Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра экономической математики, информатики и статистики (ЭМИС)

ШАБЛОНЫ КЛАССОВ

Отчет по лабораторной работе по дисциплине «Программирование»

Студентка гр.548-1 К.Е.Анисимова

«\_\_» \_\_\_\_\_\_ 2019 г.

Доцент кафедры ЭМИС

Шельмина Е. А.

«\_\_»\_\_\_\_\_\_\_ 2019г.

Томск 2019

**Лабораторная работа №9**

**Шаблоны классов**

Цель работы: освоить и применить на практике шаблоны классов

Теоритические сведения

Рассматриваемый далее пример - класс, который хранит пару значений. Функции-элементы этого класса возвращают минимальное и максимальное значения, а также позволяют определить, являются ли два значения одинаковыми. Поскольку перед нами шаблон класса, то тип значения может быть почти любым.

template <class T>

class Pair

{

T a, b;

public:

Pair (T t1, T t2);

T Max();

T Min ();

int isEqual ();

};

Пока все выглядит также , как и для шаблонов функций. Единственная разница состоит в том, что вместо описания функции используется объявление класса. Шаблоны классов становятся все более сложными, когда вы описываете принадлежащие функции класса. Вот, например, описание принадлежащей функции Min() класса Pair:

template <class T>

T Pair <T>::Min()

{return a < b ? a : b;}

Чтобы понять эту запись, давайте вернемся немного назад. Если бы Pair был обычным классом (а не шаблоном класса) и T был бы некоторым конкретным типом, то функция Min класса Pair была бы описана таким образом:

T Pair::Min()

{return a < b ? a : b;}

Для случая шаблонной версии нам необходимо, во-первых, добавить заголовок шаблона template <class T>.

Затем нужно дать имя классу. Помните, что на самом деле мы описываем множество классов - семейство Pair. Повторяя синтаксис префикса (заголовка) шаблона, экземпляр класса Pair для целых типов, можно назвать Pair<int>, экземпляр для типа double - Pair<double>, для типа Vector - Pair<Vector>. Однако в описании принадлежащей классу функции нам необходимо использовать имя класса Pair<T>. Это имеет смысл, так как заголовок шаблона говорит, что Т означает имя любого типа.

Ранее уже отмечалось, что шаблоны функций могут работать только для тех (встроенных) типов данных или классов, которые поддерживают необходимые операции. То же самое справедливо и для шаблонов классов. Чтобы создать экземпляр класса Pair для некоторого классового типа, например для класса X, этот класс должен содержать следующие общедоступные функции:

X (X &);// конструктор копирования

int operator == (X)

int operator < (X);

Три указанные функции необходимы, так как они реализуют операции, выполняемые над объектами типа T в метода X шаблона класса Pair.

Если вы собираетесь использовать некоторый шаблон класса, как узнать какие операции требуются? Если шаблон класса снабжен документацией, то эти требования должны быть в ней указаны. В противном случае придется читать первичную документацию - исходный текст шаблона. При этом учитывайте, что встроенные типы данных поддерживают все стандартные операции.

Шаблоны классов не только для типов

Параметризовать некоторый класс так, чтобы он работал для любого типа данных - это только половина того, что шаблоны обеспечивают для классов. Другой аспект состоит в том, чтобы дать возможность задания числовых параметров. Это позволяет Вам, например, создавать объекты типов "Вектор из 20 целых", "Вектор из 1000 целых" или "Вектор из 10 переменных типа double".

Основная идея проста, хотя используемый синтаксис может показаться сложным. Давайте в качестве примера рассмотрим некоторый обобщенный класс Vector. Как и класс Pair, класс Vector содержит функции Min(), Max(), isEqual(). Однако в нем может быть любое количество участников, а не два. В классе Pair число участников фиксировано и задаются они в качестве аргументов конструктора. В шаблоне Vector вместо этого используется второй параметр заголовка шаблона.

Имеется несколько вариантов использования шаблонов с параметрами-значениями для динамического размещения массивов различных размеров. Например, можно передать размер массива конструктору. Указание размеров объекта во время конструирования или путем обращения к некоторому методу действительно обеспечивает задание размера, однако такой способ не позволяет создать отдельный тип для каждого отдельного размера. Подход с использованием шаблона гарантирует, что размер становится частью типа. Так, Vector с пятью элементами типа double является типом, отличным от Vector с четырьмя элементами типа double.

Хотите ли Вы, чтобы различные размеры были различными типами, зависит от ваших нужд. Если сравнение двух векторов с четырьмя и пятью координатами не имеет особого смысла, то было бы неплохо сделать их различными типами. Вместе с тем, в случае классов для матриц, Вы, возможно, не захотите иметь особый тип для каждого размера матриц, так как, например, умножение, работает с матрицами различных размеров. Если Вы столкнетесь с подобной проблемой, то Вам потребуются только разумные проверки времени выполнения, а не контроль типов при компиляции.

Хотя числовые параметры шаблонов часто используются для задания размеров различных элементов, как это было показано для класса Vector, этим их применение не ограничивается. Например, с помощью числовых параметров можно задавать диапазоны значений графических координат в графическом классе.

