Задание 1. Разработать программу на языке C++ для следующих заданий:

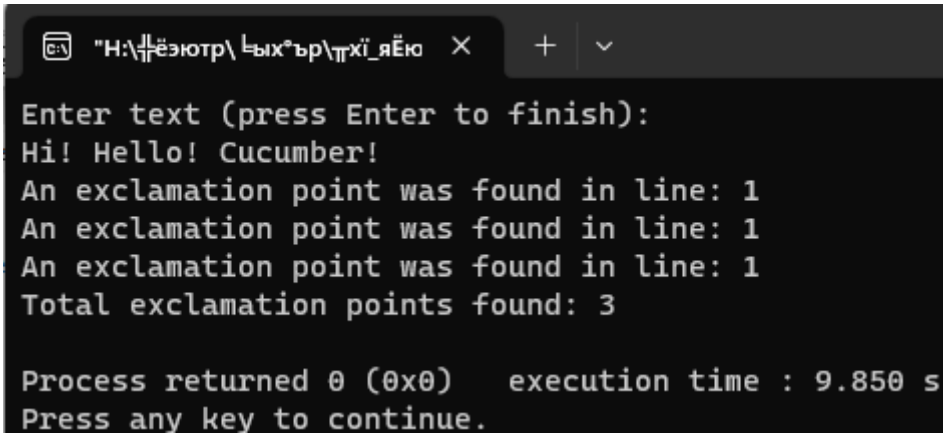
Условие задачи:

Определить содержит ли текст хотя бы один восклицательный знак и в какой строке.

Код программы:

```
#include <iostream>
#include <string>

using namespace std;

int main() {
    string text;
    int lineNumber = 1;

    cout << "Enter text (press Enter to finish):\n";
    getline(cin, text);

    int length = text.length();
    int exclamationCount = 0;  // Переменная для подсчета восклицательных знаков

    for (int i = 0; i < length; i++) {
        if (text[i] == '!') {
            cout << "An exclamation point was found in line: " << lineNumber <<
endl;
            exclamationCount++;
        }
        else if (text[i] == '\n') {
            lineNumber++;
        }
    }

    if (exclamationCount > 0) {
        cout << "Total exclamation points found: " << exclamationCount << endl;
    } else {
        cout << "No exclamation point found." << endl;
    }

    return 0;
}
```

Результат тестирования программы:



Задание 2. Структуры. Операции над комплексными числами

Условие задачи:

Даны комплексные числа

$$a = \alpha + \beta * i, b = \gamma + \sigma * i, c = \lambda + \mu * i$$

Найти комплексное число

$$d = \phi + \psi * i$$

по формуле представленной ниже.

$$d = (a * b^3 - c)/(a + b)$$

Код программы:

```
#include <iostream>

using namespace std;

struct Complex {
    double real;
    double imag;
};

// Функция сложения
Complex add(Complex a, Complex b) {
    Complex result;
    result.real = a.real + b.real;
    result.imag = a.imag + b.imag;
    return result;
}

// Функция вычитания
Complex subtract(Complex a, Complex b) {
    Complex result;
```

```
    result.real = a.real - b.real;
    result.imag = a.imag - b.imag;
    return result;
}

// Функция умножения
Complex multiply(Complex a, Complex b) {
    Complex result;
    result.real = a.real * b.real - a.imag * b.imag;
    result.imag = a.real * b.imag + a.imag * b.real;
    return result;
}

// Функция деления
Complex divide(Complex a, Complex b) {
    Complex result;
    double denominator = b.real * b.real + b.imag * b.imag;
    if (denominator != 0.0) {
        result.real = (a.real * b.real + a.imag * b.imag) / denominator;
        result.imag = (a.imag * b.real - a.real * b.imag) / denominator;
    }
    else
    {
        cout << "Error: Division by zero!" << "\n";
    }
    return result;
}

// Функция счёта куба
Complex cube(Complex num) {
    Complex result;
    result = multiply(multiply(num, num), num);
    return result;
}

int main() {
    double alpha, beta, gamma, sigma, lambda, mu;
    cout << "Enter the real part of the complex number a: ";
    cin >> alpha;
    cout << "Enter the imaginary part of the complex number a: ";
    cin >> beta;

    cout << "Enter the real part of the complex number b: ";
    cin >> gamma;
    cout << "Enter the imaginary part of the complex number b: ";
    cin >> sigma;

    cout << "Enter the real part of the complex number c: ";
    cin >> lambda;
    cout << "Enter the imaginary part of the complex number c: ";
    cin >> mu;

    Complex a = {alpha, beta};
    Complex b = {gamma, sigma};
    Complex c = {lambda, mu};
```

```
    Complex cubeB = cube(b);
    Complex product = multiply(a, cubeB);
    Complex sum = add(a, b);
    Complex result = divide(subtract(product, c), sum);

    if (sum.real != 0.0 || sum.imag != 0.0)
    {
        cout << "Solution for (a * b^3 - c) / (a + b): " << result.real << " + " << result.imag << "i" << "\n";
    }

    return 0;
}
```

