Assignment 1

IEEE754 64 bit

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

int checkSign(double *number)
{
    double check = *number;
    if (check < 0) {
        *number *= -1;
        return 1;
    } else {
        return 0;
    }
}

int binary(int number, int reversedBits[], int size) {

    int index = 0;
    int check;
    int bits[1000];

    if (number == 0) {
        bits[index++] = 0;
        return 0;
    } else {
        while (number > 0) {
            check = number % 2;
            if (check == 1) {
                bits[index++] = 1;
            } else {
                bits[index++] = 0;
            }
            number /= 2;
        }
    }

    while (number > 0) {
        check = number % 2;
        if (check == 1) {
           bits[index++] = 1;
        } else {
           bits[index++] = 0;
        }
        number /= 2;
    }

    if (size != 0 && index != 11) {
        for (int i = 0; i < size; i++) {
            reversedBits[i] = bits[(index) - i];
        }
    } else {
        for (int i = 0; i < index; i++) {
            reversedBits[i] = bits[(index - 1 ) - i];
        }
```

```
    }

    return index;
}

int decimalBinary(double number, int bits[], int start) {

    double decimalPart = number;
    double integerpart;

    int index = 0;

    if (start != 0)
    {
        index = start-1;
    }

    if (decimalPart == 0) {
        bits[index] = 0;
        return 0;
    }

    while (index < 1000) {
        decimalPart *= 2;
        if (decimalPart < 1) {
            bits[index] = 0;
        } else if (decimalPart > 1) {
            decimalPart = modf(decimalPart, &integerpart);
            bits[index] = 1;
        } else {
            bits[index] = 1;
            index++;
            break;
        }
        index++;
    }

    return index;
}

void putBit(int integerBit[], int decimalBit[], int indexInt) {

    for (int i = 0; i < indexInt-1; i++) {
        decimalBit[i] = integerBit[(i+1)];
    }
}

int searchBits(int decimalbit[]) {
    int index = 0;
    while (1) {
        if (decimalbit[index] == 1){
            index++;
            return index;
        }
        index++;
    }
    return index;
}

void printAllBinary(int sign, int exponent[], int mantissa[], int index) {

    printf("%d ", sign);
    for (int i = 0; i < 11; i++) {
        printf("%d", exponent[i]);
    }
    printf(" ");
    for (int i = index; i < 52 + index; i++) {
        printf("%d", mantissa[i]);
    }

}

int main() {

    double number;

    printf("Enter number : ");
    scanf("%lf", &number);

    int sign;  //Sign Bits
    int mantissa[1000] = {0};
```

```
    int exponent[1000] = {0};
    int indexOneBits = 0;

    if (number == 0) {
        sign = 0;
        printAllBinary(sign, exponent, mantissa, indexOneBits);
    } else if (number == -0) {
        sign = 1;
        printAllBinary(sign, exponent, mantissa, indexOneBits);
    } else {

        sign = checkSign(&number);

        double integerPart;
        double decimalPart;

        decimalPart = modf(number, &integerPart);

        int e;
        int integerBits[1000] = {0};
        int indexInt; //เก็บจำนวนหลักของ integerpart

        indexInt = binary(integerPart, integerBits, 0);


        if (integerPart > 0) {
            putBit(integerBits, mantissa, indexInt);
            decimalBinary(decimalPart, mantissa, indexInt);
            e = (indexInt - 1) + 1023;
        } else {
            decimalBinary(decimalPart, mantissa, indexInt);
            indexOneBits = searchBits(mantissa);
            e = (indexOneBits * -1) + 1023;
        }

        binary(e, exponent, 11);
        printAllBinary(sign, exponent, mantissa, indexOneBits);
    }

      return 0;
}
```

Input/Output

```
Enter number : 12.45
0 10000000010 1000111001100110011001100110011001100110011001100110011001100110

Enter number : -100.25
1 10000000101 1001000100000000000000000000000000000000000000000000000000000000

Enter number : 0.005
0 01111110111 0100011110101110000101000111101011100001010001111011
```

65070501061 อาทิตย์ นพจรูญศรี

อ้างอิงจาก

https://www.binaryconvert.com/convert_double.html?decimal=049050051052053054055056057048049050051052053046049050051052053054055056057049050051052053054055