Министерство образования и науки РФ

Тамбовский государственный технический университет

Кафедра

Отчет по лабораторной работе №1

по дисциплине «Программирование»

Вариант 20

Выполнил: студент группы

.

Проверил: .

# Тамбов 20

**Постановка задачи**

Дана матрица [n\*m]. Установить, имеются ли в ней седловые точки. Сортировать матрицу по возрастанию методом бинарных вставок. Проверить, сохранили ли седловые точки свой статус. Точка является седловой, если она min в строке и max в столбце.

**Решение**

Для решения поставленной задачи необходимо выполнить следующие шаги:

1. Создаем массив [n][m]
2. Заполняем массив рандомными числами от 1 до 100
3. Ищем седловые точки, выводим результат поиска
4. Переписываем двумерный массив в одномерный
5. Делаем бинарную сортировку одномерного массива
6. Переписываем одномерный массив в двумерный
7. Повторяем шаг 3

Принцип работы сортировки на основе бинарных вставок:

Сортировку простыми вставками можно немного улучшить: поиск "подходящего места" в упорядоченной последовательности можно вести более экономичным способом, который называется Двоичный поиск в упорядоченной последовательности. Он напоминает детскую игру "больше-меньше": после каждого сравнения обрабатываемая последовательность сокращается в два раза.

Пусть, к примеру, нужно найти место для элемента 7 в таком массиве:

[2 4 6 8 10 12 14 16 18]

Найдем средний элемент этой последовательности (10) и сравним с ним семерку. После этого все, что больше 10 (да и саму десятку тоже), можно смело исключить из дальнейшего рассмотрения:

[2 4 6 8] 10 12 14 16 18

Снова возьмем середину в отмеченном куске последовательности, чтобы сравнить ее с семеркой. Однако здесь нас поджидает небольшая проблема: точной середины у новой последовательности нет, поэтому нужно решить, который из двух центральных элементов станет этой "серединой". От того, к какому краю будет смещаться выбор в таких "симметричных" случаях, зависит окончательная реализация нашего алгоритма. Давайте договоримся, что новой "серединой" последовательности всегда будет становиться левый центральный элемент. Это соответствует вычислению номера "середины" по формуле

nomer\_sred:= (nomer\_lev + nomer\_prav)div 2

Итак, отсечем половину последовательности:

2 4 [6 8] 10 12 14 16 18

И снова:

2 4 6 [8] 10 12 14 16 18

2 4 6][8 10 12 14 16 18

Таким образом, мы нашли в исходной последовательности место, "подходящее" для нового элемента. Если бы в той же самой последовательности нужно было найти позицию не для семерки, а для девятки, то последовательность границ рассматриваемых промежутков была бы такой:

[2 4 6 8] 10 12 14 16 18

2 4 [6 8] 10 12 14 16 18

2 4 6 [8] 10 12 14 16 18

2 4 6 8][10 12 14 16 18

Из приведенных примеров уже видно, что поиск ведется до тех пор, пока левая граница не окажется правее(!) правой границы. Кроме того, по завершении этого поиска последней левой границей окажется как раз тот элемент, на котором необходимо закончить сдвиг "хвоста" последовательности.

Если мы станем искать позицию не для семерки или девятки, а для единицы:

[2 4 6 8] 10 12 14 16 18

[2] 4 6 8 10 12 14 16 18

][2 4 6 8 10 12 14 16 18

Как видим, правая граница становится неопределенной - выходит за пределы массива. Будет ли этот факт иметь какие-либо неприятные последствия? Очевидно, нет, поскольку нас интересует не правая, а левая граница.