Наследование в шаблонах классов

Шаблоны классов, как и классы, поддерживают механизм наследования. Все основные идеи наследования при этом остаются неизменными, что позволяет построить иерархическую структуру шаблонов, аналогичную иерархии классов.

Рассмотрим очень простой пример, на котором продемонстрируем, каким образом можно создать шаблон класса, производный из нашего шаблона класса Pair. Пусть это будет класс Trio, в котором к паре элементов a и b из Pair, добавим элемент c.

template <class T>

class Trio: public Pair <T>

{

T c;

public:

Trio (T t1, T t2, T t3);

};

template <class T>

Trio<T>::Trio (T t1, T t2, T t3): Pair <T> (t1, t2), c(t3)

{}

// Заметим, что вызов родительского конструктора

// также сопровождается передачей типа Т в качестве параметра.

Задания

Задание 1. Разработать шаблон класса для работы с однонаправленным некольцевым списком. Реализовать следующие действия: добавление звена в начало списка; удаление звена из начала списка; добавление звена в произвольное место списка, отличное от начала (например, после звена, указатель на которое задан); удаление звена из произвольного места списка, отличного от начала (например, после звена, указатель на которое задан); проверка, пуст ли список; очистка списка; формирование списка; печать списка. Решить задачу: составить программу, которая в списке L заменяет первое вхождение списка L1 (если такое есть) на список L2.

Скриншоты кода программы и результат представлены на рисунках 1.1 – 1.

