**Введение**

**Условие задачи**: Реализовать Класс FazzyNumber для работы с нечеткими числами, которые представляются тройками чисел (x – e1, x, x + e2). Для чисел A = (A – a1, A, A + ar) и B = (B – b1, B, B + br) арифметические операции выполняются по следующим формулам:

– сложение A + B = (A + B – a1 – b1, A + B, A + B + ar + br);

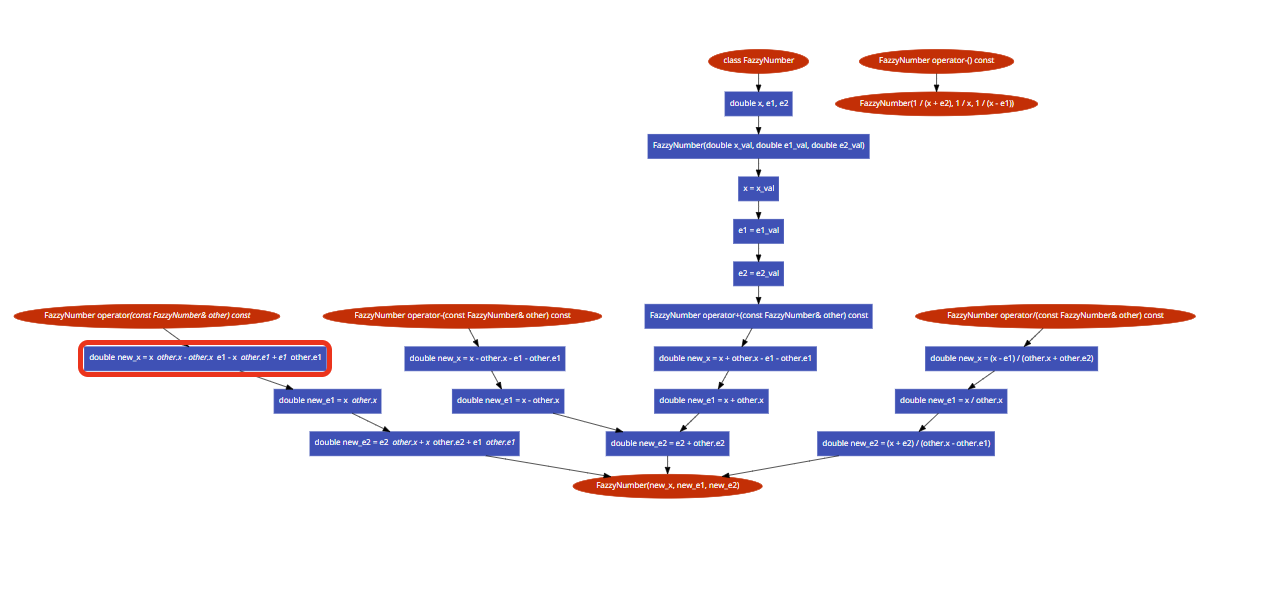
– вычитание A – B = (A – B – a1 – b1, A – B, A – B + ar + br);

– умножение A\*B = (A\*B – B\*a1 – A\*b1 + a1\*b1, A\*B, A\*B + B\*a1 + A\*b1 + a1\*b1);

– обратное число A = (1/(A + ar), 1/A, 1/(A – al)), A > 0;

– деление A/B = ((A – al)/(B + br), A/B, (A + ar)/(B – bl), B > 0.

**Суть задачи**: Данная задача состоит в реализации класса FazzyNumber для работы с нечеткими числами, представляемыми тройками чисел (x - e1, x, x + e2). Класс должен обеспечивать арифметические операции с нечеткими числами, такие как сложение, вычитание, умножение, деление и нахождение обратного числа.

****

**Решение задачи и объяснение:**

Класс FazzyNumber содержит приватные переменные x, e1 и e2, которые представляют значения нечеткого числа (x - e1, x, x + e2). Конструктор класса принимает значения x, e1 и e2 и инициализирует соответствующие переменные.

Операторы класса реализуют арифметические операции над нечеткими числами.

- Оператор + выполняет сложение двух нечетких чисел, вычитая e1 и other.e1 из суммы x и other.x, и складывая e2 и other.e2.

- Оператор - выполняет вычитание двух нечетких чисел, аналогично оператору +.

- Оператор \* выполняет умножение двух нечетких чисел, используя формулу, описанную в задаче.

- Оператор - возвращает обратное число, где каждый компонент исходного числа заменяется обратным значением (1/val).

- Оператор / выполняет деление двух нечетких чисел, используя формулу, описанную в задаче.

**Выводы решенной задачи:**

Реализация класса FazzyNumber позволяет работать с нечеткими числами и выполнять арифметические операции над ними. Класс обеспечивает удобный способ представления и манипулирования нечеткими числами.

Приложение

Листинг 1

class FazzyNumber {

private:

double x, e1, e2;

public:

FazzyNumber(double x\_val, double e1\_val, double e2\_val) {

x = x\_val;

e1 = e1\_val;

e2 = e2\_val;

}

FazzyNumber operator+(const FazzyNumber& other) const {

double new\_x = x + other.x - e1 - other.e1;

double new\_e1 = x + other.x;

double new\_e2 = e2 + other.e2;

return FazzyNumber(new\_x, new\_e1, new\_e2);

}

FazzyNumber operator-(const FazzyNumber& other) const {

double new\_x = x - other.x - e1 - other.e1;

double new\_e1 = x - other.x;

double new\_e2 = e2 + other.e2;

return FazzyNumber(new\_x, new\_e1, new\_e2);

}

FazzyNumber operator\*(const FazzyNumber& other) const {

double new\_x = x \* other.x - other.x \* e1 - x \* other.e1 + e1 \* other.e1;

double new\_e1 = x \* other.x;

double new\_e2 = e2 \* other.x + x \* other.e2 + e1 \* other.e1;

return FazzyNumber(new\_x, new\_e1, new\_e2);

}

FazzyNumber operator-() const {

return FazzyNumber(1 / (x + e2), 1 / x, 1 / (x - e1));

}

FazzyNumber operator/(const FazzyNumber& other) const {

double new\_x = (x - e1) / (other.x + other.e2);

double new\_e1 = x / other.x;

double new\_e2 = (x + e2) / (other.x - other.e1);

return FazzyNumber(new\_x, new\_e1, new\_e2);

}

};