МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

|  |  |
| --- | --- |
|  | Утвержден на заседании кафедры  «Вычислительная техника»  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 г.  Заведующий кафедрой                                              М.А. Митрохин |
|  |  |

**ОТЧЕТ ПО УЧЕБНОЙ (ЭКСПЛУАТАЦИОННОЙ) ПРАКТИКЕ**

(2023/2024 учебный год)

                                         \_\_ Прохоров Даниил Сергеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения **1**семестр 2

Период прохождения практики с 20.06.2024 по 17.07.2024

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

Руководитель практики д.т.н., профессор, Карамышева Н.С.

*(должность, ученая степень, ученое звание)*

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

|  |  |
| --- | --- |
|  | Утвержден на заседании кафедры  «Вычислительная техника»  "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 г.  Заведующий кафедрой                                              М.А. Митрохин |
|  |  |

**ИНДИВИДУАЛЬНЫЙ ПЛАН ПРОХОЖДЕНИЯ УЧЕБНОЙ (ЭКСПЛУАТАЦИОННОЙ) ПРАКТИКИ**

(2023/2024 учебный год)

                                            Прохоров Даниил Сергеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения                  1                 семестр                 2

Период прохождения практики с 20.06.2024 по 17.07.2024

Кафедра «Вычислительная техника»

Заведующий кафедрой д.т.н., профессор, Митрохин М.А.

*(должность, ученая степень, ученое звание, Ф.И.О.)*

Руководитель практики д.т.н., профессор, Карамышева Н.С.

*(должность, ученая степень, ученое звание)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Планируемая форма работы во время практики | Количество часов | Календарные сроки проведения работы | Подпись  руководителя  практики от вуза |
| 1 | Выбор темы и разработка индивидуального плана проведения работ | 26 | 20.06.24 –  24.06.24 |  |
| 2 | Подбор и изучение материала по теме работы | 26 | 24.06.24 –  26.06.24 |  |
| 3 | Установка виртуальной машины | 26 | 26.06.24 –  28.06.24 |  |
| 4 | Установка операционной системы | 30 | 01.07.24 –  04.07.24 |  |
| 5 | Разработка программы на языке Си | 30 | 04.07.24 –  09.07.24 |  |
| 6 | Тестирование и отладка | 38 | 09.07.24 –  12.07.24 |  |
| 7 | Оформление отчёта | 40 | 12.07.24 –  17.07.24 |  |
|  | **Общий объём часов** | 216 |  |  |

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЧЁТ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ЭКСПЛУАТАЦИОННОЙ) ПРАКТИКИ**

(2023/2024 учебный год)

                                            Прохоров Даниил Сергеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 20.06.2024 по 17.07.2024

Кафедра «Вычислительная техника»

Прохоров Д.С. выполнял практическое задание «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_». На первоначальном этапе

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

Бакалавр Прожорин Е.А. \_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Руководитель Карамышева Н.С.\_\_\_\_\_\_\_\_\_\_\_\_ "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

практики

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЗЫВ**

**О ПРОХОЖДЕНИИ УЧЕБНОЙ (ЭКСПЛУАТАЦИОННОЙ) ПРАКТИКИ**

(2023/2024 учебный год)

                                            Прохоров Даниил Сергеевич

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Наименование профиля подготовки «Вычислительные машины, комплексы, системы и сети»

Форма обучения – очная Срок обучения в соответствии с ФГОС – 4 года

Год обучения1семестр 2

Период прохождения практики с 20.06.2024 по 17.07.2024

Кафедра «Вычислительная техника»

В процессе выполнения практики Прохоров Д.С. решал следующие задачи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

За период выполнения практики были освоены\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_. Во время выполнения работы Прожорин Е. А. показал себя ответственным, добросовестным учеником, знающим свой предмет, имеющим представление о современном состоянии науки, владеющим современными общенаучными знаниями по информатике и вычислительной технике.

За выполнение работы Прожорин Е. А. заслуживает оценки «\_\_\_\_\_\_».

Руководитель практики д.т.н., профессор, Карамышева Н.С. « » 2024 г.

1. **Введение**Microsoft Visual Studio — это программная среда по разработке

приложений для ОС Windows, как консольных, так и с графическим

интерфейсом.

Функциональная структура среды включает в себя:

1) редактор исходного кода с поддержкой технологии IntelliSense и

возможностью простейшего рефакторинга кода;

2) отладчик кода;

3) редактор форм, предназначенный для упрощённого конструирования

графических интерфейсов;

4) веб-редактор;

5) дизайнер классов;

6) дизайнер схем баз данных.

