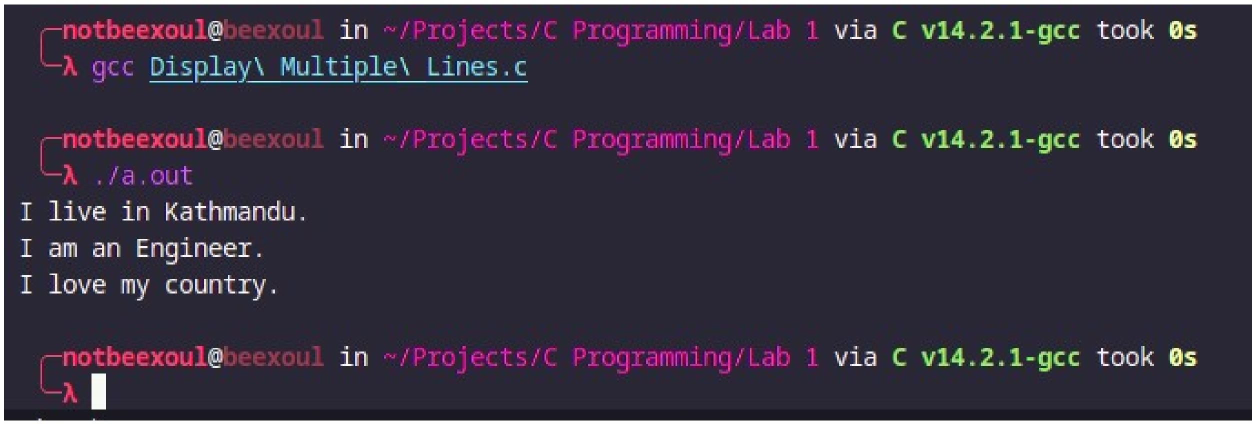
I live in Kathmandu.

I am an Engineer.

I love my country.

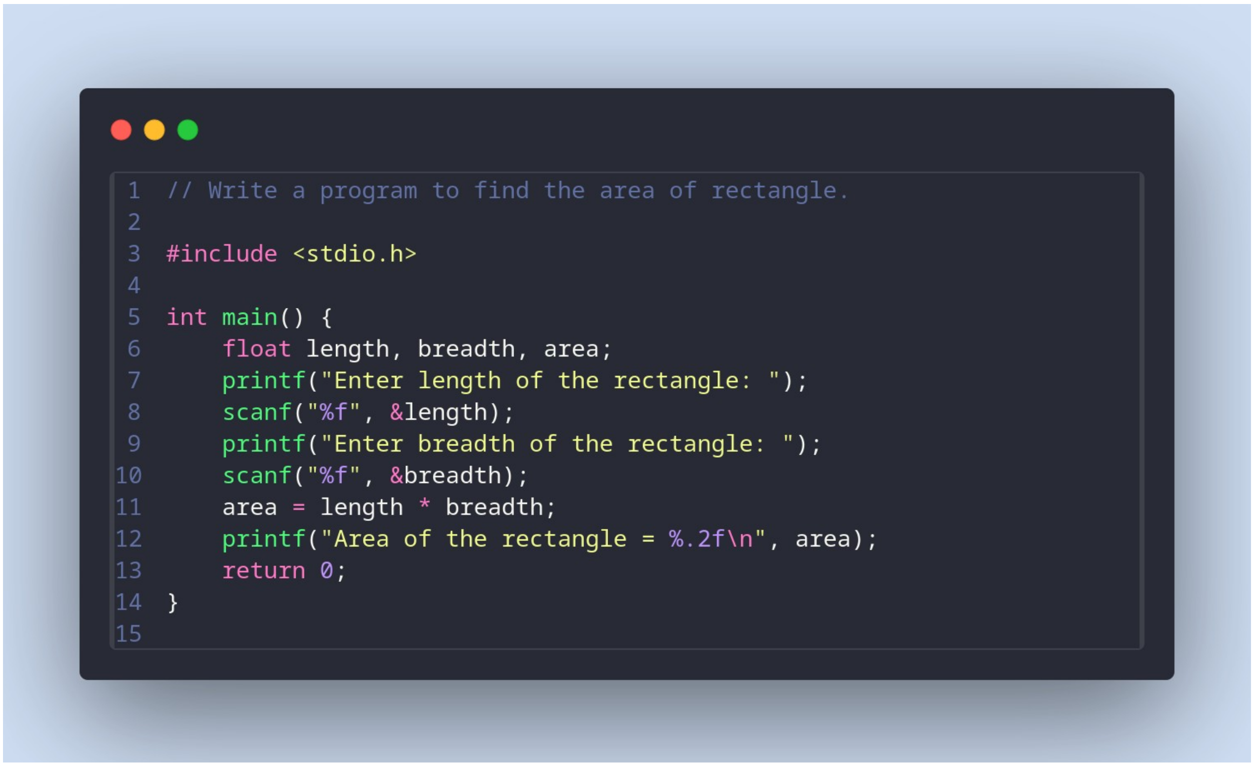


```c
/* Write a program to display the following
I live in Kathmandu.
I am an Engineer.
I love my country. */

#include <stdio.h>

int main() {
    printf("I live in Kathmandu.\n");
    printf("I am an Engineer.\n");
    printf("I love my country.\n");
    return 0;
}

