Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе № 4

по курсу «Анализ алгоритмов»

на тему: «Параллельные вычисления»

Студент ИУ7-53Б (Группа) _____ (Подпись, дата) В. Марченко (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Ю. В. Строганов (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Л. Л. Волкова (И. О. Фамилия)

Москва — 2022 г.

Оглавление

Введение **3**

1 Аналитическая часть **4**

1.1 Цели и задачи 4

1.2 Алгоритм сортировки слиянием 4

2 Конструкторская часть **6**

2.1 Описание алгоритмов 6

2.1.1 Алгоритм сортировки слиянием 6

2.1.2 Алгоритм объединения массивов 7

3 Технологическая часть **11**

3.1 Требования к программному обеспечению 11

3.2 Средства реализации 11

3.3 Реализация алгоритмов 11

3.3.1 Объединение массивов 11

3.3.2 Сортировка слиянием 13

3.3.3 Сортировка слиянием с использованием параллельных вычислений 13

3.4 Тестовые данные 14

4 Исследовательская часть **16**

4.1 Технические характеристики устройства 16

4.2 Время работы алгоритмов 16

Заключение **19**

Список использованных источников **20**

Введение

Параллелизм — это возможность одновременного выполнения более одной арифметико-логической операции или программной ветви [1]. Параллельная обработка данных, воплощая идею одновременного выполнения нескольких действий, имеет две разновидности: конвейерность и собственно параллельность [2].

Параллельная обработка. Если некое устройство выполняет одну операцию за единицу времени, то тысячу операций оно выполнит за тысячу единиц. Если предположить, что имеется пять таких же независимых устройств, способных работать одновременно и независимо, то ту же тысячу операций система из пяти устройств может выполнить уже не за тысячу, а за двести единиц времени. Аналогично, система из N устройств ту же работу выполнит примерно за $\frac{1000}{N}$ единиц времени [2].

Конвейерная обработка. Идея конвейерной обработки заключается в выделении отдельных этапов выполнения общей операции, причем каждый этап, выполнив свою работу, передает результат следующему, одновременно принимая новую порцию входных данных [2].

В рамках данной лабораторной работы параллельные вычисления будут исследоваться на материале алгоритмов сортировки.

Задача сортировки формально определяется следующим образом.

Вход: последовательность из n чисел $\langle a_1, a_2, ..., a_n \rangle$.

Выход: перестановка (изменение порядка) $\langle a'_1, a'_2, ..., a'_n \rangle$ входной последовательности таким образом, что для ее членов выполняется соотношение $a'_1 \leq a'_2 \leq ... \leq a'_n$ [3].

1 Аналитическая часть

1.1 Цели и задачи

Цель работы: изучить параллельные вычисления и реализовать алгоритм сортировки слиянием с использованием параллельных вычислений.

Задачи лабораторной работы:

1) изучить понятие параллельных вычислений;
2) реализовать последовательный алгоритм сортировки слиянием;
3) реализовать алгоритм сортировки слиянием с использованием параллельных вычислений;
4) провести сравнительный анализ времени работы алгоритмов на основе экспериментальных данных.

1.2 Алгоритм сортировки слиянием

Сортировка слиянием — алгоритм сортировки, который упорядочивает массив в определенном порядке. Эта сортировка использует принцип «разделяй и властвуй» [3]. Сначала массив делится на несколько подмассивов меньшего размера. Затем эти массивы сортируются с помощью рекурсивного вызова. Наконец, все подмассивы соединяются в один, и получается решение исходной задачи.

Этапы сортировки массива слиянием выглядят следующим образом.

1. Сортируемый массив делится на две части примерно одинакового размера.
2. Каждая из получившихся частей сортируется отдельно — рекурсивно с помощью этого же алгоритма сортировки.
3. Два упорядоченных массива соединяются в один.

Притом рекурсивное разбиение задачи на подзадачи происходит до тех пор, пока размер подмассива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).

Вывод из аналитической части

В текущем разделе был рассмотрен алгоритм сортировки слиянием. Этот алгоритм хорошо подходит для внедрения параллельных вычислений, поскольку во время работы он выделяет подмассивы, что позволяет сортировать их одновременно, используя несколько потоков.