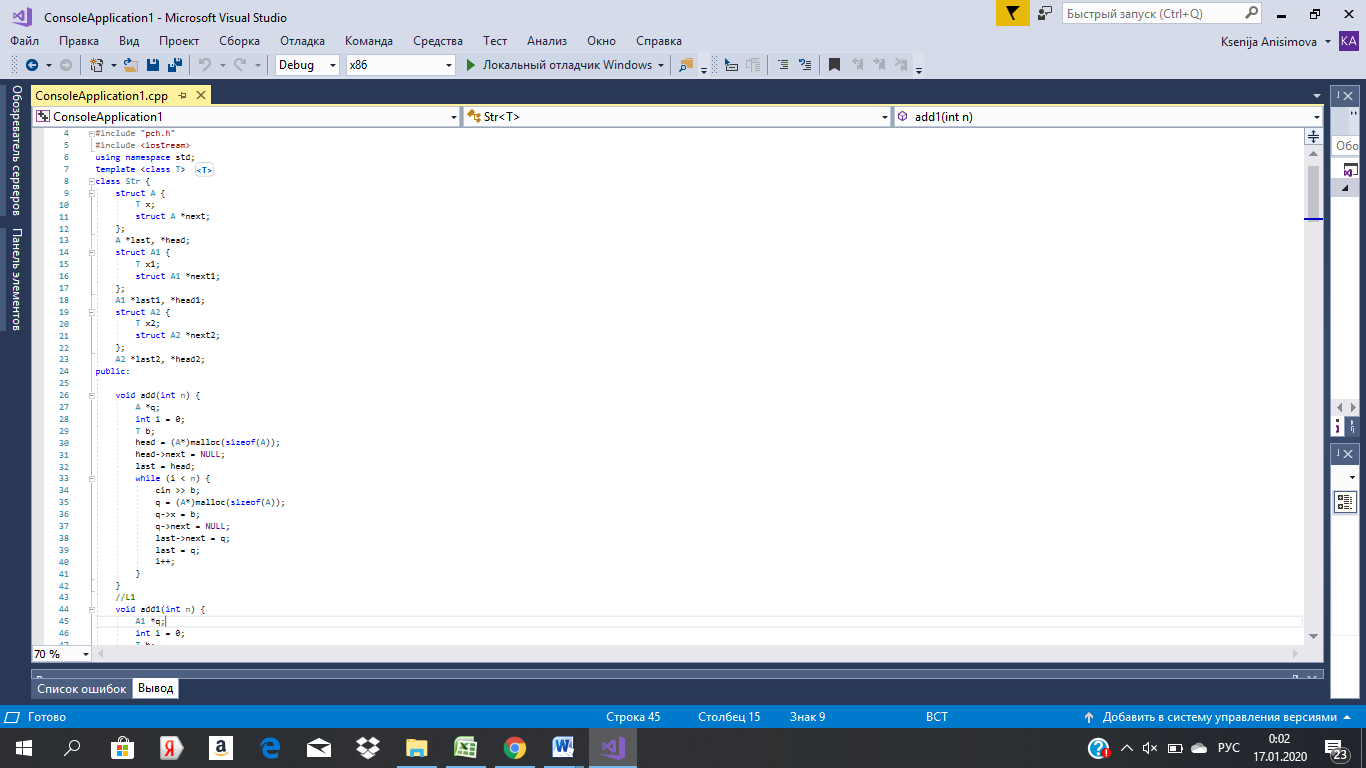


Рисунок 1.1 – скриншот кода программы

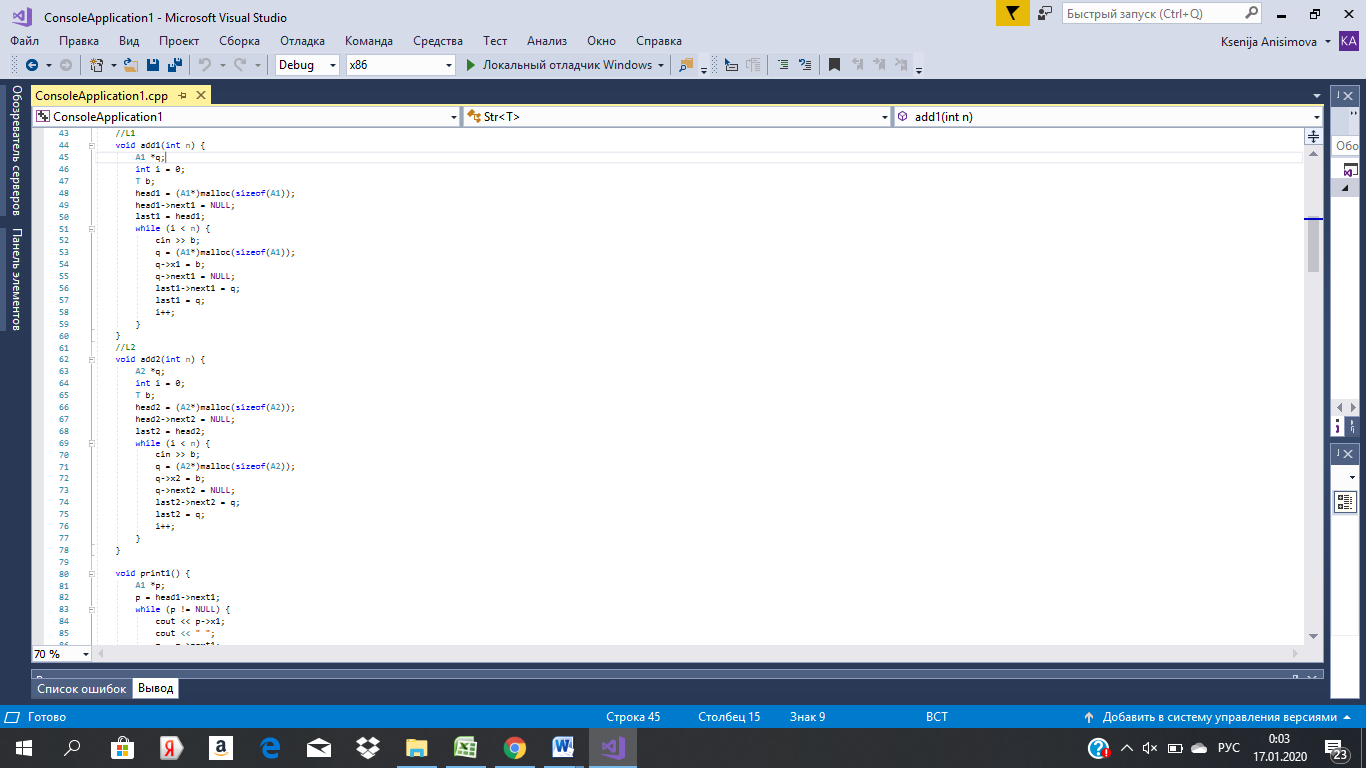


Рисунок 1.2 – скриншот кода программы

