УДК 004.4’233

ИСПОЛЬЗОВАНИЕ СРЕДСТВ ПРОФИЛИРОВАНИЯ VISUAL STUDIO ДЛЯ ОПТИМИЗАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Батталов А.И., Батталова Н.И.

Саратовский национальный исследовательский государственный университет имени Н.Г. Чернышевского, Саратов, e-mail: battalov.abbas.94@gmail.com, battalova.nur@gmail.com

В современном мире разработка программного обеспечения превратилась в одну из самых дорогостоящих индустрий. Любые ошибки и недочеты в процессе его создания могут привести к нежелательным результатам. На уровне промышленной разработки программного обеспечения недостаточно уметь программировать на высокоуровневых языках программирования, знать базовые алгоритмы и их теоретическую сложность, необходимо владеть инструментальными средствами, позволяющими проводить тестирование и выявлять проблемы производительности программного обеспечения. В данной статье рассмотрена возможность использования средств профилирования Microsoft Visual Studio .Net для оптимизации программного обеспечения. Представлены результаты профилирования приложения, реализующего фрактальное сжатие/восстановление изображений на языке программирования C#. Проанализировав полученный отчет профилировщика, удалось выявить проблемные места в приложении, предположить причины возникновения проблем, предложить и реализовать способы решения данных проблем. Изменения, внесенные в приложение, позволили улучшить его производительность в восемь раз. Следует отметить, что использование методик оптимизации программного кода не всегда приводит к улучшению производительности приложения. В статье рассмотрен пример, демонстрирующий данное утверждение. В таком случае следует отказаться от изменений, внесенных в программный код.

Ключевые слова: профилирование производительности, оптимизация, программное обеспечение

USING VISUAL STUDIO PROFILES TO OPTIMIZING THE SOFTWARE

Battalov A.I., Battalova N.I.

Saratov National Research State University named after N.G. Chernyshevskiy, Saratov, e-mail: battalov.abbas.94@gmail.com, battalova.nur@gmail.com

Software development has become one of the most expensive industries in the modern world. Any errors and shortcomings in the process of developing can lead to undesirable results. At this level of industrial software development, it is not enough to be able to program in high-level programming languages, to know basic algorithms and their theoretical complexity, one needs to know tools, which allow testing and identifying problems of the software. This article discusses the use of Microsoft Visual Studio .Net profiling tools to optimize software. The article presents the results of profiling applications that implement fractal compression/image recovery in the programming language C#. After analyzing the profiler’s report, it is possible to identify the problem areas in the application, to assume the causes of the problems, to offer and implement ways to solve these problems. The changes made to the app have improved it’s performance eight times. It should be noted that the use of methods of optimization of the program code does not always lead to improved application performance. The article presents an example that demonstrates this statement. In this case, to abandon the changes made to software code.

Keywords: performance profiling, optimization, software

В современном мире разработка программного обеспечения (ПО) превратилась в одну из самых дорогостоящих индустрий. Любые ошибки и недочеты в процессе его создания могут привести к нежелательным результатам, поэтому разработка «совершенного» кода очень важна.

В большинстве случаев даже правильно работающий код может быть усовершенствован. Причина заключается в том, что выбранный алгоритм является базовым, и при его реализации на конкретном языке программирования не в полной мере учитываются условия поставленной задачи, специфика языка программирования и схемы трансляции. Недостаточно знать базовые алгоритмы и их теоретическую сложность, уметь программировать на высокоуровневых языках программирования, нужно уметь оценивать реальное время выполнения программ [1], владеть инструментальными средствами, позволяющими проводить тестирование программного обеспечения [2, 3], определять проблемы производительности на уровне исходного кода, а также знать методики оптимизации кода и уметь применять их на практике [4].

Данная статья посвящена использованию средства профилирования Visual Studio (VS) [5] для оптимизации программного обеспечения, написанного на языке C# [6].

