

КПІ ім. Ігоря Сікорського
Факультет інформатики та обчислювальної
техніки
Кафедра інформатики та програмної інженерії

Звіт до комп'ютерного практикуму з курсу
“Основи програмування”

Прийняв
асистент кафедри ІІІ
Пархоменко А. В.
23.12.2022 р.

Виконала
студентка групи ІІІ-22
Андрєєва У.А.

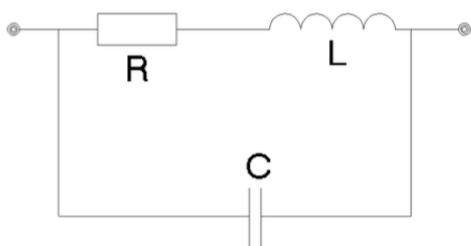
Київ 2022

Комп'ютерний практикум №8

Тема: Структури.

Завдання: Написати програму для обчислення комплексного опору заданого коливального контуру в залежності від частоти струму. Варіанти коливальних контурів наведені нижче:

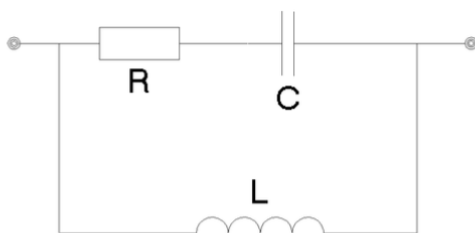
а)



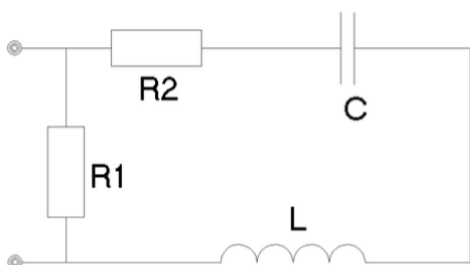
$$Z = \frac{\frac{L}{C} - i \frac{R}{\omega C}}{R + i(\omega L - \frac{1}{\omega C})}$$

б)

$$Z = \frac{\frac{L}{C} + i \frac{R}{\omega C}}{R + i(\omega L - \frac{1}{\omega C})}$$

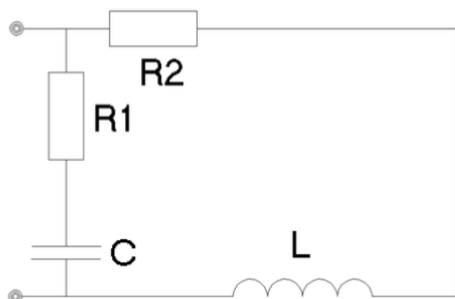


в)



$$Z = \frac{R_1 R_2 + i R_1 (\omega L - \frac{1}{\omega C})}{R_1 + R_2 + i(\omega L - \frac{1}{\omega C})}$$

з)



$$Z = \frac{R_1 R_2 + \frac{L}{C} + i(\omega L R_1 - \frac{R_2}{\omega C})}{R_1 + R_2 + i(\omega L - \frac{1}{\omega C})}$$

Тут $i = \sqrt{-1}$, кутова частота $\omega = 2\pi f$.

За вказівкою викладача визначається контур та його параметри R_1 , R_2 (Ом), L (мГн), C (мкФ).

Текст програми:

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

//typedef double;
typedef struct complexOptions {
    double realComponent;
    double complexComponent;
} complexOpt;
double checkSymbols(char *enterParameter);
void solve(complexOpt (*choice) (double, double, double, double, double), double R1, double R2,
double L, double C, double fMin, double fMax, double df);
complexOpt schemeFirstOpt (double R1, double R2, double L, double C, double omegas);
complexOpt schemeSecondOpt (double R1, double R2, double L, double C, double omegas);
complexOpt schemeThirdOpt (double R1, double R2, double L, double C, double omegas);
complexOpt schemeForthOpt (double R1, double R2, double L, double C, double omegas);

int main(){
    double R1, R2, L, C, fMin, fMax, df;
    short choice;

