Year: B.Tech II, Computer Organization Lab

***Batch (A1)***

Practical: **02**

Problem Statements

1. To declare the native data type, check the size of them and print the values if given beyond the range.

Code:

*//To declare the native data type,*

*// check the size of them and*

*// print the values if given beyond the range.*

*#include* <stdio.h>

int main()

{

    long long int x;

    printf("Enter Integer :\n");

    scanf("%lld", &x);

*if* (x >= -2147483648 && x <= 2147483647)

    {

        printf("Valid Input!\n");

        printf("The Size of Integer is %d\n", sizeof(int));

    }

*else*

    {

        printf("Invalid Input!\n");

        printf("Value Entered is Out of Range of Data Type\n");

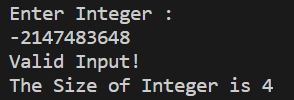
    }

*return* 0;

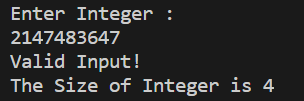
}

Sample Test Cases:

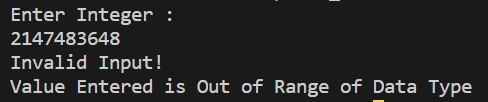
1.) [-2147483648]



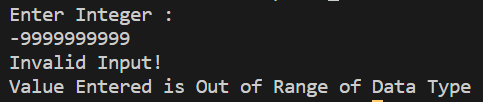
2.) [2147483647]



3.) [2147483648]



4.) [-9999999999]



2. To perform all arithmetic operations of two numbers given from the command line.

Code:

*//To perform all arithmetic operations of two numbers*

*// given from the command line.*

*#include* <stdio.h>

*#include* <stdlib.h>

*#include* <string.h>

int main(int argc, char \*argv[])

{

    int a, b;

*if* (argc < 3)

    {

        printf("Enter Two Arguments!\n");

*return* -1;

    }

    a = atoi(argv[1]);

    b = atoi(argv[2]);

    int add = a + b;

    printf("a + b = %d\n", add);

    int sub = a - b;

    printf("a - b = %d\n", sub);

    int mul = (a \* b);

    printf("a \* b = %d\n", mul);

*if* (b != 0)

    {

        int ans = (a / b);

        printf("a / b = %d\n", ans);

    }

*else*

    {

        printf("Divide by Zero Error!\n");

    }

*if* (b != 0)

    {

        int ans1 = a % b;

        printf("a %% b = %d", ans1);

    }

*else*

    {

        printf("Modulo by Zero Error!");

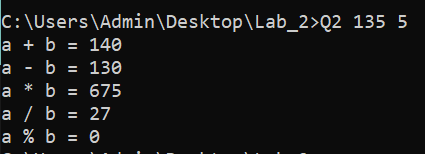
    }

*return* 0;

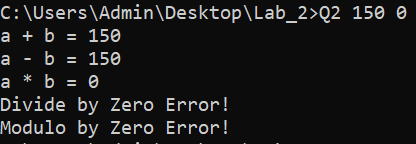
}

Sample Test Cases:

1.) [135, 5]



2.) [150, 0]



3.) Insufficient Inputs from User



3. To perform all arithmetic operations of two numbers given from the command line, but using qualifiers. Using Qualifier “const”.

Code:

*//To perform all arithmetic operations of two numbers*

*// given from the command line, but using qualifiers.*

*#include* <stdio.h>

*#include* <stdlib.h>

*#include* <string.h>

int main(int argc, char \*argv[])

{

*if* (argc > 1)

    {

        printf("Constants Value Cant Be Modified!\n");

        printf("Enter 0 Arguments Only!");

*return* -1;

    }

*// Using Qualifier*

    const double a = 250.800, b = 125.400;

    printf("a = %lf\n", a);

    printf("b = %lf\n", b);

    const double add = a + b;

    printf("a + b = %lf\n", add);

    const double sub = a - b;

    printf("a - b = %lf\n", sub);

    const double mul = (a \* b);

    printf("a \* b = %lf\n", mul);

*if* (b != 0)

    {

        const double ans = (a / b);

        printf("a / b = %lf\n", ans);

    }

*else*

    {

        printf("Divide by Zero Error!\n");

    }

    printf("Since Modulus Operator Works only for Integer Numbers\nEg : [1000, 3]\n");

    const long long int num1 = 1000, num2 = 3;