2 Конструкторская часть

2.1 Описание алгоритмов

2.1.1 Алгоритм сортировки слиянием

На рисунке 2.1 показана схема алгоритма сортировки слиянием.

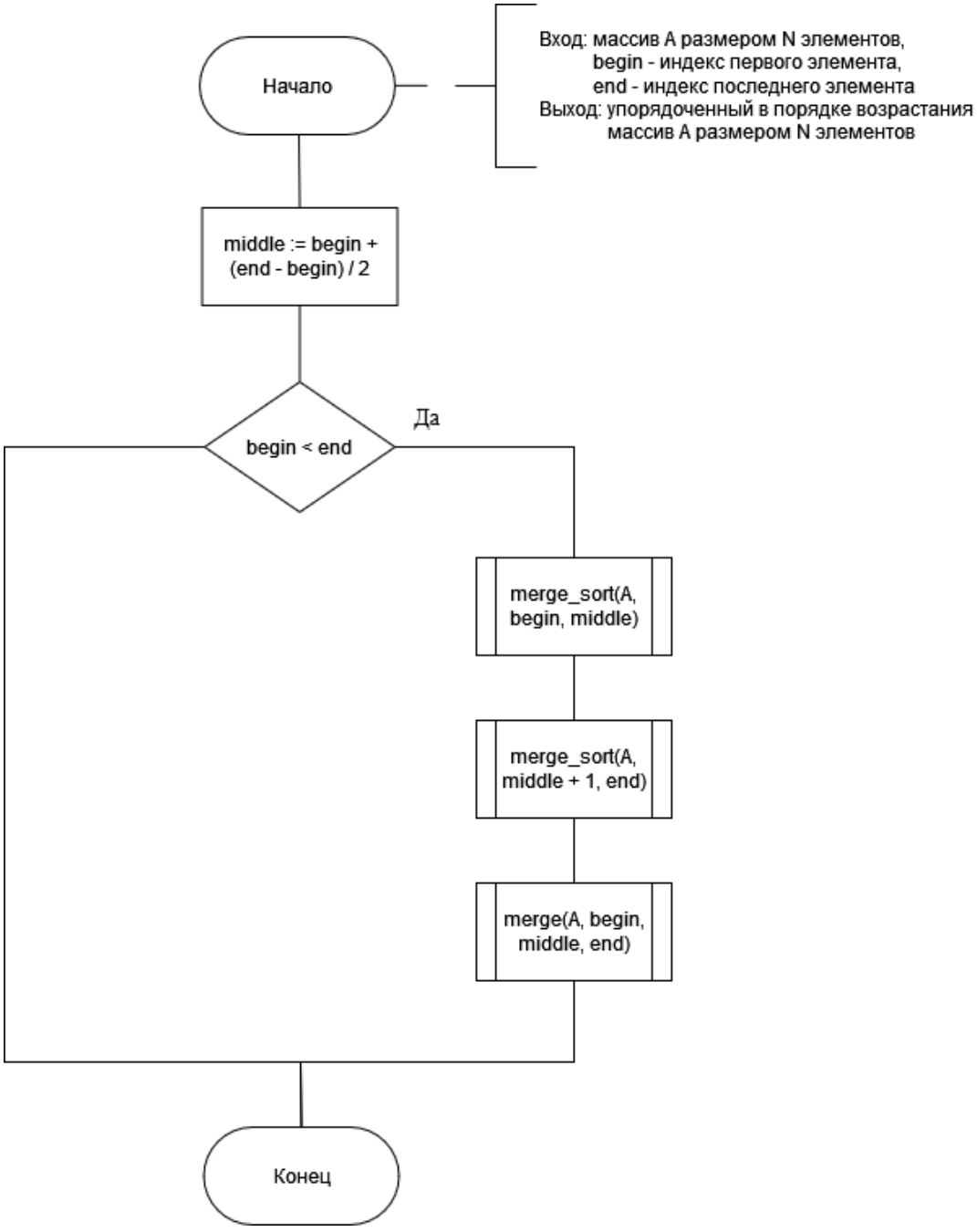

Рисунок 2.1 – Алгоритм сортировки слиянием

2.1.2 Алгоритм объединения массивов

На рисунках 2.2–2.4 показана схема алгоритма объединения массивов.