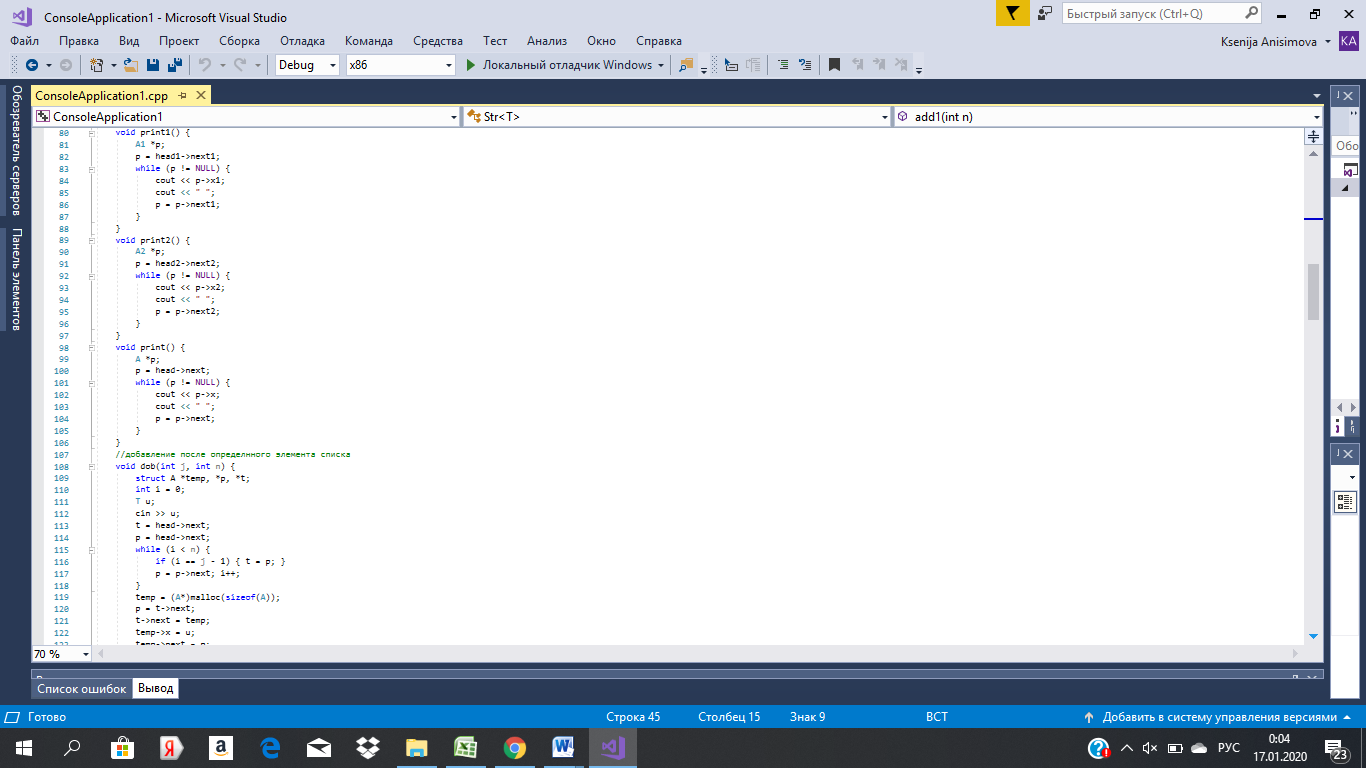


Рисунок 1.3 – скриншот кода программы

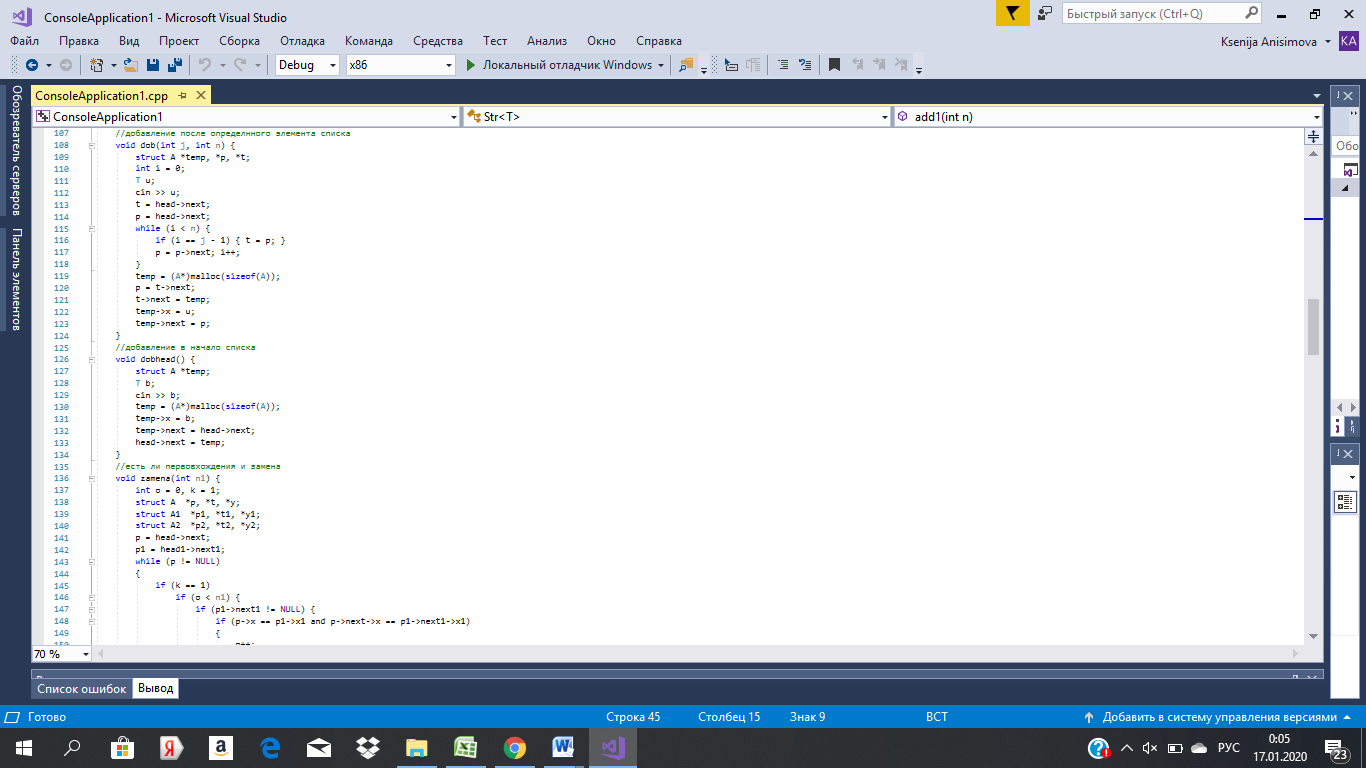