*if* (num2 != 0)

    {

        const long long int ans1 = num1 % num2;

        printf("num1 %% num2 = %lld", ans1);

    }

*else*

    {

        printf("Modulo by Zero Error!");

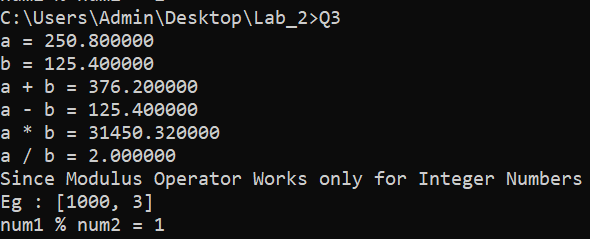
    }

*return* 0;

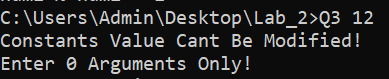
}

Sample Test Cases:

1.) [] ~ No Inputs Required



2.) Constants Value can’t be Modified Error in Case User Inputs Data



4. To display different formatting of floating point numbers.

Code:

*//To display different formatting of floating point numbers.*

*#include*<stdio.h>

int main()

{

    float num;

    printf("Enter a Number(Float Data Type) : \n");

    scanf("%f", &num);

*// 1 digits*

    printf("1 Digit Float Form : %0.1f\n", num);

*// 2 digits*

    printf("2 Digit Float Form : %0.2f\n", num);

*// 3 digits*

    printf("3 Digit Float Form : %0.3f\n", num);

*// 4 digits*

    printf("4 Digit Float Form : %0.4f\n", num);

*// 5 digits*

    printf("5 Digit Float Form : %0.5f\n", num);

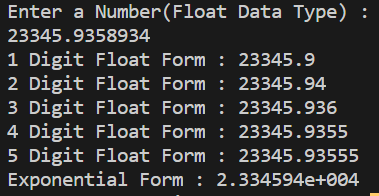
*// Exponential Form*

    printf("Exponential Form : %e\n", num);

*return* 0;

}

Sample Test Cases:



5. Perform Addition and Subtraction of two signed binary numbers given from command line.

Code:

*//Perform Addition and Subtraction of two signed binary numbers given from command line.*

*#include* <stdio.h>

*#include* <stdlib.h>

*#include* <string.h>

int fromBinary(const char \*s)

{

*return* (int)strtol(s, NULL, 2);

}

int main(int argc, char \*argv[])

{

    int num1, num2, bin\_add, bin\_sub;

*if* (argc < 3)

    {

        printf("Enter Two Arguments!\n");

*return* -1;

    }

    num1 = fromBinary(argv[1]);

    num2 = fromBinary(argv[2]);

    bin\_add = num1 + num2;

    bin\_sub = num1 - num2;

    printf("Binary Addition Result : %d\n", bin\_add);

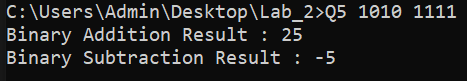
    printf("Binary Subtraction Result : %d\n", bin\_sub);

*return* 0;

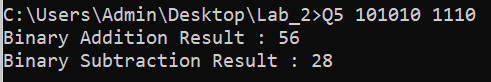
}

Sample Test Cases:

1.) [10 (1010), 15(1111)]



2.) [42 (101010), 14 (1110)]



6. Perform Multiplication for unsigned binary number.

Code:

*//Perform Multiplication for unsigned binary number*

*#include* <stdio.h>

*#include* <stdlib.h>

*#include* <string.h>

int fromBinary(const char \*s)

{

*return* (int)strtol(s, NULL, 2);

}

int main()

{

    char num1[32], num2[32];

    int n1, n2;

    printf("Input Number 1: ");

    scanf("%s", num1);

    n1 = fromBinary(num1);

    printf("Input Number 2: ");

    scanf("%s", num2);

    n2 = fromBinary(num2);

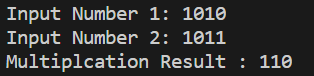
    printf("Multiplcation Result : %d\n", (n1 \* n2));

*return* 0;

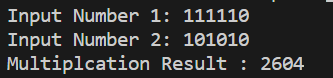
}

Sample Test Cases:

1.) [10(1010), 11(1011)]



2.) [62(111110), 42(101010)]



Submitted By:

Roll Number: U19CS012 (D-12)

Name: Bhagya Rana