Experiment No-1

Experiment name: Write a C Program to check whether a number is even or odd using user defined function that demonstrate four aspect of function call.

Objective:

To understand and implement different types of function calls in C (no argument-no return, argument-no return, no argument-return, argument-return) through a program that checks whether a number is even or odd.

- Function with no argument and no return.
- Function with argument and no return.
- Function with no argument but with return.
- Function with argument and return.

Problem Analysis:

The purpose of this program is to determine whether a given integer is even or odd using different types of user-defined function calls in C. The program helps illustrate how data is passed to and returned from functions.

Input variable	Processing variable	output variable	Header file
num (integer)	num % 2	printf()	<stdio.h>

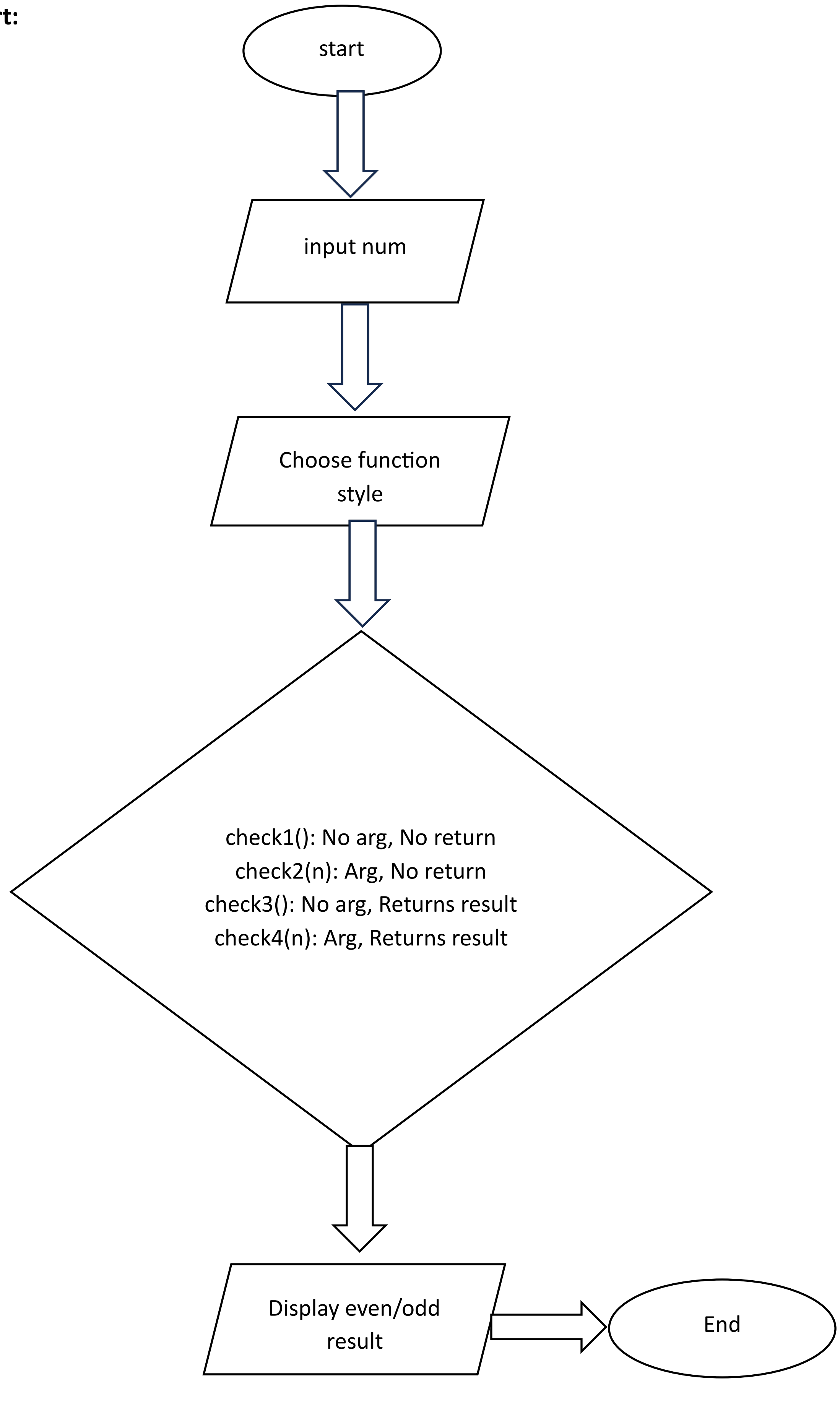
Algorithm:

Step1: start
Step2: Declare necessary variables.
Step3: Demonstrate 4 types of function calls:

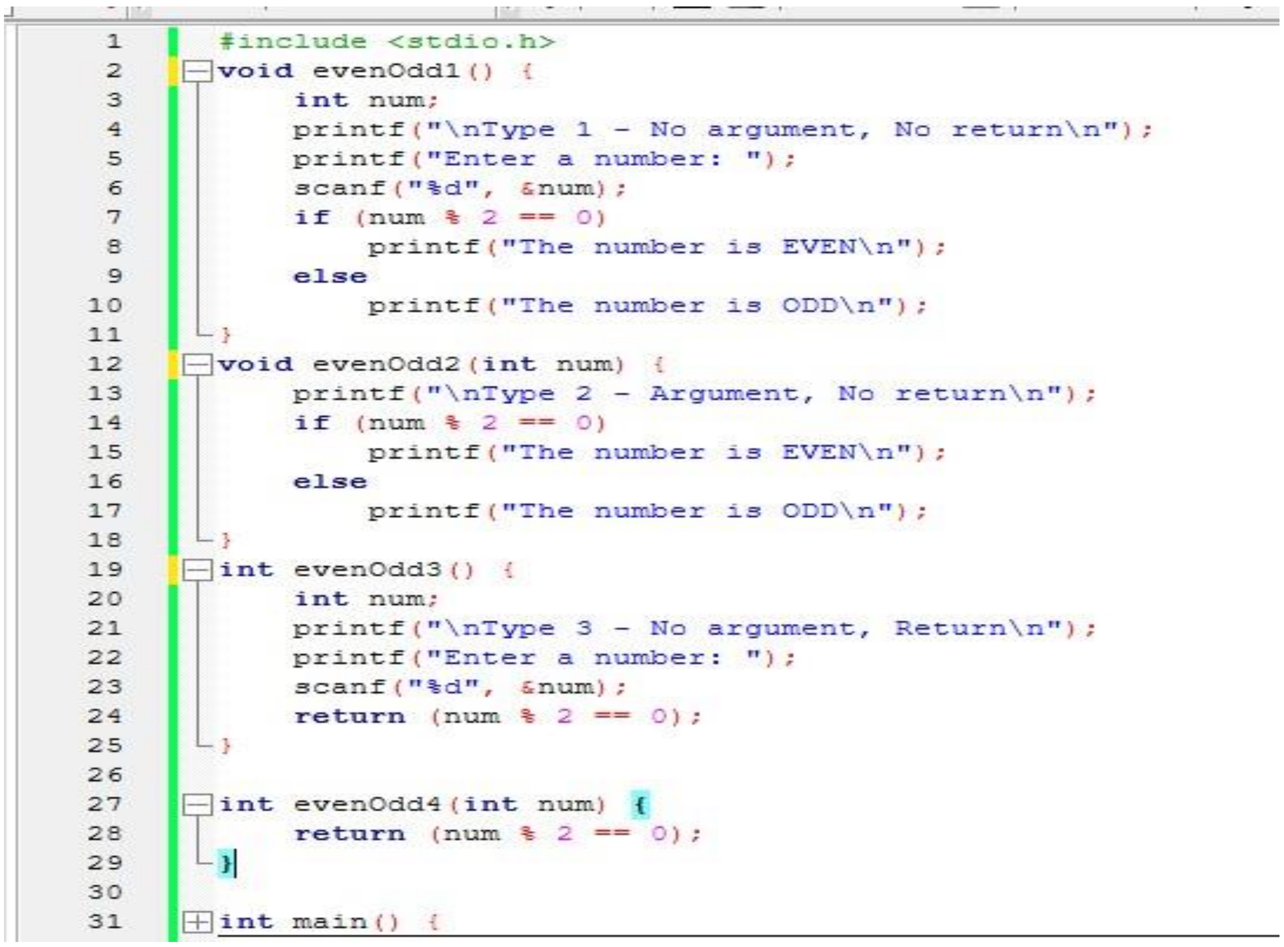
- No arguments, no return.
- Arguments, no return.
- No arguments, return.
- Arguments with return.