Результат тестирования программы:

```
C:\Users\┬ырфшьшЁ\Desktoj

Enter the real part of the complex number a: 1
Enter the imaginary part of the complex number a: 2
Enter the real part of the complex number b: 3
Enter the imaginary part of the complex number b: 4
Enter the real part of the complex number c: 5
Enter the imaginary part of the complex number c: 6
Solution for (a * b^3 - c) / (a + b): -38.7692 + 9.15385i

Process returned 0 (0x0)   execution time : 3.185 s
Press any key to continue.
```

$$\frac{(1+2\,i)\,(3+4\,i)^3-(5+6\,i)}{(1+2\,i)+(3+4\,i)}$$

Result

$$-\frac{7}{13}\,(72-17\,i)$$

Decimal approximation

$$-38.7692307692307692307692307692307692307692307692307692307\ldots + 9.1538461538461538461538461538461538461538461538461538461538461\ldots\,i$$

Задание 3. Работа с библиотекой комплексных чисел

Условие задачи:

Для заданных матриц комплексных чисел A(nxn) и B(mxm) найти

$$C = (\Delta * A - A^T) * A$$

где

$$\Delta = |B|$$

Вычислить

$$C^-1$$

Код программы:

```
#include <iostream>
#include <fstream>
#include <math.h>
#include <complex>
#include <memory.h>
using namespace std;

// Значение нуля при вычислениях
const double eps = 1E-12;

complex<double>** mul_mat(complex<double>** mat1, complex<double>** mat2, int N)
{
    complex<double>** res = new complex<double>*[N];
    int i, j, k;

    // Create the result matrix
    for (i = 0; i < N; ++i)
    {
        res[i] = new complex<double>[N];
    }

    // Perform matrix multiplication
    for (i = 0; i < N; ++i)
    {
        for (j = 0; j < N; ++j)
        {
            complex<double> v = 0.0;
            for (k = 0; k < N; ++k)
            {
                v += mat1[i][k] * mat2[k][j];
            }
            res[i][j] = v;
        }
    }

    return res;
}
```

```
//Вычисление определителя матрицы
//с комплексными коэффициентами
complex<double> determinant(complex<double> **matrica_a, int n)
{
    int i, j, k, r;
    complex<double> c, M, s, det = 1;
    complex<double> **a;
    double max;
    a = new complex<double>*[n];
    for (i = 0; i < n; i++)
        a[i] = new complex<double>[n];

    // Copy the input matrix 'matrica_a' to temporary matrix 'a'
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++)
            a[i][j] = matrica_a[i][j];
    }

    // Gaussian elimination with partial pivoting
    for (k = 0; k < n; k++) {
        max = abs(a[k][k]);
        r = k;
        // Find the row with the maximum absolute value in the current column
        for (i = k + 1; i < n; i++) {
            if (abs(a[i][k]) > max) {
                max = abs(a[i][k]);
                r = i;
            }
        }
        // Perform row swaps and eliminate elements below the pivot
        if (r != k)
            det = -det;
        for (j = 0; j < n; j++) {
            c = a[k][j];
            a[k][j] = a[r][j];
            a[r][j] = c;
        }
        for (i = k + 1; i < n; i++) {
            M = a[i][k] / a[k][k];
            for (j = k; j < n; j++)
                a[i][j] -= M * a[k][j];
        }
    }

    // Compute the determinant as the product of the diagonal elements
    for (i = 0; i < n; i++)
        det *= a[i][i];

    // Deallocate memory for the temporary matrix
    for (i = 0; i < n; i++)
        delete[] a[i];
    delete[] a;

    return det;
```

```
}

complex<double>** inv(complex<double>** mat, int n) {
    complex<double> y, w, *b, *c, **a;
    int i, j, k, r, *z;

    a = new complex<double>*[n];
    for (i = 0; i < n; ++i) {
        a[i] = new complex<double>[n];
        memcpy(a[i], mat[i], sizeof(complex<double>) * n);
    }

    z = new int[n];
    b = new complex<double>[n];
    c = new complex<double>[n];

    //initialize the original sequence numbers of columns
    for (j = 0; j < n; ++j)
        z[j] = j;

    // loop through all the rows of the matrix
    for (i = 0; i < n; ++i) {
        // In the current line find the index of the minimum
        // by absolute value of the element
        y = a[i][k = i];
        for (j = i + 1; j < n; ++j) {
            if (abs(w = a[i][j]) > abs(y)) {
                k = j;
                y = w;
            }
        }

        // The minimum element is almost zero?
        if (abs(y) < eps) {
