

## 1. Write a C program to arrange numbers using Selection Sort.

**Aim:** To write a C program to sort a given list of numbers using Selection Sort .

### Algorithm:

1. Take numbers in an array
2. Find the smallest number and put it first
3. Repeat for the remaining numbers
4. Print the sorted array

**Input:** 11 64 26 36 82

**Output:** 11 26 36 64 82

The screenshot shows the Embarcadero Dev-C++ IDE with a C program for Selection Sort. The code is as follows:

```
1 #include <stdio.h>
2 int main() {
3     int n;
4     printf("Enter number of elements: ");
5     scanf("%d", &n);
6     int a[n];
7     printf("Enter elements: ");
8     for(int i=0; i<n; i++) scanf("%d", &a[i]);
9
10    for(int i=0; i<n-1; i++){
11        int min=i;
12        for(int j=i+1; j<n; j++){
13            if(a[j]<a[min]) min=j;
14        }
15        int temp=a[i]; a[i]=a[min]; a[min]=temp;
16    }
17    printf("Sorted array: ");
18    for(int i=0; i<n; i++) printf("%d ", a[i]);
19    return 0;
20 }
```

The console output shows the program execution:

```
Enter number of elements: 5
Enter elements: 11 64 26 36 82
Sorted array: 11 26 36 64 82
.....
Process exited after 20.18 seconds with return value 0
Press any key to continue . . .
```

The bottom status bar indicates: Line: 21 Col: 1 Sel: 0 Lines: 21 Length: 476 Insert Done parsing in 0 seconds.

## 2. Duplicate in a instruction.

### Aim:

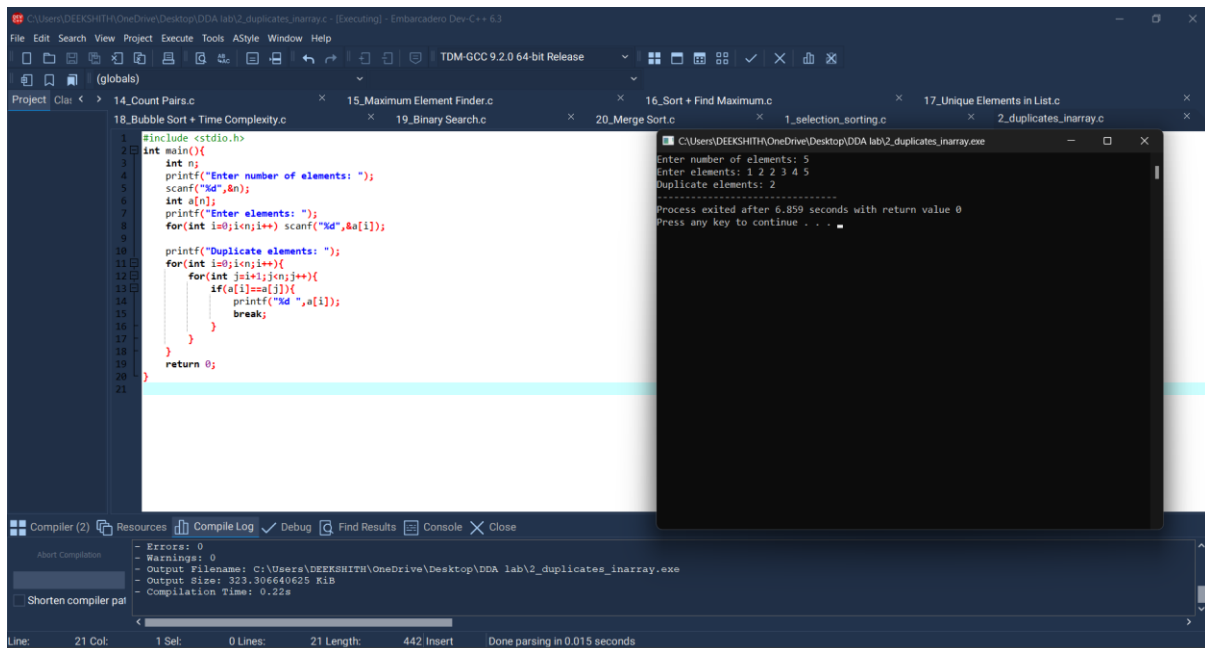
To write a C program to find duplicate elements in an array.