```

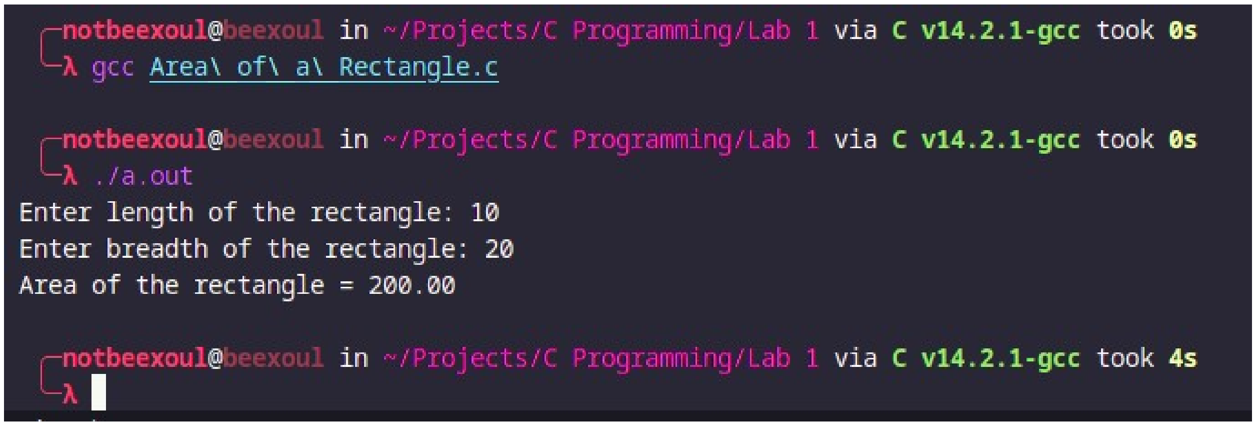


```
╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ gcc Display\ Multiple\ Lines.c

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ ./a.out
I live in Kathmandu.
I am an Engineer.
I love my country.

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ
```

3) Write a program to find the area of rectangle.

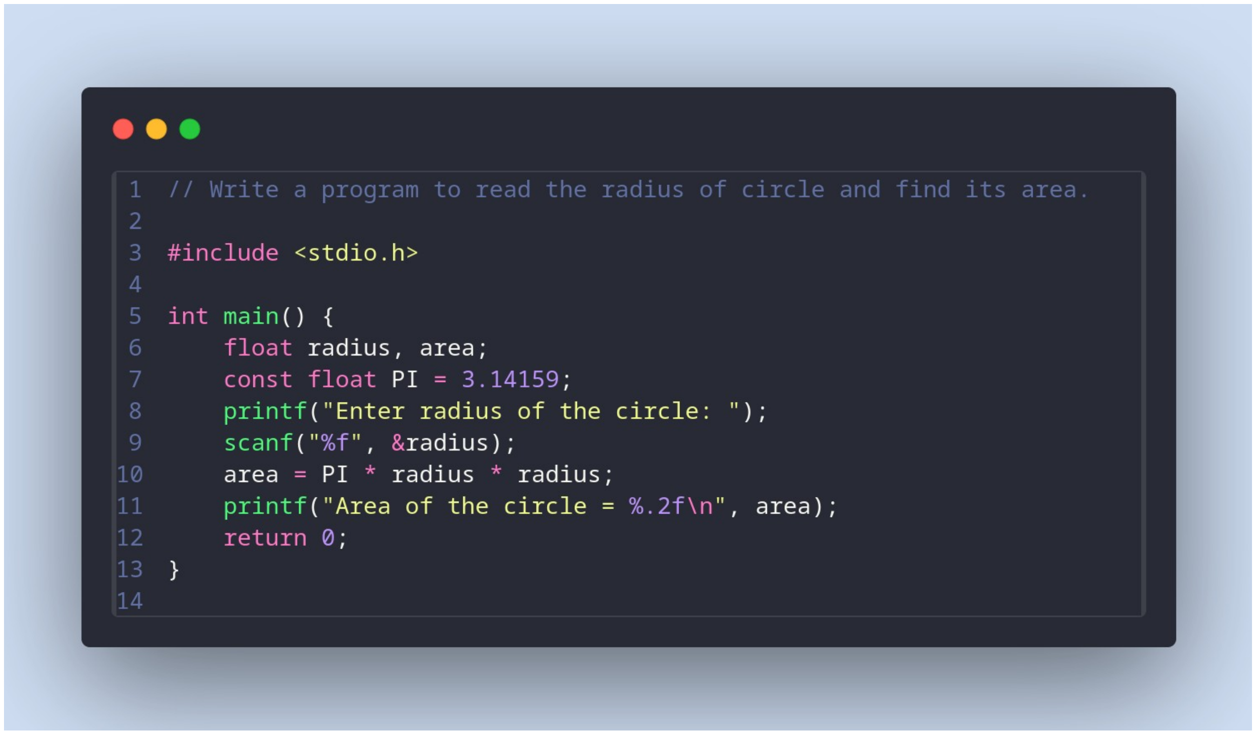


```c
// Write a program to find the area of rectangle.

#include <stdio.h>

int main() {
    float length, breadth, area;
    printf("Enter length of the rectangle: ");
    scanf("%f", &length);
    printf("Enter breadth of the rectangle: ");
    scanf("%f", &breadth);
    area = length * breadth;
    printf("Area of the rectangle = %.2f\n", area);
    return 0;
}