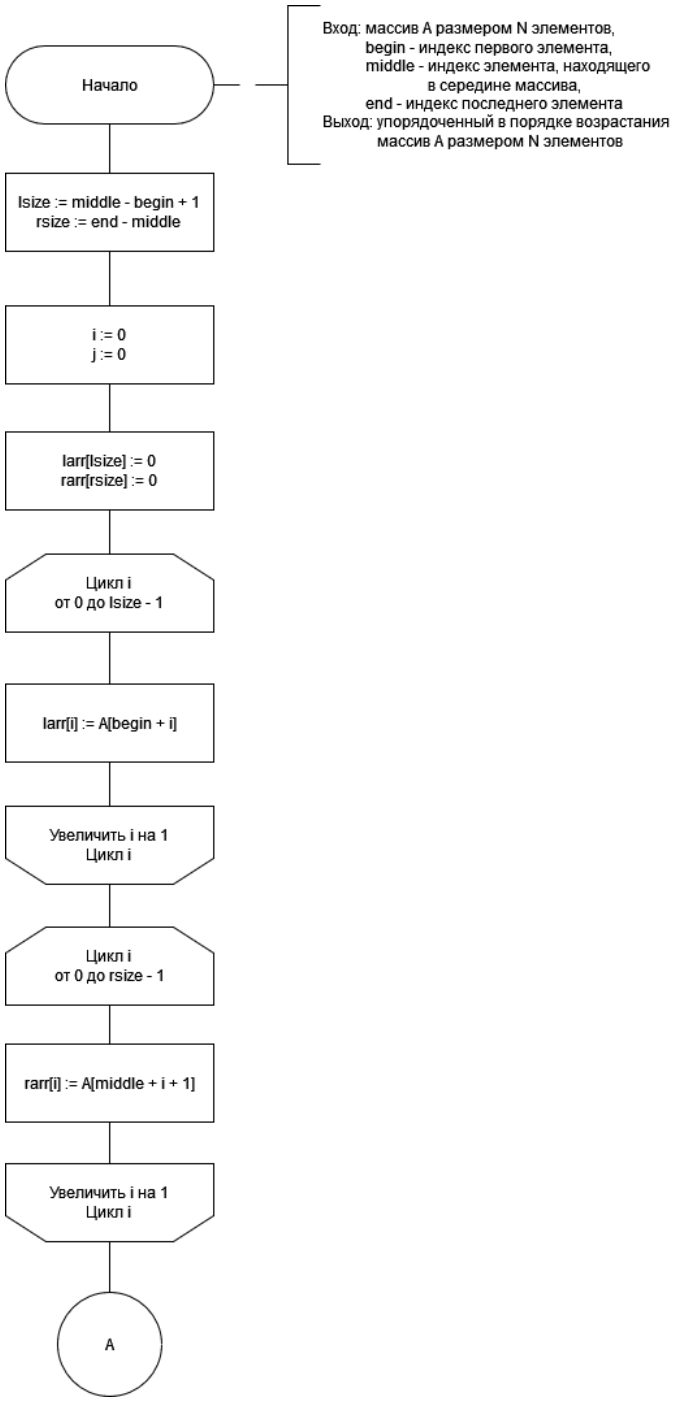

Рисунок 2.2 – Алгоритм объединения массивов — ч. 1

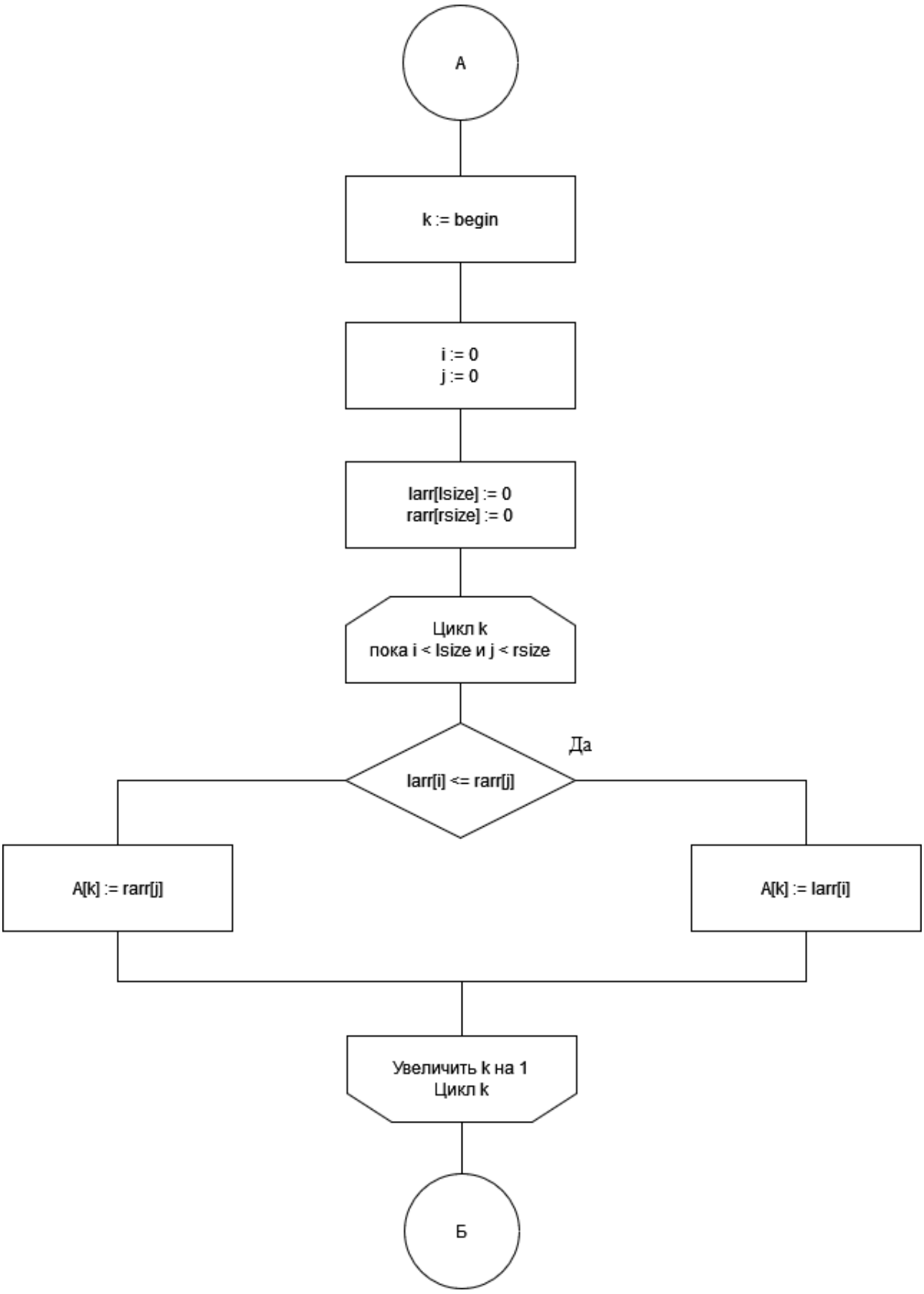

Рисунок 2.3 – Алгоритм объединения массивов — ч. 2

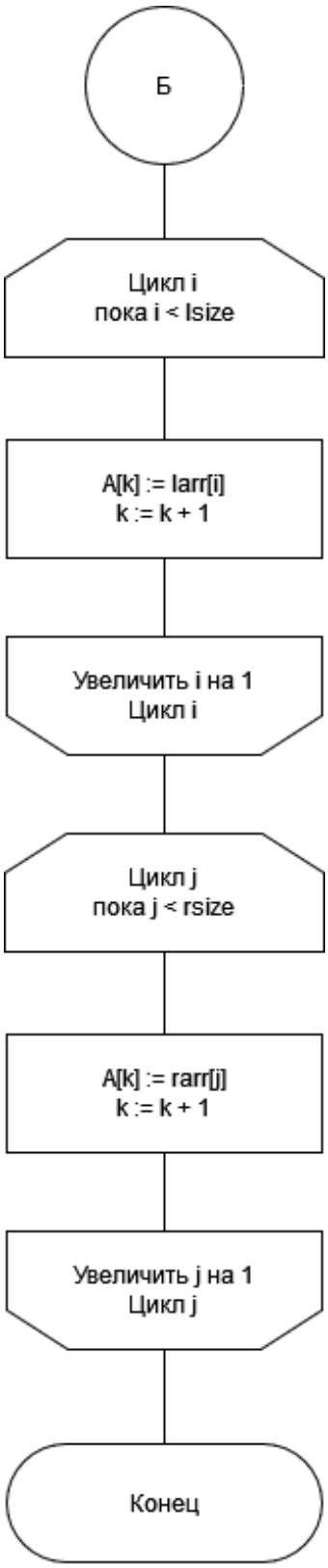

Рисунок 2.4 – Алгоритм объединения массивов — ч. 3

Вывод из конструкторской части

В текущем разделе на основе теоретических данных, полученных из аналитического раздела, была построена схемы алгоритмов сортировки слиянием и объединения массивов.

3 Технологическая часть

В текущем разделе приведены средства реализации алгоритма сортировки слиянием и объединения массивов и листинги кода.

3.1 Требования к программному обеспечению

Программа должна запрашивать у пользователя размер массива и его элементы (целые числа). Затем на экран должен быть выведен результат — отсортированный в порядке возрастания массив.

Программа должна обрабатывать ошибки (например, попытку ввода отрицательного размера массива) и корректно завершать работу с выводом информации об ошибке на экран.

3.2 Средства реализации

Для реализации программного обеспечения был выбран язык **C** ввиду следующих причин:

1) в языке есть возможность создания статических и динамических массивов;