Рисунок 1.4 – скриншот кода программы

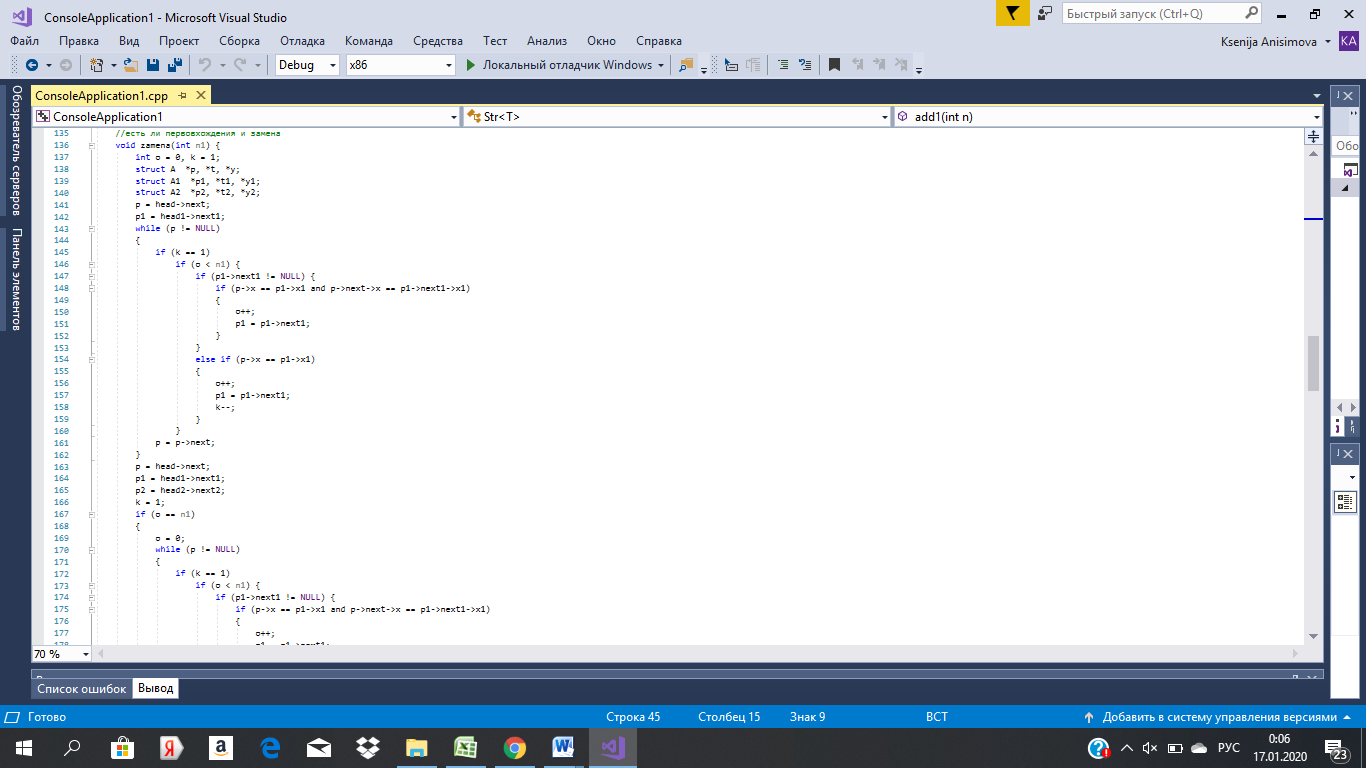


Рисунок 1.5 – скриншот кода программы

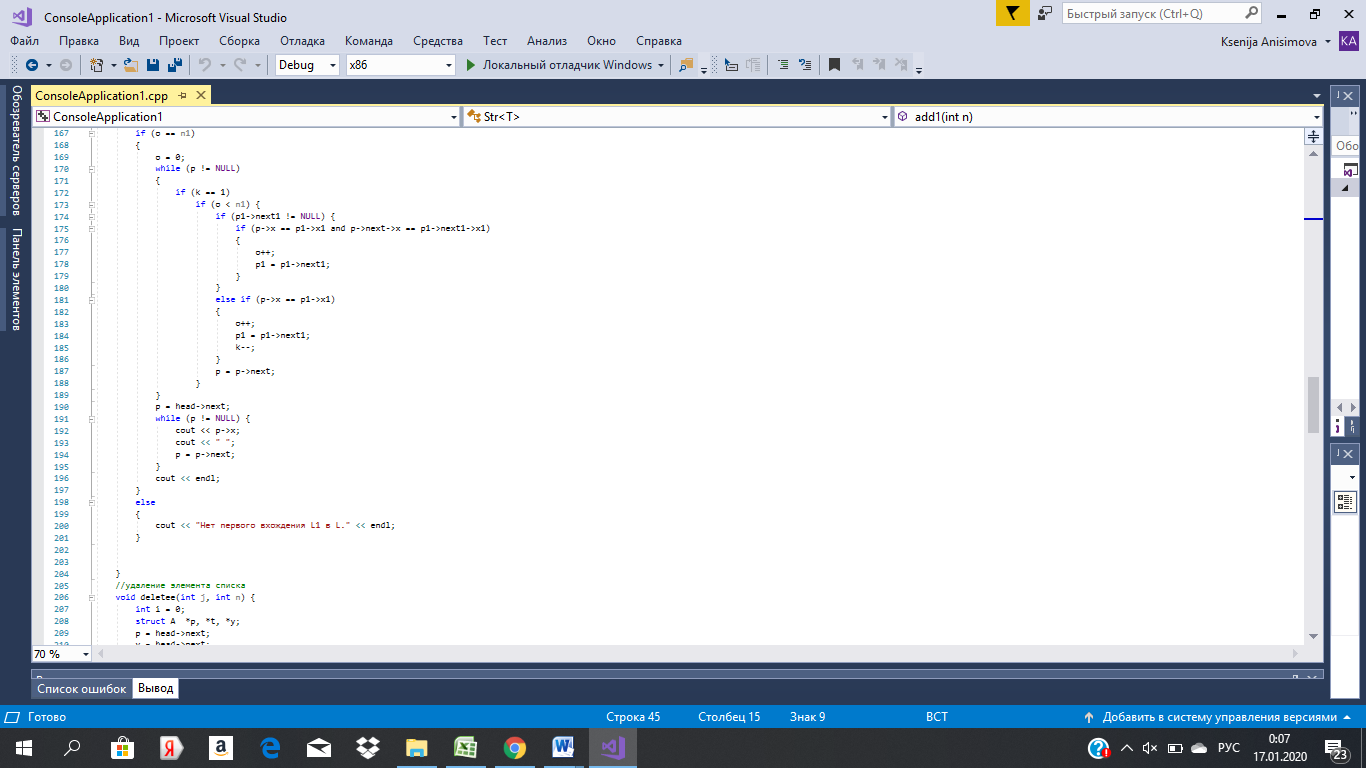