```

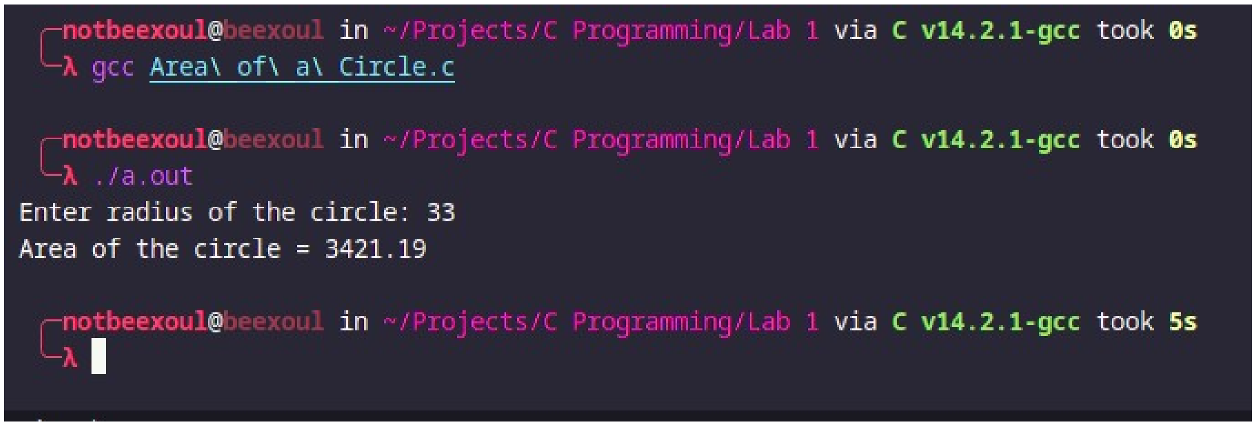


```
╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ gcc Area\ of\ a\ Rectangle.c

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ ./a.out
Enter length of the rectangle: 10
Enter breadth of the rectangle: 20
Area of the rectangle = 200.00

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 4s
╰─λ
```

4) Write a program to read the radius of circle and find its area.

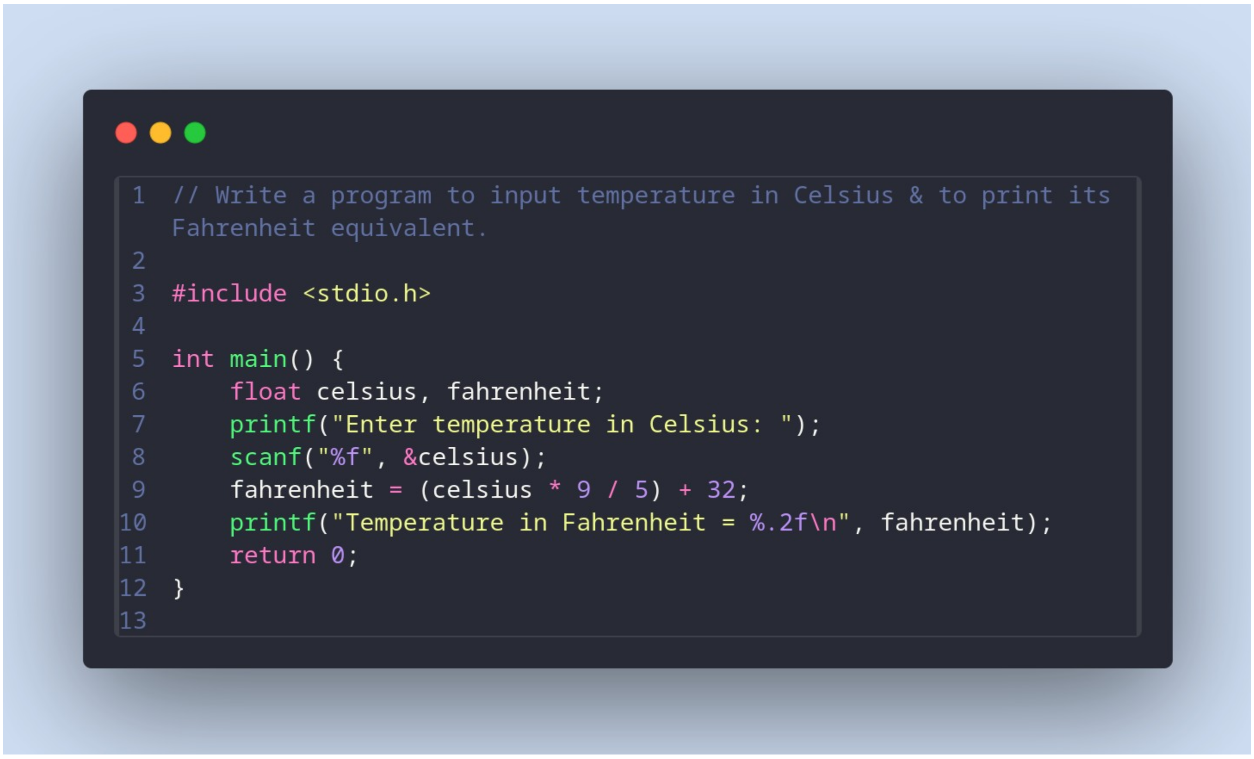


```c
// Write a program to read the radius of circle and find its area.

#include <stdio.h>

int main() {
    float radius, area;
    const float PI = 3.14159;
    printf("Enter radius of the circle: ");
    scanf("%f", &radius);
    area = PI * radius * radius;
    printf("Area of the circle = %.2f\n", area);
    return 0;
}