    printf("\tPROGRAM TO SOLVE COMPLEX RESISTANCE\n");
    R1 = checkSymbols("R1");
    R2= checkSymbols("R2");
    L= checkSymbols("L");
    C= checkSymbols("C");
    fMin= checkSymbols("fMin");
    fMax= checkSymbols("fMax");
    df= checkSymbols("df");

    do {
        printf("\n-----SELECT_YOUR_CHOICE-----\n");
        printf("\n\t1)Solving first scheme \n\t2)Solving second scheme \n\t3)Solving third scheme ");
        printf("\n\t4)Solving fourth scheme \n\t5)EXIT FROM APP");
        printf("\n-----\n");
        printf("\nEnter your choice please(1 to 4) :");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                solve(schemeFirstOpt, R1, R2, L, C, fMin, fMax, df);
                break;
            case 2:
                solve(schemeSecondOpt, R1, R2, L, C, fMin, fMax, df);
                break;
            case 3:
                solve(schemeThirdOpt, R1, R2, L, C, fMin, fMax, df);
```

```

        break;
    case 4:
        solve(schemeForthOpt, R1, R2, L, C, fMin, fMax, df);
        break;
    case 5:
        exit(0);
    default:
        printf("not an appropriate choice");
    }
} while (choice != 5);
}

double checkSymbols(char *enterParameter){
    char ch,error;
    double value;
    do {
        error = 0;
        printf("Please enter %s ->",enterParameter);
        scanf("%lf%c", &value, &ch);

        if(((ch) != '\n')||(value<=0)) {
            printf("Invalid data\n");
            error = 1;
            fflush(stdin);
        }
    } while(error);
    return value;
}

complexOpt division(complexOpt n1, complexOpt n2) { // division of two complex numbers
    complexOpt n;
    n.realComponent = (n1.realComponent * n2.realComponent + n1.complexComponent *
n2.complexComponent) / (n2.realComponent * n2.realComponent + n2.complexComponent *
n2.complexComponent);
    n.complexComponent = (n1.complexComponent * n2.realComponent - n1.realComponent *
n2.complexComponent) / (n2.realComponent * n2.realComponent + n2.complexComponent *
n2.complexComponent);
    return n;
}

complexOpt schemeFirstOpt (double R1, double R2, double L, double C, double omegas) {
    complexOpt n1, n2;
    n1.realComponent = L/C;
    n1.complexComponent = (-1) * R1 / (omegas * C);
    n2.realComponent = R1;
    n2.complexComponent = omegas * L - 1 / (omegas * C);
    return division(n1, n2);
}

complexOpt schemeSecondOpt (double R1, double R2, double L, double C, double omegas) {
    complexOpt n1, n2;
    n1.realComponent = L/C;

```

[illegible]

Введенні та одержані результати

1. Перша схема

```
Laba8 main.c
CMakeLists.txt main.c
Run: Laba8 x
PROGRAM TO SOLVE COMPLEX RESISTANCE
Please enter R1 -> 1
Please enter R2 -> 100
Please enter L -> 1
Please enter C -> 1
Please enter fMin -> 2
Please enter fMax -> 8
Please enter df -> 1

-----SELECT_YOUR_CHOICE-----

1)Solving first scheme
2)Solving second scheme
3)Solving third scheme
4)Solving fourth scheme
5)EXIT FROM APP
-----

