Отчет по Лаборатоорной работе №2

Технология программирования

Бекауов Артур Тимурович

Содержание

# 1 Цель работы

Целью работы является написание программы на с++, содержащей описание классов vect и matr, конструкторы и деструктор для каждого класса, набор оператор-функций для операций векторной алгебры и функцию main, использующую вышеописанный инструментарий.

# 2 Ход лабораторной работы

**Описание классов**

Программа, написанная мной задаёт классы vect и matr. Экземплярами vect (Рис. [1](#fig:1)), будут векторы, описываемые четырьмя полями - размерность int dim (целое число), значение double\*v (указатель на одномерный массив), номер вектора int num (целое число), и статической переменной static int count, которая ведёт общий счёт созданных векторов.

Затем описаны конструкторы и деструктор:

-vect(); - конструктор по умолчанию (создаёт единичный вектор размера 3).

-vect(vect&x); - конструктор копирования.

-vect(int n); - конструктор создания нулевого вектора размера n.

-vect(int n, double\*x); - конструктор создания вектора с заданными значениями

-~vect(); - деструктор вектора

После этого перечисленны методы класса vect и дружественные функции:

-void print(); - функция вывода вектора.

-vect operator+(vect r); - функция сложения векторов (бинарная).

-friend vect operator-(vect&l,vect&r); - друж. функция вычитания векторов (бинарная).

-vect operator-(); - функция вычитания вектора (унарная).

-vect operator=(const vect&r); - функция присвоения вектору значения другого вектора.

-double operator\*(vect&r); - функция скалярного умножения векторов.

-friend vect operator\*(double k, vect&r); - друж. функция умножения вектора на число k.

Далее описан дружественный класс matr. Сделанно это, потому что в классе matr появится метод, которому нужен будет доступ к внутренним данным класса vect (а именно к значениям вектора).

Сразу после описания класса я определил и описал статическую переменную count = 0, для её корректного функционирования внутри main.

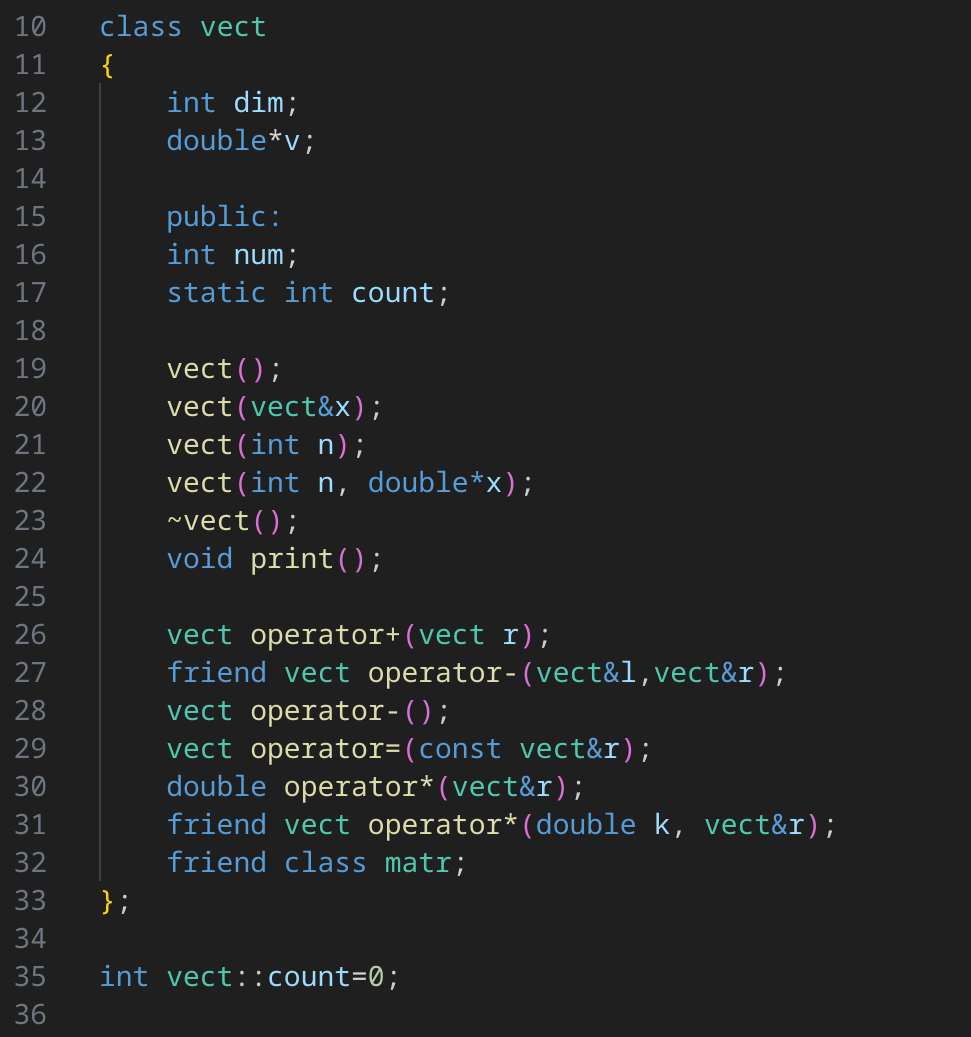


Figure 1: Описание класса vect

Экземплярами matr(Рис. [2](#fig:2)), будут квадратные матрицы, описываемые четырьмя полями (хотя можно обойтись и первыми двумя) - размерность int dim (целое число), значение double\*\*v (указатель на массив указателей на массивы вещ. чисел - т.е двумерный массив вещ. чисел), номер матрицы int num (целое число), и статической переменной static int count, которая ведёт общий счёт созданных матриц.

Затем описаны конструкторы:

-matr(); - конструктор по умолчанию (создаёт единичную матрицу 3х3).

-matr(matr&x); - конструктор копирования.

-matr(int n); - конструктор создания нулевой матрицы размера nxn.

-matr(int n, double\*\*x); - конструктор создания матрицы nxn с заданными компонентами.

-~matr(); - деструктор матрицы.

После этого перечисленны методы класса matr и дружественные функции:

-void print(); - функция вывода матрицы.

-matr operator+(matr r); - функция сложения матриц.

-matr operator-(matr&r); - функция вычитания матриц (бинарная).

-matr operator-(); - функция вычитания матрицы (унарная).

-matr operator=(const matr r); - функция присвоения матрице значения другой матрицы.

-matr operator\*(matr&r); - функция перемножения матриц.

-friend matr operator\*(double k, matr&r); - друж. функция умножения матрицы на число.

-vect operator\*(vect&r); - функция умножения матрицы на вектор столбец (результатом будет вектор).

Затем я, конечно, определил и описал статическую переменную count = 0.

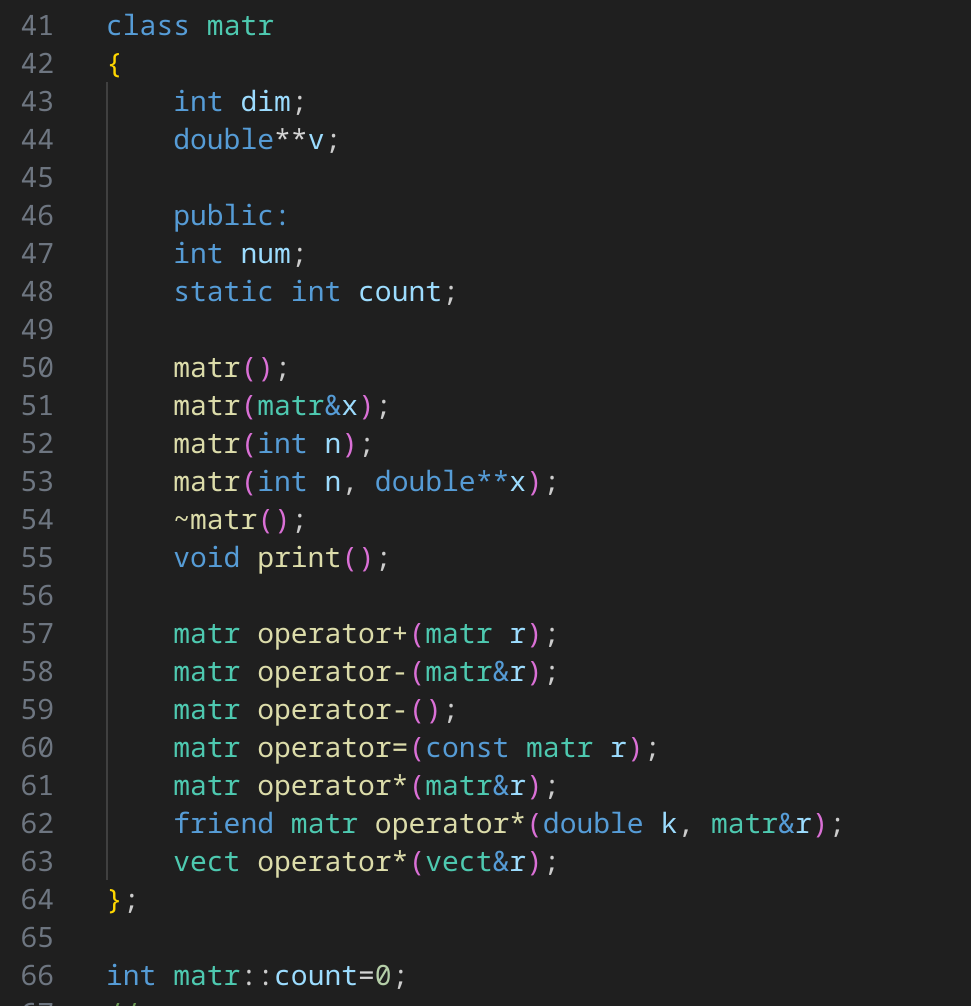


Figure 2: Описание класса matr

**Описание конструкторов, деструктора и методов класс vect**

Описание я начал с конструктора вектора по умолчанию (Рис. [3](#fig:3)). Так как это конструктор, то первым делом я увеличил значение count на 1 (потому что всякий раз, когда вызывается конструктор - создаётся новый вектор). Присвоил парамеру num создаваемого новое значение count. Таким образом созданный вектор получил свой уникальный номер, по которому я смогу к нему обращаться в описаниях действий конструкторов, деструкторов и методов класса vect (в том числе и при выводе вектора). Затем параметру dim задаю значение 3. А параметру v c помощью оператора new задаю значение указателя на динамическую память в которой находится массив размером в dim вещественных чисел. Далее с помощью цикла for заполняю массив единицами. В конце вывожу сообщение, что конструктор по умолчанию создал вектор №num. Таким образом, конструктор по умолчанию создал единичный вектор размерностью 3. (Как пример - v1(1,1,1)).