```

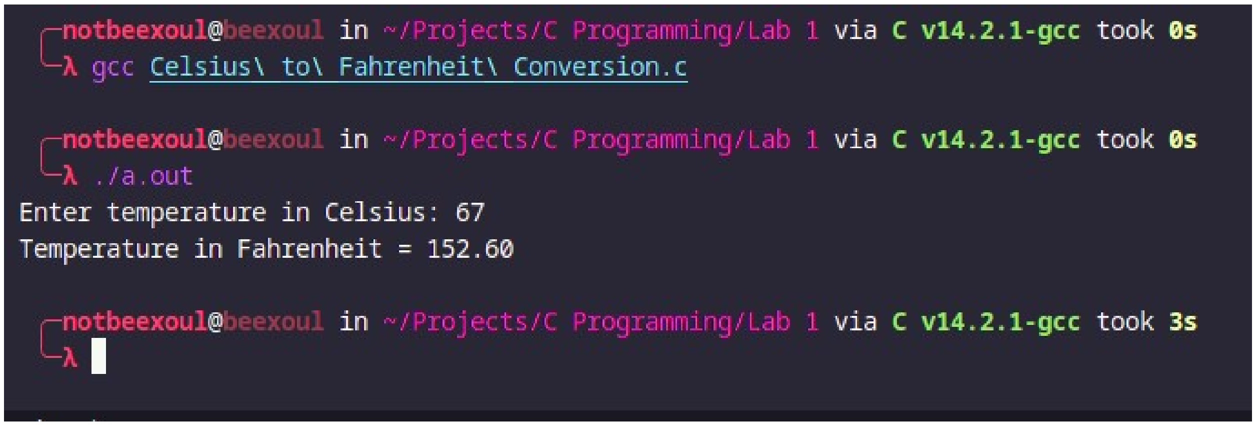


```
┌notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
└λ gcc Area\ of\ a\ Circle.c

┌notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
└λ ./a.out
Enter radius of the circle: 33
Area of the circle = 3421.19

┌notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 5s
└λ
```

5) Write a program to input temperature in Celsius & to print its Fahrenheit equivalent.

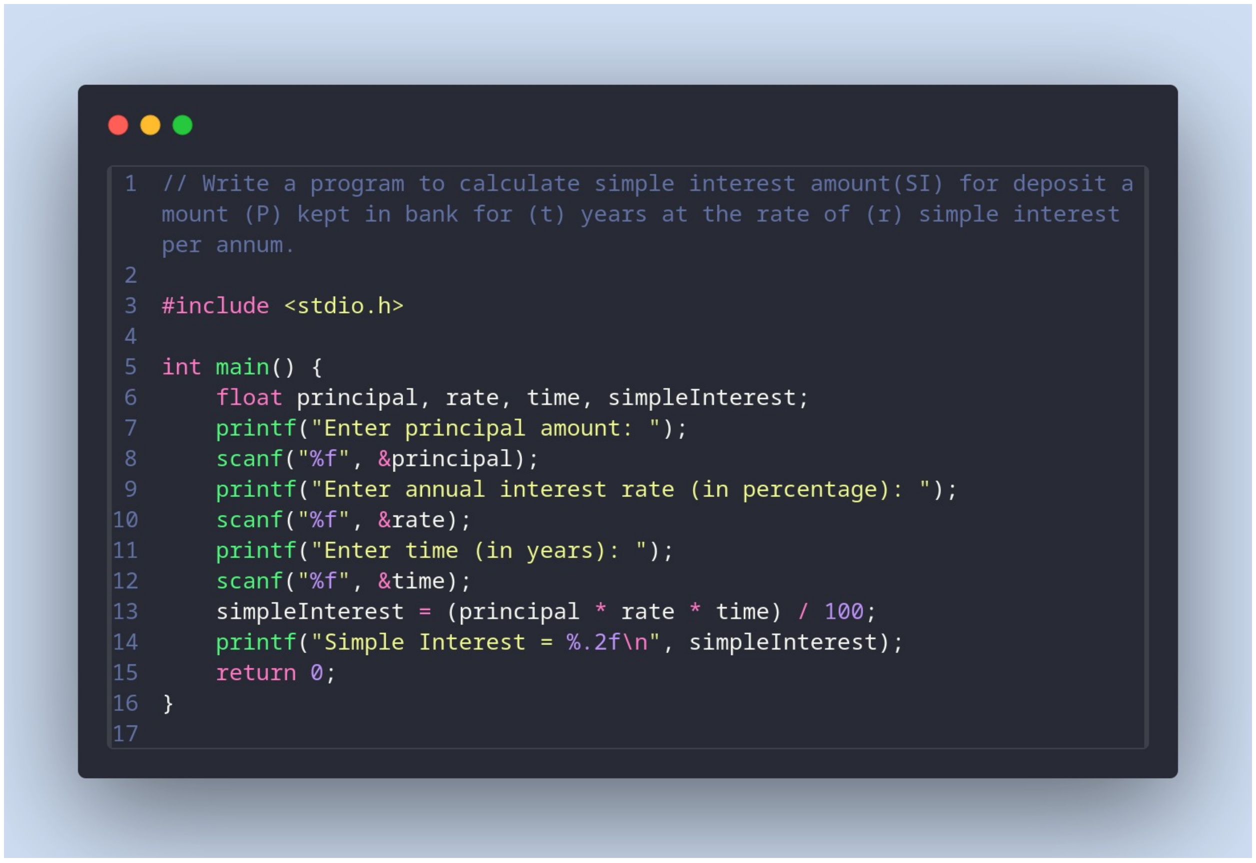


```c
// Write a program to input temperature in Celsius & to print its
Fahrenheit equivalent.

#include <stdio.h>

int main() {
    float celsius, fahrenheit;
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);
    fahrenheit = (celsius * 9 / 5) + 32;
    printf("Temperature in Fahrenheit = %.2f\n", fahrenheit);
    return 0;
}

