МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Государственное бюджетное образовательное учреждение

высшего образования

«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПУТЕЙ СООБЩЕНИЯ ИМПЕРАТОРА АЛЕКСАНДРА I»

Кафедра «ИНФОРМАЦИОННЫЕ И ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ»

Дисциплина: «Программирование(C)»

ОТЧЕТ

по лабораторной работе № 5

Вариант *19*

Выполнил студент Шефнер А.

Факультета *АИТ*

Группы *ИВБ-211*

Санкт-Петербург

2023

**Постановка задачи**

1. Создать функцию сортировки «пузырьком» итерационную и рекурсивную.

2. Создать функцию сортировки «вставками», для поиска места вставки в отсортированную часть массива применять функцию двоичного поиска.

3. Создать функцию быстрой «qsort» сортировки, отличной от приведённого на лекции.

Во всех функциях:

a) применять указатели (не индексы);

b) для сравнения элементов, применять функцию, передаваемую как указатель, в качестве параметра функции сортировки;

c) посчитать количество сравнений элементов, перестановок и (желательно) глубину рекурсии.

**Пояснения**

В процедуре main вызываются 5 функций тестирования, куда передаются различные функции сравнения элементов. Вы можете попробовать различные функции с различными сортировками или даже написать свою собственную функцию. Примерный вид такой функции описан в файлах book.h и book.c.

**Код программы**

**c\_lab\_5.c** (точка входа программы)

**#include <stdio.h>**

**#include <stdlib.h>**

**#include "book.h"**

**#include "sort.h"**

**#include "testing.h"**

**int main(int argc, char\* argv[])**

**{**

**int count;**

**book\*\* books = get\_books\_from\_file("books.txt", &count);**

**printf("Before sort:\n\n");**

**print\_books(books, count);**

**test\_info\_sort(books, count, bubble\_iter\_info, book\_compare\_pages, "\n\nBubble sort.\n\n");**

**test\_info\_sort(books, count, insertion\_iter\_info, book\_compare\_surname, "\n\nInsertion sort.\n\n");**

**test\_info\_sort\_rec(books, count, bubble\_rec\_info, book\_compare\_year, "\n\nBubble sort recursive.\n\n");**

**test\_info\_sort\_rec(books, count, insertion\_rec\_info, book\_compare\_year, "\n\nInsertion sort recursive.\n\n");**

**test\_info\_sort\_rec(books, count, quicksort\_info, book\_compare\_year, "\n\nQuick sort.\n\n");**

**for(int i = 0; i < count; i++)**

**{**

**free(books[i]);**

**}**

**free(books);**

**printf("\n");**

**system("pause"); *// NOLINT(concurrency-mt-unsafe)***

**return 0;**

**}**

**book.h** (Структура book и основные процедуры работы с массивом книг)

**#pragma once**

**#define SURNAME\_CHAR\_NUMBER 20**

**#define THEME\_CHAR\_NUMBER 50**

**#define SURNAME\_FORMAT "%20s"**

**#define THEME\_FORMAT "%50s"**

**#define YEAR\_FORMAT "%5hu"**

**#define PAGE\_FORMAT "%5hu"**

**struct book;**

**typedef struct book {**

**char surname[SURNAME\_CHAR\_NUMBER];**

**char theme[THEME\_CHAR\_NUMBER];**

**unsigned short year;**

**unsigned short page\_count;**

**} book;**

**book\*\* get\_books\_from\_file(const char\* path, int\* count);**

**int book\_compare\_year(const book\* a, const book\* b);**

**int book\_compare\_pages(const book\* a, const book\* b);**

**int book\_compare\_surname(const book\* a, const book\* b);**

**book.с**

**#define \_CRT\_SECURE\_NO\_WARNINGS**

**#include "book.h"**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**void replace\_char(char str[], char from, char to, int size)**