### Algorithm:

1. Start
2. Read n numbers into array
3. Compare each element with others
4. If any two are equal, print as duplicate
5. Stop

**Input:** 1 2 2 3 4 5

**Output:**2



```
1 #include <stdio.h>
2 int main()
3 {
4     int n;
5     printf("Enter number of elements: ");
6     scanf("%d", &n);
7     int a[n];
8     printf("Enter elements: ");
9     for(int i=0; i<n; i++) scanf("%d", &a[i]);
10
11     printf("Duplicate elements: ");
12     for(int i=0; i<n; i++){
13         for(int j=i+1; j<n; j++){
14             if(a[i]==a[j]){
15                 printf("%d ", a[i]);
16                 break;
17             }
18         }
19     }
20     return 0;
21 }
```

Enter number of elements: 5  
Enter elements: 1 2 2 3 4 5  
Duplicate elements: 2  
-----  
Process exited after 6.859 seconds with return value 0  
Press any key to continue . . .

Compiler (2) Resources Compile Log Debug Find Results Console Close  
- Errors: 0  
- Warnings: 0  
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab2\_duplicates\_inarray.exe  
- Output Size: 323.306640625 KiB  
- Compilation Time: 0.22s

### 3. Bigger Number in a Series.

#### Aim:

To write a C program to find the largest number from given numbers.

#### Algorithm:

1. Start
2. Read n numbers
3. Assume first number as max
4. Compare each number with max
5. If bigger, update max
6. Print max
7. Stop

**Input:** 1 2 4 43 22

**Output:** 43

The screenshot shows the Embarcadero Dev-C++ IDE with a C program open. The program prompts the user to enter the size of the array and the elements. It then iterates through the elements, comparing each to the current maximum and updating it if a larger number is found. The output shows the largest number is 43.

```
1 #include <stdio.h>
2 int main()
3 {
4     int n;
5     printf("Enter size: ");
6     scanf("%d", &n);
7     int a[n];
8     printf("Enter elements: ");
9     for(int i=0; i<n; i++) scanf("%d", &a[i]);
10    int max=a[0];
11    for(int i=1; i<n; i++)
12        if(a[i]>max) max=a[i];
13    printf("Largest number: %d", max);
14    return 0;
15 }
```

Output window content:

```
Enter size: 5
Enter elements: 1 2 4 43 22
Largest number: 43
.....
Process exited after 8.306 seconds with return value 0
Press any key to continue . . .
```

Compiler output:

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\3_bigger_inseries.exe
- Output Size: 323,3037109375 KiB
- Compilation Time: 0.20s
```

## 4. Recursion – Factorial of a Given Number.

### Aim:

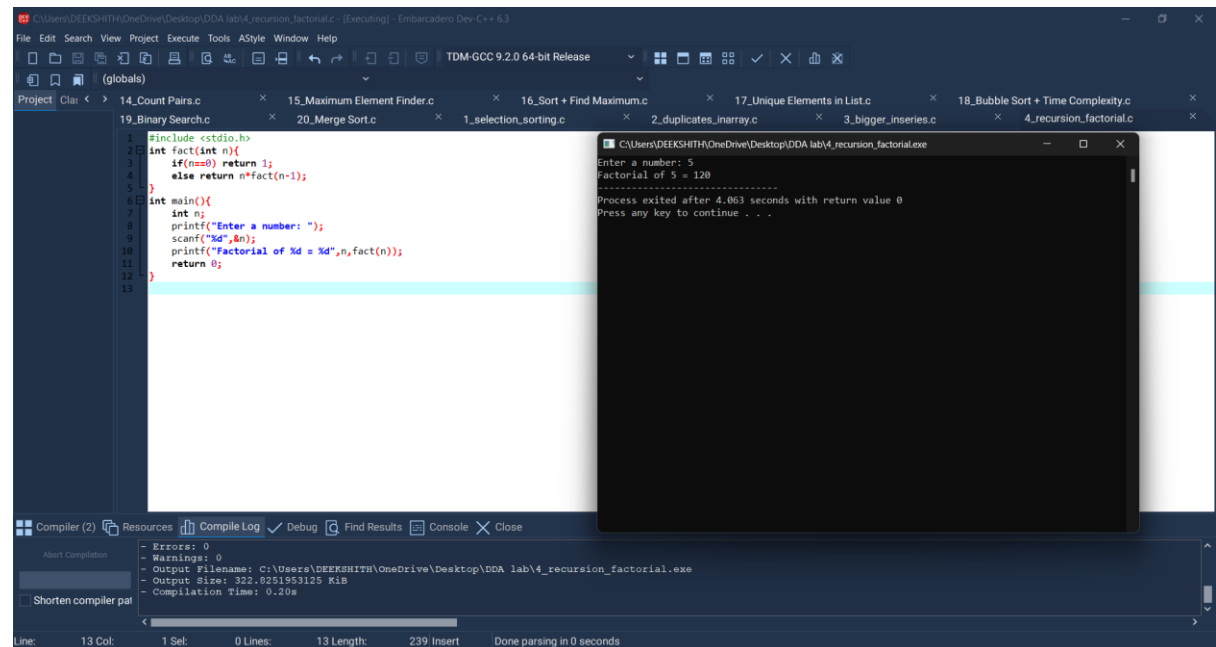
To write a C program to find the factorial of a number using recursion.

### Algorithm:

1. Start
2. Read a number n
3. If  $n==0$  or  $n==1 \rightarrow$  return 1
4. Else return  $n * \text{factorial}(n-1)$
5. Print result
6. Stop

**Input:** 5

**Output:** 120



The screenshot displays a C++ IDE with the following components:

- Editor:** Contains the C program for calculating the factorial of a number using recursion. The code is as follows:

```
1 #include <stdio.h>
2 int fact(int n){
3     if(n==0) return 1;
4     else return n*fact(n-1);
5 }
6 int main(){
7     int n;
8     printf("Enter a number: ");
9     scanf("%d",&n);
10    printf("Factorial of %d = %d",n,fact(n));
11    return 0;
12 }
13
```
- Output Window:** Shows the execution results:

```
Enter a number: 5
Factorial of 5 = 120
-----
Process exited after 4.063 seconds with return value 0
Press any key to continue . . .
```
- Compiler Output:** Located at the bottom, it shows successful compilation with no errors or warnings:

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab4_recursion_factorial.exe
- Output Size: 322,825,195,3125 KiB
- Compilation Time: 0.20s
```

## 5. Fibonacci Series.

### Aim:

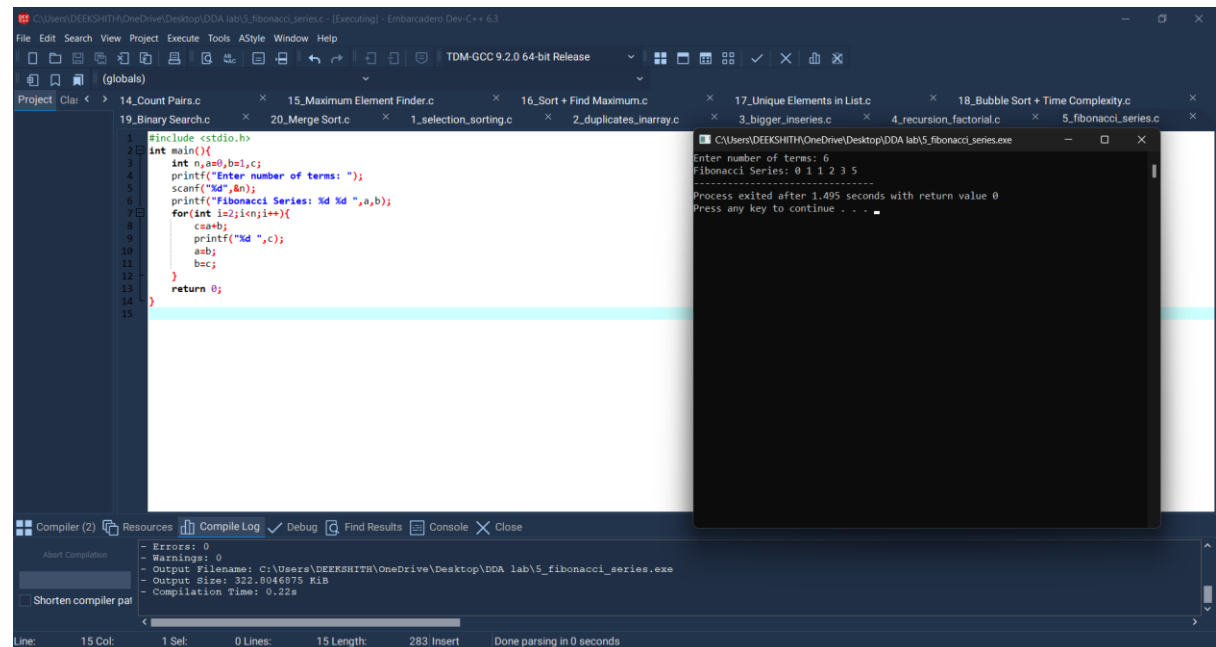
To write a C program to generate the Fibonacci series.