Step4: Each function checks whether a number is even or odd.
Step5: Print result for each method
Step6: End

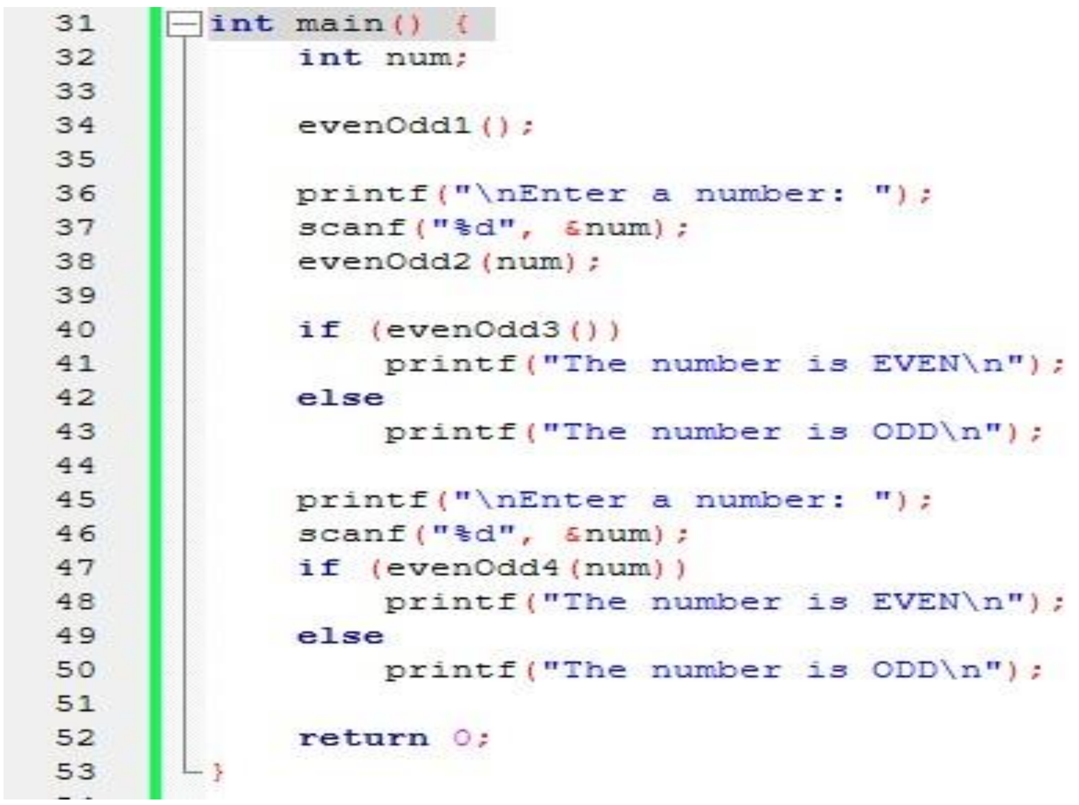
Flowchart:


start
input num
Choose function style
check1(): No arg, No return
check2(n): Arg, No return
check3(): No arg, Returns result
check4(n): Arg, Returns result
Display even/odd result
End

