Практическая работа № 3.  
Перегрузка операторов

**Цель практической работы**

Закрепление теоретических знаний по перегрузке операторов в программах на C#.

**Постановка задачи**

Перегрузка операторов в C# – это возможность определить или изменить поведение стандартных операторов для пользовательских типов данных. Это позволяет использовать операторы, такие как +, -, \*, == и другие, с вашими собственными классами или структурами, что делает код более читаемым и интуитивно понятным.

**Синтаксис**: Для перегрузки операторов используется ключевое слово operator, за которым следует знак оператора, который вы хотите перегрузить. Метод должен быть статическим, и его возвращаемый тип должен соответствовать типу результата операции.

**Ограничения**: Не все операторы могут быть перегружены. Например, операторы . (точка), :: (двойная двоеточие), ?: (тернарный оператор) и new не могут быть перегружены.

Рассмотрим пример, в котором мы создадим класс Complex, представляющий комплексные числа, и перегрузим несколько операторов для работы с ними.

public class Complex

{

public double Re { get; set; } // Вещественная часть

public double Im { get; set; } // Мнимая часть

public Complex(double re = 0, double im = 0)

{

Re = re;

Im = im;

}

// Определяем оператор сложения для объектов типа Complex

public static Complex operator +(Complex c1, Complex c2)

=> new(c1.Re + c2.Re, c1.Im + c2.Im);

// Определяем оператор вычитания для объектов типа Complex

public static Complex operator -(Complex c1, Complex c2)

=> new(c1.Re - c2.Re, c1.Im - c2.Im);

// Определяем оператор сложения Complex и double

public static Complex operator +(Complex c, double d)

=> new(c.Re + d, c.Im);

// Операции по умолчанию не коммутативны

public static Complex operator -(double d, Complex c)

=> new(d - c.Re, c.Im);

public static Complex operator +(double d, Complex c)

=> c + d;

public static Complex operator -(Complex c, double d)

=> new(c.Re - d, c.Im);

public static Complex operator \*(Complex c, double d)

=> new(d \* c.Re, d \* c.Im);

public static Complex operator \*(double d, Complex c)

=> c \* d;

public static Complex operator /(Complex c, double d)

=> new(c.Re / d, c.Im / d);

public static Complex operator \*(Complex c1, Complex c2)

=> new(c1.Re \* c2.Re - c1.Im \* c2.Im,

c1.Re \* c2.Im + c1.Im \* c2.Re);

public static Complex operator /(Complex c1, Complex c2)

{

var conjugate = c2.GetConjugate();

var c1New = c1 \* conjugate;

var c2New = c2 \* conjugate;

return c1New / c2New.Re;

}

// Сопряженное комплексное число

public Complex GetConjugate()

=> new(Re, -Im);

// Неявное преобразование вещественного числа в комплексное

public static implicit operator Complex(double d)

=> new(d, 0);

// Явное преобразование комплексного числа в вещественное с отбрасыванием мнимой части

public static explicit operator double(Complex c)

=> c.Re;

}

Работа с данным классом выглядит следующим образом:

var c1 = new Complex(1, 2);

var c2 = new Complex(4, 1);

var sum = c1 + c2; // Сложение комплексных чисел

Console.WriteLine($"Сумма {c1}+{c2}={sum}");

var prod = c1 \* c2; // Произведение комплексных чисел

Console.WriteLine($"Произведение ({c1})\*({c2})={prod}");

var num = (double)c1; // Явное (explicit) преобразование в вещественное число

Console.WriteLine($"Явное (explicit) преобразование {c1} в вещественное число {num}");

Complex c3 = num; // Неявное (implicit) преобразование в комплексное число

Console.WriteLine($"Неявное (implicit) преобразование {num} в комплексное число {c3}");

На рисунке 1 представлен результат работы с данным классом.