### Algorithm:

1. Start
2. Read n terms
3. Initialize t1=0, t2=1
4. Print t1 and t2
5. Repeat for remaining terms: next = t1+t2, print, update t1=t2, t2=next
6. Stop

**Input:** 6

**Output:** 0 1 1 2 3 5



The screenshot shows an IDE window with a C program for generating the Fibonacci series. The code is as follows:

```
1 #include <stdio.h>
2 int main()
3 {
4     int n,a=0,b=1,c;
5     printf("Enter number of terms: ");
6     scanf("%d",&n);
7     printf("Fibonacci Series: %d %d ",a,b);
8     for(int i=2;i<=n;i++){
9         c=a+b;
10        printf("%d ",c);
11        a=b;
12        b=c;
13    }
14    return 0;
15 }
```

The IDE also shows a console window with the following output:

```
Enter number of terms: 6
Fibonacci Series: 0 1 1 2 3 5
Process exited after 1.495 seconds with return value 0
Press any key to continue . . .
```

The bottom status bar of the IDE indicates: Line: 15 Col: 1 Sel: 0 Lines: 15 Length: 283 Insert Done parsing in 0 seconds.

## 6. Two Order Homogeneous Recursion.

### Aim:

To write a C program using recursion for a second-order homogeneous recurrence relation.

### Algorithm:

1. Start
2. Define recursive relation:  $F(n)=F(n-1)+F(n-2)$
3. Base cases:  $F(0)=0$ ,  $F(1)=1$
4. Print terms using recursion
5. Stop

**Input:** terms=5

**Output:** 0 1 1 2 3

The screenshot shows the Embarcadero Dev-C++ IDE with a C program for calculating the Fibonacci sequence. The code is as follows:

```
1 #include <stdio.h>
2 int fib(int n) {
3     if(n==0) return 0;
4     if(n==1) return 1;
5     return fib(n-1)+fib(n-2);
6 }
7 int main() {
8     int n,i;
9     printf("Enter number of terms: ");
10    scanf("%d",&n);
11    for(i=0;i<n;i++)
12        printf("%d ",fib(i));
13    return 0;
14 }
15
```

The output window shows the execution results:

```
Enter number of terms: 5
0 1 1 2 3
-----
Process exited after 6.447 seconds with return value 0
Press any key to continue . . .
```

The status bar at the bottom indicates: Line: 15 Col: 1 Sel: 0 Lines: 15 Length: 280 Insert Done parsing in 0.016 seconds.

