

## INTEGER RESTORING DIVISION

### EXP NO: 33

**AIM:** To write a C program to implement integer restoring division.

#### ALGORITHM:

Step 1: Start.

Step 2: Declare necessary variables: acum[], q[], and b[] to store the accumulator, quotient, and divisor in binary form.

Step 3: Prompt the user to enter two integers (dividend and divisor).

Step 4: Convert the dividend (x) and divisor (y) into binary and store them in q[] and b[] respectively.

Step 5: Form the two's complement of the divisor b[] and store it in bc[].

Step 6: Perform the division using the restoring method.

Step 7: For each iteration: a. Shift the accumulator (acum[]) left by one bit. b. Shift the quotient (q[]) left by one bit. c. Subtract the divisor from the accumulator. If the accumulator is negative, restore the accumulator by adding the divisor back and set the current quotient bit to 0. d. If the accumulator is positive or zero, set the current quotient bit to 1. Step 8: After all iterations, display the quotient and remainder in binary form.

Step 9: End.

#### PROGRAM:

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

```
int acum[100] = {0};
```

```
int q[100], b[100];
```

```
// Function to add two binary numbers
```

```
void add(int acum[], int b[], int n);
```

```
int main() {
```

```
    int x, y, i = 0;
```

```
    // Input the dividend and divisor
```

```
    printf("Enter the Number (dividend and divisor): ");
```

```
    scanf("%d%d", &x, &y);
```

```
    // Convert the dividend and divisor to binary
```

```
    while (x > 0 || y > 0) {
```

```
        q[i] = (x > 0) ? x % 2 : 0;
```

```
        x = (x > 0) ? x / 2 : x;
```

```
        b[i] = (y > 0) ? y % 2 : 0;
```

```
        y = (y > 0) ? y / 2 : y;
```

```
        i++;
```

```
}
```

```
int n = i;
```

```
int bc[50];
```

```
// Calculate two's complement of the divisor
```

```
for (i = 0; i < n; i++) {
```

```
    bc[i] = (b[i] == 0) ? 1 : 0;
```

```
}
```

```
bc[n] = 1;
```

```
// Add 1 to the two's complement of the divisor
```

```
for (i = 0; i <= n; i++) {
```

```
    if (bc[i] == 0) {
```

```
        bc[i] = 1;
```

```
        break;
```

```
    } else {
```

```
        bc[i] = 0;
```

```
    }
```

```
}
```

```
// Perform restoring division
```

```
for (i = n; i != 0; i--) {
```

```
    // Shift accumulator left
```

```
    for (int j = n; j > 0; j--) {
```

```
        acum[j] = acum[j - 1];
```

```
    }
```

```
    acum[0] = q[n - 1];
```

```
    // Shift quotient left
```

```
    for (int j = n - 1; j > 0; j--) {
```

```
        q[j] = q[j - 1];
```

```
    }
```

```

// Subtract divisor
add(acum, bc, n + 1);

if (acum[n] == 1) { // If accumulator is negative
    q[0] = 0;
    add(acum, b, n + 1); // Restore the accumulator
} else {
    q[0] = 1; // Set quotient bit to 1
}
}

// Output quotient and remainder
printf("\nQuotient: ");
for (int l = n - 1; l >= 0; l--) {
    printf("%d", q[l]);
}

printf("\nRemainder: ");
for (int l = n; l >= 0; l--) {
    printf("%d", acum[l]);
}

return 0;
}

```

```

// Function to add two binary numbers
void add(int acum[], int bo[], int n) {
    int temp = 0;
    for (int i = 0; i < n; i++) {
        int sum = acum[i] + bo[i] + temp;
        if (sum == 0) {
            acum[i] = 0;
            temp = 0;
        } else if (sum == 1) {

```

```

    acum[i] = 1;

    temp = 0;

} else if (sum == 2) {

    acum[i] = 0;

    temp = 1;

} else if (sum == 3) {

    acum[i] = 1;

    temp = 1;

}

}

}

```

## INPUT & OUTPUT:

The screenshot displays the Dev-C++ IDE with a C program named `dummy.c` open. The code implements a binary division algorithm. The console window shows the program's execution output.

```

1  #include <stdio.h>
2
3  int acum[100] = {0};
4  int q[100], b[100];
5
6  // Function to add two binary numbers
7  void add(int acum[], int b[], int n);
8
9  int main() {
10     int x, y, i = 0;
11
12     // Input the dividend and divisor
13     printf("Enter the Number (dividend and divisor): ");
14     scanf("%d%d", &x, &y);
15
16     // Convert the dividend and divisor to binary
17     while (x > 0 || y > 0) {
18         q[i] = (x > 0) ? x % 2 : 0;
19         x = (x > 0) ? x / 2 : x;
20         b[i] = (y > 0) ? y % 2 : 0;
21         y = (y > 0) ? y / 2 : y;
22         i++;
23     }
24
25     int n = i;
26     int bc[50];
27
28     // Calculate two's complement of the divisor
29     for (i = 0; i < n; i++) {
30         bc[i] = (b[i] == 0) ? 1 : 0;
31     }
32     bc[n] = 1;

```

Console Output:

```

Enter the Number (dividend and divisor): 14 3
Quotient: 0100
Remainder: 00010
-----
Process exited after 1.755 seconds with return value 0
Press any key to continue . . .

```

The bottom status bar of the IDE shows: Line: 102 Col: 1 Sel: 0 Lines: 102 Length: 2377 Insert Done p.

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