МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“Харківський політехнічний інститут”

Кафедра “Програмної інженерії та інформаційних технологій управління”

**ЗВІТ**

з лабораторної роботи №3

за курсом «Конструювання програмного забезпечення»

ВИКОНАЛИ

Студент и групи ІФ 32 В

Овчаренко С. Е.

Галайко С.С.

ПЕРЕВІРИВ

Доц. каф. АСУ

Двухглавов Д. Е.

Скляров М.О.

Харків 2015

**Тема:** Командна розробка додатків.

**Мета:** Набути навичок командної розробки додатків, вивчити питання організації взаємодії у ході розробки багатокомпонентного додатку.

**Хід роботи**

**1 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

1.1. Загальний опис розроблюваного програмного забезпечення.

Розроблюване програмне забезпечення (надалі ПЗ) являє собою засіб для порівняння наступних алгоритмів сортування: бульбашка і швидке сортування.

Безпосередньо ПЗ створюється двома особами, тому проект будується за принципом мінімальної зв’язаності його компонентів. Існує загальний каркас додатку, прив’язаний до інтерфейсу (немає потреби робити його відділеним). Кожен алгоритм сортування винесений в окремий клас. Взаємодія між інтерфейсом та класами однотипна, використовуються однаковий склад параметрів та спосіб їх передачі. Така структура дозволяє організувати правильну командну роботу.

Серед особливостей проекту слід виділити Metro інтерфейс (рисунок 3.1) та можливість порівняти швидкість роботи алгоритмів за таймером.

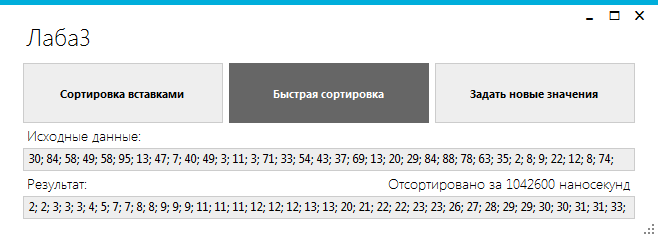


Рисунок 3.1. Інтерфейс проекту

1.2. Використані технології при розробці ПЗ

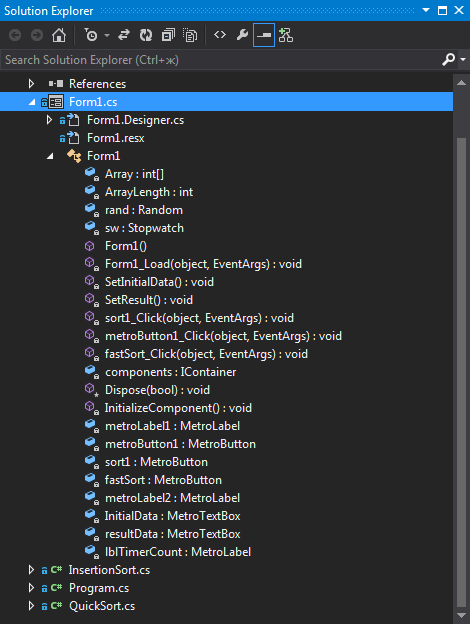
Перелік основних технологій, використовуваних в процесі створення ПЗ, зазначені нижче в таблиці 3.1.

Таблиця 3.1. Стек технологій для створення програмного продукту

|  |  |
| --- | --- |
| **Написання програмного коду** | |
| Мова розробки | С# |
| Середовище розробки | Microsoft Visual Studio 2013 |
| **Забезпечення командної розробки** | |
| Планування завдань | trello |
| Комунікація між виконавцями | Skype 7.4 |
| Система контролю версій | GitHub |
| Клієнт взаємодії с системою контролю версій | Графічний, GitHub for Windows 2.14.5 |
| **Взаємодія з замовником** | |
| Доступ до файлів проекту | GitHub |
| Документація | GitHub – README.md |

1.3. Опис компонентів проекту

На рисунку 3.2 нижче дано короткий опис важливих компонентів системи.