Enter your choice please(1 to 4) : 1

|Frequency Value|      |Resonance frequency|      |Resistance|
2.00              9.188815e-02      1.333895e-05 - i*7.974474e-02
3.00              9.188815e-02      2.637950e-06 - i*5.310133e-02
4.00              9.188815e-02      8.350069e-07 - i*3.980971e-02
5.00              9.188815e-02      3.420838e-07 - i*3.184173e-02
6.00              9.188815e-02      1.649880e-07 - i*2.653204e-02
7.00              9.188815e-02      8.906196e-08 - i*2.274034e-02
8.00              9.188815e-02      5.220860e-08 - i*1.989699e-02
```

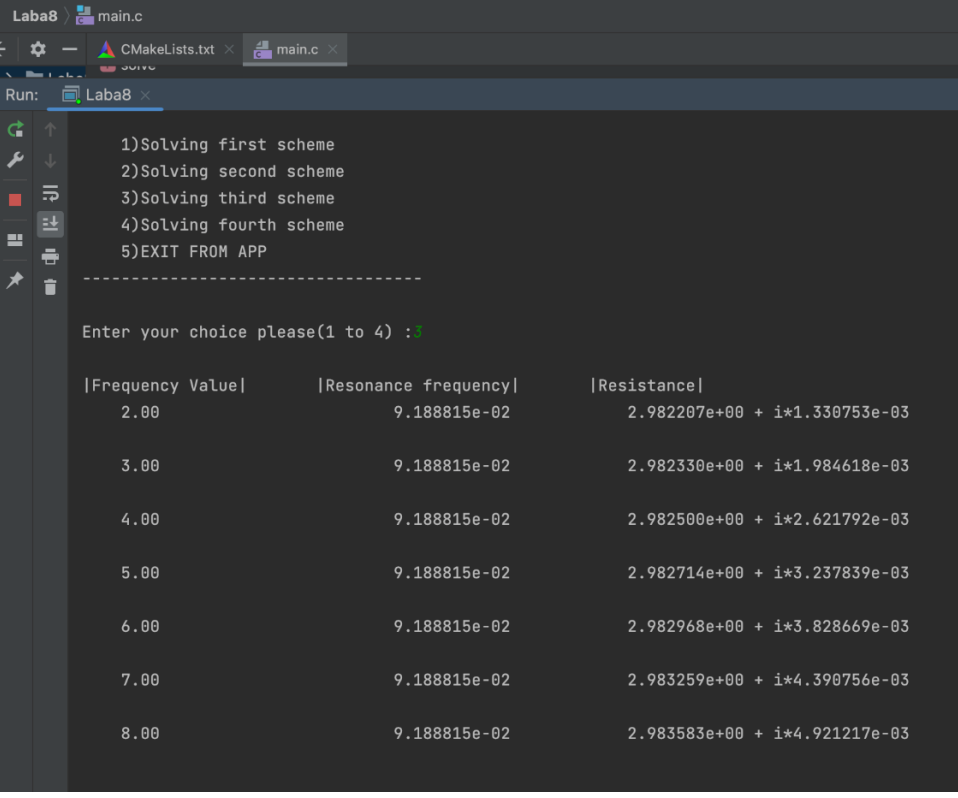
2. Друга схема

```
Laba8 main.c
CMakeLists.txt main.c
Run: Laba8 x
1)Solving first scheme
2)Solving second scheme
3)Solving third scheme
4)Solving fourth scheme
5)EXIT FROM APP
-----