Visual Studio также позволяет создавать и подключать сторонние

дополнения (плагины) для расширения функциональности практически на

каждом уровне, включая добавление поддержки систем контроля версий

исходного кода, добавление новых наборов инструментов (для

редактирования и визуального проектирования кода на предметно-

ориентированных языках программирования или инструментов для прочих

аспектов процесса разработки программного обеспечения).

В комплект входят следующие основные компоненты:

1. Visual Basic.NET – для разработки приложений на VisualBasic;
2. Visual C++ – на традиционном языке C++;
3. Visual C# – на языке C# (Microsoft);
4. Visual F# – на F# (Microsoft Developer Division).

Интегрированная среда разработки (IntegratedDevelopmentEnvironment

– IDE) Visual Studio предлагает ряд высокоуровневых функциональных

возможностей, которые выходят за рамки базового управления кодом.

Ниже перечислены основные преимущества IDE-среды Visual Studio:

1. Встроенный Web-сервер;
2. Поддержка множества языков при разработке;
3. Интуитивный стиль кодирования;
4. Более высокая скорость разработки;
5. Возможности отладки.

Язык программирования C# представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день C# является одним из самых популярных и распространенных языков.

Своими корнями он уходит в язык C и C++, а также в Java, от которых он унаследовал многие концепции и синтаксические особенности. В начале 2000-х годов язык C# был разработан компанией Microsoft под руководством Андерса Хейлсберга. Он был создан как часть платформы .NET и изначально был предназначен для разработки приложений под операционную систему Windows. Фактически, C# вобрал в себя лучшие черты своих предшественников и добавил ряд новых возможностей, таких как автоматическое управление памятью через сборку мусора и расширенные средства для работы с асинхронным программированием.

C# является мощным языком, предоставляя разработчикам богатый набор возможностей для создания современных приложений. Он широко используется для разработки настольных приложений, веб-приложений и веб-сервисов. Благодаря платформе .NET, C# предоставляет отличные средства для создания высокопроизводительных и масштабируемых решений, что делает его идеальным выбором для корпоративного программного обеспечения и облачных сервисов. Многие популярные системы управления контентом (CMS), такие как Umbraco и Sitecore, написаны на C#.

Однако применение данного языка не ограничивается только корпоративными приложениями. C# активно используется в разработке игр, благодаря мощному движку Unity, который является одним из лидеров на рынке игровых движков. С помощью Unity и C# можно создавать игры для различных платформ, включая мобильные устройства, консоли и ПК.

Также в последнее время C# нашел применение в мобильной разработке, особенно благодаря фреймворку Xamarin, который позволяет создавать кроссплатформенные мобильные приложения для iOS и Android на основе единого кода. Кроме того, C# используется для создания микросервисов и различных вспомогательных сервисов, которые обслуживают веб-приложения и облачные системы.

В общем, C# – это язык широкого пользования, на котором можно создавать практически любые виды программ.

1. **Разработка и описание алгоритма**

Сортировка Шелла (или метод Шелла) – это один из первых алгоритмов сортировки, разработанный Дональдом Шеллом в 1959 году. Этот алгоритм является улучшенной версией сортировки вставками и работает по принципу разделения и последующей сортировки подмассивов, что позволяет существенно уменьшить количество перестановок.

**Основные принципы работы алгоритма:**

1. Разделение массива на подмассивы:

Алгоритм начинается с разделения исходного массива на несколько подмассивов, которые образуются путем выбора элементов через определенный интервал (gap). Например, если интервал равен 5, то подмассивы будут состоять из элементов, расположенных через каждые 5 позиций.

2. Сортировка подмассивов:

Каждый подмассив сортируется с помощью сортировки вставками. Поскольку эти подмассивы относительно небольшие, сортировка вставками на них работает очень эффективно.

3. Уменьшение интервала (gap):

После сортировки всех подмассивов текущего интервала, интервал уменьшается и процесс повторяется. Обычно, интервал делится на определенное число (например, на 2 или на 2.2), пока не станет равным 1.

4. Конечная сортировка:

Когда интервал становится равным 1, массив полностью сортируется сортировкой вставками. На этом этапе массив уже практически отсортирован благодаря предыдущим шагам, что делает последнюю сортировку вставками очень быстрой.

**Пример работы алгоритма:**

Рассмотрим массив [23, 29, 15, 19, 31, 7, 9, 5, 2].

1. Выбираем начальный интервал (gap). Пусть это будет 4.

Подмассивы: [23, 31], [29, 7], [15, 9], [19, 5], [2]

2. Сортируем каждый подмассив:

[23, 31] -> [23, 31]

[29, 7] -> [7, 29]

[15, 9] -> [9, 15]

[19, 5] -> [5, 19]

[2] -> [2]

3. Новый интервал: gap = 2.

