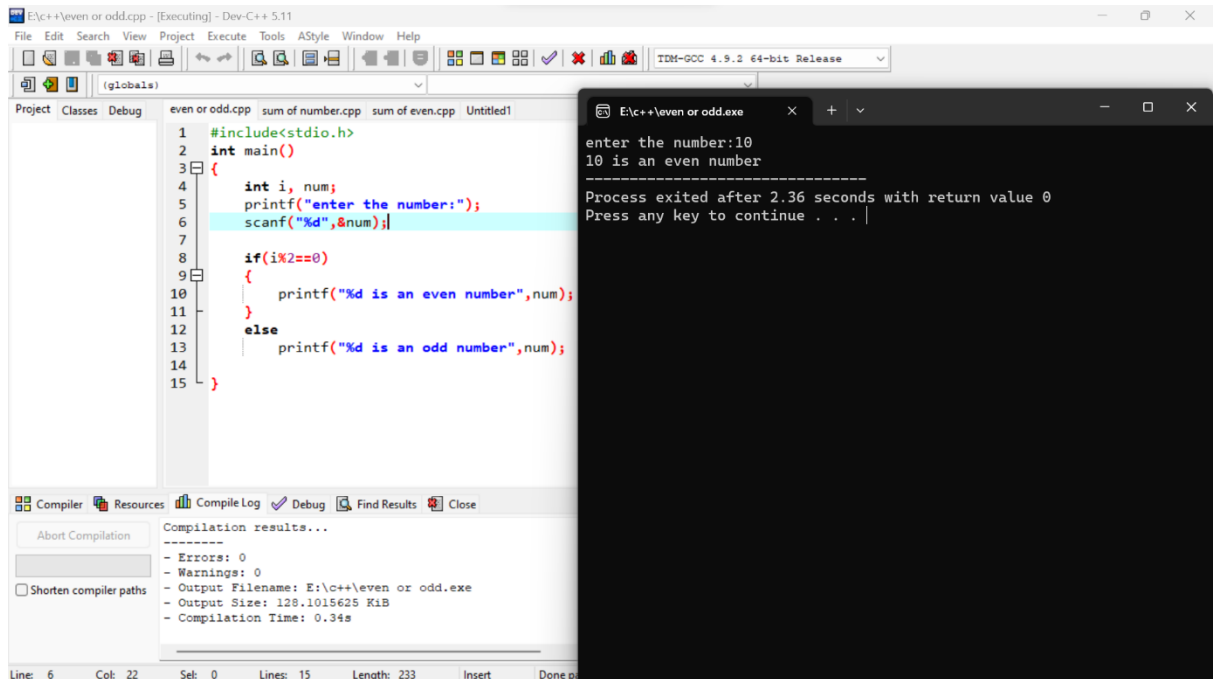


DAY - 1

DATA STRUCTURES – BASIC C PROGRAMMING - 1

1. Write a c program to check the given numbers is even or odd.



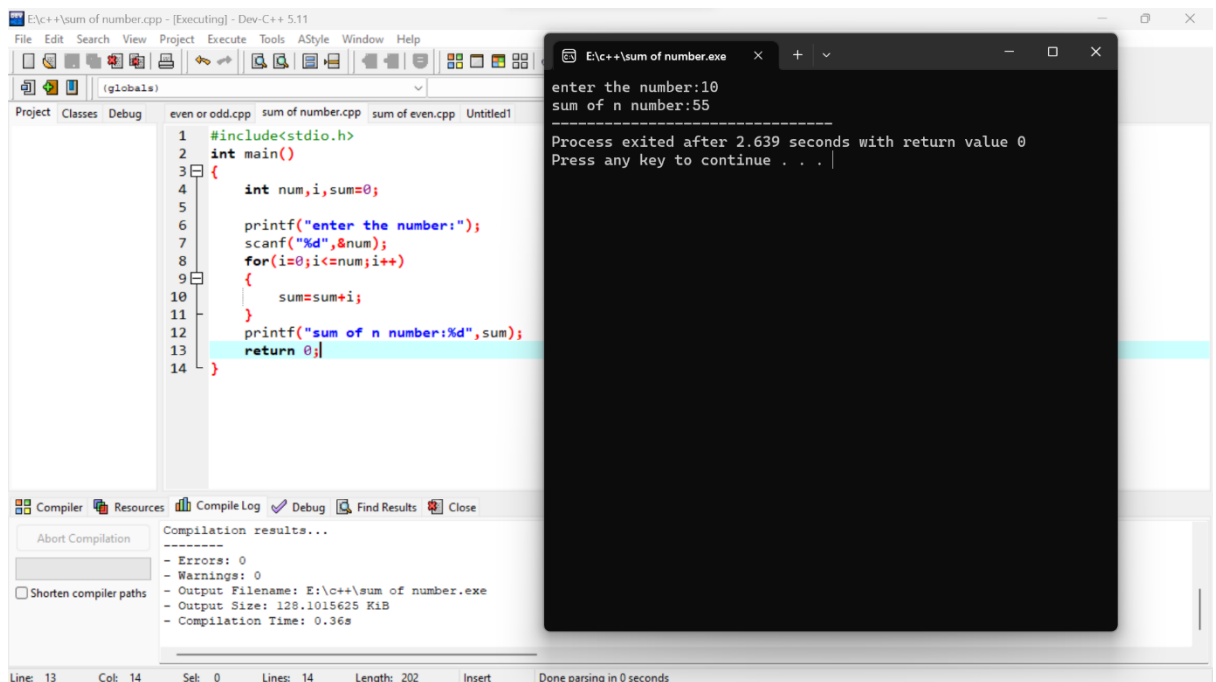
```
1 #include<stdio.h>
2 int main()
3 {
4     int i, num;
5     printf("enter the number:");
6     scanf("%d",&num);
7
8     if(i%2==0)
9     {
10        printf("%d is an even number",num);
11    }
12    else
13        printf("%d is an odd number",num);
14
15 }
```