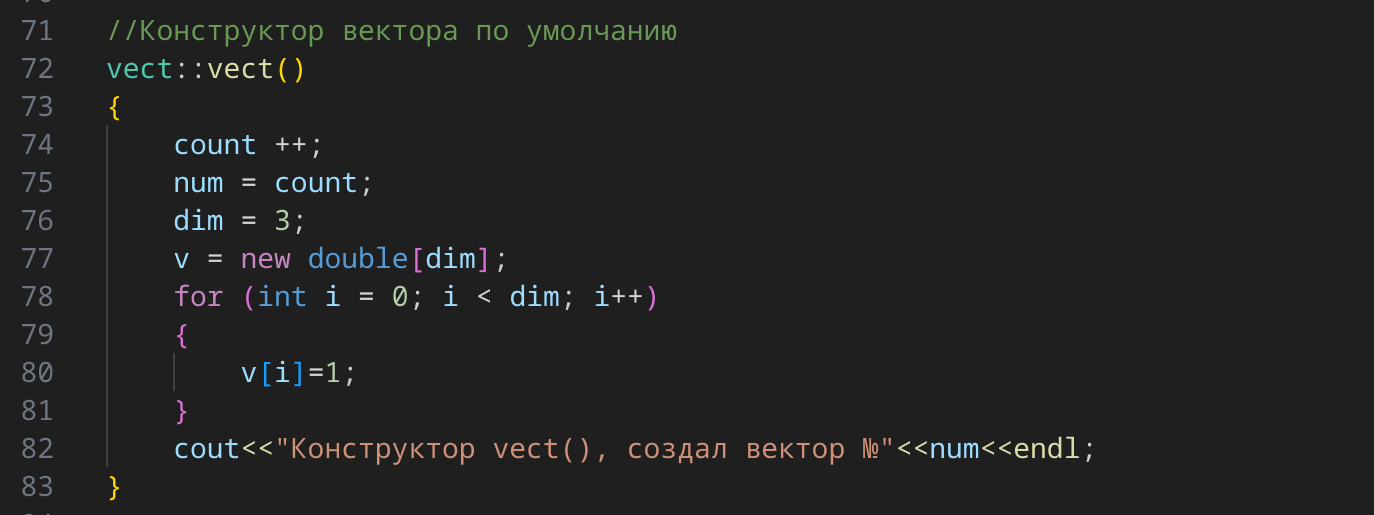


Figure 3: vect: Конструктор по умолчанию

После - я описал конструктор копирования вектора (Рис. [4](#fig:4)), который получает на вход ссылку на объект класса vect - r. Увеличиваю значение count на 1 и присваиваю его параметру num. Затем пареметру dim присваиваю значение dim вектора r (r.dim). Выделяем динамическую память под массив v, и каждому из значений массива v[i] присваиваю соответствующее значение r.v[i]. В конце вывожу сообщение, что конструктор копирования создал вектор №num. Таким образом, конструктор по умолчанию создал вектор-копию r.

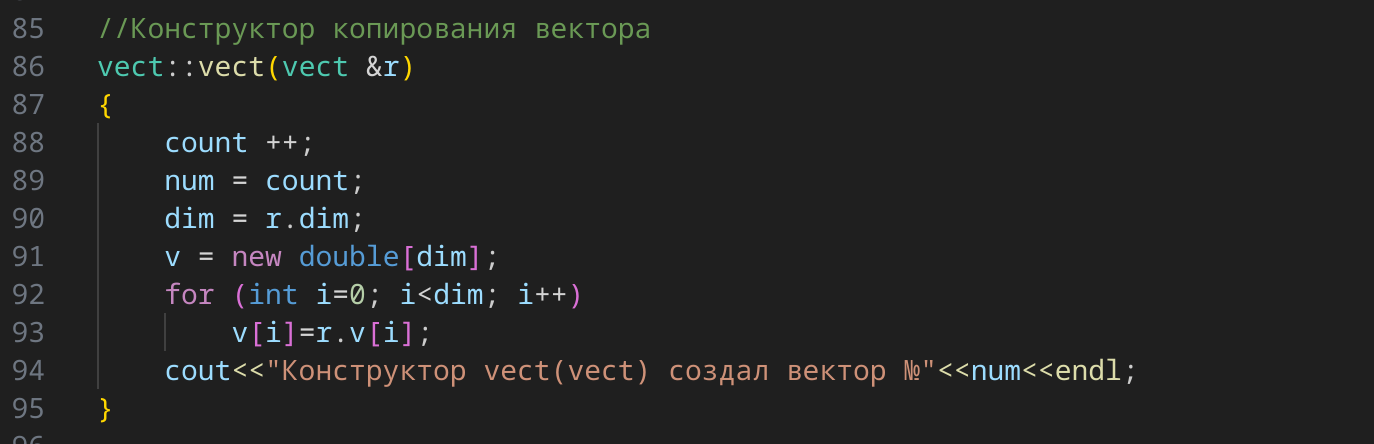


Figure 4: vect: Конструктор копирования

Далее я описал конструктор создания нулевого вектора (Рис. [5](#fig:5)), который получает на вход целое число n - размер будущего вектора. Увеличиваю значение count на 1 и присваиваю его параметру num. Затем параметру dim присваиваю значение n. Выделяем динамическую память под массив v, и каждому из значений массива v[i] присваиваю значение 0. В конце вывожу сообщение, что конструктор создания нулевого вектора создал вектор №num. Таким образом, конструктор создания нулевого вектора создал нулевой вектор размерностью n.

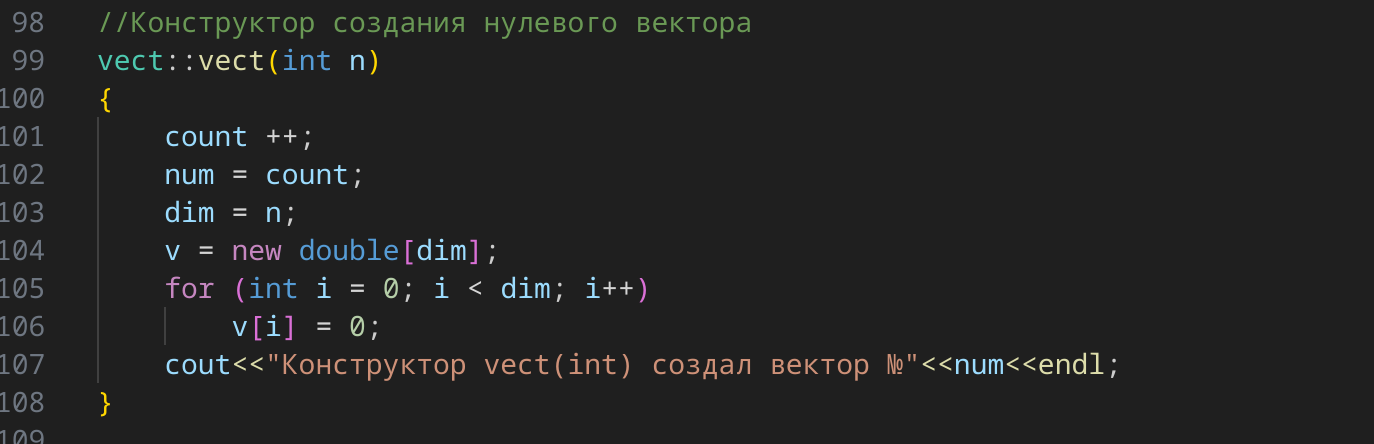


Figure 5: vect: Конструктор создания нулевого вектора

После - я описал конструктор создания вектора с компонентами (Рис. [6](#fig:6)), который полуает на вход целое число n - размер будущего вектора и указатель на массив вещественных числе \*x - компоненты будущего вектора. Увеличиваю значение count на 1 и присваиваю его параметру num. Затем пареметру dim присваиваю значение n. Выделяем динамическую память под массив v, и каждому из значений массива v[i] присваиваю значение x[i]. В конце вывожу сообщение, что конструктор создания вектора с компонентами создал вектор №num. Таким образом, конструктор создания вектора с компонентами создал вектор размерностью n, с компонентами массива x.

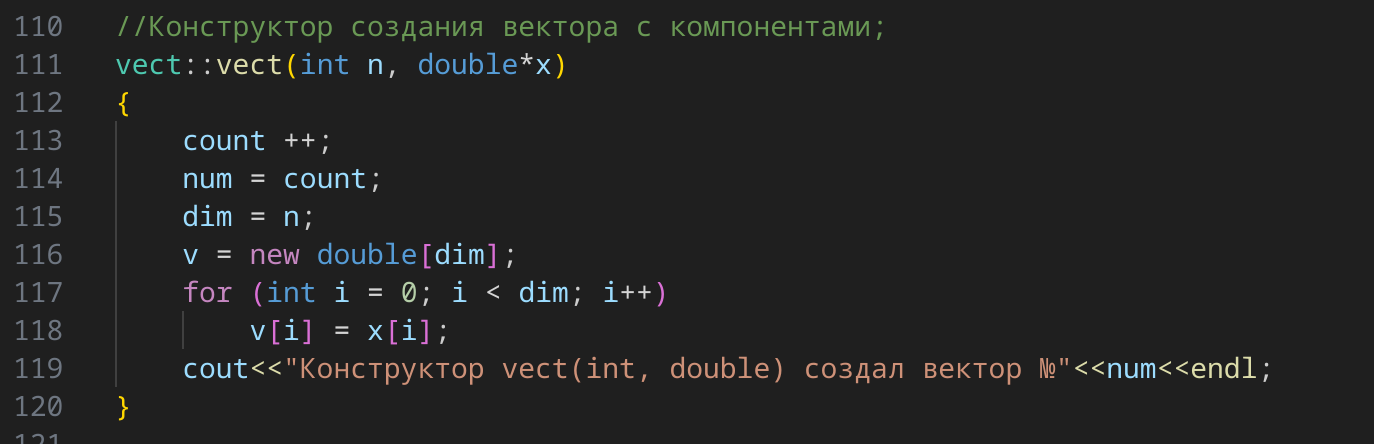


Figure 6: vect: Конструктор создания вектора с компонентами

Далее я описал деструктор вектора (Рис. [7](#fig:7)). С помощью оператора delete освобождаю динамическую память, на которую указывет указатель v. Параметру dim присваиваю значеник 0. В конце вывожу сообщение, что деструктор вектора ликвидировал вектор №num. Таким образом деструктор вектора ликвидировал вектор №num и освободил соотствующую ему динамическую память.