## 7. Leap Year

### Aim:

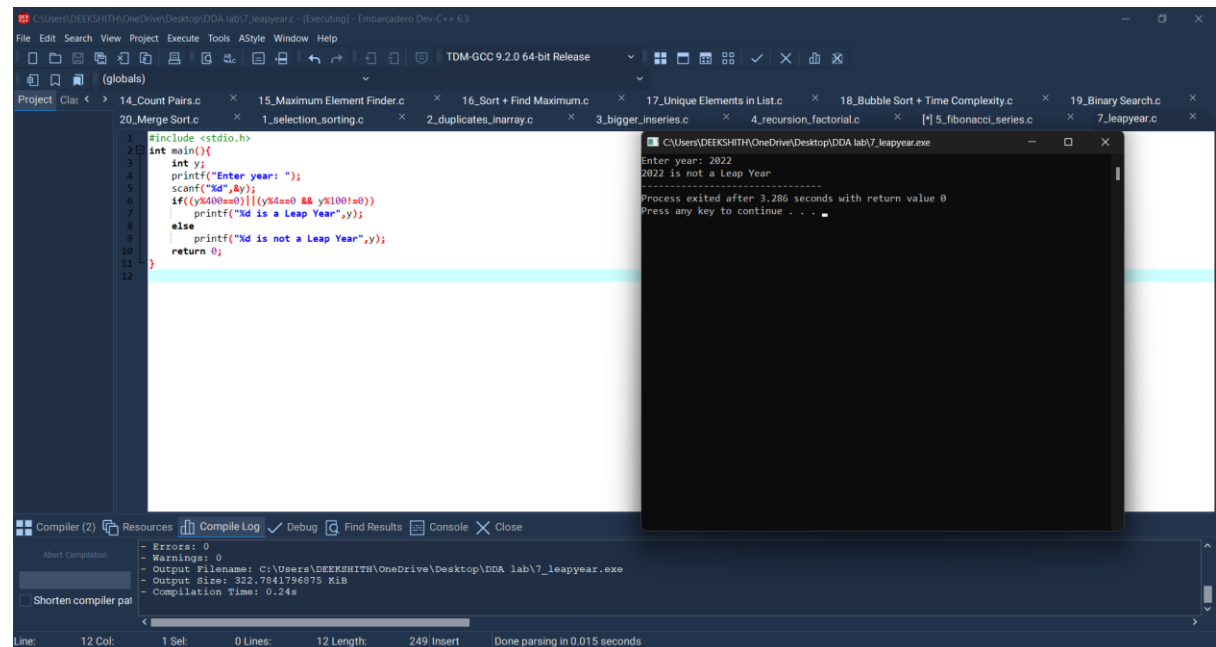
To write a C program to check whether a year is a leap year.

### Algorithm:

1. Start
2. Read year
3. If divisible by 400 → leap year
4. Else if divisible by 4 but not by 100 → leap year
5. Else not a leap year
6. Stop

**Input:** year=2022

**Output:** 2022 is not a leap year



The screenshot shows the Embarcadero Dev-C++ IDE. The main window displays a C program for checking leap years. The code is as follows:

```
1 #include <stdio.h>
2 int main()
3 {
4     int y;
5     printf("Enter year: ");
6     scanf("%d", &y);
7     if((y%400==0)||((y%4==0 && y%100!=0)))
8         printf("%d is a Leap Year", y);
9     else
10        printf("%d is not a Leap Year", y);
11    return 0;
12 }
```

A console window titled "C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\7\_leapyear.exe" is open, showing the program's execution. It prompts "Enter year: 2022" and outputs "2022 is not a Leap Year". Below the console, the compiler output shows 0 errors and 0 warnings, with the output file named "C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\7\_leapyear.exe".