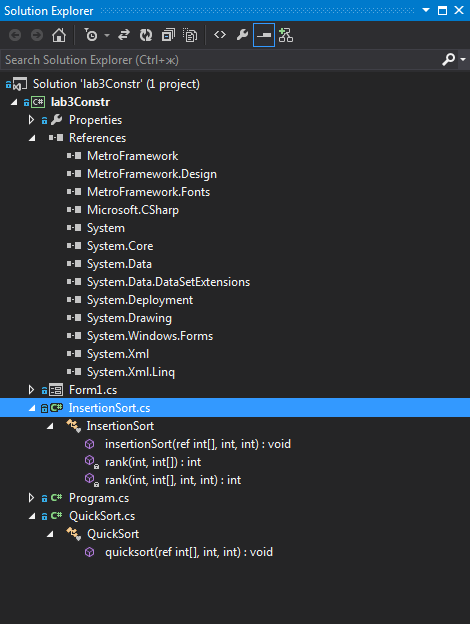
Перелік змінних

Обробники вхідних даних для передачі методам та вихідних для представлення користувачеві

Обробники натиснення кнопок

Елементи інтерфейсу

Рисунок 3.2. Структура проекту у VisualStudio (каркас системи)



Перелік бібліотек для створення Metro інтерфейсу

Клас-обробник сортування вставками

Клас-обробник швидкого сортування

Рисунок 3.2. Структура проекту у VisualStudio (бібліотеки та класи-обробники)

**2 ОРГАНІЗАЦІЯ КОМАНДНОЇ РОБОТИ**

2.1. Розподіл обов’язків

В рамках командної даного ПЗ було виділено наступні ролі:

Галайко С. С. – керівник проекту, розробник, дизайнер. Обов’яки: постановка задач, координування інших членів команди; створення каркасу ПЗ та основного репозиторію; створення дизайну; внесення виправлень в роботу інших членів команди.

Скляров М. О. – розробник. Обов’язки: розробка класів-обробників (програмування алгоритмів сортування).

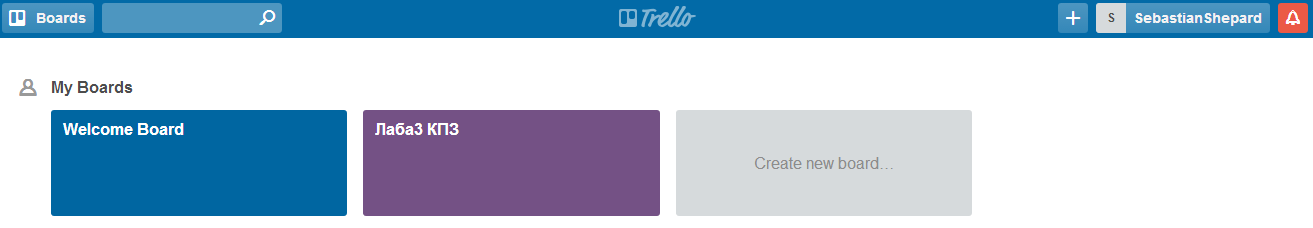
Овчаренко С. Е. – технічний писатель, відповідальний за збір та конфігурації. Обов’язки: оформлення документації, збір і відладка компонентів системи.