А что будет, если мы захотим добавить 21?. Проверим это:

2 4 6 8 10 [12 14 16 18]

2 4 6 8 10 12 14 [16 18]

2 4 6 8 10 12 14 16 [18]

2 4 6 8 10 12 14 16 18][

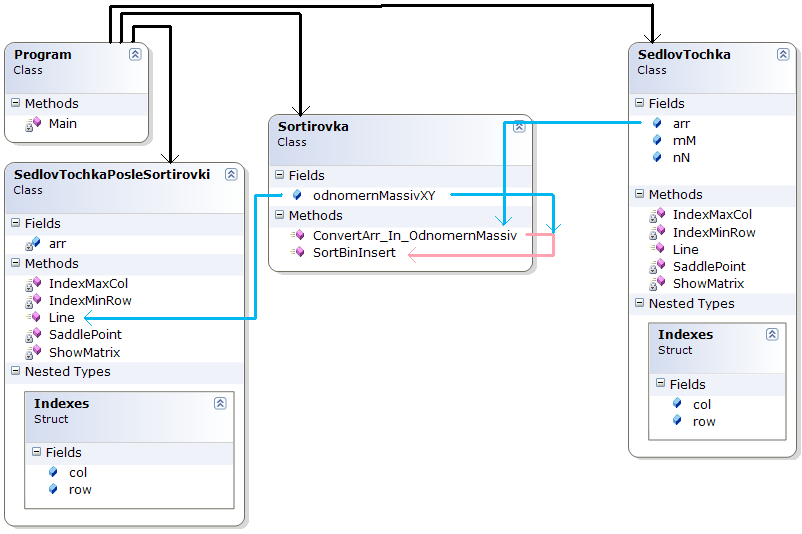
Кажется, будто все плохо: левая граница вышла за пределы массива; непонятно, что нужно сдвигать...

Вспомним, однако, что в реальности на (N+1)-й позиции как раз и находится вставляемый элемент (21). Таким образом, если левая граница вышла за рассматриваемый диапазон, получается, что ничего сдвигать не нужно. Вообще же такие действия выглядят явно лишними, поэтому от них стоит застраховаться, введя одну дополнительную проверку в текст алгоритма.

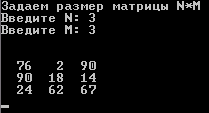
**Код программы**

Для написания программы использовалась среда разработки Visual Studio 2008 с анализатором кода ReSharper. Язык разработки C#.

Общая структура программы:

В главном методе Main() подключаются классы поиска седловых точек, сортировки и повторного поиска с изменёнными начальными значениями.

После управление переходит в класс SedlovTochka. В методе Line() создается матрица размера [n][m], где n,m вводятся пользователем с клавиатуры. Далее, тамже матрица[n][m] заполняется рандомно числами от 1 до 100.

Следующим выполняется метод ShowMatrix(int[][] arr) который принимает сформированный массив[n][m] на входе и выводит построчно содержимое этого массива в виде таблицы.

Далее выполнятся два метода IndexMinRow() и IndexMaxCol() производящие поиск всех минимальных значений по строкам и всех максимальных значений по столбцам соответственно. В данной матрице результат следующий:

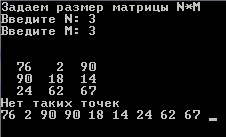
Следующий метод SaddlePoint().

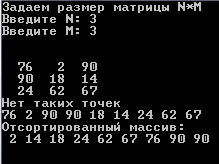
Он производит сравнивание результатов, полученных в методах IndexMinRow() и IndexMaxCol(). Т.е. ели одно число попадется в двух методах, то это число станет седловой точкой, иначе матрица неимеет таковых.

В итоге видим результат сравнения, говорящий об отсутствии подходящих значений.

На этом выполнение класса SedlovTochka завершается.

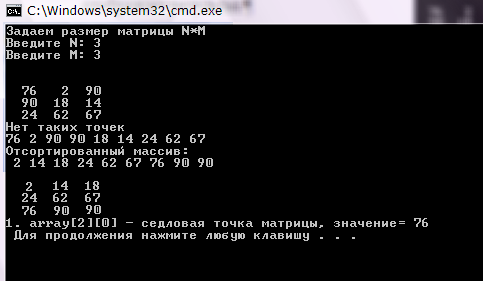
Следующим управление переходит в класс Sortirovka. Этот класс содержит два метода, один из которых переводит двумерный массив из класса SedlovTochka в одномерный odnomernMassivXY[ ], а после производится сортировка методом бинарных вставок.

Метод ConvertArr\_In\_OdnomernMassiv() переписывает все значения из arr[n][m] в массив odnomernMassivXY[ ] с последующим отображением полученных результатов.

