\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Computer Programming Lab

CEN-392

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Program 12

Code :-

#include <iostream>

using namespace std;

int lenght(char complex[])

{

    int len = 0;

    while (complex[len] != '\0')

        len++;

    return len;

}

void reverse(char str[])

{

    int len = lenght(str);

    for (int i = 0; i < len / 2; i++)

    {

        char ch = str[i];

        str[i] = str[len - 1 - i];

        str[len - 1 - i] = ch;

    }

}

bool check\_decimal(char str[])

{

    int itr = 0;

    while (str[itr] != '\0')

    {

        if (str[itr++] == '.')

            return true;

    }

    return false;

}

void String\_Integer(char str[], int s, int n, float arr[], int indx)

{

    int ten\_pow = 1;

    arr[indx] = 0;

    while (n >= s)

    {

        arr[indx] += ten\_pow \* (str[n--] - '0');

        ten\_pow \*= 10;

    }

}

void String\_Decimal(char str[], int s, int n, float arr[], int indx)

{

    float ten\_pow = 0.1;

    while (s <= n)

    {

        arr[indx] += ten\_pow \* (str[s++] - '0');

        ten\_pow /= 10;

    }

}

void Addition(float real[], float imaginary[])

{

    cout << "Addition Operation Is Selected..."

         << "\n";

    float r = real[0] + real[1], img = imaginary[0] + imaginary[1];

    cout << "Addition : | " << r;

    if (img > 0)

        cout << " + ";

    cout << img << "i |\n";

}

void Subtract(float real[], float imaginary[])

{

    cout << "Subtraction Operation Is Selected..."

         << "\n";

    float r = real[0] - real[1], img = imaginary[0] - imaginary[1];

    cout << "Subtraction : | " << r;

    if (img > 0)

        cout << " + ";

    cout << img << "i |\n";

}

void Multiply(float real[], float imaginary[])

{

    cout << "Multiplication Operation Is Selected..."

         << "\n";

    float r = real[0] \* real[1] - imaginary[0] \* imaginary[1];

    float img = real[0] \* imaginary[1] + imaginary[0] \* real[1];

    cout << "Multiplication : | " << r;

    if (img > 0)

        cout << " + ";

    cout << img << "i |\n";

}

void Division(float real[], float imaginary[])

{

    cout << "Devision Operation Is Selected..."

         << "\n";

    float devide = real[1] \* real[1] + imaginary[1] \* imaginary[1];

    float r = real[0] \* real[1] + imaginary[0] \* imaginary[1];

    float img = real[1] \* imaginary[0] - imaginary[1] \* real[0];

    r /= devide;

    img /= devide;

    cout << "Division : |" << r;

    if (img > 0)

        cout << " + ";

    cout << img << "i |\n";

}

void Menu()

{

    cout << "\n\_\_\_\_String\_Operations\_\_\_\_\n";

    cout << "1.Add\n";

    cout << "2.Subtract\n";

    cout << "3.Multiply\n";

    cout << "4.Devide\n";

    cout << "5.Exit\n";

    cout << "Enter Your Choice : ";

}

void AnsBar()

{

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

}

bool Options(float real[], float imaginary[])

{

    int opt;

    fflush(stdin);

    cin >> opt;

    AnsBar();

    switch (opt)

    {

    case 1:

        Addition(real, imaginary);

        break;

    case 2:

        Subtract(real, imaginary);

        break;

    case 3:

        Multiply(real, imaginary);

        break;

    case 4:

        Division(real, imaginary);

        break;

    case 5:

        return 0;

    default:

        cout << "Invalid Input!\nTry Again!\n";

    }

    AnsBar();

    return 1;

}

int main()

{

    system("cls");

    cout << "\_\_\_\_\_Vicky\_Gupta\_20BCS070\_\_\_\_\_\n\n";

    char complex[100];

    cout << "Enter The Complex Number : \n";

    gets(complex);

    float real[2], imaginary[2];

    int itr = 0;

    int clen = lenght(complex);

    int complex\_counter = 0;

    int prev\_indx = 0;

    while (clen > itr)

    {

        int citr = itr; // complex iterator

        while (complex[citr] != ',' && complex[citr] != '\0')

            citr++;

        itr = citr + 1;

        citr--;

        char r[50], img[50];

        int iitr = 0, ritr = 0; // imaginary iterator real iterator

        if (complex[citr] == 'i')

        {

            citr--;

            while (citr >= prev\_indx && complex[citr] != '+' && complex[citr] != '-')

                img[iitr++] = complex[citr--];

            if (citr >= prev\_indx)

            {

                img[iitr++] = complex[citr--];

            }

            img[iitr] = '\0';

            reverse(img);

        }

        else

            img[0] = '\0';

        if (citr > prev\_indx)

        {

            while (citr >= prev\_indx && complex[citr] != '+' && complex[citr] != '-')

                r[ritr++] = complex[citr--];

            if (citr >= prev\_indx)

            {

                r[ritr++] = complex[citr--];

            }

            r[ritr] = '\0';

            reverse(r);

        }

        else

            r[0] = '\0';

        if (r[0] != '\0') // for real

        {

            bool isDecimal = check\_decimal(r);

            if (isDecimal)

            {

                int decimal\_index = 0;

                while (r[decimal\_index] != '.')

                    decimal\_index++;

                if (r[0] == '+' || r[0] == '-')

                    String\_Integer(r, 1, decimal\_index - 1, real, complex\_counter);

                else

                    String\_Integer(r, 0, decimal\_index - 1, real, complex\_counter);

                String\_Decimal(r, decimal\_index + 1, lenght(r) - 1, real, complex\_counter);

            }

            else

            {

                if (r[0] == '+' || r[0] == '-')

                    String\_Integer(r, 1, lenght(r) - 1, real, complex\_counter);

                else

                    String\_Integer(r, 0, lenght(r) - 1, real, complex\_counter);

            }

            if (r[0] == '-')

                real[complex\_counter] = -real[complex\_counter];

        }

        else

            real[complex\_counter] = 0;

        if (img[0] != '\0') // for imaginary

        {

            bool isDecimal = check\_decimal(img);

            if (isDecimal)

            {

                int decimal\_index = 0;

                while (img[decimal\_index] != '.')

                    decimal\_index++;

                if (img[0] == '+' || img[0] == '-')

                    String\_Integer(img, 1, decimal\_index - 1, imaginary, complex\_counter);

                else

                    String\_Integer(img, 0, decimal\_index - 1, imaginary, complex\_counter);

                String\_Decimal(img, decimal\_index + 1, lenght(img) - 1, imaginary, complex\_counter);

            }

            else

            {

                if (img[0] == '+' || img[0] == '-')

                    String\_Integer(img, 1, lenght(img) - 1, imaginary, complex\_counter);

                else

                    String\_Integer(img, 0, lenght(img) - 1, imaginary, complex\_counter);

            }

            if (img[0] == '-')

                imaginary[complex\_counter] = -imaginary[complex\_counter];

        }

        else

            imaginary[complex\_counter] = 0;

        complex\_counter++;

        prev\_indx = itr;

    }

    cout << "\nComplex Number \n";

    for (int i = 0; i < 2; i++)

    {

        cout << i + 1 << ". " << real[i] << " " << imaginary[i] << "i\n";

    }

    cout << "\n";

    while (true)

    {

        Menu();

        if (!Options(real, imaginary))

            break;

    }

    cout << "Exiting...\n";

    AnsBar();

    return 0;

}

Output :-

Text

Description automatically generated

Text

Description automatically generated