Рисунок 1.6 – скриншот кода программы

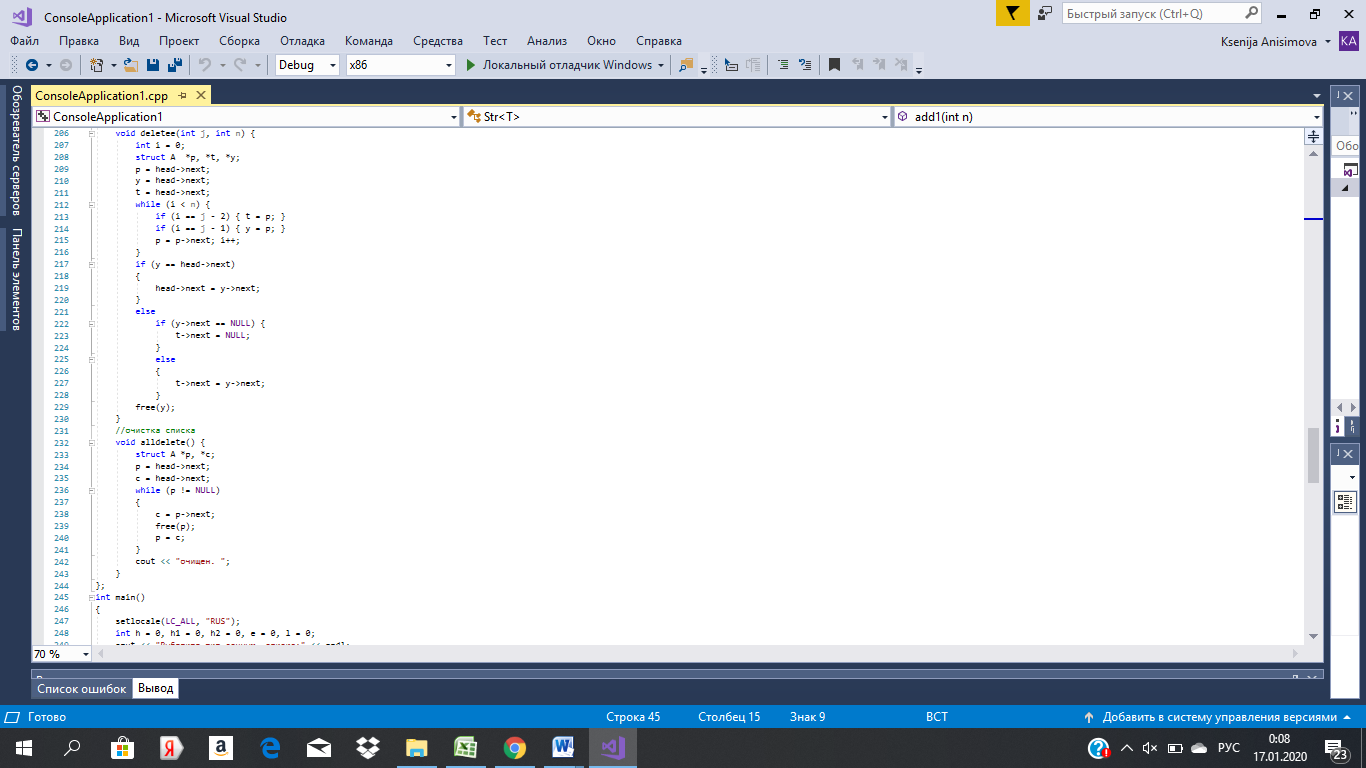


Рисунок 1.7 – скриншот кода программы