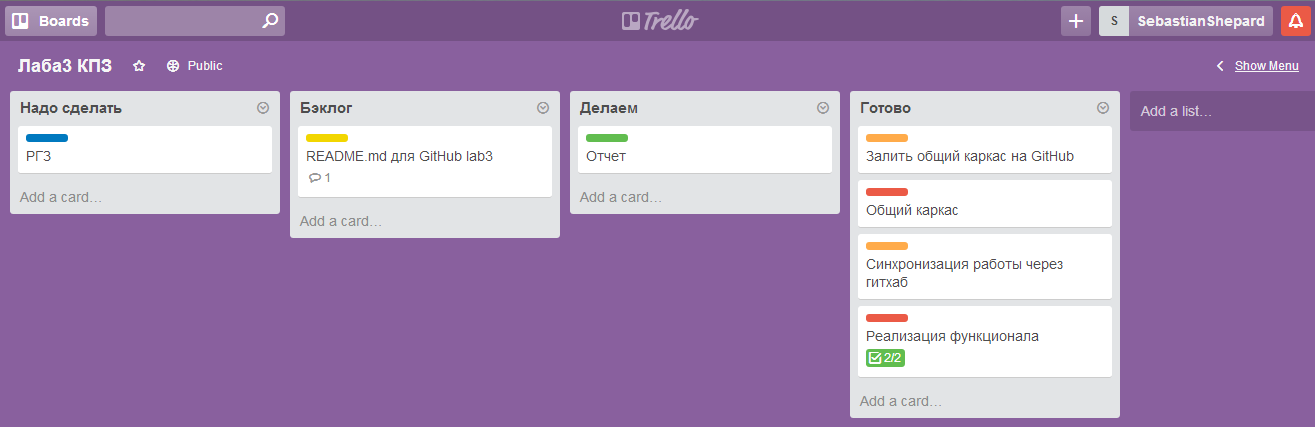
2.2. Комунікації та планування

Планування здійснювалось за допомогою онлайн-сервісу Trello, принцип роботи з яким засновано на використанні «дошки завдань» (рисунок).

Керівником створюється дошка завдань з довільним набором етапів, яка зазвичай відноситься або до одного проекту, або до одної ітерації виконання проекту. За допомогою розсилки електронних листів він відкриває учасникам проекту доступ до дошки завдань.

Керівником проекту завдання відносяться на плановий етап, а потім в «беклог» продукту – перелік завдань, необхідних для успішного завершення розробки ПЗ. По мірі виконання Керівник перевіряє завдання та переміщує їх на наступні етапи. За бажанням керівник може дати учасникам можливість самостійно ставити нові задачі.





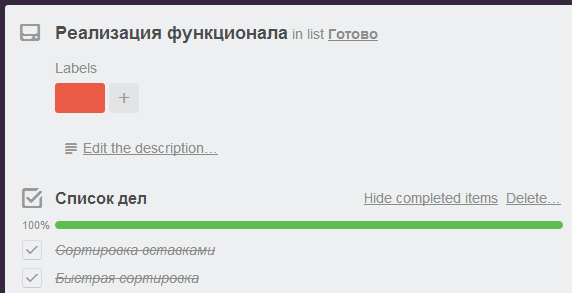
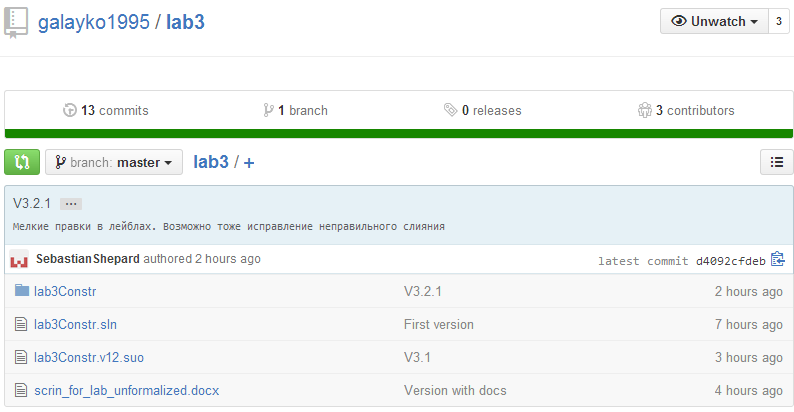


Рисунок 3.3. Планування задач в Trello

Для спілкування між учасниками проекту використовувався програмний продукт від Microsoft – Skype. Вибір був саме таким через те, що даний продукт надає необхідну функцію обміну повідомленнями чи файлами безкоштовно, при цьому дозволяє виконувати демонстрацію дисплею одного з користувачів, що допомагає вирішувати деякі індивідуальні проблеми при написанні програмного коду та роботою з репозиторієм.

2.3. Організація роботи з системою контролю версій

Для організації командного написання програмного коду було обрано GitHub у парі з клієнтом на основі графічного інтерфейсу для Windows. Керівник проекту створив основний репозиторій, запросивши для розробки двох інших членів команди. Члени команди за потреби створювали свою вітку проекту, де реалізовували відведену в задачах частину функціоналу. Відповідальний за збір та конфігурацію забезпечував злиття версій та виконання невеликих виправлень у коді. На рисунку 3.4 показана статистика завантажень нових версій проекту у головний репозиторій. Для швидкого початку роботи з GitHub було обрано графічний інтерфейс замість традиційного консольного. Крім того, GitHub забезпечує можливість ведення зручної документації для проекту та відкривати обмежений доступ для завантаження результуючого проекту. Також є можливість відсліковування всіх правок коду та їх коментування.