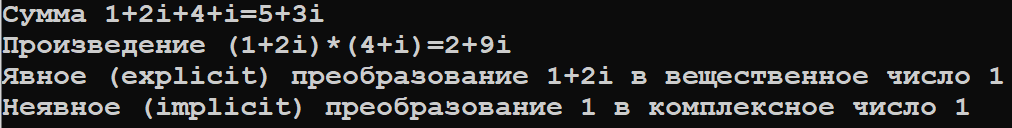


Рисунок 1 – Результат работы с классом комплексных чисел

**Задание на практическую работу**

1. Разработать библиотеку классов в соответствии с вариантом и консольное приложение, использующее данную библиотеку.
2. В библиотеке необходимо разработать класс и перегрузить соответствующие операторы класса согласно варианту.
3. В консольном приложении необходимо продемонстрировать работу с классом.
4. Защита работы включает демонстрацию работы программы на подготовленном примере.

**Варианты заданий на практическую работу**

1. **Класс Вектор (Vector)**:
   * Перегрузка операторов + (сложение векторов)
   * Перегрузка операторов - (вычитание векторов)
   * Перегрузка оператора \* (умножение вектора на скаляр)
   * Перегрузка оператора \* (скалярное произведение векторов)
   * Перегрузка оператора == (сравнение векторов на равенство)
   * Перегрузка оператора != (сравнение векторов на неравенство)
   * Перегрузка оператора [] (доступ к компонентам вектора)
2. **Класс Матрица (Matrix)**:
   * Перегрузка операторов + (сложение матриц)
   * Перегрузка операторов - (вычитание матриц)
   * Перегрузка оператора \* (умножение матриц)
   * Перегрузка оператора \* (умножение матрицы на скаляр)
   * Перегрузка оператора [] (доступ к элементам матрицы)
3. **Класс Время (Time)**:
   * Перегрузка операторов + (сложение временных интервалов)
   * Перегрузка операторов - (вычитание временных интервалов)
   * Перегрузка оператора \* (умножение временного интервала на скаляр)
   * Перегрузка оператора == (сравнение временных интервалов на равенство)
   * Перегрузка оператора != (сравнение временных интервалов на неравенство)
   * Перегрузка оператора < (сравнение временных интервалов)
   * Перегрузка оператора > (сравнение временных интервалов)
4. **Класс Цвет (Color)**:
   * Перегрузка операторов + (смешивание цветов)
   * Перегрузка операторов - (вычитание одного цвета из другого)
   * Перегрузка оператора \* (умножение цвета на скаляр для изменения яркости)
   * Перегрузка оператора == (сравнение цветов на равенство)
   * Перегрузка оператора != (сравнение цветов на неравенство)
   * Перегрузка оператора [] (доступ к компонентам цвета, например, RGB)
5. **Класс Дата (Date)**:
   * Перегрузка операторов + (добавление дней к дате)
   * Перегрузка операторов - (вычитание дней из даты)
   * Перегрузка оператора - (вычисление разницы между двумя датами)
   * Перегрузка оператора == (сравнение дат на равенство)
   * Перегрузка оператора != (сравнение дат на неравенство)
   * Перегрузка оператора < (сравнение дат)
   * Перегрузка оператора > (сравнение дат)
6. **Класс Диапазон (Range)**:
   * Перегрузка операторов + (расширение диапазона)
   * Перегрузка операторов - (уменьшение диапазона)
   * Перегрузка оператора \* (умножение диапазона на скаляр)
   * Перегрузка оператора == (сравнение диапазонов на равенство)
   * Перегрузка оператора != (сравнение диапазонов на неравенство)
   * Перегрузка оператора [] (проверка, содержит ли диапазон значение)

**Пример разработки приложения**

**Шаг 1. Создание структуры класса**

1. Создайте новый класс с именем Complex
2. Добавьте два публичных свойства:
   * Re (вещественная часть, тип double)
   * Im (мнимая часть, тип double)
3. Создайте конструктор с параметрами по умолчанию (0, 0)