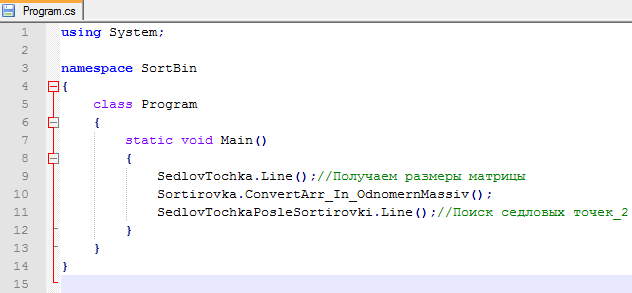
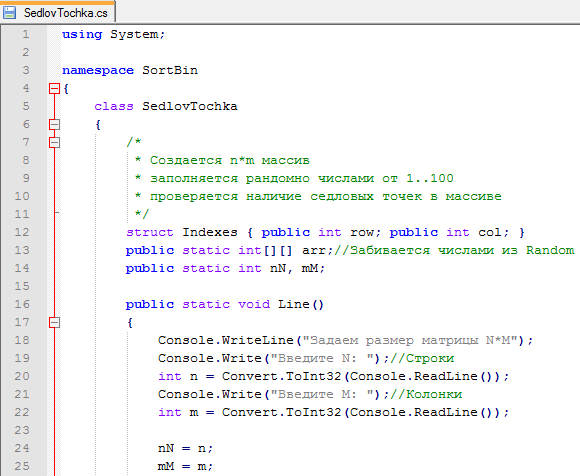
После выполнится метод SortBinInsert(odnomernMassivXY) который на вход принимает массив из предыдущего метода, выполняет сортировку по возрастанию и отображает результат.

Работа класса на этом завершится.

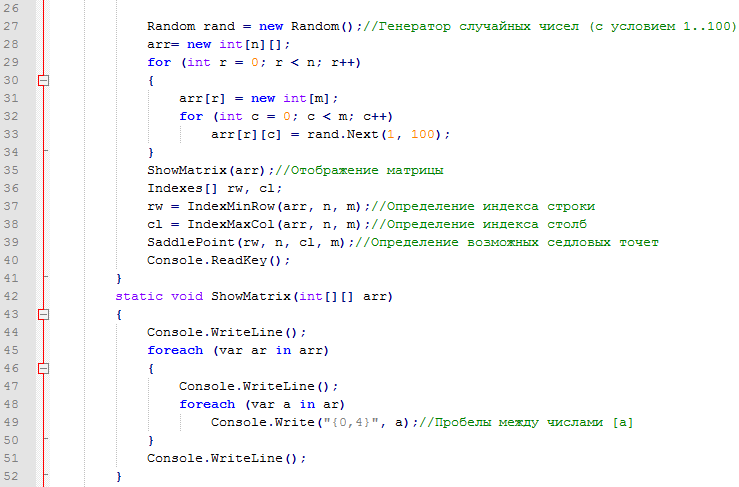
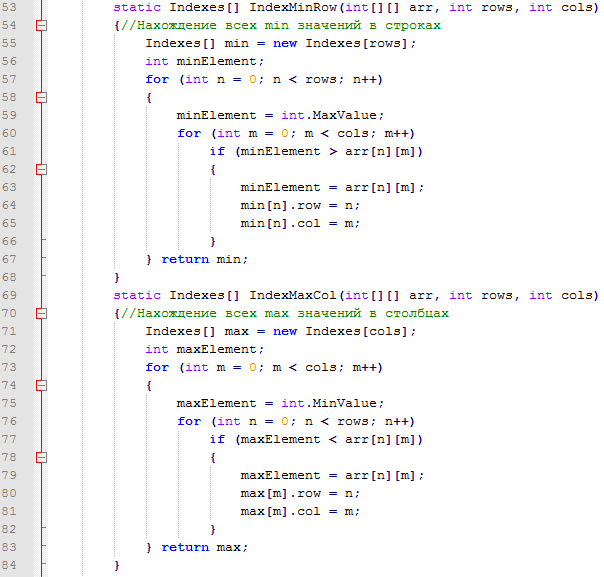
Следующим управление возьмет класс SedlovTochkaPosleSortirovki. Принцип работы методов этого класса схож с классом SedlovTochka. Разница лишь в том, что вместо заполнения массива arr[r][c] рандомными числами его заполняют значениями из массива odnomernMassivXY[ ]. Далее происходит поиск седловых точек на основе новых отсортированных значений и отображение их в случае нахождения.

