Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра ПЗ

Лабораторна робота №7, варіант № 19
з дисципліни «Основи програмування»

Виконала: ст. 1ПІ-25Б Семенов В.О. .
Перевірив: доцент Решетнік О.О.

Вінниця – 2025

Тема: Структури та об'єднання

Мета: Опанувати створення та використання структур та об'єднань, робота зі складними структурами даних.

Завдання 19. Увести структуру COMPLEX для опису поняття комплексне число. Реалізувати функції: COMPLEX mul(COMPLEX a, COMPLEX b) – множення чисел, COMPLEX div(COMPLEX a, COMPLEX b) – ділення чисел, COMPLEX pow(COMPLEX base, COMPLEX exp) – піднесення до степені.

Декомпозиція завдання

1. Оголошення структури COMPLEX
    - Визначає комплексне число з дійсною та уявною частинами.
    - Містить перевантаження операторів для арифметичних операцій.
2. Перевантаження операторів
    - + та - – додавання і віднімання комплексних чисел.
    - * – множення комплексних чисел.
    - / – ділення комплексних чисел з перевіркою ділення на нуль.
    - ^ – піднесення комплексного числа до цілого степеня (позитивного та від’ємного).
3. Функція вводу числа з перевіркою
    - Забезпечує правильне введення дійсних чисел від користувача.
    - Повторно запитує, якщо введено некоректне значення.
4. Функція виводу комплексного числа
    - Форматує число у вигляді a + bi або a - bi.
    - Обробляє знак уявної частини, щоб уникнути подвійного мінуса.
5. Введення користувачем даних
    - Запитує дійсну та уявну частину двох комплексних чисел.
    - Запитує цілий степінь для оператора піднесення до степеня.
6. Виконання операцій над комплексними числами
    - Обчислюються всі перевантажені операції (+, -, *, /, ^).

7. Вивід результатів

    - Виводяться всі результати у зрозумілому форматі, показуючи дійсну і уявну частину.

Код програми показано на лістингу.

Приклади роботи програми можна побачити на рисунку 1

Лістинг 1 – Програма для роботи з комплексними числами

```
#include <iostream>
#include <limits>
#include <cmath>

struct COMPLEX {
    double re;
    double im;

    COMPLEX operator+(const COMPLEX& other) const
    {
        return { re + other.re, im + other.im };
    }

    COMPLEX operator-(const COMPLEX& other) const
    {
        return { re - other.re, im - other.im };
    }

    COMPLEX operator*(const COMPLEX& other) const
    {
        return { re * other.re - im * other.im,
                 re * other.im + im * other.re };
    }

    COMPLEX operator/(const COMPLEX& other) const
    {
        double denom = other.re * other.re + other.im *
other.im;
        if (denom == 0.0)
        {
            std::cerr << "Error: division by zero\n";
            return { 0.0, 0.0 };
        }
        return { (re * other.re + im * other.im) / denom,
                 (im * other.re - re * other.im) / denom };
    }

    COMPLEX operator^(int n) const
```

Продовження лістингу 1

```
    {
        COMPLEX result{ 1.0, 0.0 };
        COMPLEX base = *this;
        if (n == 0) return result;
        bool negative = false;
        if (n < 0)
        {
            negative = true;
            n = -n;
        }
        for (int i = 0; i < n; i++)
        {
            result = result * base;
        }
        if (negative)
        {
            result = COMPLEX{ 1.0,0.0 } / result;
        }
        return result;
    }
};

double inputDouble(const std::string& prompt)
{
    double val;
    std::cout << prompt;
    std::cin >> val;
    return val;
}

void printComplex(const COMPLEX& z)
{
    std::cout << z.re;
    if (z.im >= 0)
    {
        std::cout << " + " << z.im << "i";
    }
    else
    {
        std::cout << " - " << -z.im << "i";
    }
    std::cout << '\n';
}

int main()
{
    std::cout << "Enter first complex number:\n";
    COMPLEX a;
    a.re = inputDouble("  Real part: ");
    a.im = inputDouble("  Imaginary part: ");
```