2) есть возможность создавать потоки с помощью библиотечного вызова **pthread_create()**;
3) для считывания данных и вывода их на экран в стандартной библиотеке существуют соответственно функции **scanf()** и **printf()**.

Таким образом, с помощью языка **C** можно реализовать программное обеспечение, которое соответствует перечисленным выше требованиям.

3.3 Реализация алгоритмов

3.3.1 Объединение массивов

В листинге 3.1 показана реализация алгоритма объединения массивов.

Листинг 3.1 – Реализация алгоритма объединения массивов

```
1 int merge(int *const array, const int begin_pos,
```

```
2             const int middle_pos, const int end_pos)
3 {
4     int left_size = middle_pos - begin_pos + 1;
5     int right_size = end_pos - middle_pos, i, j;
6     int *left_array = (int *)malloc(left_size * sizeof(int));
7     if (left_array == NULL)
8         return -1;
9     int *right_array = (int *)malloc(right_size * sizeof(int));
10    if (right_array == NULL)
11    {
12        free(left_array);
13        return -1;
14    }
15    for (i = 0; i < left_size; i++)
16        left_array[i] = array[begin_pos + i];
17    for (i = 0; i < right_size; i++)
18        right_array[i] = array[middle_pos + i + 1];
19    int k = begin_pos;
20    i = j = 0;
21    while (i < left_size && j < right_size)
22    {
23        if (left_array[i] <= right_array[j])
24            array[k++] = left_array[i++];
25        else
26            array[k++] = right_array[j++];
27    }
28    while (i < left_size)
29        array[k++] = left_array[i++];
30    while (j < right_size)
31        array[k++] = right_array[j++];
32    free(left_array);
33    free(right_array);
34    return 0;
35 }
```

3.3.2 Сортировка слиянием

В листинге 3.2 показана реализация последовательного алгоритма сортировки слиянием.

Листинг 3.2 – Реализация последовательного алгоритма сортировки слиянием

```
void merge_sort(int *const array, const int begin_pos, const int
   end_pos)
{
    int middle_pos = begin_pos + (end_pos - begin_pos) / 2;
    if (begin_pos < end_pos)
    {
        merge_sort(array, begin_pos, middle_pos);
        merge_sort(array, middle_pos + 1, end_pos);
        merge(array, begin_pos, middle_pos, end_pos);
    }
}
```

3.3.3 Сортировка слиянием с использованием параллельных вычислений

В листинге 3.3 показана реализация вспомогательной структуры для использования потоков.