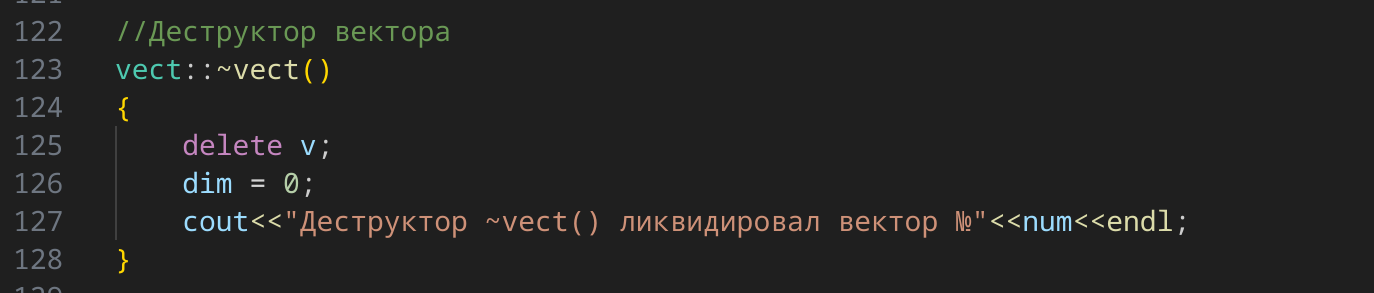


Figure 7: vect: Деструктор вектора

Затем я описал функцию вывода вектора (Рис. [8](#fig:8)). Вывожу сообщение, указывающее номер выводимого вектора и его размерность. Потом с помощью цикла for вывожу каждое из значений компонент вектора.

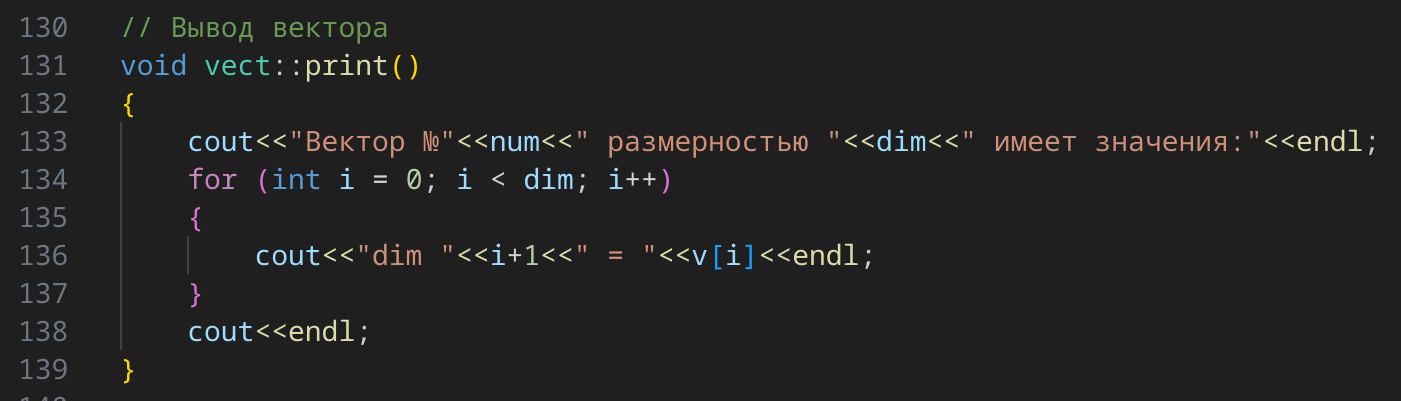


Figure 8: vect: Функция print

После - я описал функция сложения векторов (Рис. [9](#fig:9)), которая получает на вход вектор r (Функция бинарная, первый (левый вектор) - тот, к которму применяется данный метод). С помощью конструктора создания нулевого вектора создаю вектор tmp размерностью dim. Далее в цикле for меняю значения массива tmp.v[i]=0 на сумму v[i]+r.v[i]. В конце вывожу сообщение, что создан вектор со значением суммы векторов №num + №r.num. (“Создан” я написал потому, что вектор tmp внутри данного метода действительно создаётся конструктором копирования). Возвращается вектор tmp (Результатом работы метода является объект класса vect).

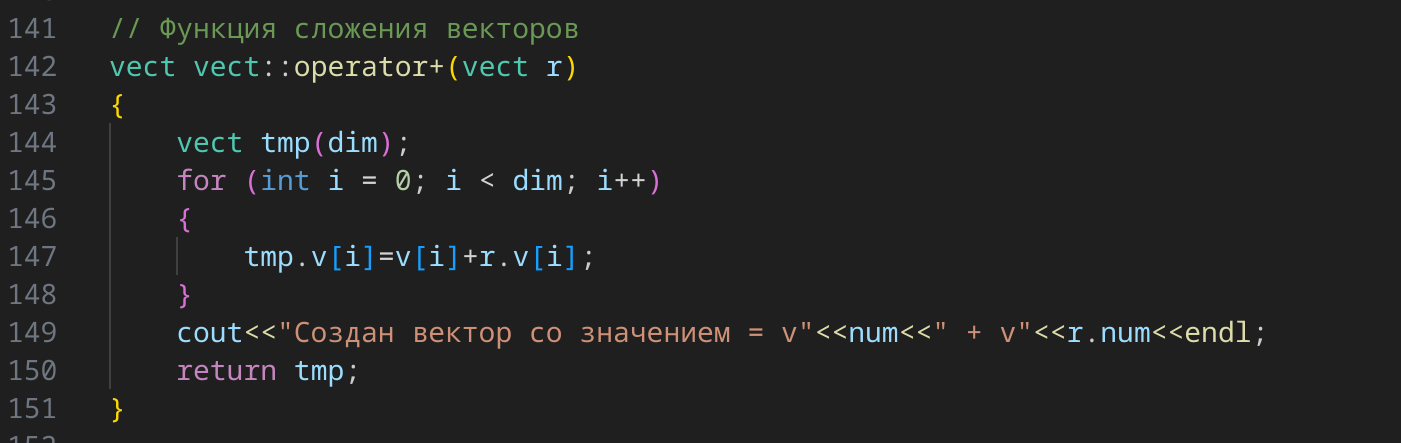


Figure 9: vect: Функция сложения векторов

Затем я описал функцию бинарного вычитания векторов (Рис. [10](#fig:10)), которая получает на вход ссылки на уменьшаемый вектор l и вычитаемый вектор r. С помощью конструктора создания нулевого вектора создаю вектор tmp размерностью l.dim. Далее в цикле for меняю значения массива tmp.v[i]=0 на разность l.v[i]-r.v[i]. В конце вывожу сообщение, что создан вектор со значением разности векторов №l.num - №r.num.Возвращается вектор tmp.

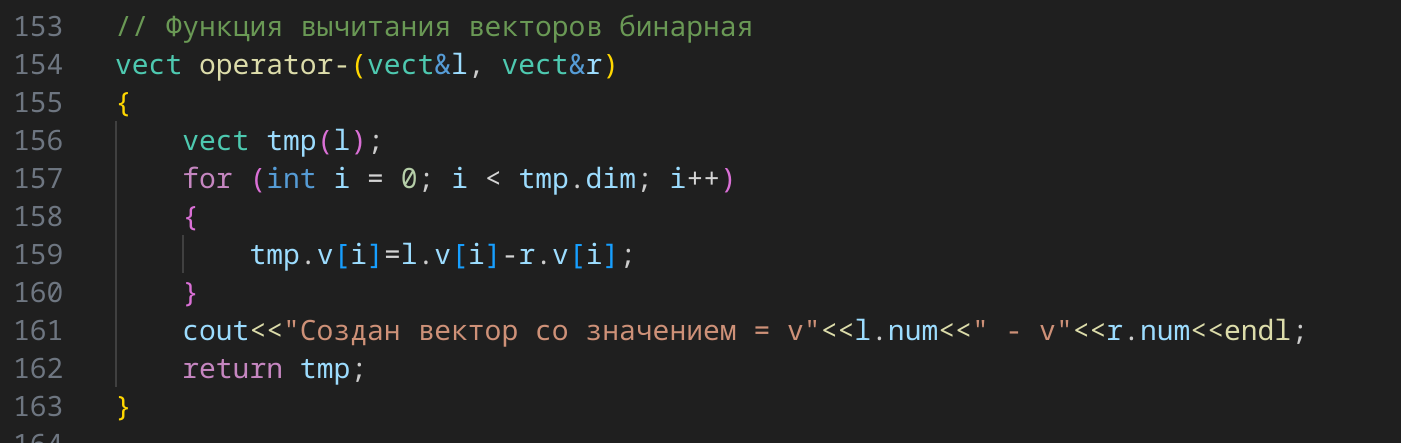


Figure 10: vect: Функция вычитания векторов бинарная

Потом я описал функцию унарного вычитания вектора (Рис. [11](#fig:11)). С помощью конструктора создания нулевого вектора создаю вектор tmp размерностью dim. Далее в цикле for меняю значения массива tmp.v[i]=0 на -1 \* v[i]. В конце вывожу сообщение, что создан вектор со значением -1 \* №num.Возвращается вектор tmp.

