**SINGLE PRECISION REPRESENTATION:**

**EXP NO:35**

**AIM:**To write a C program to implement single precision representation.

**APPARATUS:** DEV C++

**ALGORITHM:**

**1)Step-1:** First the registers are initialized with corresponding values (Q = Dividend, M= Divisor, A = 0, n = number of bits in dividend)

**2)Step-2:** Then the content of register A and Q is shifted left as if they are a single unit

**3)Step-3:** Then content of register M is subtracted from A and result is stored in A

**4)Step-4:** Then the most significant bit of the A is checked if it is 0 the least significant bit of Q is set to 1 otherwise if it is 1 the least significant bit of Q is set to 0 and value of register A is restored i.e., the value of A before the subtraction with M

**5)Step-5:** The value of counter n is decremented

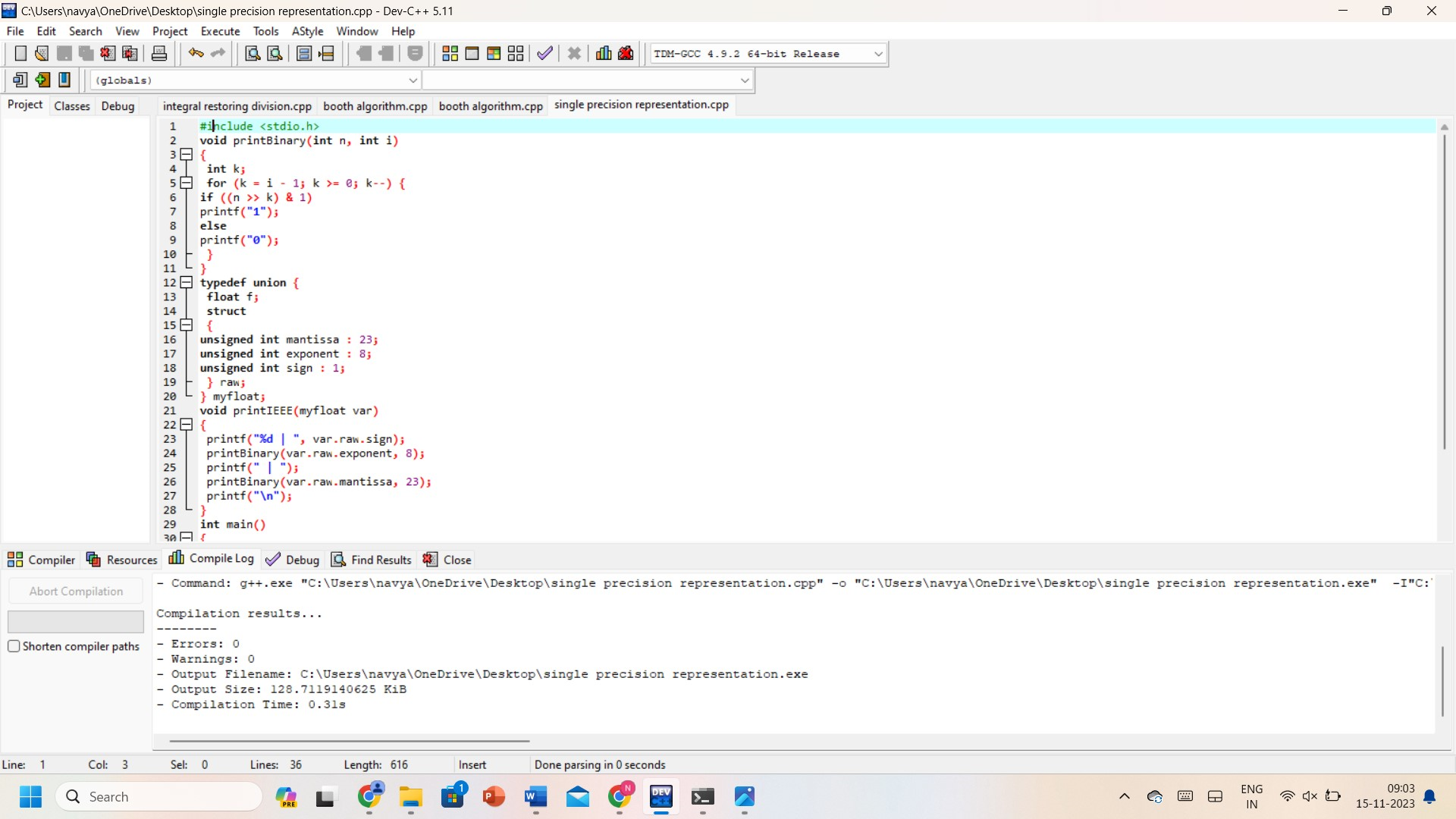
**6)Step-6:** If the value of n becomes zero, we get of the loop otherwise we repeat from step 2

**7)Step-7:** Finally, the register Q contain the quotient and A contain remainder

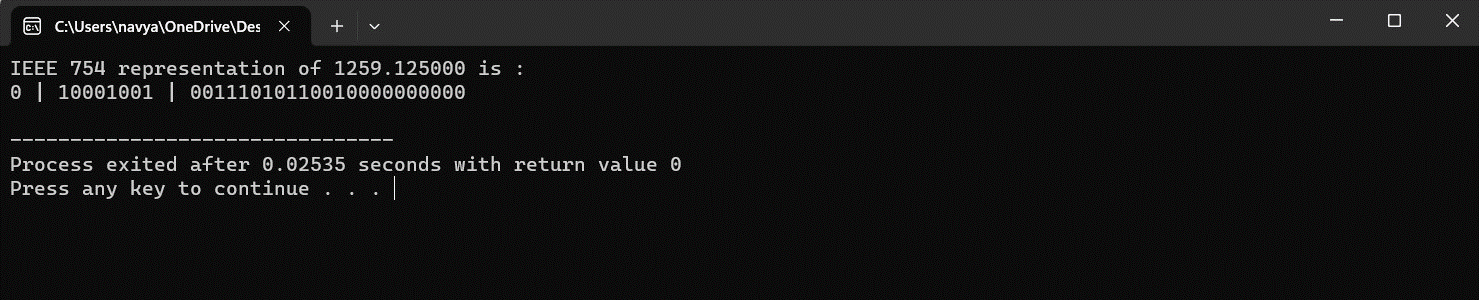
**PROGRAM:**

#include <stdio.h>  
void print Binary(int n, int i)  
{  
 int k;  
 for (k = i - 1; k >= 0; k--) {   
if ((n >> k) & 1)    
printf("1");  
else    
printf("0");  
 }  
}  
typedef union {  
 float f;  
 struct  
 {  
unsigned int mantissa : 23;  
unsigned int exponent : 8;  
unsigned int sign : 1;  
 } raw;  
} myfloat;  
void printIEEE(myfloat var)  
{  
 printf("%d | ", var.raw.sign);  
 print Binary(var.raw.exponent, 8);  
 printf(" | ");  
 print Binary(var.raw.mantissa, 23);  
 printf("\n");  
}  
int main()  
{  
myfloat var;  
var.f = 1259.125;  
printf("IEEE 754 representation of %f is : \n",var.f);  
 printIEEE(var);  
 return 0;  
}

**INPUT:**



**OUTPUT:**



**RESULT:** Thus, the program was executed successfully using DevC++.