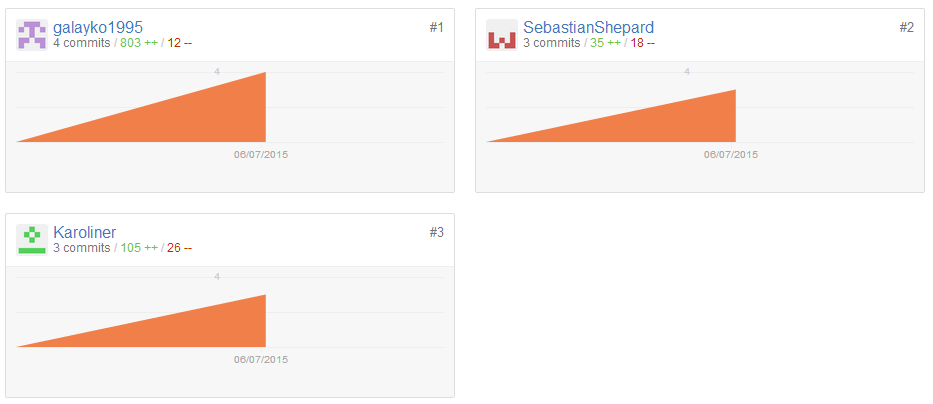
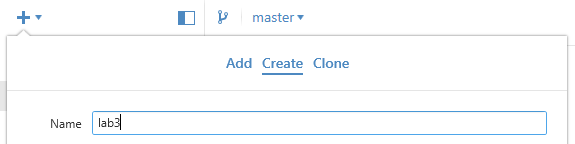


Рисунок 3.4. Статистика завантажень коду у головний репозиторій

**3 КОМАНДНЕ ВИКОНАННЯ ПРОЕКТУ**

3.1. Виконання завдання (Галайко С.С.)

Як показано на рисунку 3.5 нижче, керівник проекту створив основний репозиторій та запросив інших учасників проекту. Після цього був написаний основний каркас ПЗ (лістинг 3.1) разом з інтерфейсом та створений відповідно перший комміт. При написанні коду було задано імена для майбутніх класів обробників алгоритмів сортування та шаблон їх виклику, створено таймер для порівняння швидкості алгоритмів, методи для генерації початкових та представлення результуючих даних. В результаті на репозиторій було завантажено першу версію проекту.



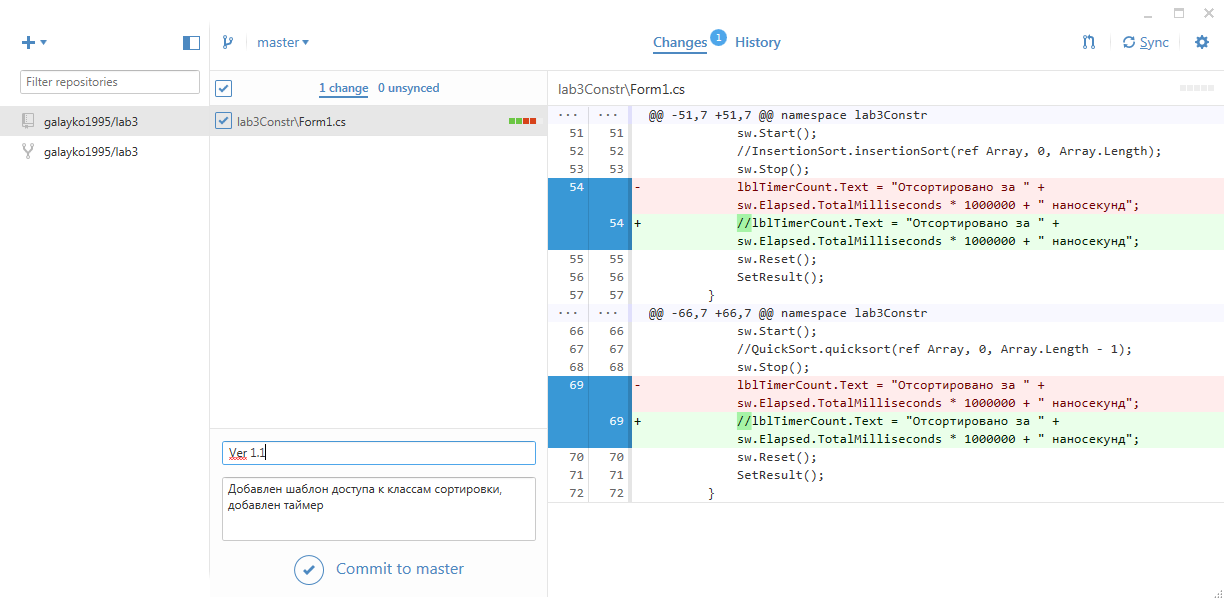


Рисунок 3.5. Робота з GitHub (Галайко С.С.)