Подмассивы: [23, 15, 31, 7, 19, 9, 5, 2]

4. Сортируем каждый подмассив:

[23, 15, 31, 7, 19, 9, 5, 2] -> [15, 23, 7, 31, 9, 19, 2, 5]

5. Новый интервал: gap = 1.

Полностью сортируем массив вставками:

[15, 7, 9, 2, 5, 19, 23, 31] -> [2, 5, 7, 9, 15, 19, 23, 31]

**Сложность алгоритма:**

Время выполнения сортировки Шелла зависит от выбора последовательности интервалов. В худшем случае сложность алгоритма может достигать O(n^2), однако на практике она обычно значительно ниже. Для некоторых оптимальных последовательностей интервалов (например, последовательность Хиббарда, Седжвика) сложность может приближаться к O(n log^2 n), что делает сортировку Шелла одной из самых эффективных среди простых алгоритмов сортировки.

Сортировка Шелла является хорошим выбором для сортировки небольших и средних массивов и демонстрирует превосходную производительность по сравнению с простыми методами, такими как сортировка вставками или пузырьковая сортировка.

1. **Описание программы:**

### Обмен элементов Метод Swap

static void Swap(ref int a, ref int b)

{

(a, b) = (b, a);

}

Этот метод обменивает значения двух переменных. Ключевое слово ref означает, что параметры a и b передаются по ссылке, что позволяет методу изменить их значения напрямую.

(a, b) = (b, a); - Используется кортеж для одновременного обмена значений a и b.

### Сортировка Шелла Метод ShellSort

static int[] ShellSort(int[] array)

{

var d = array.Length / 2;

while (d >= 1)

{

for (var i = d; i < array.Length; i++)

{

var j = i;

while ((j >= d) && (array[j - d] > array[j]))

{

Swap(ref array[j], ref array[j - d]);

j = j - d;

}

}

d = d / 2;

}

return array;

}

**Инициализация шага (расстояния)** d:

var d = array.Length / 2;

Начальное значение шага d устанавливается как половина длины массива.

**Внешний цикл** while:

while (d >= 1)

Цикл продолжается, пока d не станет меньше 1. В каждой итерации значение d уменьшается вдвое.

**Цикл** for **по элементам массива**:  
for (var i = d; i < array.Length; i++)

Этот цикл проходит по элементам массива, начиная с индекса d до конца массива.

**Внутренний цикл** while **для сравнения и обмена элементов**:

var j = i;

while ((j >= d) && (array[j - d] > array[j]))

{

Swap(ref array[j], ref array[j - d]);

j = j - d;

}

В этом цикле происходит сравнение элементов, отстоящих друг от друга на расстояние d. Если элемент array[j - d] больше элемента array[j], то они обмениваются местами. Это продолжается до тех пор, пока не будет достигнут предел индекса d или элементы в нужной позиции.

**Уменьшение шага** d:

d = d / 2;

После завершения внутреннего цикла шаг d уменьшается вдвое, и процесс повторяется до тех пор, пока d не станет меньше 1.

**Возврат отсортированного массива**:

return array;

Метод возвращает отсортированный массив.

### Общее описание

Алгоритм сортировки Шелла является улучшенной версией сортировки вставками. Он работает следующим образом:

* Изначально используется большой шаг (расстояние) для сравнения и обмена элементов.
* На каждом этапе шаг уменьшается, и элементы, отстоящие друг от друга на это расстояние, сравниваются и при необходимости меняются местами.
* В конечном итоге шаг уменьшается до 1, и происходит окончательная сортировка вставками.

Этот подход позволяет уменьшить количество обменов и сравнений по сравнению с обычной сортировкой вставками, особенно на ранних этапах, когда шаг велик.

1. **Заключение**

На основании анализа данных, полученных в результате тестирования алгоритма сортировки Шелла, можно сделать следующие выводы:

Время, затраченное на выполнение программы, увеличивается нелинейно относительно количества элементов. Хотя сортировка Шелла демонстрирует улучшенную производительность по сравнению с сортировкой вставками, её эффективность зависит от начального выбора шага (расстояния между сравниваемыми элементами).

При малых размерах массивов сортировка Шелла выполняется быстрее за счет уменьшения количества необходимых обменов. Однако, с увеличением количества элементов наблюдается тенденция к субквадратичному росту времени выполнения, особенно в худшем случае.

Таким образом, можно сказать, что алгоритм сортировки Шелла не обладает строгой линейной зависимостью времени работы от количества элементов, но показывает значительные улучшения по сравнению с простыми алгоритмами сортировки для массивов среднего размера.