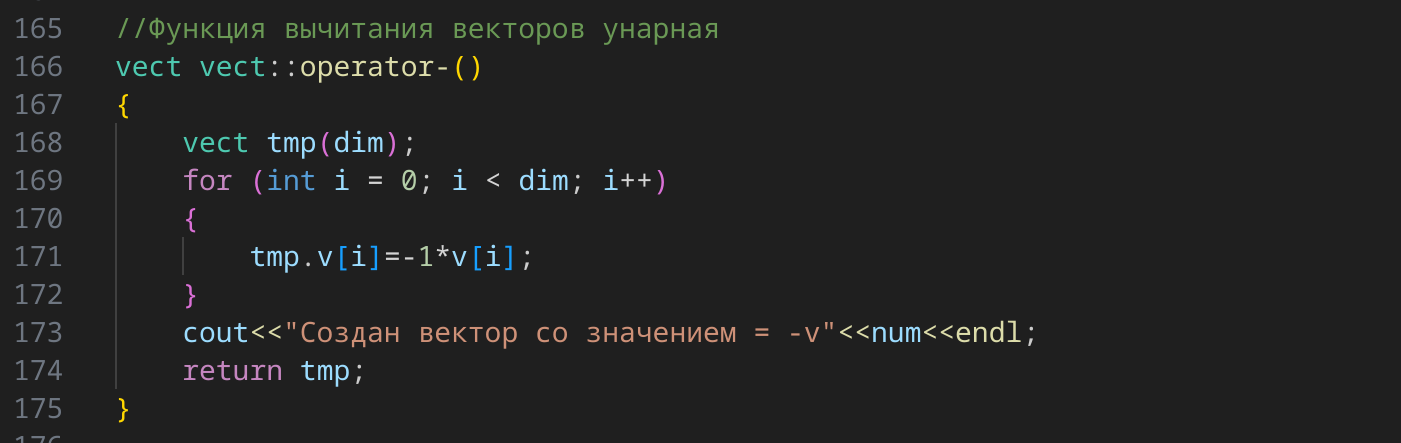


Figure 11: vect: Функция вычитания вектора унарная

После этого я описал функцию присвоения вектору значения другого вектора (Рис. [12](#fig:12)), которая получает на вход ссылку на const вектор r. Параметру dim вектора, к которому применён метод, задаём значение r.dim. Затем с помощью цикла for значениям v[i] присваиваем r.v[i]. В конце вывожу сообщение, что вектору num присвоено значение вектора r.num. Возвращается вектор, к которому применён метод (с помощью указателя this).

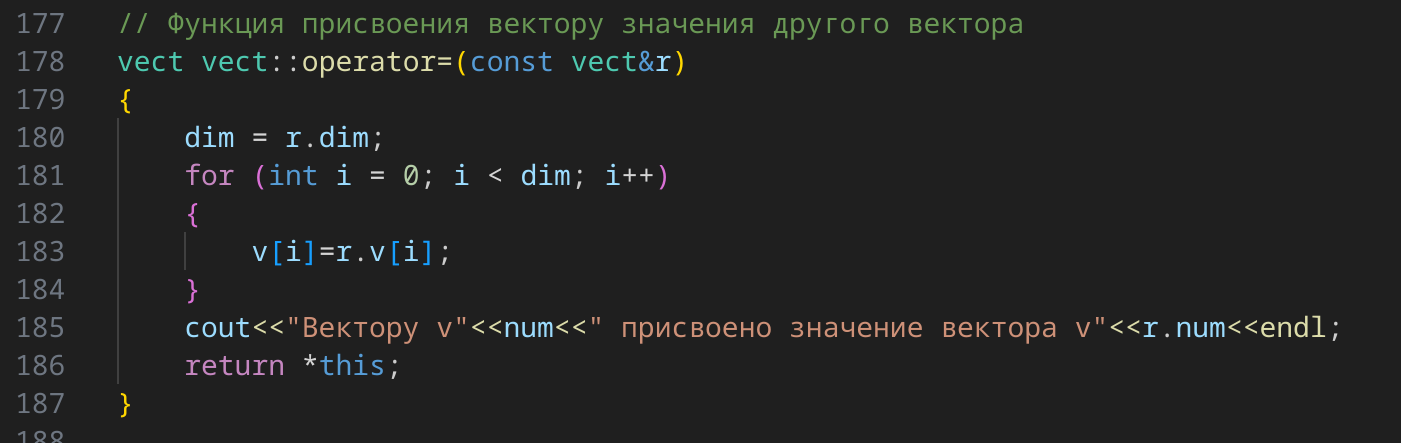


Figure 12: vect: Функция присвоения

Далее я описал функцию скалярного умножения векторов (Рис. [13](#fig:13)), которая получает на вход ссылку на вектор r. Объявляю и описываю вещественное число tmp = 0. Потом с помощью цикла for, суммирую в tmp все произведения вида v[i]*r.v[i]. В конце вывожу сообщение, что создана скалярная переменная со значением №num*  №r.num. Возвращаю значение tmp.

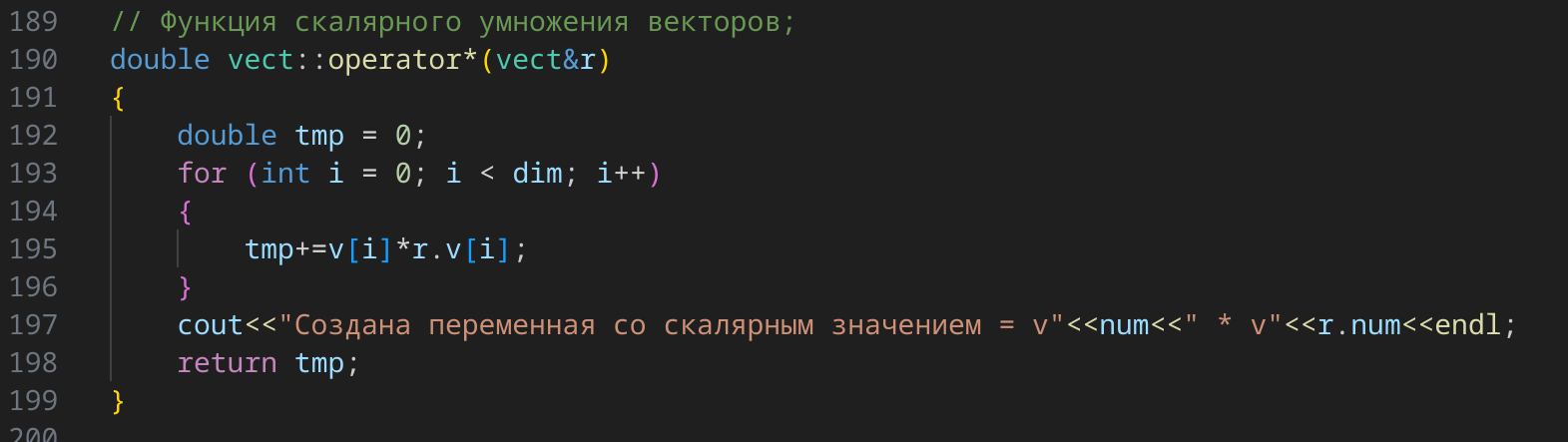


Figure 13: vect: Функция присвоения

Затем я описал функцию умножения вектора на число (Рис. [14](#fig:14)), которая получает на вход вещественное число k, и ссылку на вектор r. С помощью конструктора создания нулевого вектора создаю вектор tmp размерностью r.dim. Далее в цикле for меняю значения массива tmp.v[i]=0 на k \* r.v[i]. В конце вывожу сообщение, что создан вектор со значением k \* №num.Возвращается вектор tmp.

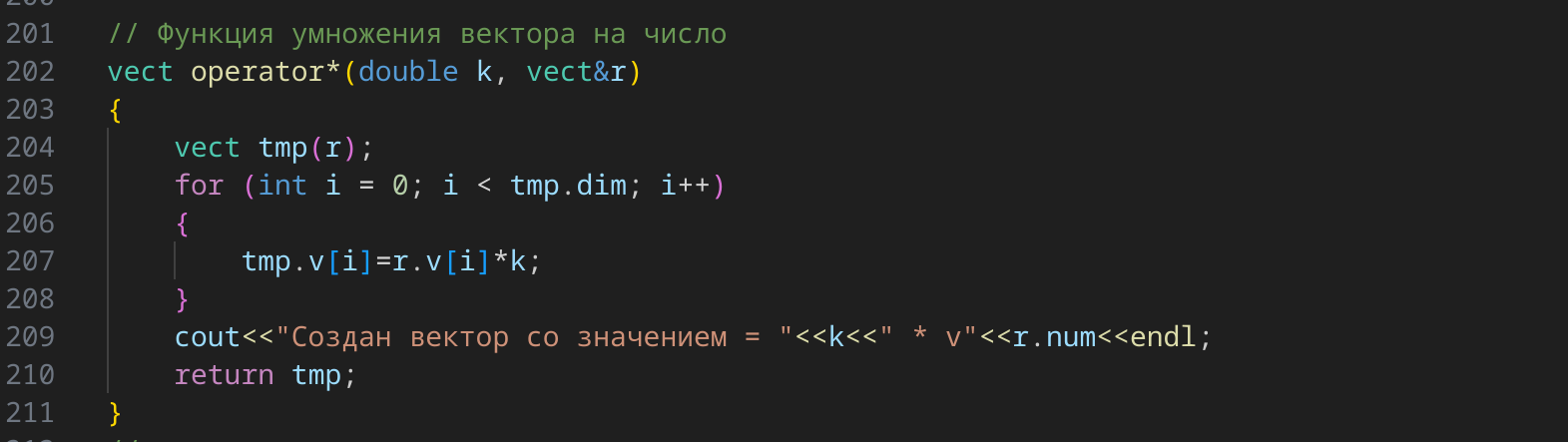


Figure 14: vect: Функция умножения вектора на число

**Описание конструкторов, деструктора и методов класс matr**

Описание я начал с конструктора матрицы по умолчанию (Рис. [15](#fig:15)). Увеличиваю значение count на 1 и присваиваю его параметру num. Затем пареметру dim присваиваю значение 3 (таким образом матрица будет 3х3). А параметру v c помощью оператора new задаю значение указателя на динамическую память в которой находится массив размером в dim из указателей на массивы вещественных чисел - таким образом v - указатель на двумерный массив вещественных чисел. Далее с помощью двух циклов for заполняю массив единицами по основной диагонали (где i=j, т.е номер строки равен номеру ряда) и нулями на всех остальных позициях. Таким образом, конструктор по умолчанию создал единичную матрицу размерностью 3х3.