Продовження лістингу 1

```
    std::cout << "Enter second complex number:\n";
    COMPLEX b;
    b.re = inputDouble("  Real part: ");
    b.im = inputDouble("  Imaginary part: ");

    int n;
    std::cout << "Enter integer exponent for first number (a ^
n): ";
    std::cin >> n;

    std::cout << "\nResults:\n";
    std::cout << "a + b = "; printComplex(a + b);
    std::cout << "a - b = "; printComplex(a - b);
    std::cout << "a * b = "; printComplex(a * b);
    std::cout << "a / b = "; printComplex(a / b);
    std::cout << "a ^ " << n << " = "; printComplex(a ^ n);

    return 0;
}
```

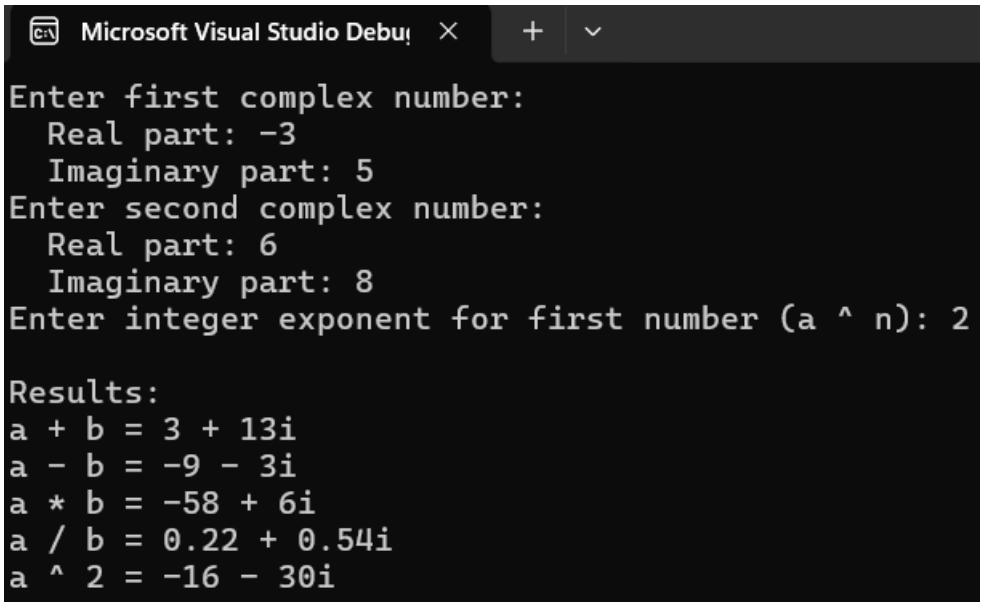

Рисунок 1 – робота програми при правильному введені

Контрольні запитання:

4. Як звернутися до елемента масиву, що входить у структуру? До структури, що є елементом масиву?

- Через крапку або стрілку, комбіновану з індексом: спочатку вказується елемент масиву, потім поле структури.

- До структури в масиві звертаються через індекс масиву, потім через крапку до її полів.

5. Що потрібно для включення структури в структуру великого розміру?

- Необхідно оголосити внутрішню структуру до її використання.
- Можна включати або як об’єкт, або як вказівник для економії пам’яті, якщо структура велика.

6. Яка область дії змінних і функцій програми?

- Локальні змінні – видимі тільки в блоці або функції, де оголошені.
- Глобальні змінні і функції – доступні у всій програмі після оголошення.
- Статичні змінні – зберігають значення між викликами і можуть мати обмежену область видимості.

Висновок: Структури та об'єднання дозволяють об’єднувати різнорідні дані в один логічний блок, спрощуючи роботу зі складними даними. Вивчення їх створення та використання дає змогу ефективно організовувати інформацію та будувати більш гнучкі програми.