Enter your choice please(1 to 4) : 1

|Frequency Value|      |Resonance frequency|      |Resistance|
2.00              9.188815e-02      1.262507e-02 - i*7.873901e-02
3.00              9.188815e-02      5.621036e-03 - i*5.280298e-02
4.00              9.188815e-02      3.163781e-03 - i*3.968379e-02
5.00              9.188815e-02      2.025397e-03 - i*3.177725e-02
6.00              9.188815e-02      1.406744e-03 - i*2.649472e-02
7.00              9.188815e-02      1.033622e-03 - i*2.271683e-02
8.00              9.188815e-02      7.914151e-04 - i*1.988125e-02
```

3. Третья схема



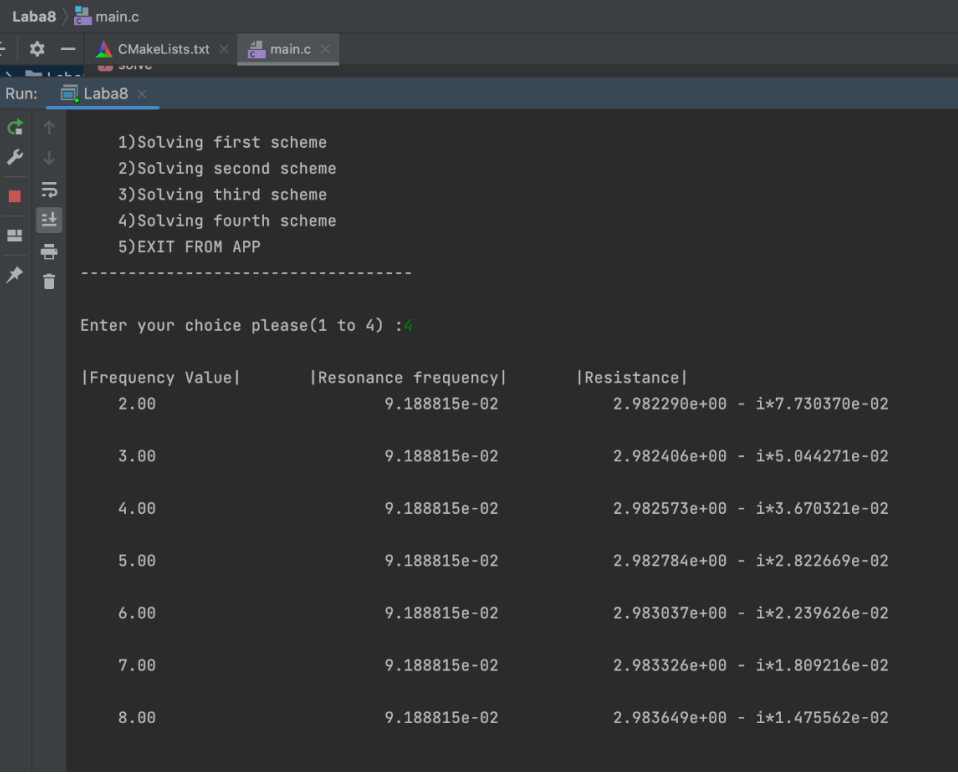
```
1)Solving first scheme
2)Solving second scheme
3)Solving third scheme
4)Solving fourth scheme
5)EXIT FROM APP

-----

Enter your choice please(1 to 4) : 3

|Frequency Value|      |Resonance frequency|      |Resistance|
2.00              9.188815e-02      2.982207e+00 + i*1.330753e-03
3.00              9.188815e-02      2.982330e+00 + i*1.984618e-03
4.00              9.188815e-02      2.982500e+00 + i*2.621792e-03
5.00              9.188815e-02      2.982714e+00 + i*3.237839e-03
6.00              9.188815e-02      2.982968e+00 + i*3.828669e-03
7.00              9.188815e-02      2.983259e+00 + i*4.390756e-03
8.00              9.188815e-02      2.983583e+00 + i*4.921217e-03
```

4. Четвертая схема



```
1)Solving first scheme
2)Solving second scheme
3)Solving third scheme
4)Solving fourth scheme
5)EXIT FROM APP

-----

Enter your choice please(1 to 4) : 4

|Frequency Value|      |Resonance frequency|      |Resistance|
2.00              9.188815e-02      2.982290e+00 - i*7.730370e-02
3.00              9.188815e-02      2.982406e+00 - i*5.044271e-02
4.00              9.188815e-02      2.982573e+00 - i*3.670321e-02
5.00              9.188815e-02      2.982784e+00 - i*2.822669e-02
6.00              9.188815e-02      2.983037e+00 - i*2.239626e-02
7.00              9.188815e-02      2.983326e+00 - i*1.809216e-02
8.00              9.188815e-02      2.983649e+00 - i*1.475562e-02
```

Висновок:

Отже, я написала програму для знаходження комплексного опору заданого коливального контуру в залежності від частоти струму. Для різних коливальних контурів, застосовуючи структуру як опис типу `complexOptions` з двома елементами. Програма працює коректно та виводить відповідь в експоненціальній формі.