Листинг 3.3 – Реализация вспомогательной структуры для использования потоков

```
typedef struct
{
    int begin_pos;
    int end_pos;
    int busy;
    int *array;
} task_t;
```

В листинге 3.4 показана реализация алгоритма сортировки слиянием с использованием параллельных вычислений.

Листинг 3.4 – Реализация алгоритма сортировки слиянием с использованием параллельных вычислений

```
void *merge_sort_thread(void *arg)
{
```

```
3       task_t *task = (task_t *)arg;
4       int begin_pos = task->begin_pos, end_pos = task->end_pos;
5       int middle_pos = begin_pos + (end_pos - begin_pos) / 2;
6       if (begin_pos < end_pos)
7       {
8           merge_sort(task->array, begin_pos, middle_pos);
9           merge_sort(task->array, middle_pos + 1, end_pos);
10          merge(task->array, begin_pos, middle_pos, end_pos);
11      }
12      task->busy = 0;
13      return NULL;
14  }
```

3.4 Тестовые данные

В таблице 3.1 приведены тестовые данные для функции, реализующей алгоритм сортировки слиянием.

Тесты выполнялись по методологии черного ящика (модульное тестирование). Все тесты пройдены успешно.

Таблица 3.1 – Тестовые данные

Описание теста	**Входной массив**	**Ожидаемый результат**	**Выходной массив**
Пустой массив	NULL	NULL	NULL
Один элемент в массиве	-25	-25	-25
Упорядоченный по возрастанию массив	-282, -50, 32, 54, 76, 108	-282, -50, 32, 54, 76, 108	-282, -50, 32, 54, 76, 108
Упорядоченный по убыванию массив	982, 654, 54, 3, -19, -89, -320	-320, -89, -19, 3, 54, 654, 982	-320, -89, -19, 3, 54, 654, 982
Случайно упорядоченный массив	10, -20, 32, -89, -76, -10, 89, 35, 197	-89, -76, -20, -10, 10, 32, 35, 89, 197	-89, -76, -20, -10, 10, 32, 35, 89, 197

Вывод из технологической части

В данном разделе был написан исходный код алгоритмов объединения массивов и сортировки слиянием (последовательного и с использованием параллельных вычислений). Описаны тесты и приведены результаты тестирования.

4 Исследовательская часть

4.1 Технические характеристики устройства

Технические характеристики устройства, на котором было проведено измерение времени работы алгоритмов:

1) операционная система Windows 11 Pro x64;
2) оперативная память 16 ГБ;
3) процессор Intel® Core™ i7-4790K @ 4.00 ГГц.

4.2 Время работы алгоритмов

Время работы функции замерено с помощью ассемблерной инструкции **rdtsc**, которая читает счетчик Time Stamp Counter и возвращает 64-битное количество тиков с момента последнего сброса процессора.

В таблице 4.1 приведено время работы в тиках функции, реализующей алгоритм сортировки слиянием, в зависимости от количества потоков при различном количестве элементов во входном массиве. На рисунке 4.1 изображена зависимость времени работы в тиках функции, реализующей алгоритм сортировки слиянием, от количества потоков при различном количестве элементов во входном массиве.

Таблица 4.1 – Время работы в тиках алгоритма сортировки слиянием в зависимости от количества потоков при различном количестве элементов во входном массиве

Количество потоков	**100**	**200**	**300**	**400**	**500**	**600**	**700**
1	15493	15512	17082	17175	17337	17976	18384
2	53227	70352	104176	120203	122818	153112	188254
3	99208	101051	137593	157357	201546	216977	222431
4	100561	137027	160311	205088	234043	288583	338209
5	102323	152047	203392	245139	272109	316746	343309
6	109825	179446	225440	260116	314463	416150	424402
7	138963	172710	288395	297874	348914	446906	470084
8	137146	214064	240727	314933	383970	440488	497007