Лістинг 3.1 Каркас проекту

namespace lab3Constr

{

public partial class Form1 : MetroForm

{

int[] Array;

int ArrayLength;

Random rand = new Random();

System.Diagnostics.Stopwatch sw = new System.Diagnostics.Stopwatch();

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

ArrayLength = 100;

Array = new int[ArrayLength];

SetInitialData();

}

private void SetInitialData()

{

InitialData.Text = "";

for (int i = 0; i < Array.Length; i++)

{

Array[i] = rand.Next(ArrayLength);

InitialData.Text += Array[i] + "; ";

}

}

private void SetResult()

{

resultData.Text = "";

for (int i = 0; i < Array.Length; i++)

resultData.Text += Array[i] + "; ";

}

private void sort1\_Click(object sender, EventArgs e)

{

sw.Start();

//InsertionSort.insertionSort(ref Array, 0, Array.Length);

sw.Stop();

//lblTimerCount.Text = "Отсортировано за " + sw.Elapsed.TotalMilliseconds \* 1000000 + " наносекунд";

sw.Reset();

SetResult();

}

private void metroButton1\_Click(object sender, EventArgs e)

{

SetInitialData();

}

private void fastSort\_Click(object sender, EventArgs e)

{

sw.Start();

//QuickSort.quicksort(ref Array, 0, Array.Length - 1);

sw.Stop();

//lblTimerCount.Text = "Отсортировано за " + sw.Elapsed.TotalMilliseconds \* 1000000 + " наносекунд";

sw.Reset();

SetResult();

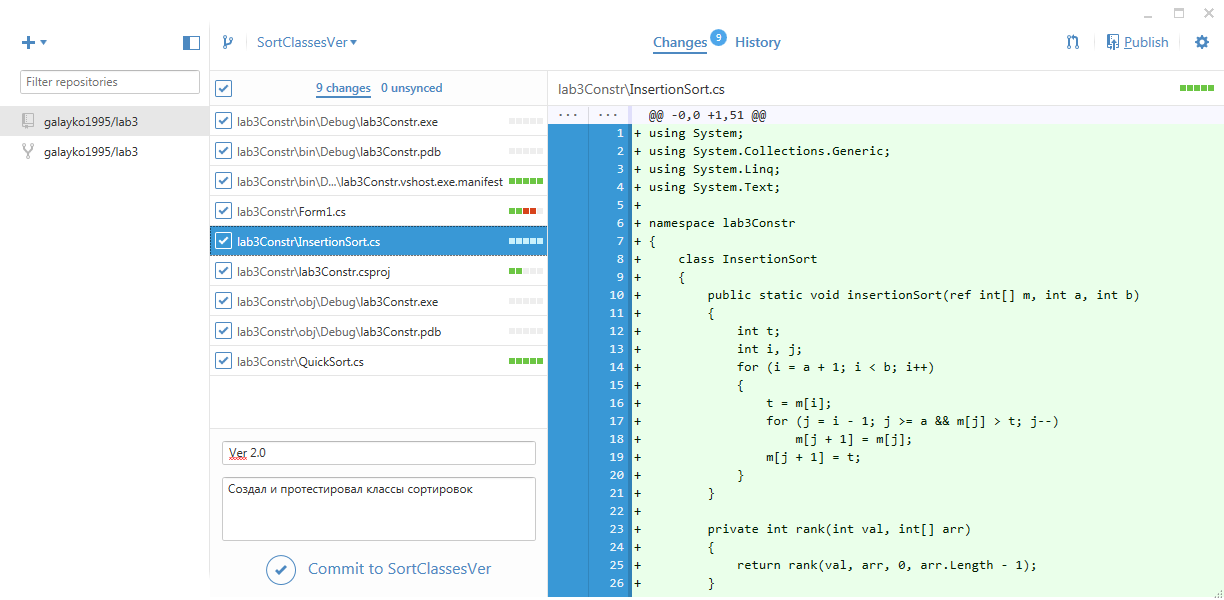
}

}

}

3.2. Виконання завдання (Скляров М.О.)