**Пояснение:**

* Это основа для представления комплексного числа в форме Re + Im\*i

**Шаг 2. Реализация базовых операторов**

1. Перегрузите оператор сложения (+) для:
   * Complex + Complex
   * Complex + double
   * double + Complex (коммутативная версия)
2. Перегрузите оператор вычитания (-) для тех же комбинаций

**Важно:**

* Для Complex+Complex складывайте соответствующие компоненты
* Для операций с double мнимая часть сохраняется

**Шаг 3. Реализация умножения и деления**

1. Перегрузите оператор умножения (\*):
   * Для Complex\*Complex используйте формулу:

Copy

Download

(a+bi)\*(c+di) = (ac-bd) + (ad+bc)i

* + Реализуйте также умножение на double

1. Перегрузите оператор деления (/):
   * Для Complex/Complex используйте умножение на сопряженное
   * Реализуйте деление на double

**Совет:**

* Создайте метод GetConjugate() для получения сопряженного числа

public class Complex

{

public double Re { get; set; } // Вещественная часть

public double Im { get; set; } // Мнимая часть

public Complex(double re = 0, double im = 0)

{

Re = re;

Im = im;

}

// Определяем оператор сложения для объектов типа Complex

public static Complex operator +(Complex c1, Complex c2)

=> new(c1.Re + c2.Re, c1.Im + c2.Im);

// Определяем оператор вычитания для объектов типа Complex

public static Complex operator -(Complex c1, Complex c2)

=> new(c1.Re - c2.Re, c1.Im - c2.Im);

// Определяем оператор сложения Complex и double

public static Complex operator +(Complex c, double d)

=> new(c.Re + d, c.Im);

// Операции по умолчанию не коммутативны

public static Complex operator -(double d, Complex c)

=> new(d - c.Re, c.Im);

public static Complex operator +(double d, Complex c)

=> c + d;

public static Complex operator -(Complex c, double d)

=> new(c.Re - d, c.Im);

public static Complex operator \*(Complex c, double d)

=> new(d \* c.Re, d \* c.Im);

public static Complex operator \*(double d, Complex c)

=> c \* d;

public static Complex operator /(Complex c, double d)

=> new(c.Re / d, c.Im / d);

public static Complex operator \*(Complex c1, Complex c2)

=> new(c1.Re \* c2.Re - c1.Im \* c2.Im,

c1.Re \* c2.Im + c1.Im \* c2.Re);

public static Complex operator /(Complex c1, Complex c2)

{

var conjugate = c2.GetConjugate();

var c1New = c1 \* conjugate;

var c2New = c2 \* conjugate;

return c1New / c2New.Re;

}

// Сопряженное комплексное число

public Complex GetConjugate()

=> new(Re, -Im);

// Неявное преобразование вещественного числа в комплексное

public static implicit operator Complex(double d)

=> new(d, 0);

// Явное преобразование комплексного числа в вещественное с отбрасыванием мнимой части

public static explicit operator double(Complex c)

=> c.Re;

}

**Шаг 5. Тестирование класса**

1. Создайте несколько комплексных чисел
2. Проверьте все операции:
   * Сложение/вычитание
   * Умножение/деление
   * Преобразования типов
3. Убедитесь, что результаты соответствуют математическим ожиданиям

**Пример тестов:**

* (1+2i) + (3+4i) = 4+6i
* (2+i) \* (1+3i) = -1+7i

Работа с данным классом выглядит следующим образом:

var c1 = new Complex(1, 2);

var c2 = new Complex(4, 1);

var sum = c1 + c2; // Сложение комплексных чисел

Console.WriteLine($"Сумма {c1}+{c2}={sum}");

var prod = c1 \* c2; // Произведение комплексных чисел

Console.WriteLine($"Произведение ({c1})\*({c2})={prod}");

var num = (double)c1; // Явное (explicit) преобразование в вещественное число

Console.WriteLine($"Явное (explicit) преобразование {c1} в вещественное число {num}");

Complex c3 = num; // Неявное (implicit) преобразование в комплексное число

Console.WriteLine($"Неявное (implicit) преобразование {num} в комплексное число {c3}");

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»