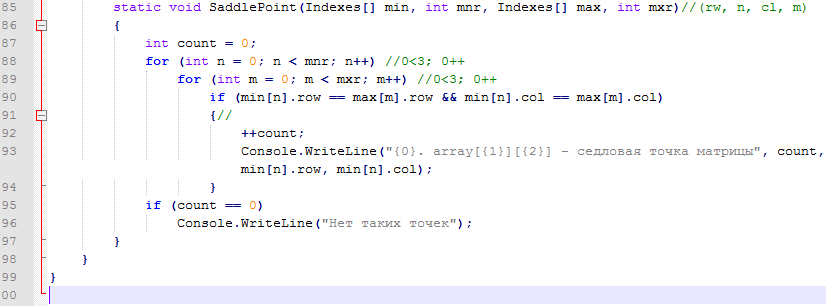


Листинг Program.cs:

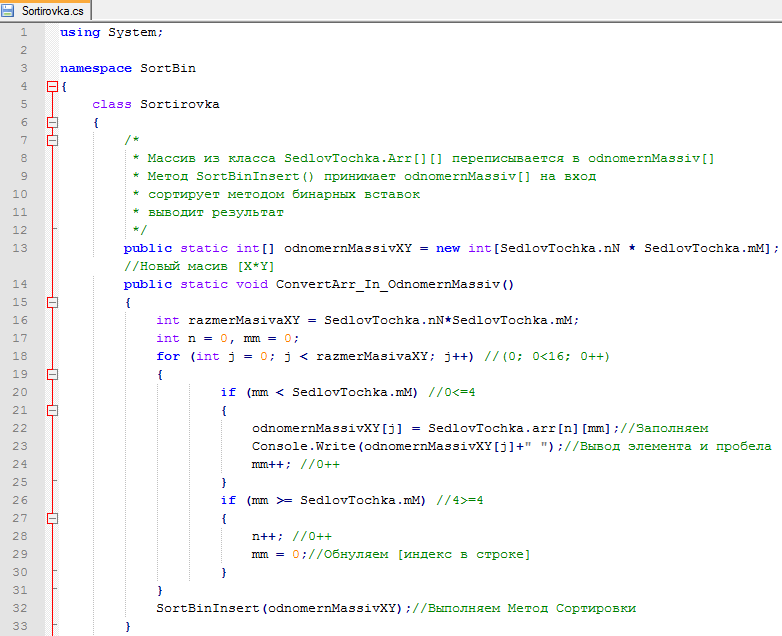


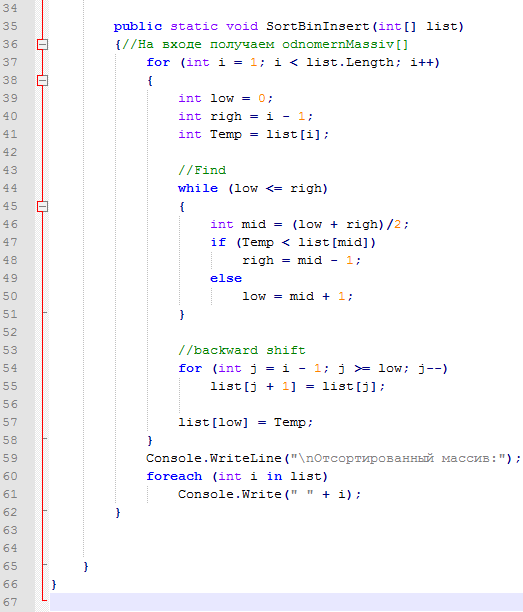
Листинг SedlovTochka.cs:

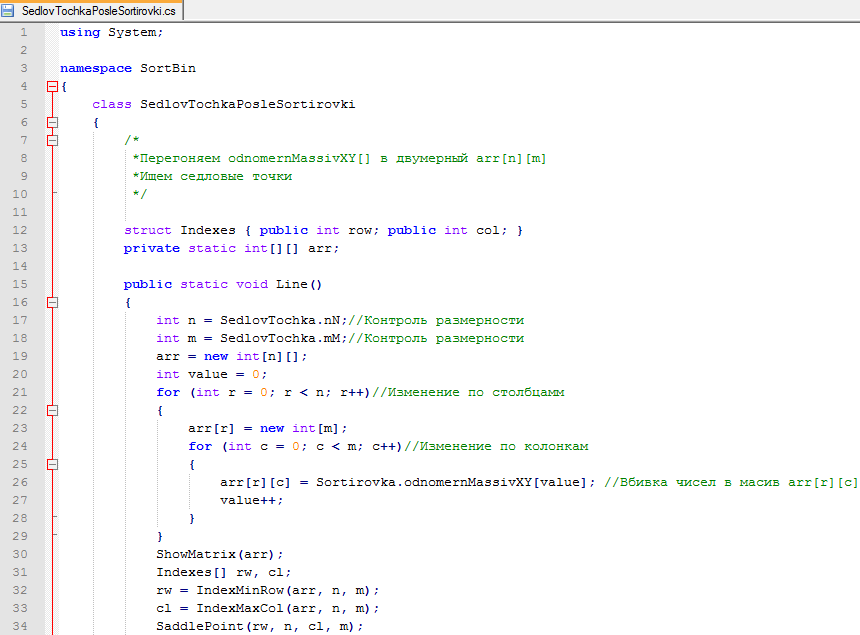


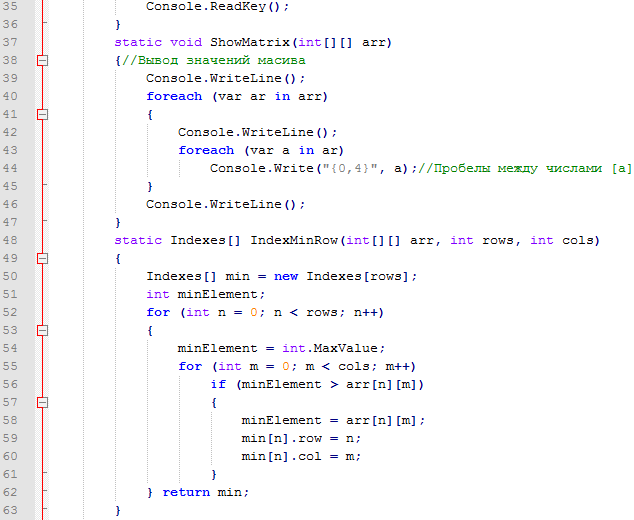


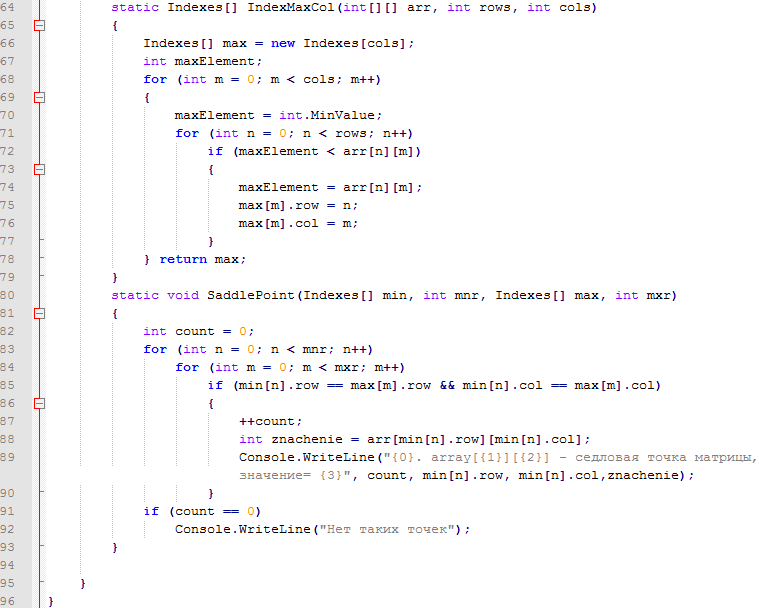
Листинг Sortirovka.cs:





Листинг SedlovTochkaPosleSortirovki.cs:





Итог работы программы:

