1. Написав програму котра ілюструє роботі всіх видів операторів

1

```c
#include <stdio.h>

// Функція для перетворення десяткового числа в двійковий формат
void decimalToBinary(unsigned int num, char *binaryStr) {
    for (int i = 0; i < 32; i++) {
        binaryStr[31 - i] = (num & (1 << i)) ? '1' : '0';
    }
    binaryStr[32] = '\0'; // Завершення рядка
}

int main()
{
    int str;
    printf("Choose the type of operator (arithmetical(1), logical(2), by bit(3)): ");
    scanf("%d", &str);

    switch (str) {
        case 1: {
            double num1, num2, result;
            int arithmeticalOperation;
            printf("Choose the type of operator: (+(1),-(2),/(3),*(4),%(5)): ");
            scanf("%d", &arithmeticalOperation);
            printf("Select two operands: ");
            scanf("%lf", &num1);
            scanf("%lf", &num2);


            switch (arithmeticalOperation) {
                case 1:  // '+' operation
                    result = num1 + num2;
                    printf("%.2lf + %.2lf = %.2lf\n", num1, num2, result);
                    break;
                case 2:  // '-' operation
                    result = num1 - num2;
                    printf("%.2lf - %.2lf = %.2lf\n", num1, num2, result);
                    break;
                case 3:  // '*' operation
                    result = num1 * num2;
                    printf("%.2lf * %.2lf = %.2lf\n", num1, num2, result);
                    break;
                case 4:  // '/' operation
                    if (num2 != 0) {
                        result = num1 / num2;
                        printf("%.2lf / %.2lf = %.2lf\n", num1, num2, result);
                    } else {
                        printf("Error! Division by zero is not allowed.\n");
                    }
                    break;


                default:
                    printf("Error! Invalid operator.\n");
            }
            break;
        }
        case 2: {
            double num3, num4;
            int logicalOperation;
            printf("Choose the type of operator: (&&(1), ||(2), !(3)): ");
            scanf("%d", &logicalOperation);
            printf("Select two operands: ");
            scanf("%lf", &num3);
            scanf("%lf", &num4);

            switch (logicalOperation) {
                case 1:  // Logical AND
                    if (num3 > 0 && num4 > 0) {
                        printf("Both num3 and num4 are positive numbers.\n");
                    }
                    break;
                case 2:  // Logical OR
                    if (num3 > 0 || num4 > 0) {
                        printf("At least one of num3 or num4 is positive.\n");
                    }
```

```
                    }
                    break;
                case 3:  // Logical NOT
                    if (!num3) {
                        printf("num3 is zero (logical NOT of
                            num3).\n");
                    }
                    break;
                default:
                    printf("Error! Invalid operator.\n");
            }
            break;
        }
        case 3: {
            unsigned int num1, num2, result;
            char binaryStr1[33], binaryStr2[33];

            printf("Select two decimal numbers:\n");
            printf("Enter first decimal number: ");
            scanf("%u", &num1);
            printf("Enter second decimal number: ");
            scanf("%u", &num2);

            // Перетворення в двійковий формат
            decimalToBinary(num1, binaryStr1);
            decimalToBinary(num2, binaryStr2);


            printf("Binary representation of first number: %s\n"
                , binaryStr1);
            printf("Binary representation of second number:
                %s\n", binaryStr2);

            int bitwiseOperation;
            printf("Choose the type of bitwise operator: (AND(1
                ), OR(2), XOR(3), NOT(4), LSHIFT(5), RSHIFT(6)):
                ");
            scanf("%d", &bitwiseOperation);

            switch (bitwiseOperation) {
                case 1:  // Bitwise AND
                    result = num1 & num2;
                    printf("num1 AND num2: %u (Binary: ", result
                        );
                    decimalToBinary(result, binaryStr1);
                    printf("%s)\n", binaryStr1);
                    break;
                case 2:  // Bitwise OR
                    result = num1 | num2;
                    printf("num1 OR num2: %u (Binary: ", result
                        );
                    decimalToBinary(result, binaryStr1);
                    printf("%s)\n", binaryStr1);
                    break;
                case 3:  // Bitwise XOR

                    result = num1 ^ num2;
                    printf("num1 XOR num2: %u (Binary: ", result
                        );
                    decimalToBinary(result, binaryStr1);
                    printf("%s)\n", binaryStr1);
                    break;
                case 4:  // Bitwise NOT
                    result = ~num1;
                    printf("NOT num1: %u (Binary: ", result);
                    decimalToBinary(result, binaryStr1);
                    printf("%s)\n", binaryStr1);
                    break;
                case 5:  // Left Shift
                    result = num1 << 1; // Зсув вліво на 1
                    printf("num1 << 1: %u (Binary: ", result);
                    decimalToBinary(result, binaryStr1);
                    printf("%s)\n", binaryStr1);
                    break;
                case 6:  // Right Shift
                    result = num1 >> 1; // Зсув вправо на 1
                    printf("num1 >> 1: %u (Binary: ", result);
                    decimalToBinary(result, binaryStr1);
                    printf("%s)\n", binaryStr1);
                    break;
                default:
                    printf("Error! Invalid operator.\n");
            }
            break;
```

```
                    printf("NOT num1: %u (Binary: ", result);
                    decimalToBinary(result, binaryStr1);
                    printf("%s)\n", binaryStr1);
                    break;
                case 5:  // Left Shift
                    result = num1 << 1; // Зсув вліво на 1
                    printf("num1 << 1: %u (Binary: ", result);
                    decimalToBinary(result, binaryStr1);
                    printf("%s)\n", binaryStr1);
                    break;
                case 6:  // Right Shift
                    result = num1 >> 1; // Зсув вправо на 1
                    printf("num1 >> 1: %u (Binary: ", result);
                    decimalToBinary(result, binaryStr1);
                    printf("%s)\n", binaryStr1);
                    break;
                default:
                    printf("Error! Invalid operator.\n");
            }
            break;
        }
        default:
            printf("Error! Invalid operator.\n");
    }

    return 0;
}