            // delete occupied memory
            delete[] b;
            delete[] c;
            delete[] z;
            for (i = 0; i < n; ++i)
                delete[] a[i];
            delete[] a;
            cout << ("Inverse matrix cannot be computed");
            exit(0);
        }

        y = 1.0 / y;
        for (j = 0; j < n; ++j) {
            c[j] = a[j][k];
            a[j][k] = a[j][i];
            a[j][i] = -c[j] * y;
            b[j] = (a[i][j] *= y);
        }

        //create the rearrangement of the indexes according to
        // with the found main element
        a[i][i] = y;
```

```
        j = z[i];
        z[i] = z[k];
        z[k] = j;

        for (k = 0; k < n; ++k) {
            if (k != i) {
                for (j = 0; j < n; ++j) {
                    if (j != i) {
                        a[k][j] -= b[j] * c[k];
                    }
                }
            }
        }
    }

    // Final formation of the inverse matrix
    for (i = 0; i < n; ++i) {
        while ((k = z[i]) != i) {
            for (j = 0; j < n; ++j) {
                w = a[i][j];
                a[i][j] = a[k][j];
                a[k][j] = w;
            }
            r = z[i];
            z[i] = z[k];
            z[k] = r;
        }
    }

    // delete occupied memory
    delete[] b;
    delete[] c;
    delete[] z;

    // return
    return a;
}


int main()
{
    // Русификация
    setlocale(LC_ALL,"Rus");

    // Ввод i,j, N - размерность матриц
    int i,j, N;

    // Инициализация матриц
    complex <double> **A,**B,**C;

    ifstream f;

    // Открытие файла
    f.open("Matrix.txt");
    f>>N;
```

```
    // Создание матриц
    A = new complex <double> *[N];
    B = new complex <double> *[N];
    C = new complex <double> *[N];
    for(i=0;i<N;i++)
    {
        A[i] = new complex <double> [N];
        B[i] = new complex <double> [N];
        C[i] = new complex <double> [N];
    }

    // Запись элементов в матрицу A
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
            {
                f>>A[i][j];
            }
    }

    // Вывод матрицы A
    cout<<"Матрица A:\n";
    for(i=0;i<N;cout<<endl,i++)
    {
        for(j=0;j<N;j++)
        {
            cout<<A[i][j]<<"\t";
        }
    }

    cout << "\n";

    // Запись элементов в матрицу B
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
        {
            f>>B[i][j];
        }
    }

    // Вывод матрицы B
    cout<<"Матрица B:\n";
    for(i=0;i<N;cout<<endl,i++)
    {
        for(j=0;j<N;j++)
        {
            cout<<B[i][j]<<"\t";
        }
    }

    cout << "\n";

    // Нахождение матрицы C

    // det(B) * A
```

```
    for(i = 0; i < N; ++i)
    {
        for(j = 0; j < N; ++j)
        {
            C[i][j] = (determinant(B,N) * A[i][j] - A[j][i]);
        }
    }
    cout << "\n";

    cout << "Matrix C: \n";
    for(i = 0; i < N; ++i)
    {
        for(j = 0; j < N; ++j)
        {
            cout << mul_mat(C, A, N)[i][j] << "\t";
        }
        cout << "\n";
    }

    cout << "Inv matrix: \n";
    for(i = 0; i < N; ++i)
    {
        for(j = 0; j < N; ++j)
        {
            cout << inv(C, N)[i][j] << "\t";
        }
        cout << "\n";
    }

    for(i = 0; i < N; ++i)
    {
        delete A[i];
        delete B[i];
        delete C[i];
    }
    delete[] A;
    delete[] B;
    delete[] C;

    return 0;
}
```

Результат тестирования программы:

```
Матрица A:
(1,2)   (2,3)   (3,1.54)
(2,5)   (3,7)   (4,10)
(1.5,3.25)      (1.7,-3.94)     (6.23,11.17)

Матрица B:
(6.23,-1.97)    (0.19,0.22)     (0.16,0.28)
(0.22,0.29)     (11,12) (6.72,-1.13)
(5,1)   (1.4,-1.76)     (4.5,2.3)


Matrix C:
(-21311,1613.83)        (-13366.6,1874.4)       (-50032.9,13121.8)
(-53766,-14584.1)       (-32940.6,14737.8)      (-140498,-24606.3)
(-25921.1,8032.19)      (20626.3,44499.9)       (-92938.7,22927)
Inv matrix:
(-0.00160807,0.000278168)       (0.000637256,-0.000389414)      (-0.000262881,-3.82422e-05)
(0.000329197,-0.000193717)      (-0.000239139,4.46329e-05)      (0.000160998,7.87251e-05)
(0.000574172,4.07984e-06)       (-0.000247347,4.0967e-05)       (1.63747e-05,-3.37212e-05)

Process returned 0 (0x0)   execution time : 0.036 s
Press any key to continue.
```