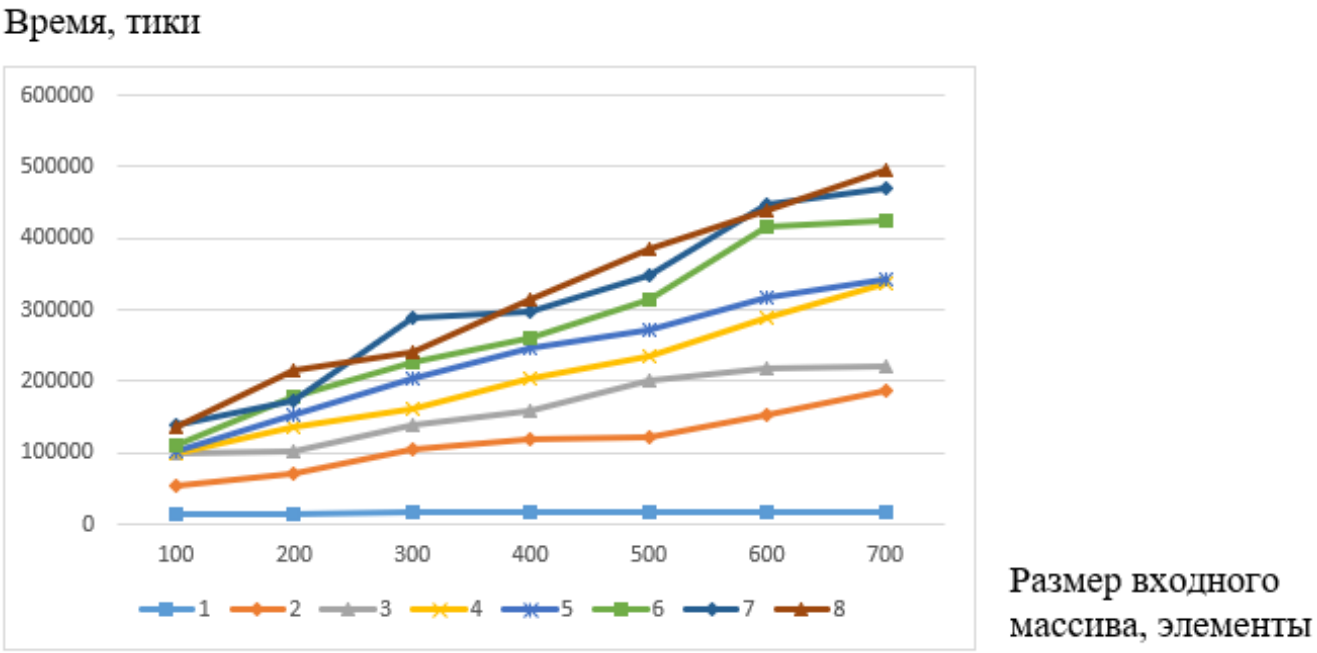

Рисунок 4.1 – Зависимость времени работы в тиках функции, реализующей алгоритм сортировки слиянием, от количества потоков при различном количестве элементов во входном массиве

Вывод из исследовательской части

Согласно полученным при проведении эксперимента данным, быстрее всего работает последовательный алгоритм сортировки слиянием. Это объясняется тем, что при увеличении количества потоков в конце работы алгоритма нужно соединять большее количество массивов. Таким образом, наиболее эффективным можно считать однопоточный алгоритм сортировки слиянием.

Заключение

В рамках данной лабораторной работы была достигнута поставленная цель: были изучены параллельные вычисления и реализован алгоритм сортировки слиянием с использованием параллельных вычислений.

Решены все поставленные задачи:

1) изучено понятие параллельных вычислений;
2) реализован последовательный алгоритм сортировки слиянием;
3) реализован алгоритм сортировки слиянием с использованием параллельных вычислений;
4) проведен сравнительный анализ времени работы алгоритмов на основе экспериментальных данных.

Исходя из данных, полученных в исследовательской части, наболее эффективной является однопоточная реализация алгоритма сортировки слиянием. Чем больше потоков — тем дольше работает алгоритм. Соответственно, использование параллельных вычислений в данном алгоритме является нерациональным.

Список использованных источников

1. *Шпаковский Г. И.* Коротко о параллельном программировании и аппаратуре // Минск. — 2013. — С. 5.
2. *Антонов А. С.* Введение в параллельные вычисления // Московский государственный университет им. М. В. Ломоносова. — 2002. — С. 17—18.
3. *Кормен Т.*, *Лейзерсон Ч.* Алгоритмы: построение и анализ // Вильямс. — 2011. — С. 23—24.