```

2.написав програму з введеням числа та виведенням її адреси та значення через вказівнк

```
#include <stdio.h>

int main() {
    int number;
    int *pointer = &number;


    printf("Enter an integer: ");
    scanf("%d", &number);

    printf("The value of the variable 'number': %d\n", *pointer);
    printf("The address of the variable 'number': %p\n", (void *)pointer);

    return 0;
}

```

3.Написав програму розв’язку квадратного рівняння з коефіцієнтами які вводяться з консолі

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

int main() {
    double a, b, c;
    double discriminant, root1, root2, realPart, imaginaryPart;


    printf("Enter coefficients a, b and c: ");
    scanf("%lf %lf %lf", &a, &b, &c);


    if (a == 0) {
        printf("Coefficient 'a' cannot be zero in a quadratic
            equation.\n");
        return 1;
    }


    discriminant = b * b - 4 * a * c;


    if (discriminant > 0) {

        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("Roots are real and different.\n");
        printf("Root 1 = %.2lf\n", root1);
```

```c
    if (discriminant > 0) {

        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
        printf("Roots are real and different.\n");
        printf("Root 1 = %.2lf\n", root1);
        printf("Root 2 = %.2lf\n", root2);
    } else if (discriminant == 0) {

        root1 = -b / (2 * a);
        printf("Roots are real and the same.\n");
        printf("Root = %.2lf\n", root1);
    } else {
       i
        realPart = -b / (2 * a);
        imaginaryPart = sqrt(-discriminant) / (2 * a);
        printf("Roots are complex and different.\n");
        printf("Root 1 = %.2lf + %.2lfi\n", realPart,
            imaginaryPart);
        printf("Root 2 = %.2lf - %.2lfi\n", realPart,
            imaginaryPart);
    }

    return 0;
}
```