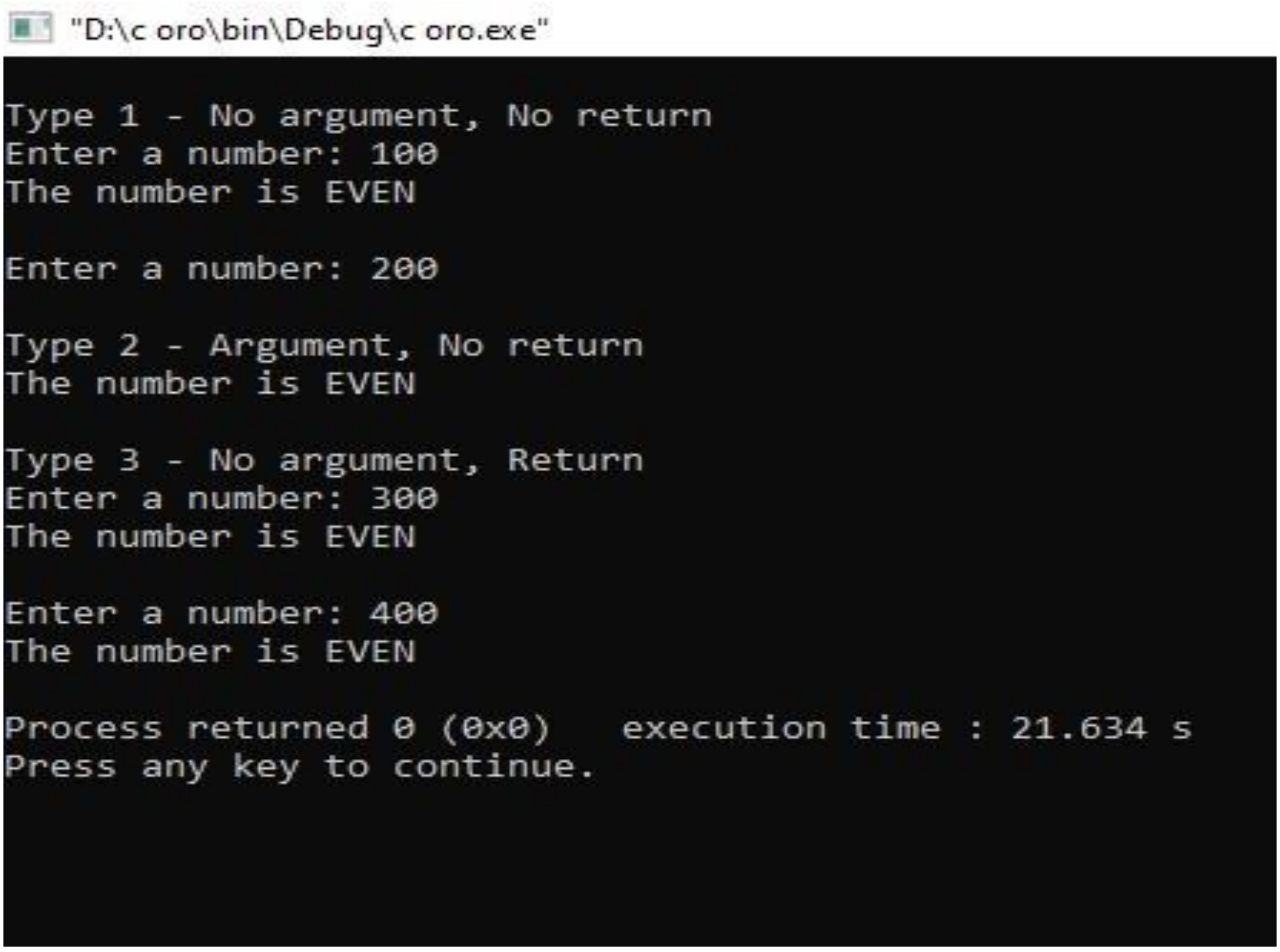
Source code:



```c
#include <stdio.h>
void evenOdd1() {
    int num;
    printf("\nType 1 - No argument, No return\n");
    printf("Enter a number: ");
    scanf("%d", &num);
    if (num % 2 == 0)
        printf("The number is EVEN\n");
    else
        printf("The number is ODD\n");
}
void evenOdd2(int num) {
    printf("\nType 2 - Argument, No return\n");
    if (num % 2 == 0)
        printf("The number is EVEN\n");
    else
        printf("The number is ODD\n");
}
int evenOdd3() {
    int num;
    printf("\nType 3 - No argument, Return\n");
    printf("Enter a number: ");
    scanf("%d", &num);
    return (num % 2 == 0);
}

int evenOdd4(int num) {
    return (num % 2 == 0);
}

int main() {
```



```c
int main() {
    int num;

    evenOdd1();

    printf("\nEnter a number: ");
    scanf("%d", &num);
    evenOdd2(num);

    if (evenOdd3())
        printf("The number is EVEN\n");
    else
        printf("The number is ODD\n");

    printf("\nEnter a number: ");
    scanf("%d", &num);
    if (evenOdd4(num))
        printf("The number is EVEN\n");
    else
        printf("The number is ODD\n");

    return 0;
}
```



```
"D:\c oro\bin\Debug\c oro.exe"

Type 1 - No argument, No return
Enter a number: 100
The number is EVEN

Enter a number: 200

Type 2 - Argument, No return
The number is EVEN

Type 3 - No argument, Return
Enter a number: 300
The number is EVEN

Enter a number: 400
The number is EVEN

Process returned 0 (0x0)   execution time : 21.634 s
Press any key to continue.
```