```

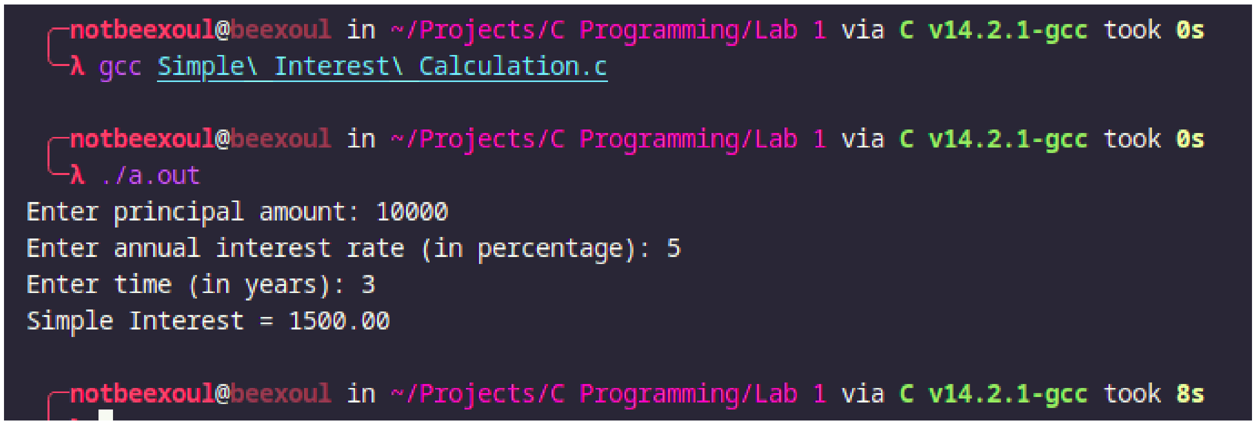


```
╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ gcc Celsius\ to\ Fahrenheit\ Conversion.c

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ ./a.out
Enter temperature in Celsius: 67
Temperature in Fahrenheit = 152.60

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 3s
╰─λ
```

6) Write a program to calculate simple interest amount (SI) for deposit amount (P) kept in bank for (t) years at the rate of (r) simple interest per annum.

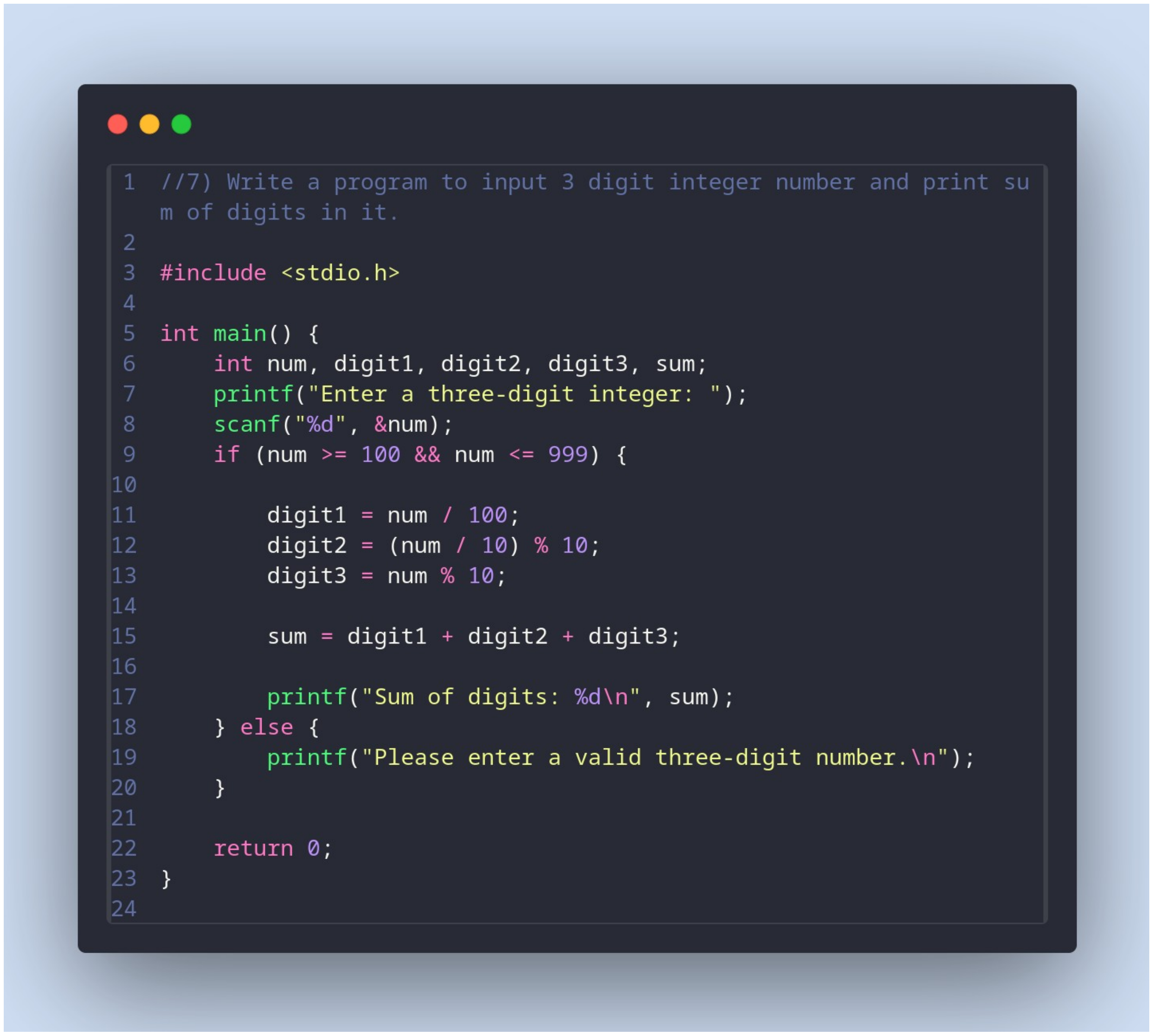


```c
// Write a program to calculate simple interest amount(SI) for deposit a
mount (P) kept in bank for (t) years at the rate of (r) simple interest
per annum.

#include <stdio.h>

int main() {
    float principal, rate, time, simpleInterest;
    printf("Enter principal amount: ");
    scanf("%f", &principal);
    printf("Enter annual interest rate (in percentage): ");
    scanf("%f", &rate);
    printf("Enter time (in years): ");
    scanf("%f", &time);
    simpleInterest = (principal * rate * time) / 100;
    printf("Simple Interest = %.2f\n", simpleInterest);
    return 0;
}
```

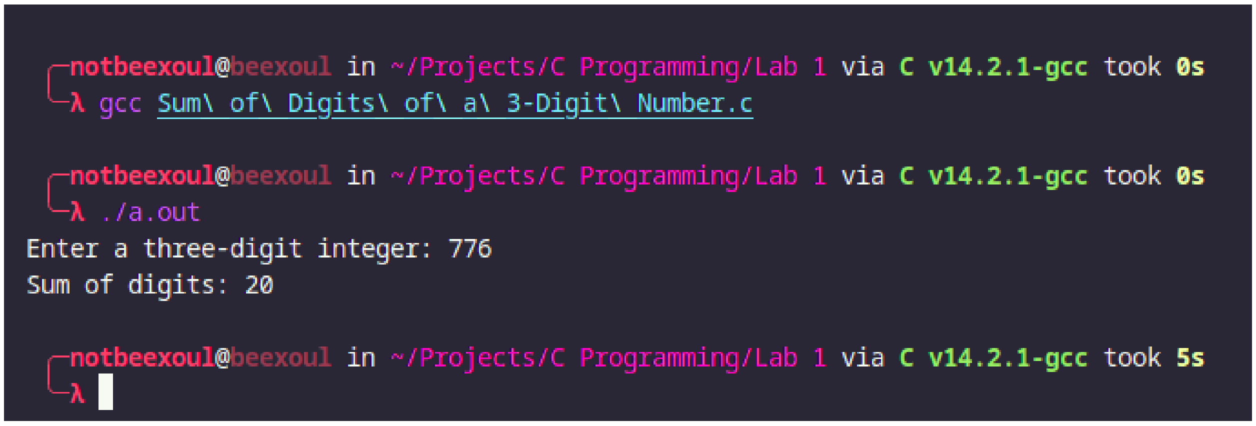


```
╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ gcc Simple\ Interest\ Calculation.c

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ ./a.out
Enter principal amount: 10000
Enter annual interest rate (in percentage): 5
Enter time (in years): 3
Simple Interest = 1500.00

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 8s
```

7) Write a program to input 3 digits integer numbers and print sum of digits in it.

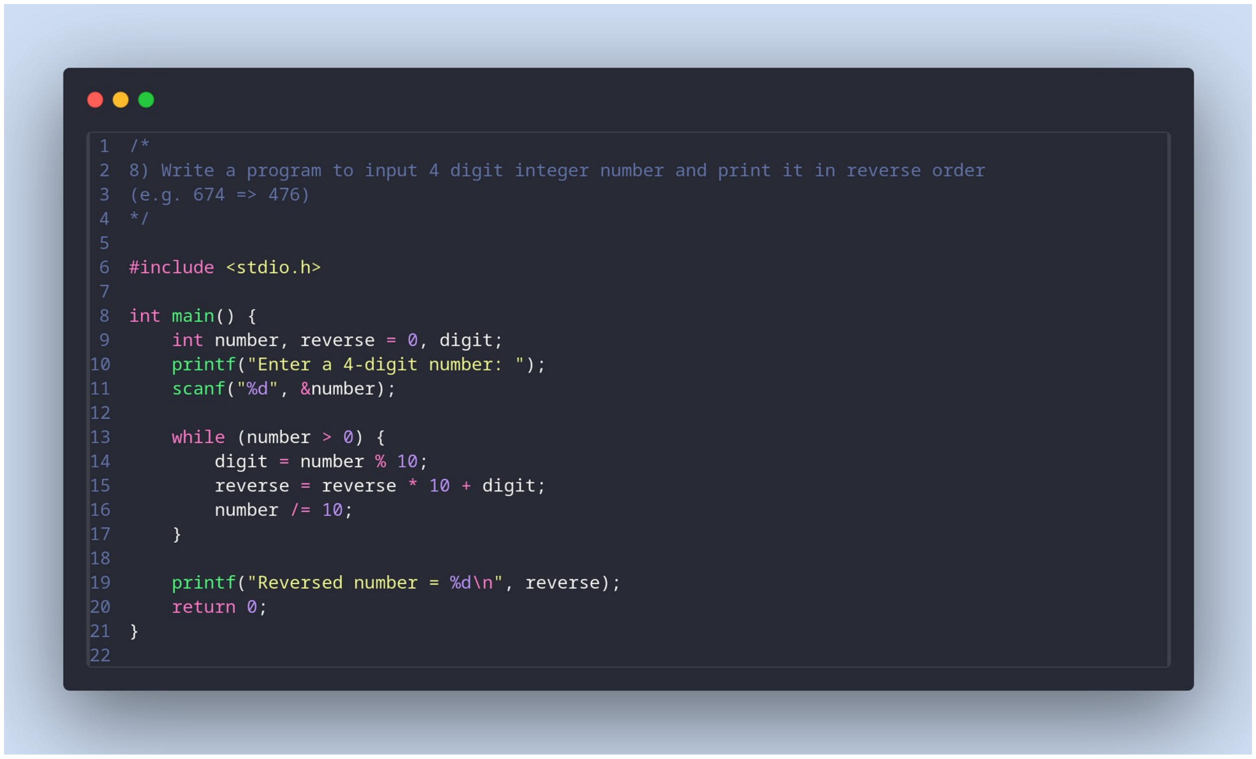


```c
//7) Write a program to input 3 digit integer number and print sum of digits in it.

#include <stdio.h>

int main() {
    int num, digit1, digit2, digit3, sum;
    printf("Enter a three-digit integer: ");
    scanf("%d", &num);
    if (num >= 100 && num <= 999) {

        digit1 = num / 100;
        digit2 = (num / 10) % 10;
        digit3 = num % 10;

        sum = digit1 + digit2 + digit3;

        printf("Sum of digits: %d\n", sum);
    } else {
        printf("Please enter a valid three-digit number.\n");
    }

    return 0;
}