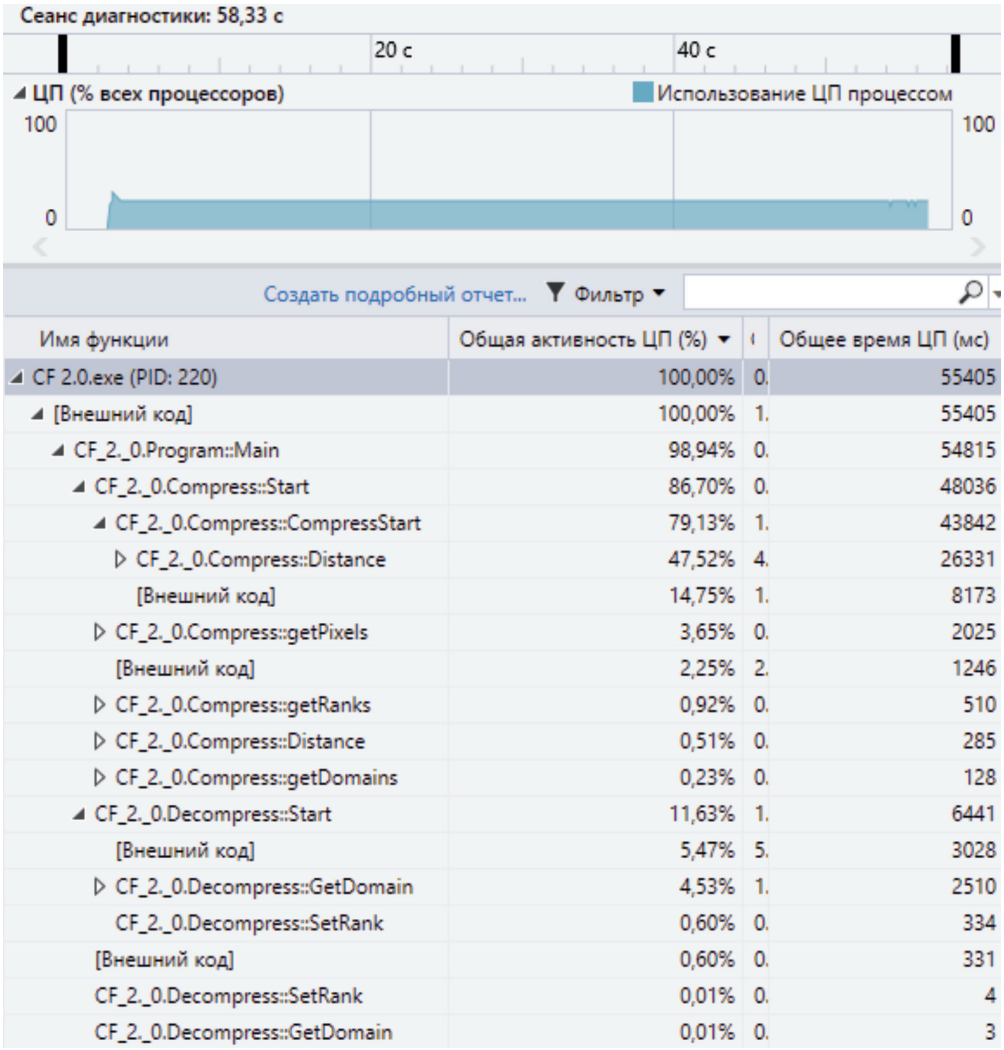
В работе [7] нами была представлена методика применения средств профилирования VS на примере оптимизации кода приложения, решающего задачу о «красивом тексте» [8] с использованием регулярных выражений и Linq-запросов языка C#. Данная методика позволила выявить «про-

блемные» места в приложении с точки зрения использования центрального процессора (ЦП). Применив к проблемным местам «методику минимизация объема работы, выполняемой внутри цикла» [4], нам удалось сократить время работы приложения в 4 раза.