Discussion:

This program helps us understand how user-defined functions work in C.It checks if a number is even or odd using four different styles of calling a function. Each one handles input/output a bit differently, which shows how flexible and powerful functions are in C programming.

In short:

- Functions can take input from inside or from main.
- They can return a value or just print it directly.
- This is a clean and useful way to organize code.

Experiment No-2

Experiment name: Write a C program using function (With Arguments and with Return) to calculate and display the total amount given that
Total amount= p * (1 +r)n

Where p = principle amount, r= rate of interest, n = period.

Objective: To write a C program using a function with arguments and return value that calculates the total amount based on the formula:

Total Amount = $P \times (1 + r)^n$

Where:

- P = Principal amount
- r = Rate of interest
- n = Period (years)

Problem Analysis:

To calculate the total amount after a certain period with interest, we use this formula:

Total = $P \times (1 + r)^n$

We need to Take 3 inputs: principal (P), rate (r), and time (n) .Use a function with arguments and return value . Return the total amount to main() and display it.

Algorithm:

Step1: Start the program

Step2: Take input: principal, rate, and period from the user

Step3: Call a function with these values as arguments

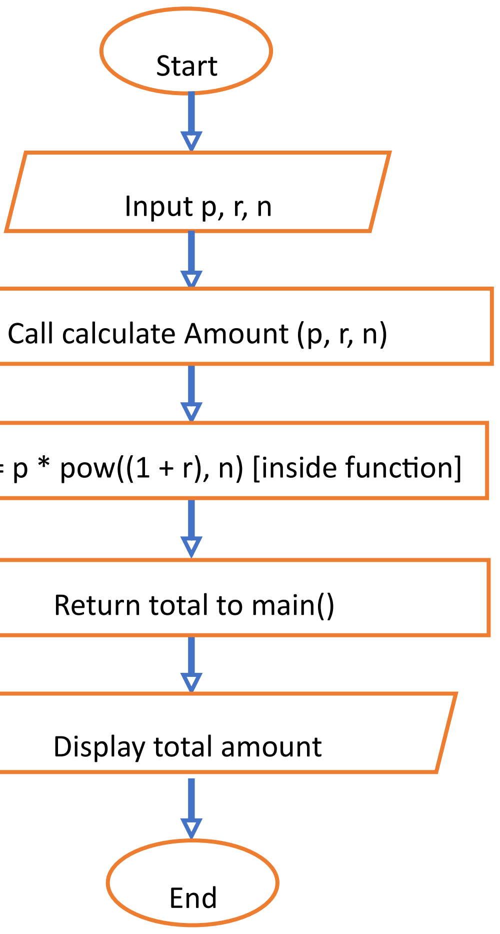
Step4: In the function, calculate total amount using the formula