```

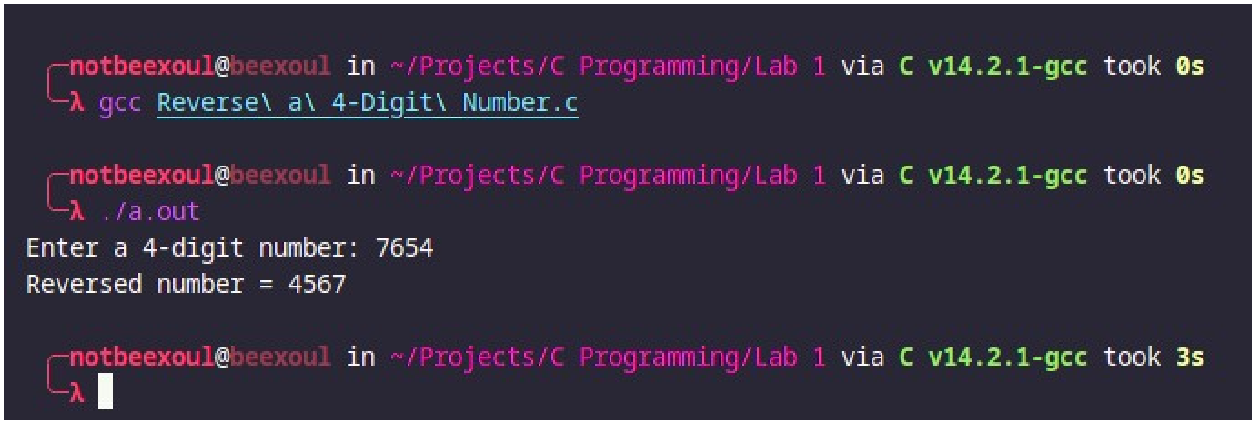


```
╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ gcc Sum\ of\ Digits\ of\ a\ 3-Digit\ Number.c

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
╰─λ ./a.out
Enter a three-digit integer: 776
Sum of digits: 20

╭─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 5s
╰─λ
```

8) Write a program to input 4 digits integer number and print it in reverse order (e.g. 674 => 476)