Продемонстрируем использование данной методики на более сложной задаче. Дано приложение ColorFractus, реализующее фрактальное сжатие/восстановление изображений [9, 10]. Используя средства профилирования VS, необходимо: выявить проблемные места в приложении; предположить причины проблем; предложить и реализовать способы решения проблем; показать, что внесенные изменения в приложение ColorFractus, действительно привели к оптимизации.

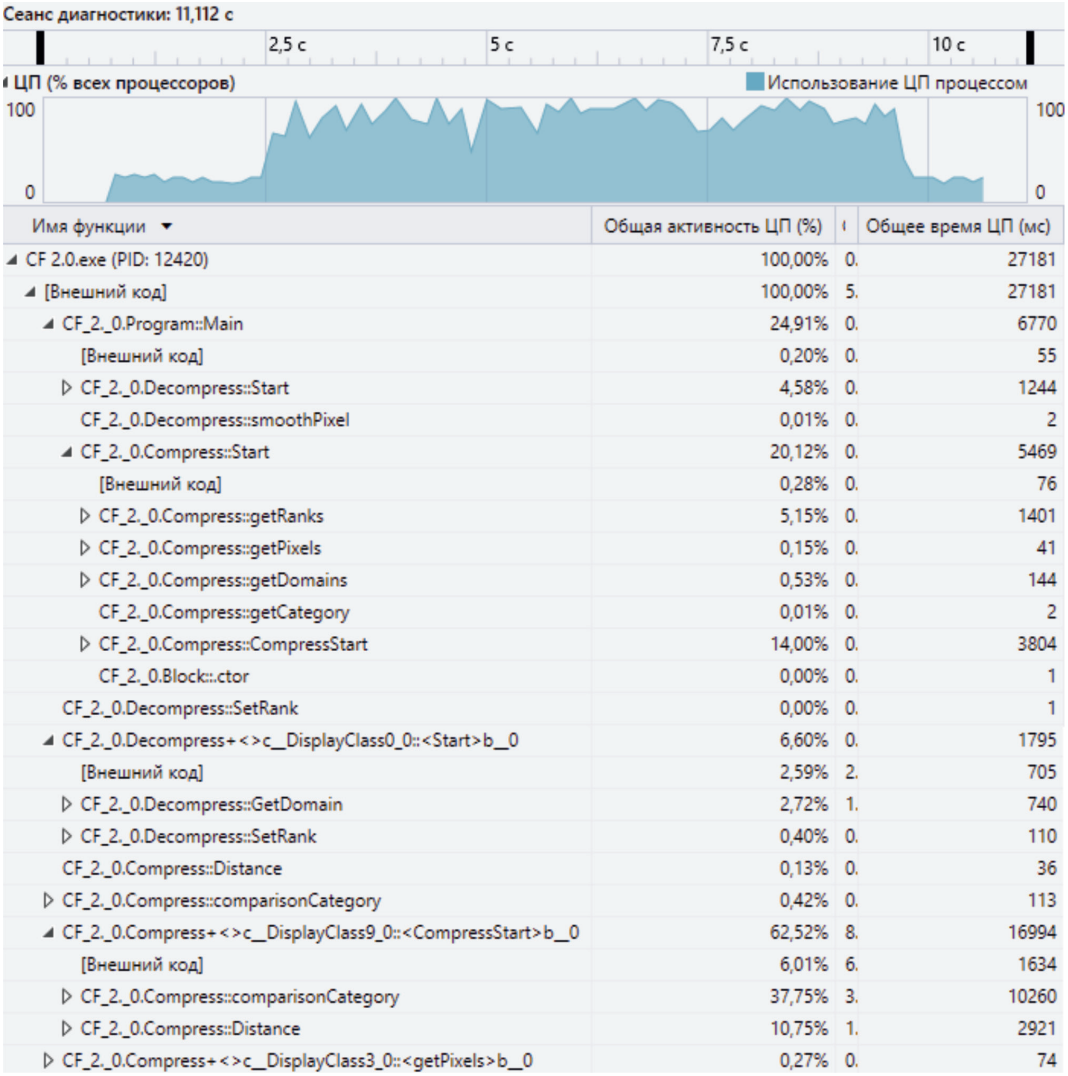
Использование средств профилирования VS начинается с настройки сеанса анализа производительности. Для этой цели нужно открыть код программы (решение) в среде VS. Выбрать конфигурацию «Выпуск»/«Release» (запуск от имени администратора). Установить флажок «Выборка циклов ЦП» или «Инструментирование», и нажать кнопку «Готово».

Для запуска процесса сбора данных о производительности в меню «Анализ» необходимо выбрать «Профилировщик производительности», установить флажок «Мастер производительности» и нажать кнопку «Запуск». Исследуемое решение запускается, и VS начинает собирать данные о его производительности в реальном времени, которые записываются специальный файл (*.vsp). После завершения сбора данных файл с отчетом отображается в окне «Отчет о производительности». Для приложения ColorFractus мы получили следующий отчет о производительности (рис. 1).



Имя функции	Общая активность ЦП (%)		Общее время ЦП (мс)
CF 2.0.exe (PID: 220)	100,00%	0.	55405
[Внешний код]	100,00%	1.	55405
CF_2._0.Program::Main	98,94%	0.	54815
CF_2._0.Compress::Start	86,70%	0.	48036
CF_2._0.Compress::CompressStart	79,13%	1.	43842
CF_2._0.Compress::Distance	47,52%	4.	26331
[Внешний код]	14,75%	1.	8173
CF_2._0.Compress::getPixels	3,65%	0.	2025
[Внешний код]	2,25%	2.	1246
CF_2._0.Compress::getRanks	0,92%	0.	510
CF_2._0.Compress::Distance	0,51%	0.	285
CF_2._0.Compress::getDomains	0,23%	0.	128
CF_2._0.Decompress::Start	11,63%	1.	6441
[Внешний код]	5,47%	5.	3028
CF_2._0.Decompress::GetDomain	4,53%	1.	2510
CF_2._0.Decompress::SetRank	0,60%	0.	334
[Внешний код]	0,60%	0.	331
CF_2._0.Decompress::SetRank	0,01%	0.	4
CF_2._0.Decompress::GetDomain	0,01%	0.	3

Рис. 1. Статистические данные, полученные методом выборки



Имя функции	Общая активность ЦП (%)		Общее время ЦП (мс)
CF 2.0.exe (PID: 12420)	100,00%	0.	27181
[Внешний код]	100,00%	5.	27181
CF_2._0.Program::Main	24,91%	0.	6770
[Внешний код]	0,20%	0.	55
CF_2._0.Decompress::Start	4,58%	0.	1244
CF_2._0.Decompress::smoothPixel	0,01%	0.	2
CF_2._0.Compress::Start	20,12%	0.	5469
[Внешний код]	0,28%	0.	76
CF_2._0.Compress::getRanks	5,15%	0.	1401
CF_2._0.Compress::getPixels	0,15%	0.	41
CF_2._0.Compress::getDomains	0,53%	0.	144
CF_2._0.Compress::getCategory	0,01%	0.	2
CF_2._0.Compress::CompressStart	14,00%	0.	3804
CF_2._0.Block::.ctor	0,00%	0.	1
CF_2._0.Decompress::SetRank	0,00%	0.	1
CF_2._0.Decompress+<>c__DisplayClass0_0::<Start>b__0	6,60%	0.	1795
[Внешний код]	2,59%	2.	705
CF_2._0.Decompress::GetDomain	2,72%	1.	740
CF_2._0.Decompress::SetRank	0,40%	0.	110
CF_2._0.Compress::Distance	0,13%	0.	36
CF_2._0.Compress::comparisonCategory	0,42%	0.	113
CF_2._0.Compress+<>c__DisplayClass9_0::<CompressStart>b__0	62,52%	8.	16994
[Внешний код]	6,01%	6.	1634
CF_2._0.Compress::comparisonCategory	37,75%	3.	10260
CF_2._0.Compress::Distance	10,75%	1.	2921
CF_2._0.Compress+<>c__DisplayClass3_0::<getPixels>b__0	0,27%	0.	74

Рис. 2. Статистические данные о производительности оптимизированного кода

Проанализировав данные, представленные на рис. 1, можно сделать следующие выводы:

1. Выполнение программы нагружает ЦП не более чем на 25%. Компьютер, на котором проводились замеры, имеет 4-поточный процессор. Это означает, что программа выполняется только в одном потоке процессора.

2. Метод сжатия (Compress.Start) выполняется почти 87% от всего времени работы приложения. Это подтверждает то, что фрактальное сжатие является несимметричным алгоритмом сжатия, так как время сжатия изображения больше времени восстановления.

3. В методе Compress.Start вызывается метод Compress.CompressStart, который, перебирая все ранговые блоки, находит подходящий доменный блок для каждого рангового и занимает 80% времени от всего выполнения программы. Это означает, что данный метод является самым дорогостоящим с точки зрения скорости выполнения.

4. Внутри метода Compress.CompressStart вызывается метод Compress.Distance, который предназначен для сравнения блоков с помощью подсчета дистанции (числовая характеристика, оценивающаяся функцией среднеквадратического отклонения) между блоками. Суммарное время выполнения всех вывозов данной функции занимает около 48%.

Решением указанных проблем мы видим: в предварительной классификации ранговых и доменных блоков по признаку распределения яркости отдельных пикселей

блока; в распараллеливании циклов перебора ранговых блоков, во время поиска наиболее подобного доменного блока и во время восстановления изображения, так как эти процессы являются независимыми; в сокращении количества вызовов метода поиска дистанции между блоками; в добавлении кеширования, принципом которого будет сохранение доменного блока для конкретного ключа, и присваивании последующему ранговому блоку, который имеет такой ключ. Все предложения по изменению кода приложения ColorFractus были нами реализованы в полном объеме.

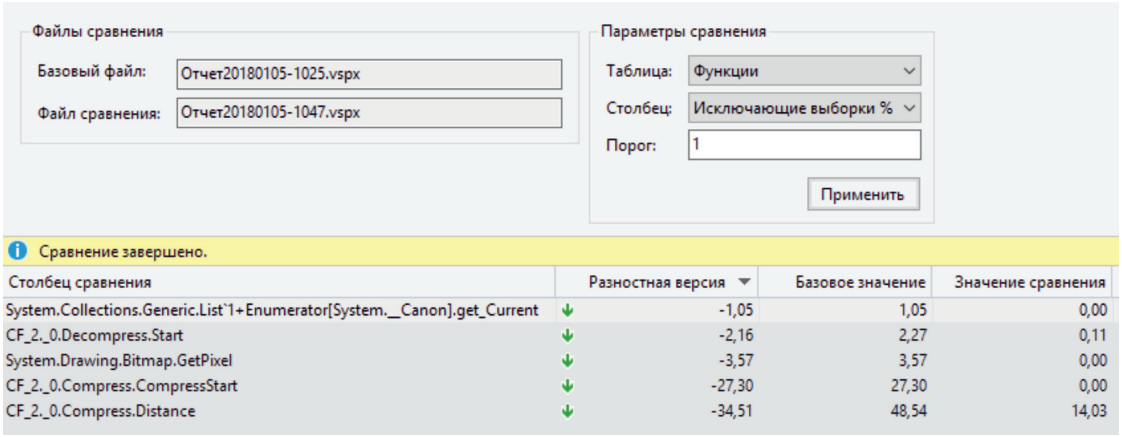


Рис. 3. Фрагменты сравнения двух отчетов производительности

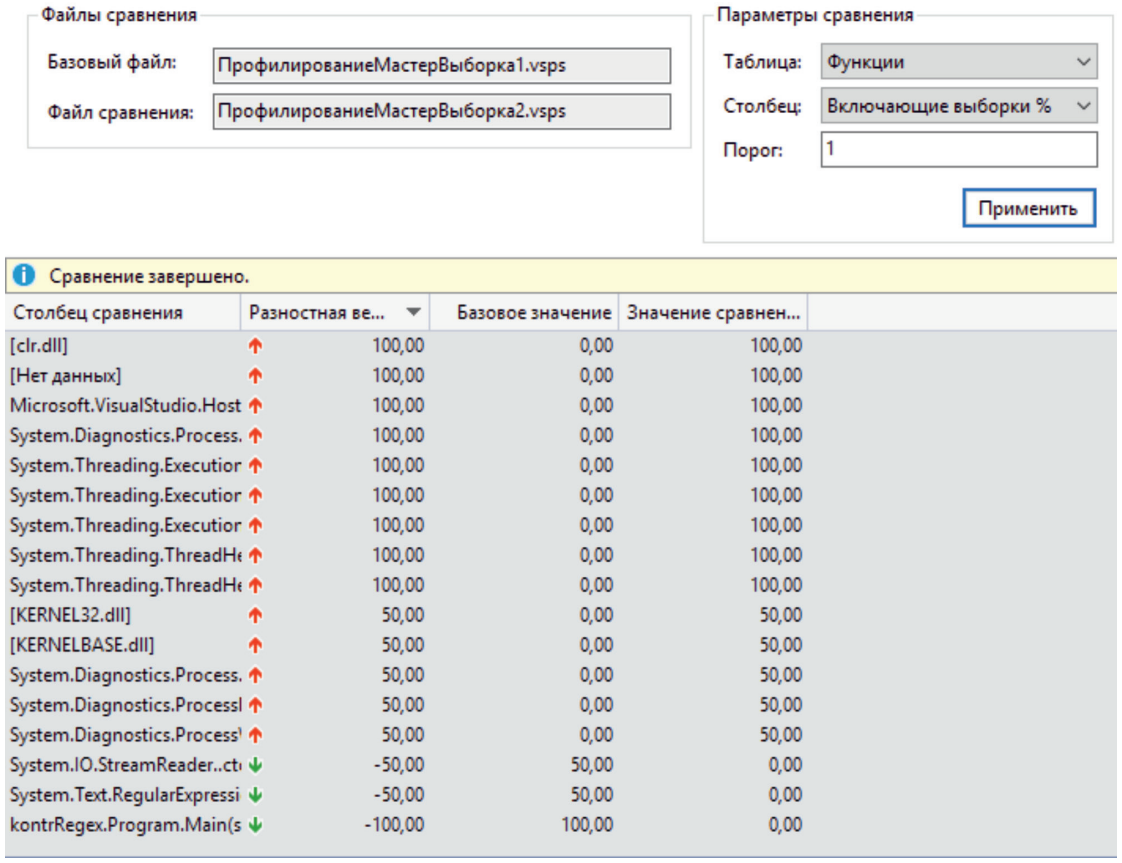


Рис. 4. Фрагмент сравнения отчетов производительности (отрицательный результат)

Чтобы убедиться в том, что внесенные изменения в исходный код привели к его оптимизации, повторно запустим профилировщик производительности, а также сравним полученные отчеты (до внесения изменений в код приложения и после). Полученные данные представлены на рис. 2 и 3.

Проанализировав данные, представленные на рис. 2 и 3, можно сделать следующие выводы:

1. Большую часть времени выполнения приложения ColorFractus ЦП загружен на 100%. Это означает, что некоторые участки кода выполняются параллельно и задействованы все 4 ядра (потока) ЦП.

2. Время выполнения метода Compress. Start составляет 5,5 секунд в отличие от 48 секунд в исходной программе.

3. Время выполнения метода Decompress.Start составляет чуть более одной секунды в отличие от 6,4 секунды в исходной программе.

4. Полное время выполнения и сжатия/восстановления изображения уменьшилось с 55 секунд до 6,8, что дает увеличение в скорости сжатия приблизительно в 8 без потери качества обрабатываемого изображения.

Таким образом, мы убедились, что внесенные изменения в приложение ColorFractus действительно привели к его оптимизации.

Следует отметить, что не всегда удается оптимизировать приложения. Так, например, на этапе сравнения отчетов о производительности в ходе оптимизации приложения, позволяющего находить все вещественные числа в больших объемах текстовой информации с помощью регулярных выражений, были получены данные, представленные на рис. 4.

Проанализировав полученные данные, можно сделать вывод, что наши действия, направленные на оптимизацию кода ухудшили производительность приложения, поэтому от таких «оптимизаций» следует отказаться.

В заключение следует отметить, что средства профилирования VS – это удобный и доступный для разработчиков программного обеспечения помощник в решении проблемы разработки совершенного кода. Обучение студентов IT-направлений навыкам использования средства профилирования VS, для оптимизации программного кода, поможет выпускникам вузов в их будущей профессиональной деятельности.

Список литературы

1. Кудрина Е.В. Роль оценки реального времени выполнения программы на примере алгоритмов поиска делителей натурального числа / Е.В. Кудрина, В.Р. Кузьмина // Информационные технологии в образовании: материалы VII Всерос. научно-практ. конференции. – Саратов: ООО «Издательский центр «Наука», 2015. – С. 49–54.

2. Лукашова М.А. Применение методов тестирования программного обеспечения на практике / М.А. Лукашова, Е.В. Кудрина // Информационные технологии в образовании: материалы VIII Международ. научно-практ. конференции. – Саратов: ООО «Издательский центр «Наука», 2016. – С. 237–241.

3. Ефимов М.С. Анализатор покрытия кода Unit-тестами для Visual Studio/ М.С. Ефимов, Е.В. Кудрина, А.В. Кузнецов // Информационные технологии в образовании: материалы VII Всерос.научно-практ. конференции. – Саратов: ООО «Издательский центр «Наука», 2015. – С. 214–218.

4. Макконнелл С. Совершенный код. Мастер-класс / С. Макконнелл. – М.: Издательство Русская редакция, 2010. – 896 с.

5. Общие представление о способах профилирования [Электронный ресурс] // Microsoft: Developer Network. URL: https://msdn.microsoft.com/ru-ru/library/dd264994.aspx#instrumentation (дата обращения: 10.01.2018).

6. Огнева М.В. Программирование в среде Visual Studio .Net: разработка приложений на языке C#/ М.В. Огнева, Е.В. Кудрина. – Саратов: Издательство «КУБиК», 2010. – 545 с.

7. Батталова Н.А. Оптимизация программного кода с использованием средств профилирования visual studio / Н.А. Батталова // Информационные технологии в образовании: материалы IX Всерос. научно-практ. конференции. – Саратов: ООО «Издательский центр «Наука», 2017. – С. 113–117.

8. Задача D «Красивый текст» [Электронный ресурс]// Летняя компьютерная школа 2013. Регулярные выражения. URL: http://hist.leenr.ru/camps/lksh/2013.Winter/ejudge.lksh.ru/ C/05/problems.pdf (дата обращения: 10.01.2018).

9. Батталов А.И. Фрактальное сжатие цветных изображений // Научные исследования студентов Саратовского государственного университета: материалы итоговой студенческой научной конференции. – Саратов: Изд-во Сарат. ун-та, 2016. – С. 31–32.

10. Батталов А.И.. Разработка и реализация быстрого алгоритма фрактального сжатия цветных изображений / А.И. Батталов, Е.В. Кудрина // Информационные технологии в образовании: материалы VIII Международ. научно-практ. конференции. – Саратов: ООО «Издательский центр «Наука», 2016. – С. 21–25.