enter the number:10
10 is an even number

Process exited after 2.36 seconds with return value 0
Press any key to continue . . .

Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: E:\c++\even or odd.exe
- Output Size: 128.1015625 KiB
- Compilation Time: 0.34s

2. Write a c program to find the sum of first n numbers using for loop.



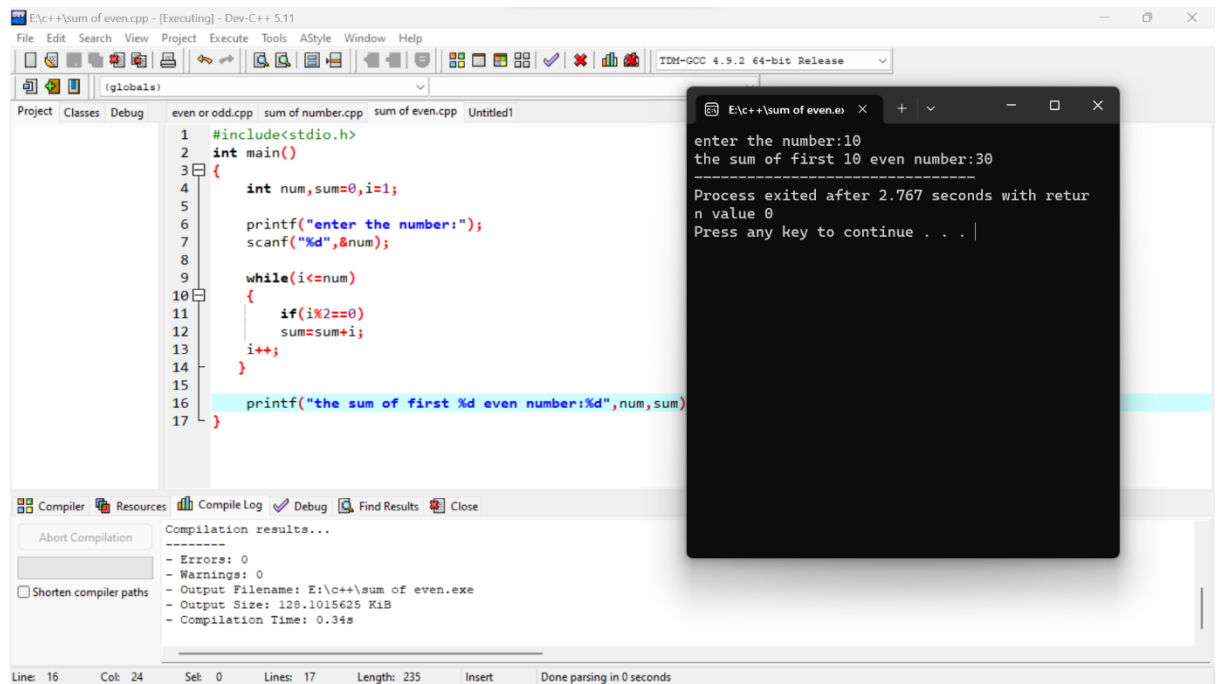
```
1 #include<stdio.h>
2 int main()
3 {
4     int num,i,sum=0;
5
6     printf("enter the number:");
7     scanf("%d",&num);
8     for(i=0;i<=num;i++)
9     {
10        sum=sum+i;
11    }
12    printf("sum of n number:%d",sum);
13    return 0;
14 }
```

enter the number:10
sum of n number:55

Process exited after 2.639 seconds with return value 0
Press any key to continue . . .

Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: E:\c++\sum of number.exe
- Output Size: 128.1015625 KiB
- Compilation Time: 0.36s

3. Write a c program to find sum of even numbers using while loop.



```
1 #include<stdio.h>
2 int main()
3 {
4     int num,sum=0,i=1;
5
6     printf("enter the number:");
7     scanf("%d",&num);
8
9     while(i<=num)
10    {
11        if(i%2==0)
12            sum=sum+i;
13        i++;
14    }
15
16    printf("the sum of first %d even number:%d",num,sum);
17 }
```

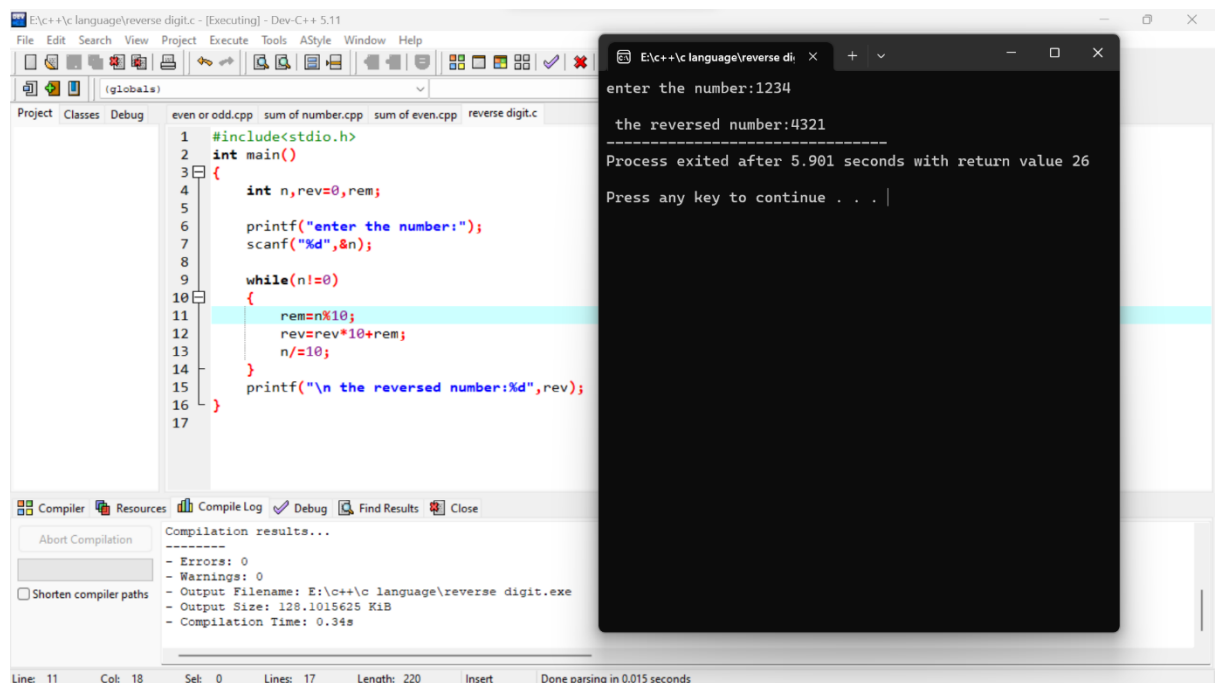
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: E:\c++\sum of even.exe
- Output Size: 128.1015625 KiB
- Compilation Time: 0.34s

enter the number:10
the sum of first 10 even number:30

Process exited after 2.767 seconds with return value 0
Press any key to continue . . . |