## 8. Swapping of Numbers.

### Aim:

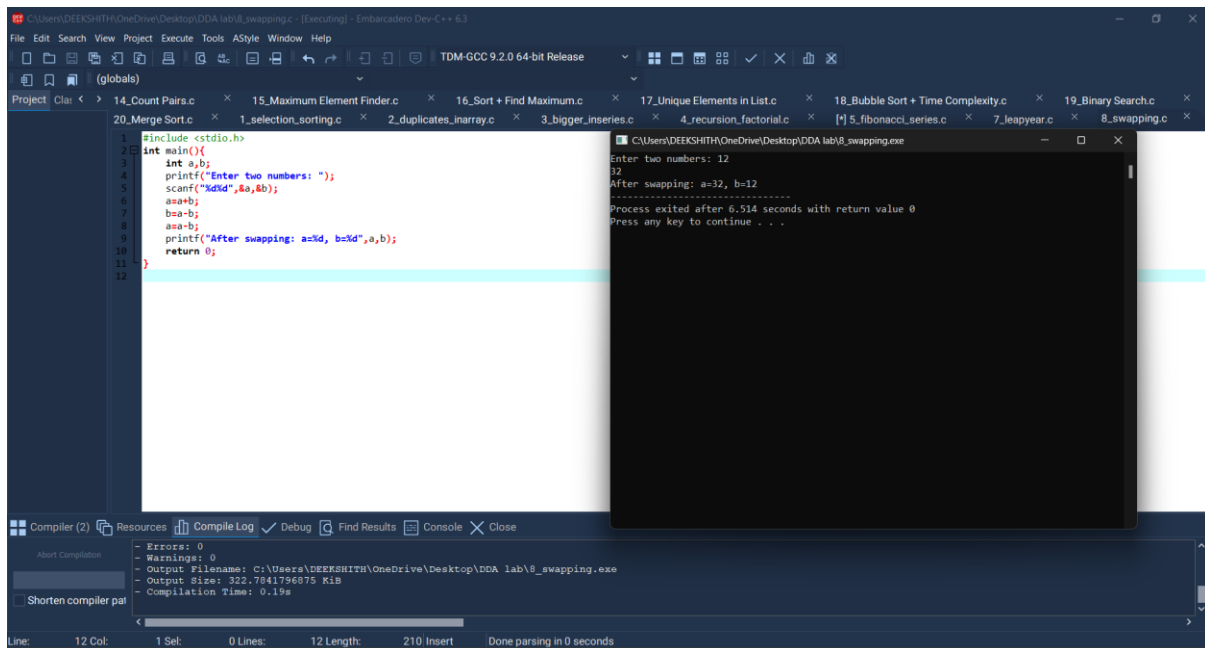
To write a C program to swap two numbers.

### Algorithm:

1. Start
2. Read two numbers a and b
3. Swap using temp variable (or without)
4. Print swapped values
5. Stop

**Input:** a=12 b=32

**Output:** a=32 b=12



The screenshot displays an IDE window titled "C:\Users\DEEKSHITHA\OneDrive\Desktop\DDA lab\8\_swapping.c - [Executing] - Embarcadero Dev-C++ 6.3". The code editor shows the following C program:

```
1 #include <stdio.h>
2
3 int main(){
4     printf("Enter two numbers: ");
5     scanf("%d%d",&a,&b);
6     a=a+b;
7     b=a-b;
8     a=a-b;
9     printf("After swapping: a=%d, b=%d",a,b);
10    return 0;
11 }
12
```

The console window shows the execution output:

```
Enter two numbers: 12
32
After swapping: a=32, b=12
-----
Process exited after 6.514 seconds with return value 0
Press any key to continue . . .
```

The bottom status bar indicates "Line: 12 Col: 1 Sel: 0 Lines: 12 Length: 210 Insert Done parsing in 0 seconds".