Step5: Return the total amount

Step6: Display the result in main()

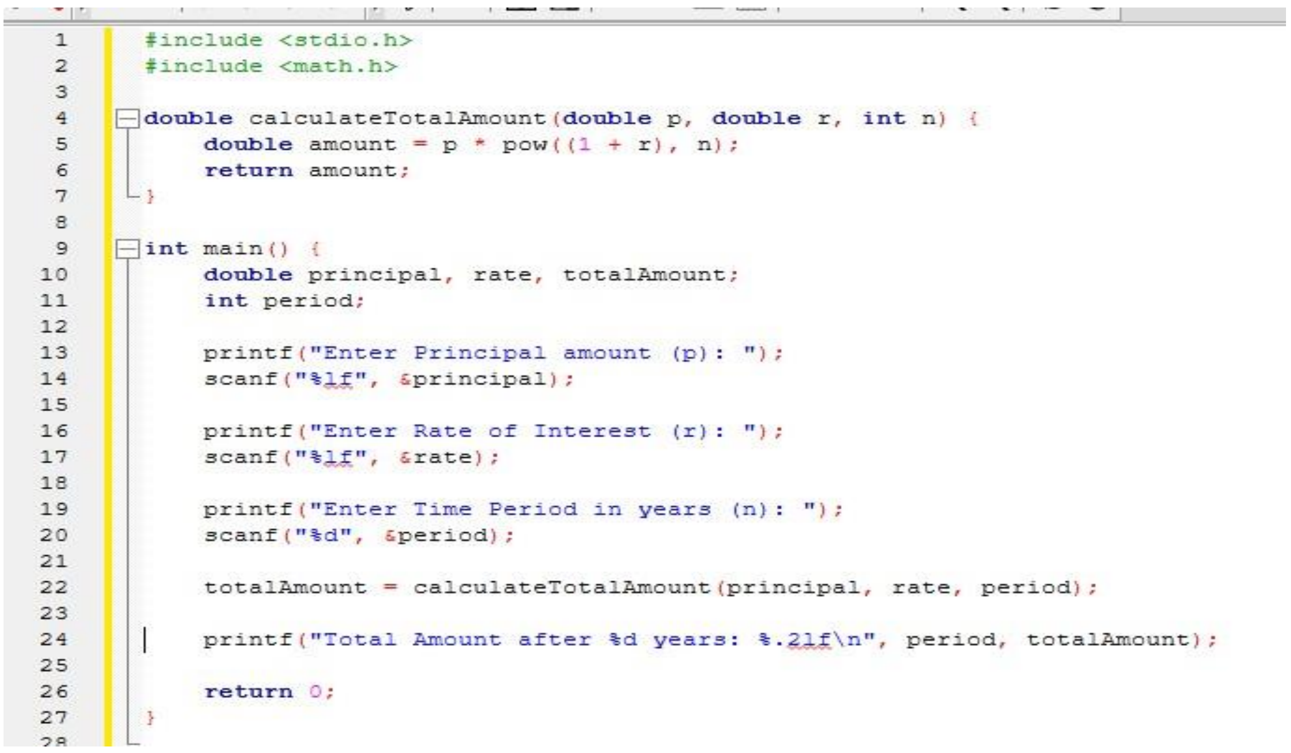
Step7: End the program

Flowchart:



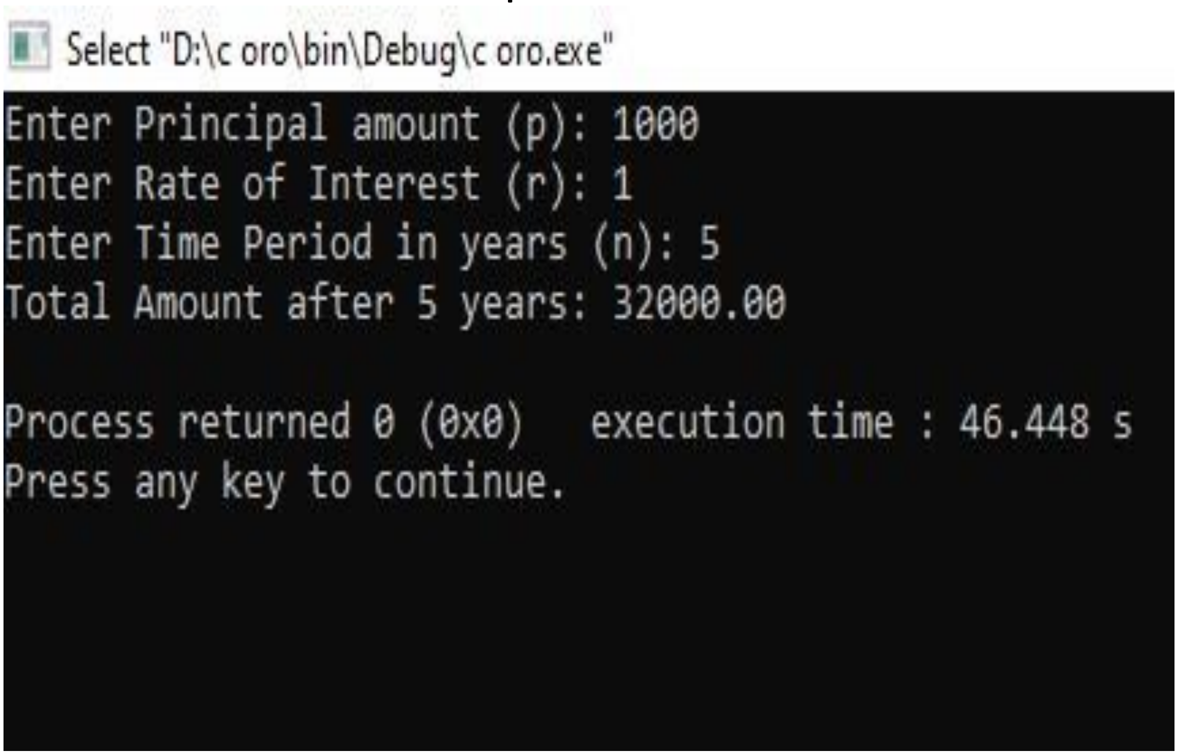
Start
Input p, r, n
Call calculate Amount (p, r, n)
total = p * pow((1 + r), n) [inside function]
Return total to main()
Display total amount
End

Source Code:



```c
#include <stdio.h>
#include <math.h>

double calculateTotalAmount(double p, double r, int n) {
    double amount = p * pow((1 + r), n);
    return amount;
}

int main() {
    double principal, rate, totalAmount;
    int period;

    printf("Enter Principal amount (p): ");
    scanf("%lf", &principal);

    printf("Enter Rate of Interest (r): ");
    scanf("%lf", &rate);

    printf("Enter Time Period in years (n): ");
    scanf("%d", &period);

    totalAmount = calculateTotalAmount(principal, rate, period);

    printf("Total Amount after %d years: %.2lf\n", period, totalAmount);

    return 0;
}
```

Experiment No-2



Select "D:\c oro\bin\Debug\c oro.exe"

```
Enter Principal amount (p): 1000
Enter Rate of Interest (r): 1
Enter Time Period in years (n): 5
Total Amount after 5 years: 32000.00

Process returned 0 (0x0)   execution time : 46.448 s
Press any key to continue.
```

Discussion:

This program takes the values of principal, rate, and time period as inputs and uses a function with arguments and return value to calculate the total amount using the compound interest formula. It uses the pow() function from the math.h library to handle the exponent part. The separation of calculation into a function makes the code clean and reusable. The program works correctly for any valid numeric input and shows how powerful user-defined functions can be in real-wold calculations.