4. Write a c program to reverse a given number.



```
1 #include<stdio.h>
2 int main()
3 {
4     int n,rev=0,rem;
5
6     printf("enter the number:");
7     scanf("%d",&n);
8
9     while(n!=0)
10    {
11        rem=n%10;
12        rev=rev*10+rem;
13        n/=10;
14    }
15    printf("\n the reversed number:%d",rev);
16 }
17
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: E:\c++\c language\reverse digit.exe
- Output Size: 128.1015625 KiB
- Compilation Time: 0.34s

enter the number:1234
the reversed number:4321

Process exited after 5.901 seconds with return value 26
Press any key to continue . . . |

5. Write a c program to check the given numbers is palindrome or not.

The screenshot shows the Dev-C++ IDE with a C program to check if a number is a palindrome. The program is named `palindrm.c` and is located at `E:\c++\c language\palindrm.c`. The code is as follows:

```
1 int main()
2 {
3     int n, rev=0, original, rem;
4     printf("enter the number:");
5     scanf("%d", &n);
6     original=n;
7
8     while(n!=0)
9     {
10        rem=n%10;
11        rev=rev*10+rem;
12        n/=10;
13    }
14
15    if(original==rev)
16        printf("\n %d is a palindrome", original);
17    else
18        printf("\n %d is not a palindrome", original);
19 }
20
21 }
```

The compilation results show 0 errors and 0 warnings. The output filename is `E:\c++\c language\palindrm.exe`. The output size is 128.6015625 KiB, and the compilation time is 0.33s.

The execution output shows the user entering the number 11111, and the program outputting "11111 is a palindrome". The process exited after 2.292 seconds with a return value of 2.

6. Write a c program to check the given number is Armstrong or not.

The screenshot shows the Dev-C++ IDE with a C program to check if a number is an Armstrong number. The program is named `armstrong.c` and is located at `E:\c++\c language\armstrong.c`. The code is as follows:

```
1 #include<stdio.h>
2 int main()
3 {
4     int num, org, rem, res=0;
5
6     printf("enter the number:");
7     scanf("%d", &num);
8     org=num;
9
10    while(org !=0)
11    {
12        rem=org%10;
13        res+=rem*rem*rem;
14        org/=10;
15    }
16    if(res==num)
17        printf("\n %d is armstrong number", num);
18    else
19        printf("\n %d is not an armstrong number", num);
20 }
```

The compilation results show 0 errors and 0 warnings. The output filename is `E:\c++\c language\armstrong.exe`. The output size is 128.6015625 KiB, and the compilation time is 0.34s.

The execution output shows the user entering the number 371, and the program outputting "371 is armstrong number". The process exited after 3.606 seconds with a return value of 25.

7. Write a c program to find factorial of given number without recursion.

The screenshot shows the Dev-C++ IDE with a C program for calculating factorial without recursion. The code is as follows:

```
1 #include<stdio.h>
2 int main()
3 {
4     int fact=1,num,i;
5
6     printf("enter the number:");
7     scanf("%d",&num);
8
9     for(i=1;i<=num;i++)
10    {
11        fact=fact*i;
12    }
13    printf("the factorial of %d is %d",num,fact);
14 }
```

The output window shows the program's execution:

```
enter the number:5
the factorial of 5 is 120
-----
Process exited after 1.688 seconds with return value 24
Press any key to continue . . . |
```

Compilation results:

```
-----
- Errors: 0
- Warnings: 0
- Output Filename: E:\c++\c language\factorial.exe
- Output Size: 128.1015625 KiB
- Compilation Time: 0.33s
```

8. Write a c program to find factorial of given number with recursion.

The screenshot shows the Dev-C++ IDE with a C program for calculating factorial using recursion. The code is as follows:

```
1 #include<stdio.h>
2 long factorial(int n)
3 {
4     if (n == 0)
5         return 1;
6     else
7         return(n * factorial(n-1));
8 }
9
10 int main()
11 {
12     int number;
13     long fact;
14     printf("Enter a number: ");
15     scanf("%d", &number);
16
17     fact = factorial(number);
18     printf("Factorial of %d is %d \n", number, fact);
19 }
```

The output window shows the program's execution:

```
enter the number:10
the factorial of 10 is 3628800
-----
Process exited after 5.116 seconds with return value 29
Press any key to continue . . . |
```

Compilation results:

```
-----
- Errors: 0
- Warnings: 0
- Output Filename: E:\c++\c language\factorial recursion.exe
- Output Size: 128.6513671875 KiB
- Compilation Time: 0.34s
```

9. Write a c program to generate Fibonacci series without recursion.

The screenshot shows the Dev-C++ IDE with a C program for generating the Fibonacci series without recursion. The code is as follows:

```
1 #include<stdio.h>
2 int main()
3 {
4     int t1=0,t2=1,n,i;
5     int nxt=t1+t2;
6
7     printf("enter the number:");
8     scanf("%d",&n);
9
10    printf("fibonacci series of %d, %d ",t1 ,t2);
11    for(i=3;i<=n;i++)
12    {
13        printf(" %d",nxt);
14        t1=t2;
15        t2=nxt;
16        nxt=t1+t2;
17    }
18 }
19 }
```

The compilation results show 0 errors and 0 warnings. The output file is E:\c++\c language\fibonacci.exe, with a size of 128.1015625 KiB and a compilation time of 0.38s.

The execution output is as follows:

```
enter the number:10
fibonacci series of 0, 1 1 2 3 5 8 13 21 34
-----
Process exited after 3.165 seconds with return value 10
Press any key to continue . . .
```

10. Write a c program to generate Fibonacci series with recursion.

The screenshot shows the Dev-C++ IDE with a C program for generating the Fibonacci series with recursion. The code is as follows:

```
1 #include<stdio.h>
2 int printFibonacci(int n)
3 {
4     int n1=0,n2=1,n3;
5     if(n>0)
6     {
7         n3 = n1 + n2;
8         n1 = n2;
9         n2 = n3;
10        printf("%d ",n3);
11        printFibonacci(n-1);
12    }
13 }
14 int main()
15 {
16     int n;
17     printf("Enter the number of elements: ");
18     scanf("%d",&n);
19
20     printf("Fibonacci Series: ");
```

The compilation results show 0 errors and 0 warnings. The output file is E:\c++\c language\fibonacci recursion.exe, with a size of 128.6630859375 KiB and a compilation time of 0.33s.

The execution output is as follows:

```
enter the number:10
fibonacci series of 0, 1 1 2 3 5 8 13 21 34
-----
Process exited after 3.432 seconds with return value 10
Press any key to continue . . .
```

11. Write a c program to search element in an array using linear search.

The screenshot shows the Dev-C++ IDE with a C program for linear search. The code is as follows:

```

5  #include <stdio.h>
6
7  int main()
8  {
9      int elementCount;
10     printf("Enter Number of Elements in Array\n");
11     scanf("%d", &elementCount);
12     printf("Enter %d numbers \n", elementCount);
13
14     for(counter = 0; counter < elementCount; counter++){
15         scanf("%d", &inputArray[counter]);
16     }
17
18     printf("Enter a number to serach in Array\n");
19     scanf("%d", &num);
20
21     for(counter = 0; counter < elementCount; counter++){
22         if(inputArray[counter] == num){
23             printf("Number %d found at index %d\n",
24                   num, counter);
25             break;
26         }
27     }
28
29     if(counter == elementCount)
30     {
31         printf("Number %d Not Present in Input Array\n", num);
32     }
33
34     return 0;
35 }

```

The execution output is shown in a separate window:

```

Enter Number of Elements in Array
6
Enter 6 numbers
1 2 4 5 6 7
Enter a number to serach in Array
5
Number 5 found at index 3
=====
Process exited after 21.1 seconds with return
value 0
Press any key to continue . . .

```

The compilation results show 0 errors and 0 warnings. The output filename is E:\c++\c language\linear search.exe, the output size is 128.7705078125 KiB, and the compilation time is 1.88s.

12. Write a c program to search element in an array using binary search.

The screenshot shows the Dev-C++ IDE with a C program for binary search. The code is as follows:

```

1  #include <stdio.h>
2  int main()
3  {
4      int c, first, last, middle, n, search, array[100];
5      printf("Enter number of elements\n");
6      scanf("%d", &n);
7      printf("Enter %d integers\n", n);
8      for (c = 0; c < n; c++)
9      {
10         scanf("%d", &array[c]);
11     }
12     printf("Enter value to find\n");
13     scanf("%d", &search);
14     first = 0;
15     last = n - 1;
16     middle = (first+last)/2;
17     while( first <= last )
18     {
19         if ( array[middle] < search )
20             first = middle + 1;
21         else if ( array[middle] == search )
22         {
23             printf("Number %d found at location %d\n", search, middle+1);
24             break;
25         }
26         else
27             last = middle - 1;
28         middle = (first + last)/2;
29     }
30 }

```

The execution output is shown in a separate window:

```

Enter number of elements
5
Enter 5 integers
2
3
4
5
6
Enter value to find
4
4 found at location 3.
=====
Process exited after 9.08 seconds with return value 0
Press any key to continue . . .

```

The compilation results show 0 errors and 0 warnings. The output filename is E:\c++\c language\binary serach.exe, the output size is 128.7705078125 KiB, and the compilation time is 0.52s.