Input interpretation

$$\begin{vmatrix} 6.23 - 1.97\,i & 0.19 + 0.22\,i & 0.16 + 0.28\,i \\ 0.22 + 0.29\,i & 11 + 12\,i & 6.72 - 1.13\,i \\ 5 + i & 1.4 - 1.76\,i & 4.5 + 2.3\,i \end{vmatrix}$$

Result

$$292.679\ldots + 533.312\ldots\,i$$

Test = det(B) * A:

```
Матрица A:
(1,2)   (2,3)   (3,1.54)
(2,5)   (3,7)   (4,10)
(1.5,3.25)      (1.7,-3.94)     (6.23,11.17)

Матрица B:
(6.23,-1.97)    (0.19,0.22)     (0.16,0.28)
(0.22,0.29)     (11,12) (6.72,-1.13)
(5,1)   (1.4,-1.76)     (4.5,2.3)

Test
(-773.946,1118.67)      (-1014.58,1944.66)      (56.7351,2050.66)
(-2081.2,2530.02)       (-2855.15,3648.69)      (-4162.41,5060.03)
(-1294.25,1751.17)      (2598.8,-246.523)       (-4133.71,6591.75)
```

Test = det(B)*A - trans(A)

```
Матрица A:
(1,2)    (2,3)    (3,1.54)
(2,5)    (3,7)    (4,10)
(1.5,3.25)        (1.7,-3.94)      (6.23,11.17)

Матрица B:
(6.23,-1.97)     (0.19,0.22)      (0.16,0.28)
(0.22,0.29)      (11,12) (6.72,-1.13)
(5,1)    (1.4,-1.76)      (4.5,2.3)

Test
(-774.946,1116.67)       (-1016.58,1939.66)       (55.2351,2047.41)
(-2083.2,2527.02)        (-2858.15,3641.69)       (-4164.11,5063.97)
(-1297.25,1749.63)       (2594.8,-256.523)        (-4139.94,6580.58)
```

C = (det(B)*A - trans(A)) * A

```
Matrix C:
(-21311,1613.83)         (-13366.6,1874.4)        (-50032.9,13121.8)
(-53766,-14584.1)        (-32940.6,14737.8)       (-140498,-24606.3)
(-25921.1,8032.19)       (20626.3,44499.9)        (-92938.7,22927)
```

	C_1	C_2	C_3
1	-21310.97585+1613.827075i	-13366.56693+1874.392706i	-50032.884827+13121.773527i
2	-53766.0575-14584.1525i	-32940.6852+14737.8024i	-140497.661-24606.3836i
3	-25921.09+8032.149i	20626.2582+44499.8906i	-92938.655+22926.9166i

C = C^(-1)

```
Matrix C:
(-21311,1613.83)         (-13366.6,1874.4)        (-50032.9,13121.8)
(-53766,-14584.1)        (-32940.6,14737.8)       (-140498,-24606.3)
(-25921.1,8032.19)       (20626.3,44499.9)        (-92938.7,22927)
Inv matrix:
(0.00135495,0.000980804)         (-0.000780985,-0.000251906)      (0.000173582,0.000293699)
(-0.000406477,-9.06698e-05)      (0.000195901,-6.66165e-06)       (-5.07701e-05,-6.28984e-05)
(-0.000393923,-0.000468375)      (0.000247727,0.000156194)        (-3.74789e-05,-0.000114427)
```

	B_1	B_2	B_3
1	0.0013549425031875009881+0.0009807983022193726475i	-0.0007809804129599262199 8-0.0002519051256377174963 2i	0.0001735806600385330522 1+0.0002936983977542934219 7i
2	-0.0004064748349666937 2817-0.0000906691943405770247 7i	0.0001958999216805524432 3-0.0000066614840931612352 45i	-0.0000507698399934368163 67-0.0000628983811373999985 71i
3	-0.0003939206552962791061-0.0004683723785712345484 1i	0.0002477256176727560835 2+0.0001561934865384064831 5i	-0.0000374785455316350661 76-0.0001144263916771719945 6i