Як показано на рисунку 3.6, розробник створив власну гілку проекту. В ній він додав спочатку два класи-обробники (лістинг 3.2), для яких пізніше виконав комміт. В результаті перевірки правильності роботи сортування, команда прийняла рішення виконати злиття з основною гілкою проекту. При злитті гілок можуть виникати певні конфлікти. Проблема вирішується синхронізацією новішої гілки з локальним диском і коммітом змін в основну гілку.



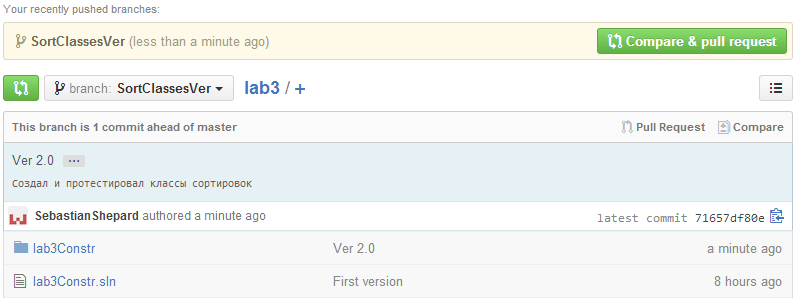
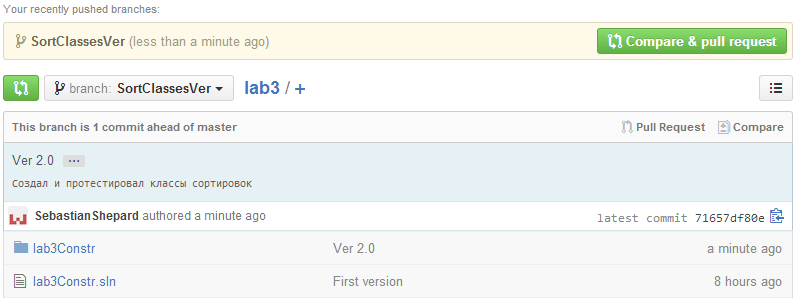




Рисунок 3.5. Робота з GitHub, публікування нової гілки (Скляров М.О.)

Лістинг 3.1 Обробники сортування

class InsertionSort

{

public static void insertionSort(ref int[] m, int a, int b)

{

int t;

int i, j;

for (i = a + 1; i < b; i++)

{

t = m[i];

for (j = i - 1; j >= a && m[j] > t; j--)

m[j + 1] = m[j];

m[j + 1] = t;

}

}

private int rank(int val, int[] arr)

{

return rank(val, arr, 0, arr.Length - 1);

}

private static int rank(int val, int[] arr, int lo, int hi)

{

if (lo > hi)

return -1;

int mid = lo + (hi - lo) / 2;

if (val < arr[mid])

{

return rank(val, arr, lo, mid - 1);

}

else if (val > arr[mid])

{

return rank(val, arr, mid + 1, hi);

}

else

{

return mid;

}

}

}

class QuickSort

{

public static void quicksort(ref int[] a, int l, int r)

{

int temp;

int x = a[l + (r - l) / 2];

//запись эквивалентна (l+r)/2,

//но не вызввает переполнения на больших данных

int i = l;

int j = r;

//код в while обычно выносят в процедуру particle

while (i <= j)

{

while (a[i] < x) i++;

while (a[j] > x) j--;

if (i <= j)

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

i++;

j--;

}

}

if (i < r) quicksort(ref a, i, r);

if (l < j) quicksort(ref a, l, j);

}

}

3.2. Виконання завдання (Овчаренко С.Е.)

Як показано на рисунку 3.7, для репозиторію було створено описання у виді README.MD. Крім того, було заповнено Wiki-дані (документацію) на код проекту.

3. Организация взаимодействия с командой в ходе разработки (указываются основные мероприятия по взаимодействию, как выполнялась постановка задач и отчет о результатах выполнения, какие средства для этого использовались).

4. Порядок и особенности разработки программного кода в рамках своей части работы.

5. Средства тестирования корректности кода и функциональности программы.