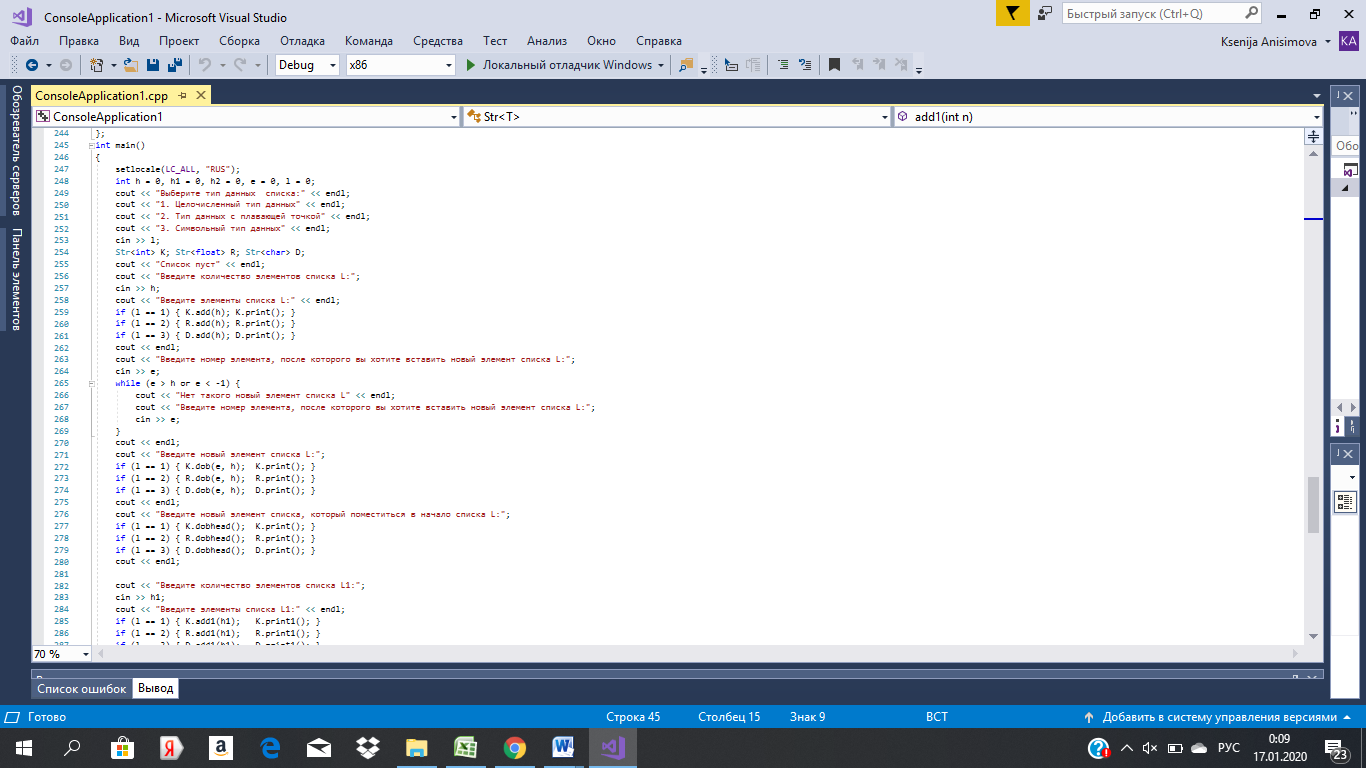


Рисунок 1.8 – скриншот кода программы

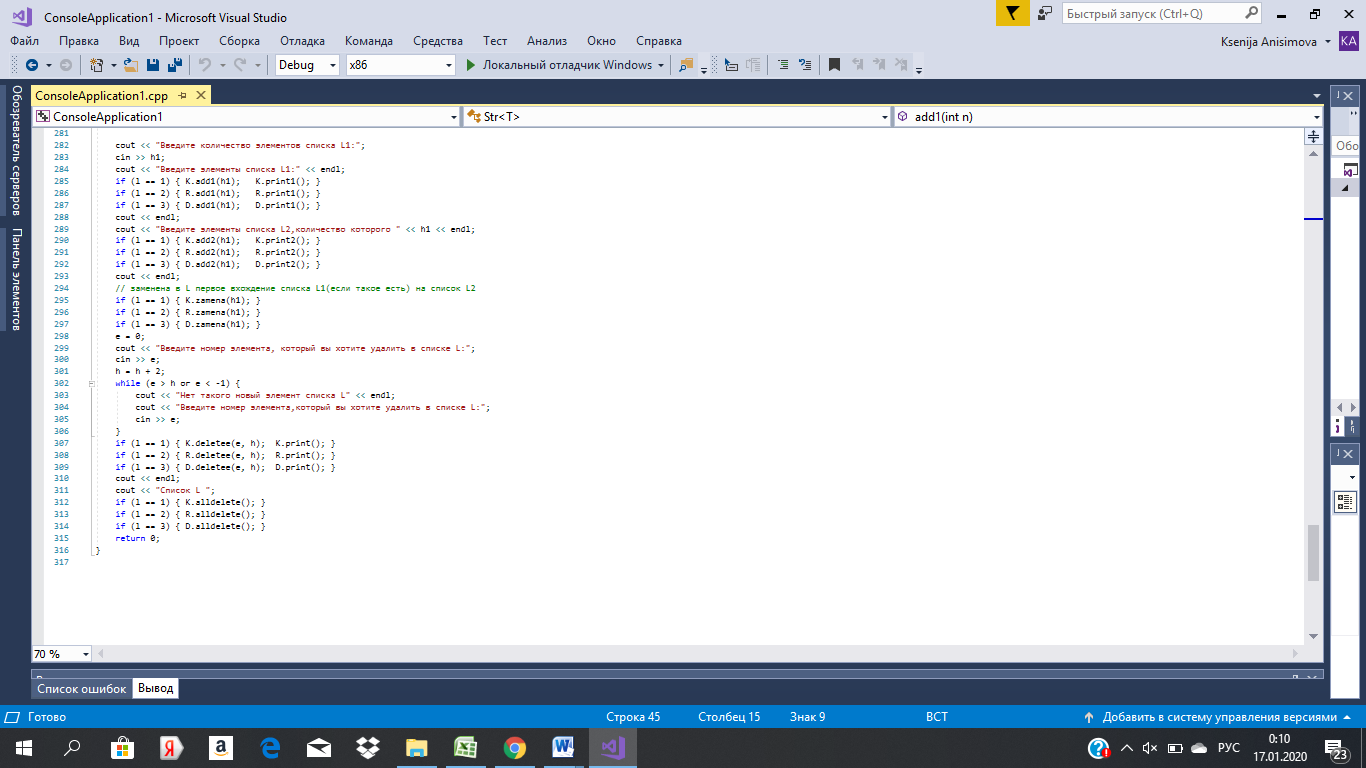


Рисунок 1.9 – скриншот кода программы