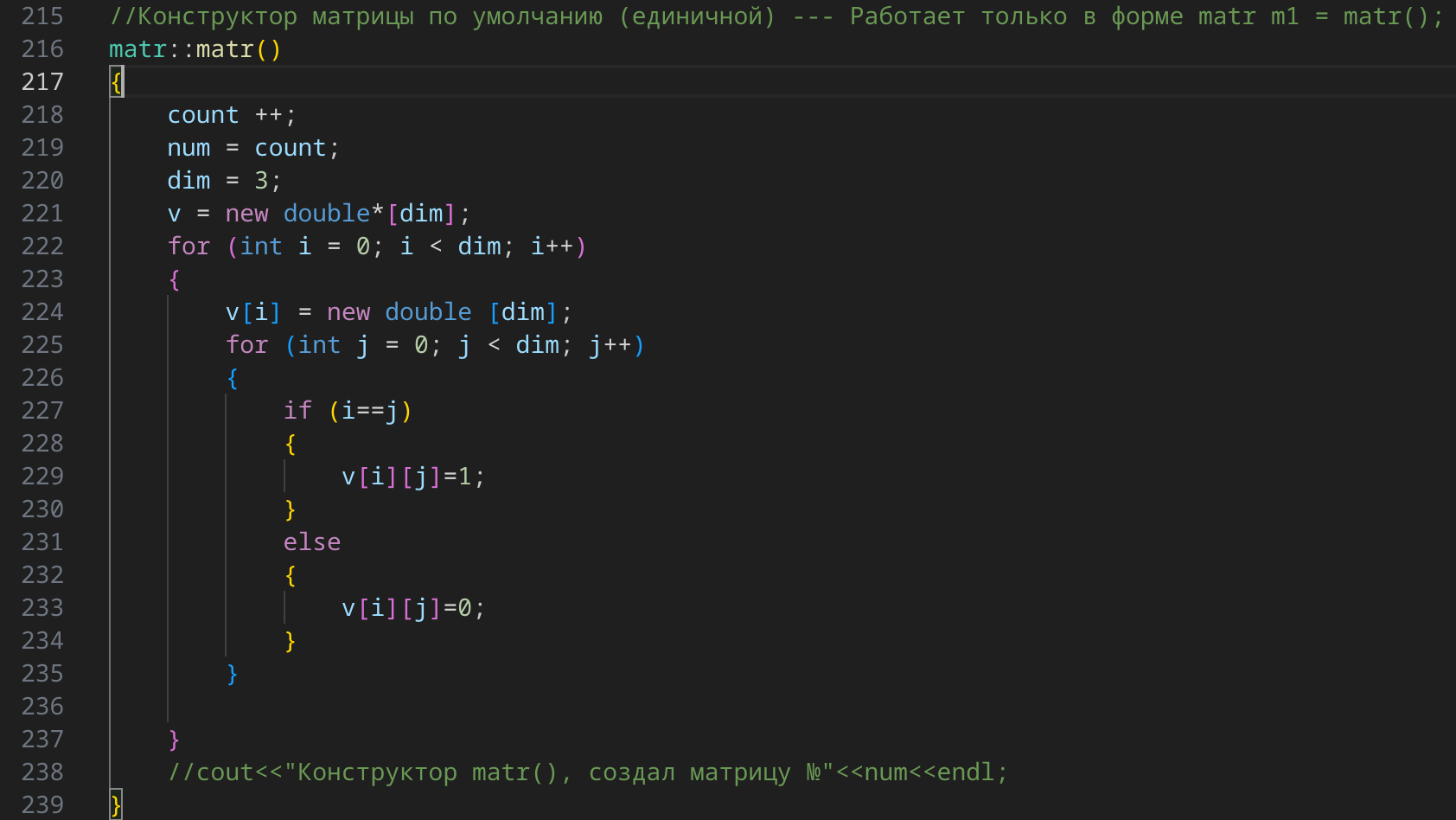


Figure 15: matr: Конструктор по умолчанию

После - я описал конструктор копирования матрицы (Рис. [16](#fig:16)), который получает на вход ссылку на объект класса matr - r. Увеличиваю значение count на 1 и присваиваю его параметру num. Затем параметру dim присваиваю значение dim матрицы r (r.dim). Выделяем динамическую память под двумерный массив v, и каждому из значений массива v[i][j] присваиваю соответствующее значение r.v[i][j]. Таким образом, конструктор по умолчанию создал матрицу-копию r.

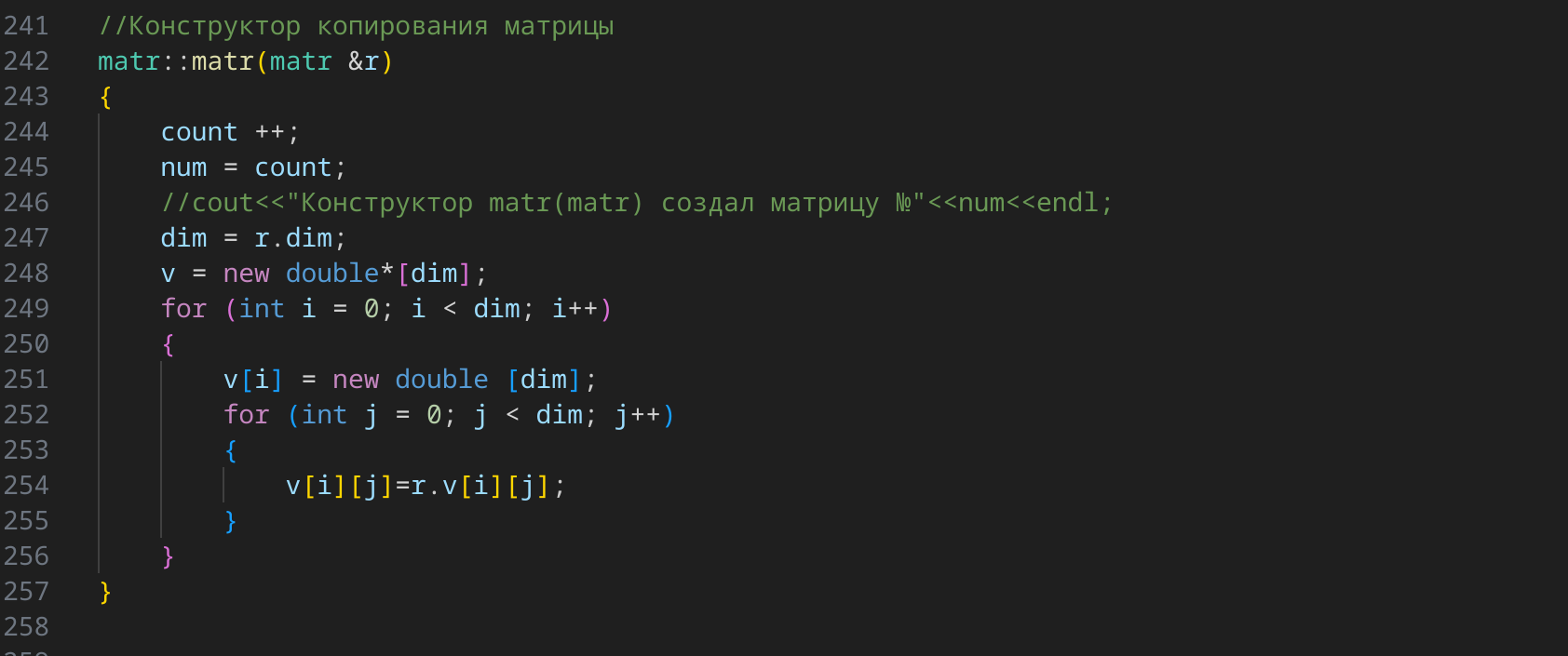


Figure 16: matr: Конструктор копирования

Далее я описал конструктор создания нулевой матрицы (Рис. [17](#fig:17)), который получает на вход целое число n - размер будущей матрицы. Увеличиваю значение count на 1 и присваиваю его параметру num. Затем пареметру dim присваиваю значение n. Выделяем динамическую память под двумерный массив v, и каждому из значений массива v[i][j] присваиваю значение 0. Таким образом, конструктор создания нулевой матрицы создал нулевую матрицу размерностью nхn.