## 9. Identifying Palindrome

### Aim:

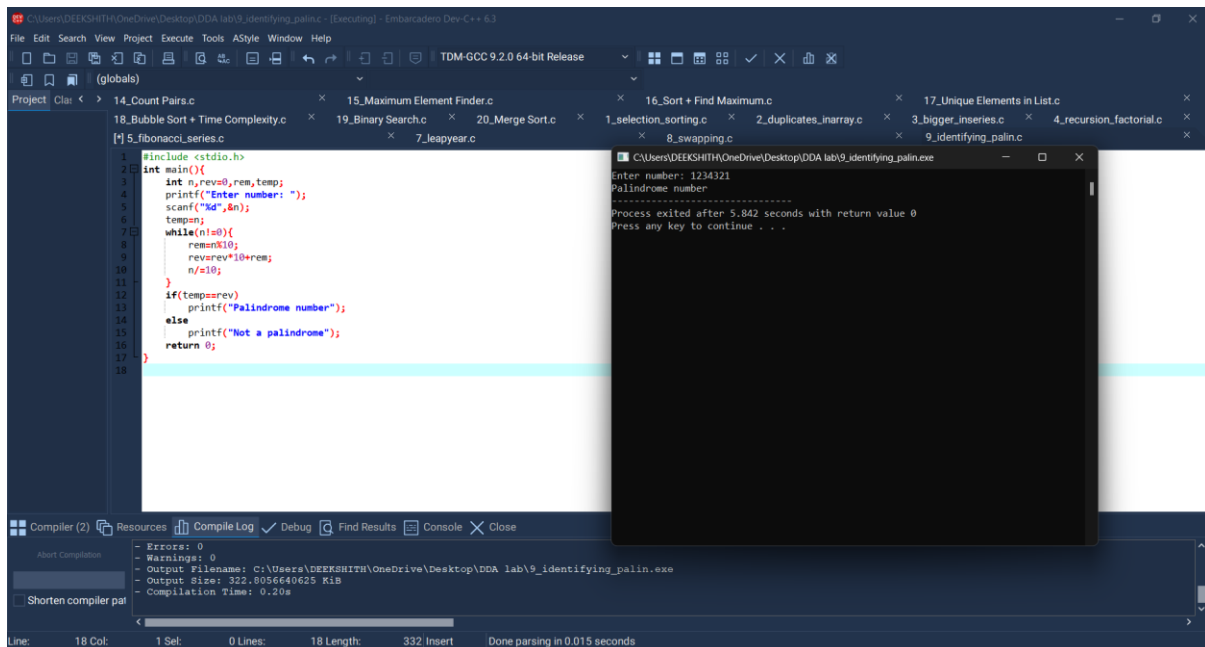
To write a C program to check whether a number is a palindrome.

### Algorithm:

1. Start
2. Read a number n
3. Reverse digits of n
4. If reverse = original  $\rightarrow$  palindrome
5. Else not palindrome
6. Stop

**Input:** 1234321

**Output:** palindrome number



The screenshot shows a code editor with a C program for identifying a palindrome. The program uses a while loop to reverse the digits of the input number and then compares the reversed number with the original. The output window shows the program's execution for the input 1234321, confirming it is a palindrome.

```
1 #include <stdio.h>
2 int main()
3 {
4     int n, rev=0, rem, temp;
5     printf("Enter number: ");
6     scanf("%d", &n);
7     temp=n;
8     while(n!=0)
9     {
10        rem=n%10;
11        rev=rev*10+rem;
12        n/=10;
13    }
14    if(temp==rev)
15        printf("Palindrome number");
16    else
17        printf("Not a palindrome");
18    return 0;
19 }
```

Output window content:

```
Enter number: 1234321
Palindrome number
Process exited after 5.842 seconds with return value 0
Press any key to continue . . .
```

Compiler output:

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab9_identifying_palin.exe
- Output Size: 322,805,664,9625 KiB
- Compilation Time: 0.20s
```

## 10. Prime Number

### Aim:

To write a C program to check whether a number is prime.

### Algorithm:

1. Start
2. Read n
3. If  $n \leq 1 \rightarrow$  not prime
4. Check divisibility from 2 to  $n/2$
5. If divisible  $\rightarrow$  not prime
6. Else  $\rightarrow$  prime
7. Stop

**Input:** 37

**Output:** prime number

The screenshot shows the Embarcadero Dev-C++ IDE with a C program open. The program prompts the user to enter a number and checks if it is prime. The output window shows the execution results for the input 37.

```
#include <stdio.h>
#include <math.h>
int main()
{
    int n, flag=0;
    printf("Enter number: ");
    scanf("%d", &n);
    if(n<=1) flag=1;
    for(int i=2; i<=sqrt(n); i++)
        if(n%i==0) { flag=1; break; }
    if(flag==0)
        printf("%d is a Prime number", n);
    else
        printf("%d is not a Prime number", n);
    return 0;
}
```

Compiler (2) Resources Compile Log Debug Find Results Console Close

Errors: 0  
Warnings: 0  
Output Filename: C:\Users\DEEKSHITH\OneDrive\Desktop\DDA lab\10\_primenumber.exe  
Output Size: 327,052,734,375 KiB  
Compilation Time: 0.19s

Line: 16 Col: 1 Sel: 0 Lines: 16 Length: 352 Insert Done parsing in 0.031 seconds

Enter number: 37  
37 is a Prime number  
Process exited after 3.403 seconds with return value 0  
Press any key to continue . . .