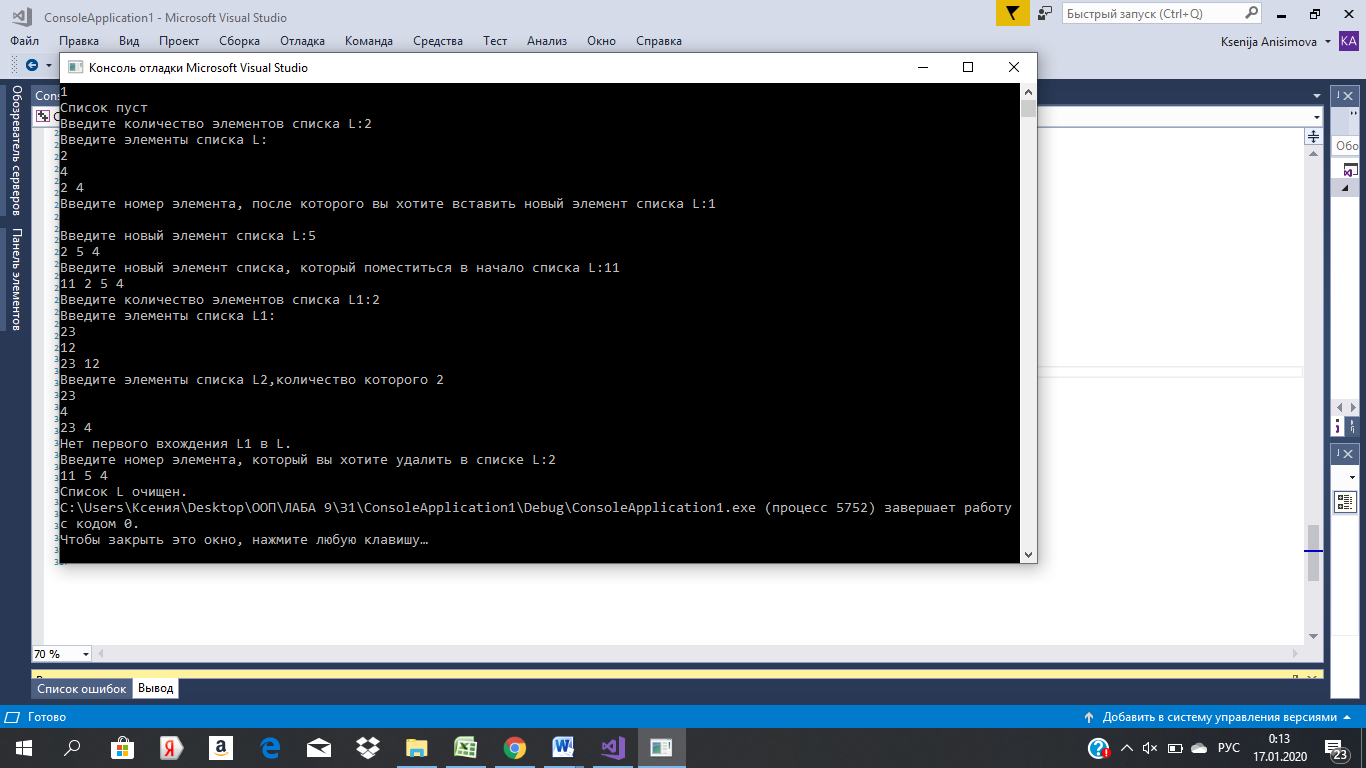


Рисунок 1.10 – скриншот результата программы

Вывод: освоены и применены на практике шаблоны классов.