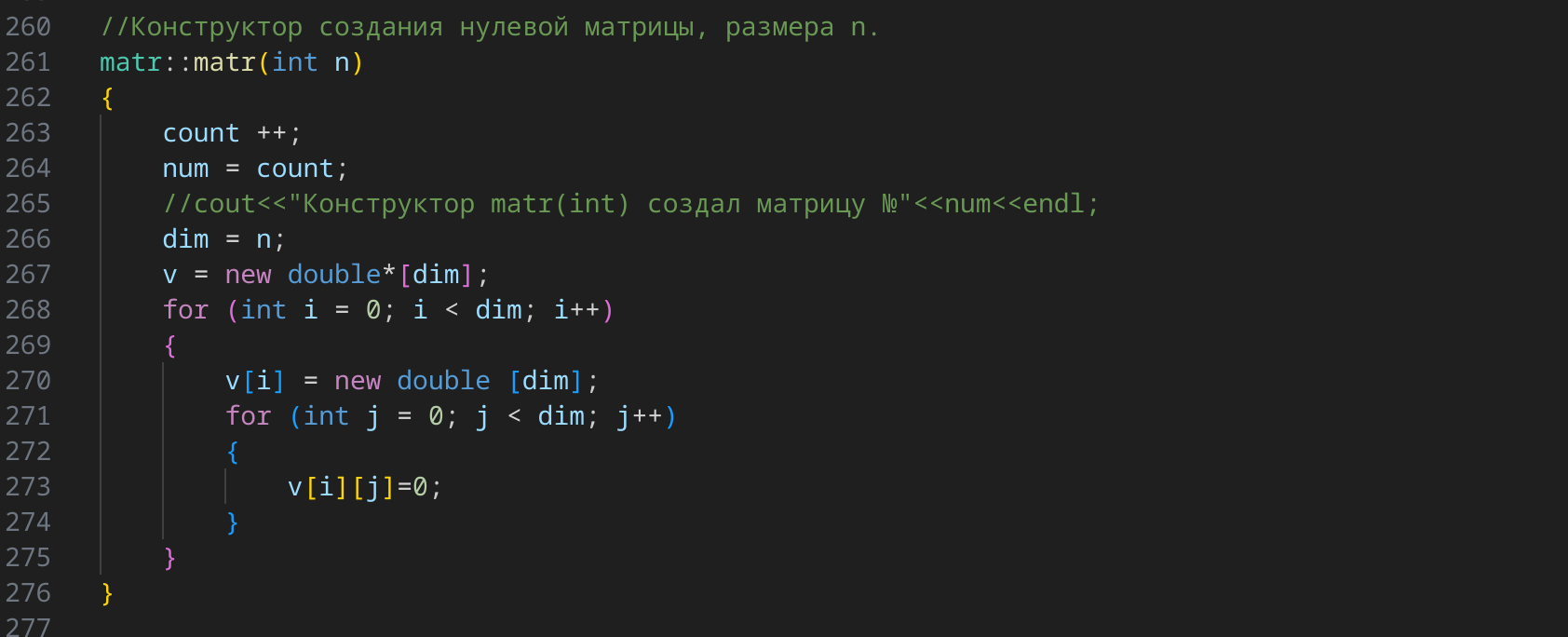


Figure 17: matr: Конструктор создания нулевой матрицы

После - я описал конструктор создания матрицы с компонентами (Рис. [18](#fig:18)), который полуает на вход целое число n - размер будущей матрицы и указатель на двумерный массив вещественных числе \*\*x - компоненты будущей матрицы. Увеличиваю значение count на 1 и присваиваю его параметру num. Затем пареметру dim присваиваю значение n. Выделяем динамическую память под двумерный массив v, и каждому из значений массива v[i][j] присваиваю значение x[i][j]. Таким образом, конструктор создания матрицы с компонентами создал матрицу размерностью nxn, с компонентами двумерного массива x.

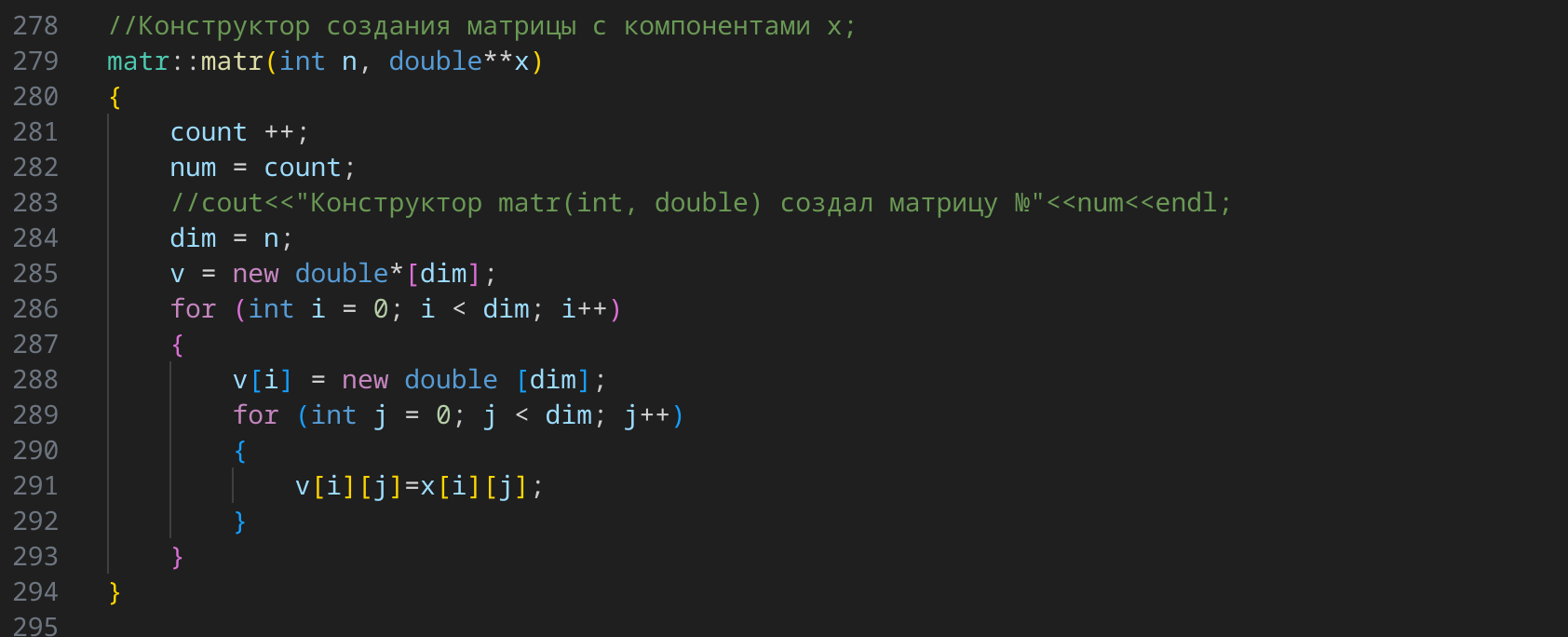


Figure 18: matr: Конструктор создания матрицы с компонентами

Далее я описал деструктор матрицы (Рис. [19](#fig:19)). С помощью цикла for и оператора delete освобождаю динамическую память, на которую указывет указатель v. Таким образом деструктор вектора ликвидировал матрицу №num и освободил соотствующую ей динамическую память.

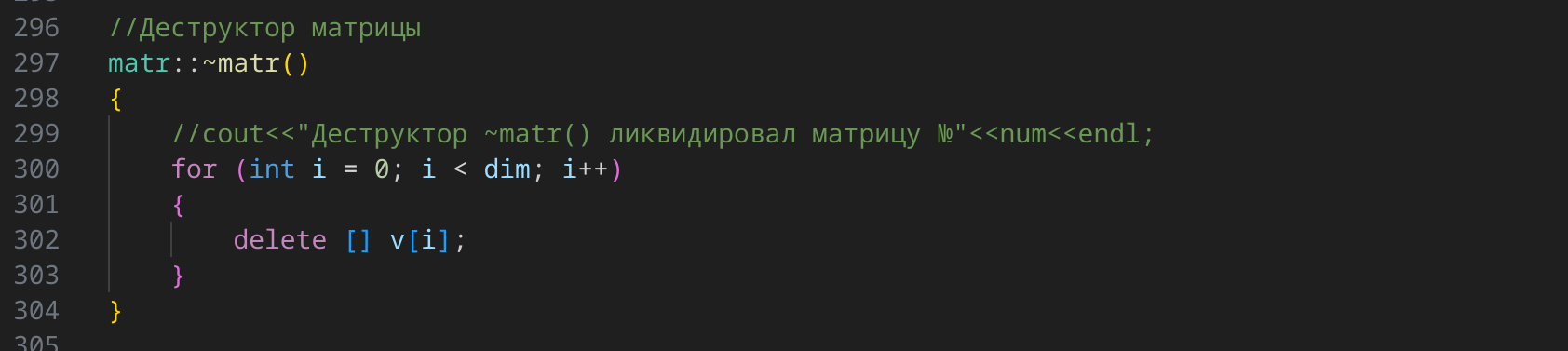


Figure 19: matr: Деструктор матрицы

Затем я описал функцию вывода матрицы (Рис. [20](#fig:20)). Вывожу сообщение, указывающее номер выводимой матрицы и её размерность. Потом с помощью двух циклов for вывожу каждое из значений компонент матрицы.

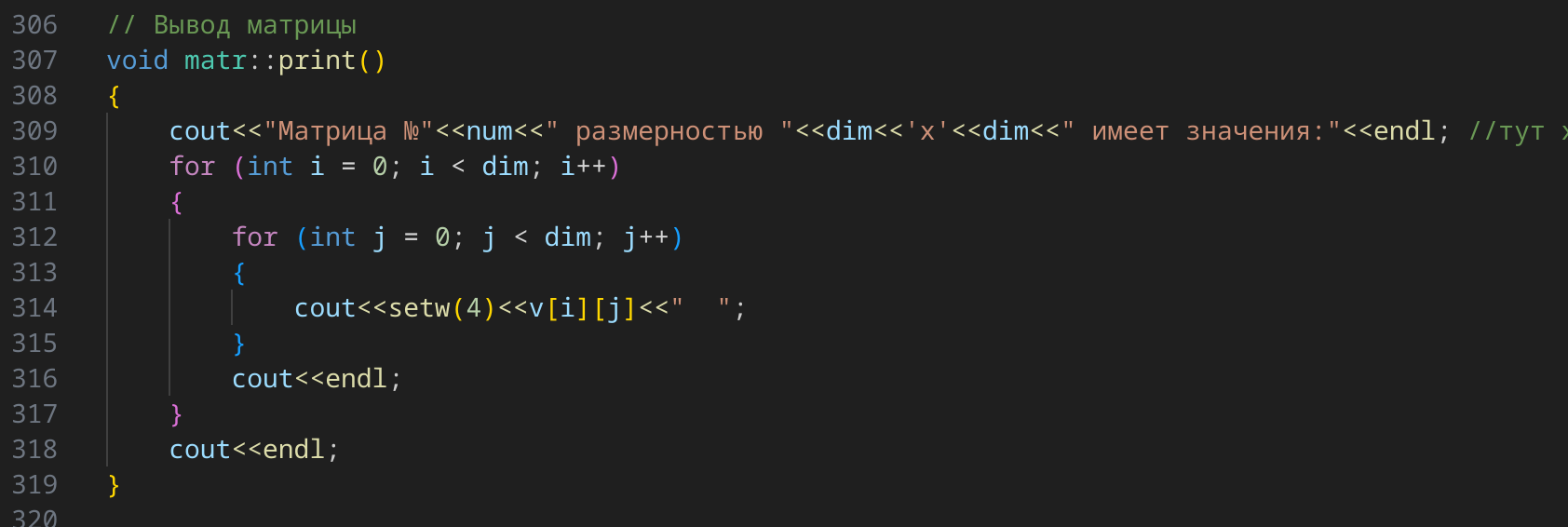


Figure 20: matr: Функция print

После - я описал функцию сложения матриц (Рис. [21](#fig:21)), которая получает на вход матрицу r. С помощью конструктора создания нулевой матрицы создаю матрицу tmp размерностью dimxdim. Далее в цикле for меняю значения массива tmp.v[i][j]=0 на сумму v[i][j]+r.v[i][j]. Возвращается матрица tmp (Результатом работы метода является объект класса matr).

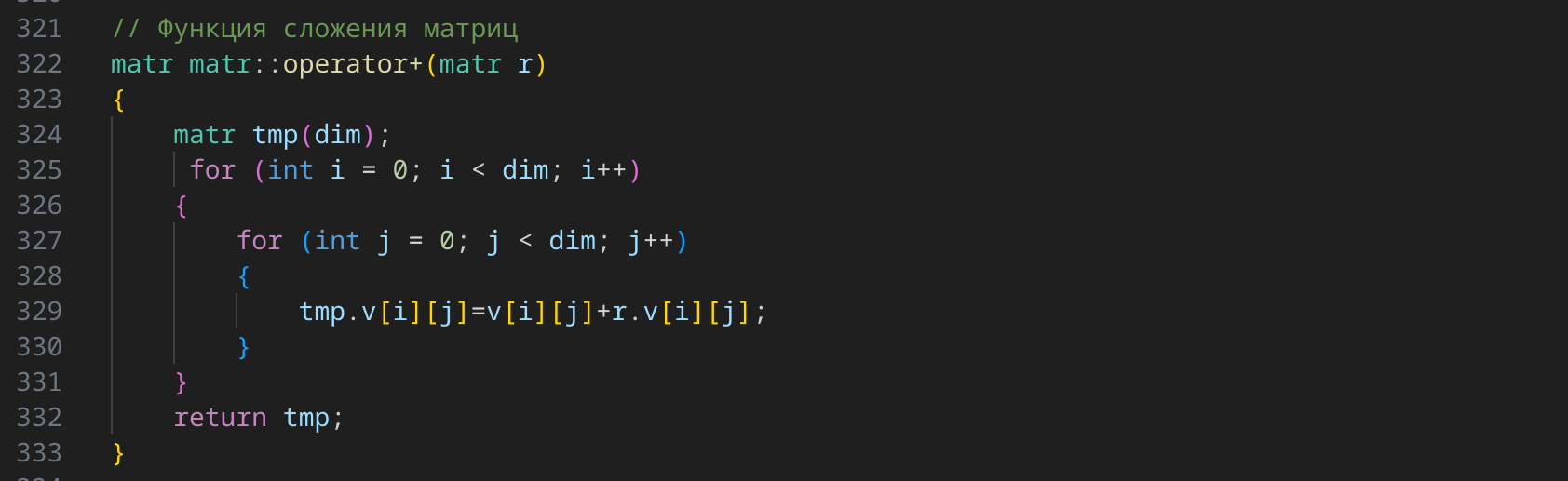


Figure 21: matr: Функция сложения матриц

Затем я описал функцию бинарного вычитания матриц (Рис. [22](#fig:22)), которая получает на вход ссылку на вычитаемую матрицу r. С помощью конструктора создания нулевой матрицы создаю матрицу tmp размерностью dimxdim. Далее в циклах for меняю значения массива tmp.v[i][j]=0 на разность v[i][j]-r.v[i][j]. Возвращается матрица tmp.

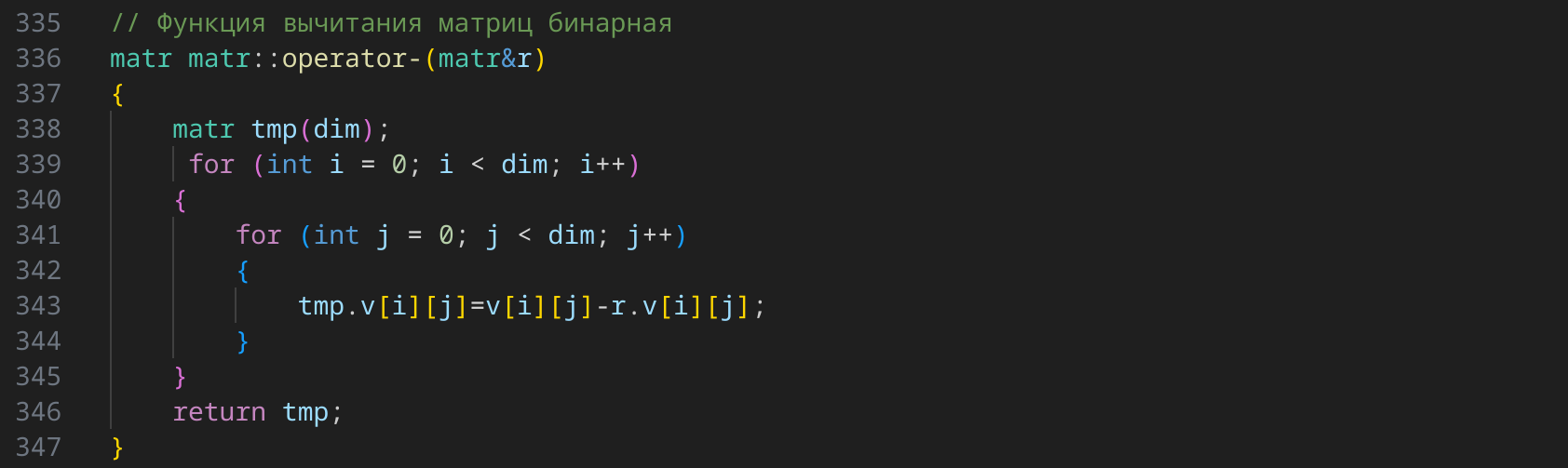


Figure 22: matr: Функция вычитания матриц бинарная

Потом я описал функцию унарного вычитания матрицы (Рис. [23](#fig:23)). С помощью конструктора создания нулевой матрицы создаю матрицу tmp размерностью dimxdim. Далее в циклах for меняю значения массива tmp.v[i]=0 на -1 \* v[i]. Возвращается матрица tmp.

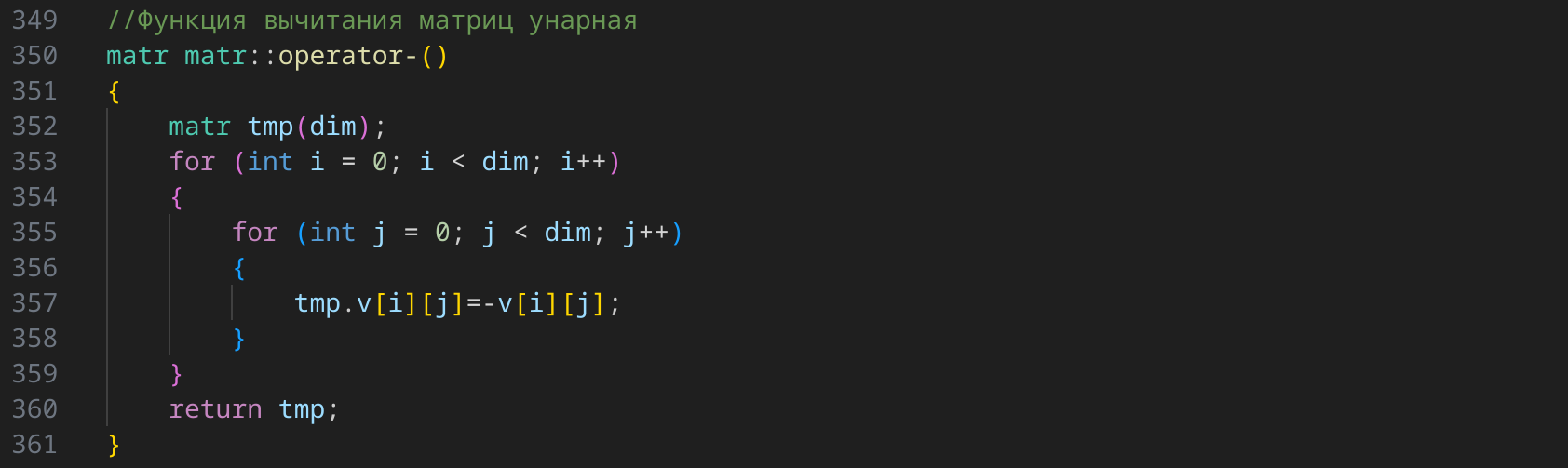


Figure 23: matr: Функция вычитания матрицы унарная

После этого я описал функцию присвоения матрице значения другой матрицы (Рис. [24](#fig:24)), которая получает на вход ссылку на const матрицу r. Параметру dim матрицы, к которому применён метод, задаём значение r.dim. Затем с помощью циклов for значениям v[i][j] присваиваем r.v[i][j]. Возвращается матрица, к которой применён метод (с помощью указателя this).