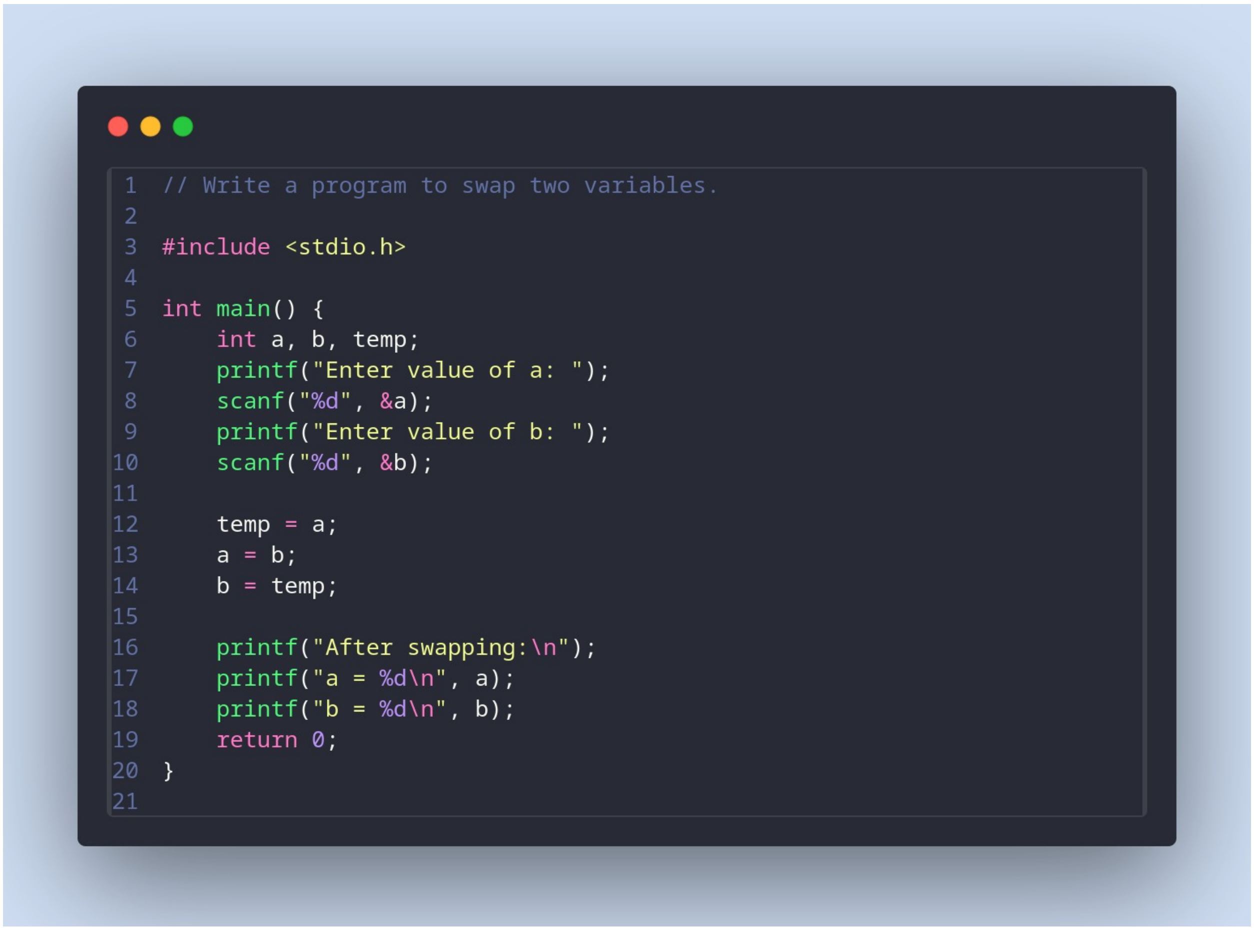


```c
/*
8) Write a program to input 4 digit integer number and print it in reverse order
(e.g. 674 => 476)
*/

#include <stdio.h>

int main() {
    int number, reverse = 0, digit;
    printf("Enter a 4-digit number: ");
    scanf("%d", &number);

    while (number > 0) {
        digit = number % 10;
        reverse = reverse * 10 + digit;
        number /= 10;
    }

    printf("Reversed number = %d\n", reverse);
    return 0;
}
```

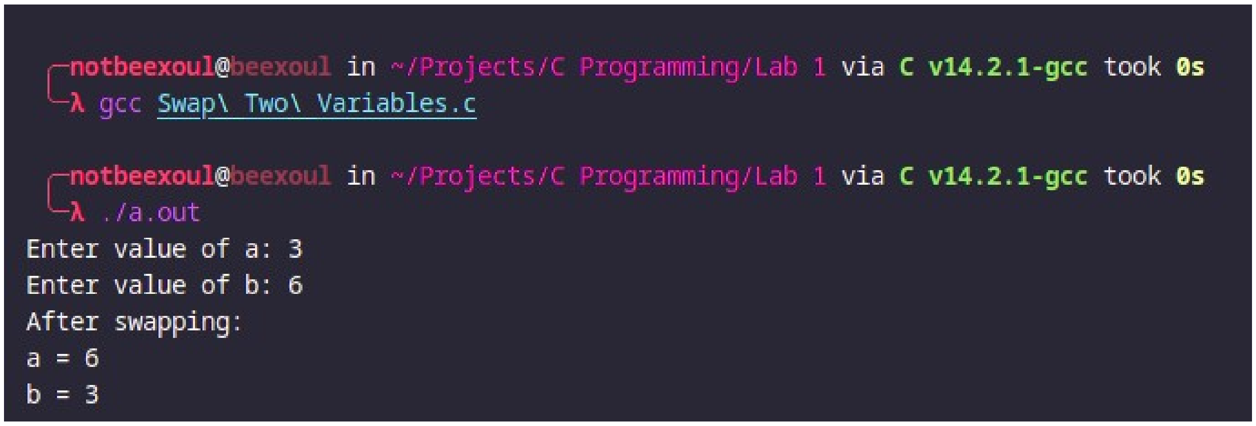


```
┌notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
└λ gcc Reverse\ a\ 4-Digit\ Number.c

┌notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
└λ ./a.out
Enter a 4-digit number: 7654
Reversed number = 4567

┌notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 3s
└λ
```

9) Write a program to swap two variables.



```c
// Write a program to swap two variables.

#include <stdio.h>

int main() {
    int a, b, temp;
    printf("Enter value of a: ");
    scanf("%d", &a);
    printf("Enter value of b: ");
    scanf("%d", &b);

    temp = a;
    a = b;
    b = temp;

    printf("After swapping:\n");
    printf("a = %d\n", a);
    printf("b = %d\n", b);
    return 0;
}

```



```
┌─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
└─λ gcc Swap\ Two\ Variables.c

┌─notbeexoul@beexoul in ~/Projects/C Programming/Lab 1 via C v14.2.1-gcc took 0s
└─λ ./a.out
Enter value of a: 3
Enter value of b: 6
After swapping:
a = 6
b = 3
```

Conclusion

The lab exercises provided a foundational understanding of C programming by introducing key concepts and hands-on problem-solving techniques. Through these exercises, we became familiar with the basic syntax of C, including the use of variables, constants, header files, and the main function. Additionally, we learned how to structure programs effectively, use mathematical formulas, and implement logic for various computations such as area calculation, temperature conversion, and reversing numbers. This practical experience not only enhanced our programming skills but also demonstrated the importance of C as a versatile language for solving real-world problems systematically.