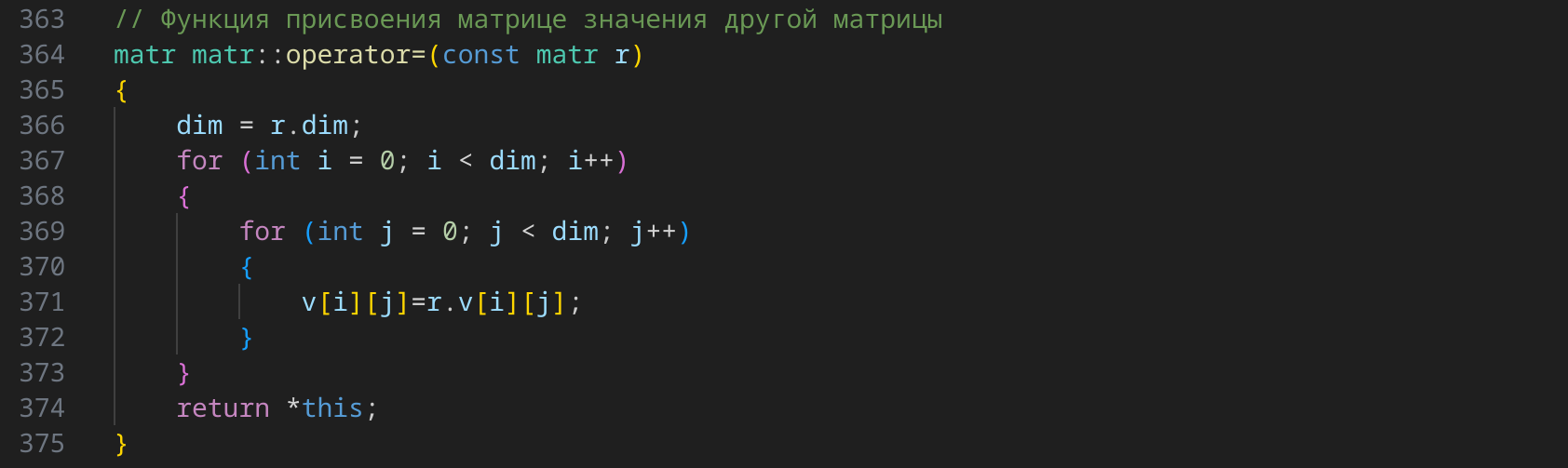


Figure 24: matr: Функция присвоения

Далее я описал функцию перемножения матриц (Рис. [25](#fig:25)), которая получает на вход ссылку на матрицу r. С помощью конструктора создания нулевой матрицы создаю матрицу tmp размерностью dimxdim. Далее с помощью трёх циклов for в каждом значении tmp.v[i][j] суммируем значения вида v[i][k]\*r.v[k][j] (где k принадлежит [0;dim-1]). Возвращается матрица tmp.

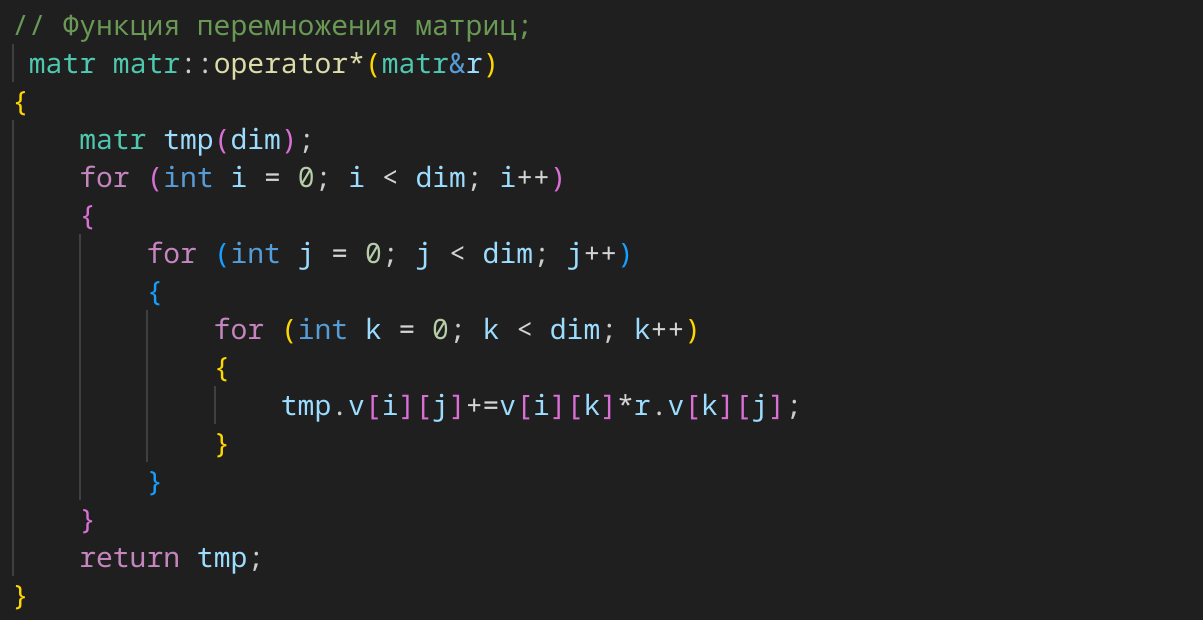


Figure 25: matr: Функция присвоения

Затем я описал функцию умножения матрицы на число (Рис. [26](#fig:26)), которая получает на вход вещественное число k, и ссылку на матрицу r. С помощью конструктора создания нулевой матрицы создаю матрицу tmp размерностью r.dimxr.dim. Далее в циклах for меняю значения массива tmp.v[i][j]=0 на k \* r.v[i][j]. Возвращается матрица tmp.

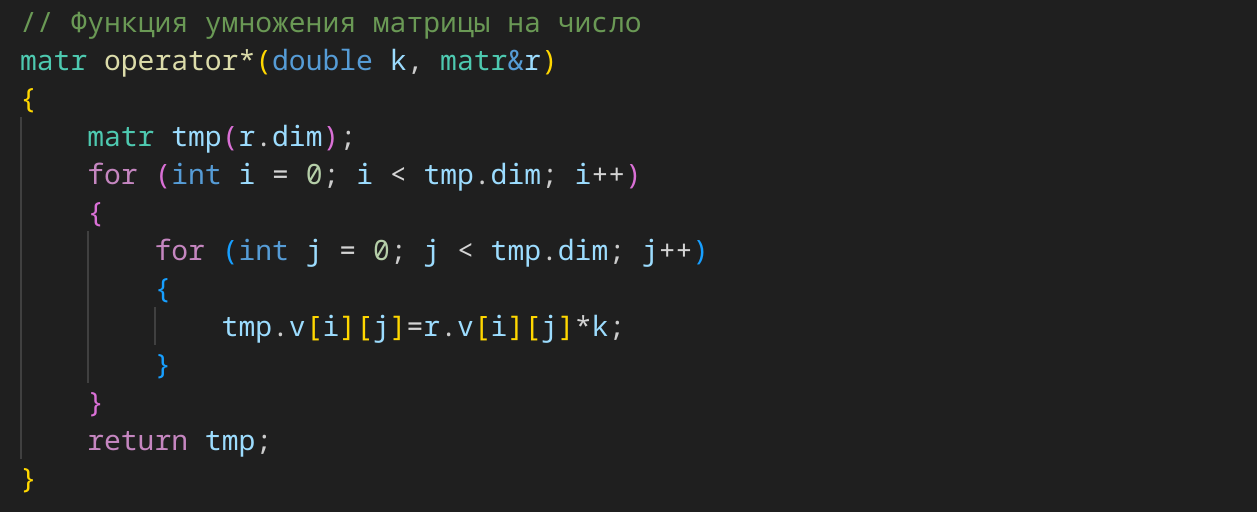


Figure 26: matr: Функция умножения матрицы на число

Затем я описал функцию умножения матрицы на вектор-столбец (Рис. [27](#fig:27)), которая получает на вход ссылку на вектор r. С помощью конструктора создания нулевого вектора создаю вектор tmp размерностью dim. Далее в циклах for меняю значения вектора tmp.v[i]=0 на сумму всех элементов вида v[i][j]\*r.v[i] (где j принадлежит [0;dim-1]). Возвращается вектор tmp.

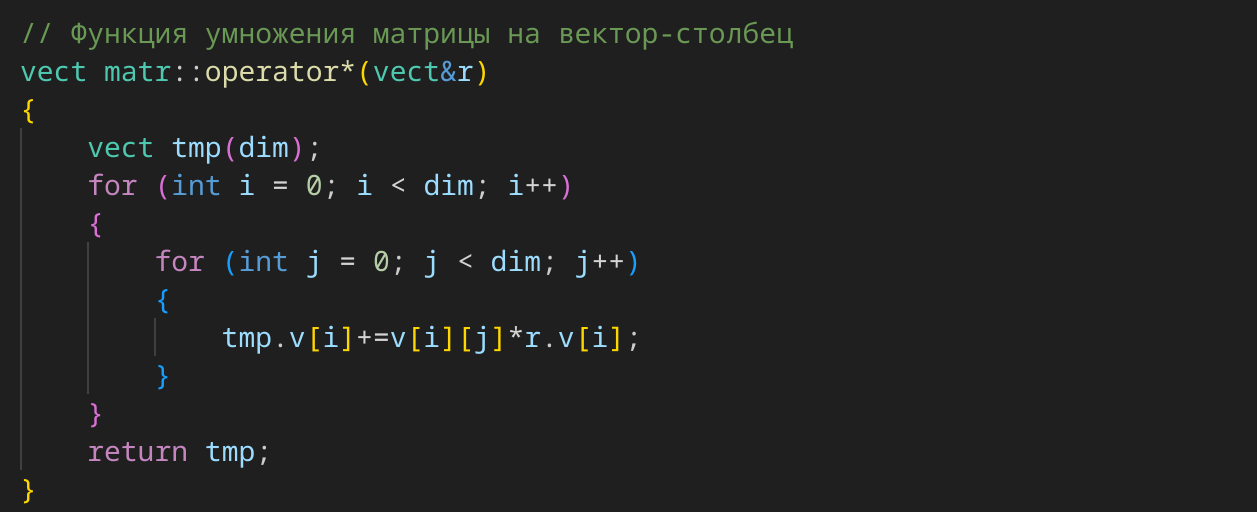


Figure 27: matr: Функция умножения матрицы на вектор-столбец

**Функция main**

В функции main я привёл пример работы каждого конструктора и метода классов vect и matr. Также привёл 2 более сложных примера, показывающих, что работает двойное присваивание и сложные операции. ( К примеру matr1 + 3\*mart2).

# 3 Выводы

В ходе лабораторной работы я написал программу на с++, содержащую описание классов vect и matr, конструкторы и деструктор для каждого класса, набор оператор-функций для операций векторной алгебры и функцию main, использующую вышеописанный инструментарий.