**{**

**for(int i = 0; i < size; i++)**

**{**

**if(str[i] == from) str[i] = to;**

**}**

**}**

**book\*\* get\_books\_from\_file(const char\* path, int\* count)**

**{**

**FILE\* file;**

**file = fopen(path, "r");**

**fscanf(file, "%i\n", count);**

**book\*\* books = malloc(sizeof(book\*) \* \*count);**

**for(int i = 0; i < \*count; i++)**

**{**

**books[i] = (book\*)malloc(sizeof(book));**

**char\* surname[SURNAME\_CHAR\_NUMBER];**

**char\* theme[THEME\_CHAR\_NUMBER];**

**unsigned short year;**

**unsigned short page\_count;**

**fscanf(file, SURNAME\_FORMAT " " THEME\_FORMAT " " YEAR\_FORMAT " " PAGE\_FORMAT "\n",**

**surname,**

**theme,**

**&year,**

**&page\_count**

**);**

**strcpy(books[i]->surname, surname);**

**strcpy(books[i]->theme, theme);**

**replace\_char(books[i]->theme, '\_', ' ', THEME\_CHAR\_NUMBER);**

**books[i]->year = year;**

**books[i]->page\_count = page\_count;**

**}**

**fclose(file);**

**return books;**

**}**

**int book\_compare\_year(const book\* a, const book\* b)**

**{**

**if(a->year < b->year) return -1;**

**if(a->year > b->year) return 1;**

**return 0;**

**}**

**int book\_compare\_pages(const book\* a, const book\* b)**

**{**

**if(a->page\_count < b->page\_count) return -1;**

**if(a->page\_count > b->page\_count) return 1;**

**return 0;**

**}**

**int book\_compare\_surname(const book\* a, const book\* b)**

**{**

**return strcmp(a->surname, b->surname);**

**}**

**sort.h** (функции сортировки)

**#pragma once**

**void bubble\_iter(void\*\* begin, void\*\* end, int(\*compare)(void\*, void\*));**

**void bubble\_rec(void\*\* begin, void\*\* end, int(\*compare)(void\*, void\*));**

**void insertion\_iter(void\*\* begin, void\*\* end, int(\*compare)(void\*, void\*));**

**void insertion\_rec(void\*\* begin, void\*\* end, int(\*compare)(void\*, void\*));**

**void quicksort(void\*\* begin, void\*\* end, int(\*compare)(void\*, void\*));**

**void bubble\_iter\_info(void\*\* begin, void\*\* end, int(\*compare)(void\*, void\*),**

**int\* swap\_count, int\* cmp\_count**

**);**

**void bubble\_rec\_info(void\*\* begin, void\*\* end, int(\*compare)(void\*, void\*),**

**int\* swap\_count, int\* cmp\_count, int current\_rec, int\* max\_rec**

**);**

**void insertion\_iter\_info(void\*\* begin, void\*\* end, int(\*compare)(void\*, void\*),**

**int\* swap\_count, int\* cmp\_count**

**);**

**void insertion\_rec\_info(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*),**

**int\* swap\_count, int\* cmp\_count, int current\_rec, int\* max\_rec);**

**void quicksort\_info(void\*\* begin, void\*\* end, int(\*compare)(void\*, void\*),**

**int\* swap\_count, int\* cmp\_count, int current\_rec, int\* max\_rec**

**);**

**sort.с**

**#include "sort.h"**

**void S\_swap(void\*\* a, void\*\*b)**

**{**

**void\* tmp = \*a;**

**\*a = \*b;**

**\*b = tmp;**

**}**

**void bubble\_iter(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*))**

**{**

**while(begin < end)**

**{**

**void\*\* iter = begin;**

**while(iter < end - 1)**

**{**

**if(compare(\*iter, \*(iter + 1)) == 1) S\_swap(iter, iter + 1);**

**iter++;**

**}**

**end--;**

**}**

**}**

**void bubble\_rec(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*))**

**{**

**if(begin >= end) return;**

**void\*\* iter = begin;**

**while(iter < end - 1)**

**{**

**if(compare(\*iter, \*(iter + 1)) == 1) S\_swap(iter, iter + 1);**

**iter++;**

**}**

**bubble\_rec(begin, end - 1, compare);**

**}**

**void insertion\_iter(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*))**

**{**

**void\*\* iter\_1 = begin + 1;**

**while(iter\_1 < end)**

**{**

**void\* key = \*iter\_1;**

**void\*\* iter\_2 = iter\_1 - 1;**

**while (iter\_2 >= begin && compare(\*iter\_2, key) == 1)**

**{**

**\*(iter\_2 + 1) = \*iter\_2;**

**iter\_2--;**

**}**

**\*(iter\_2 + 1) = key;**

**iter\_1++;**

**}**

**}**

**void insertion\_rec\_impl(void\*\* begin, void\*\* end, void\*\* iter\_end, int(\* compare)(void\*, void\*))**

**{**

**if(end <= iter\_end) return;**

**void\* key = \*iter\_end;**

**void\*\* iter\_2 = iter\_end - 1;**

**while (iter\_2 >= begin && compare(\*iter\_2, key) == 1)**

**{**

**\*(iter\_2 + 1) = \*iter\_2;**

**iter\_2--;**

**}**

**\*(iter\_2 + 1) = key;**

**insertion\_rec\_impl(begin, end, iter\_end + 1, compare);**

**}**

**void insertion\_rec(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*))**

**{ insertion\_rec\_impl(begin, end, begin + 1, compare); }**

**void\*\* quicksort\_partition(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*))**

**{**

**void\* pivot = \*end;**

**void\*\* pivot\_ptr = begin;**

**for(void\*\* iter\_i = begin; iter\_i < end; iter\_i++)**

**{**

**if(compare(\*iter\_i, pivot) == -1)**

**{**

**S\_swap(pivot\_ptr, iter\_i);**

**pivot\_ptr++;**

**}**

**}**

**S\_swap(pivot\_ptr, end);**

**return pivot\_ptr;**

**}**

**void quicksort\_impl(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*))**

**{**

**if(begin >= end) return;**

**void\*\* pivot\_ptr = quicksort\_partition(begin, end, compare);**

**quicksort\_impl(begin, pivot\_ptr - 1, compare);**

**quicksort\_impl(pivot\_ptr + 1, end, compare);**

**}**

**void quicksort(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*))**

**{ quicksort\_impl(begin, end - 1, compare); }**

**void bubble\_iter\_info(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*), int\* swap\_count, int\* cmp\_count)**

**{**

**while(begin < end)**

**{**

**void\*\* iter = begin;**

**while(iter < end - 1)**

**{**

**(\*cmp\_count)++;**

**if(compare(\*iter, \*(iter + 1)) == 1)**

**{**

**(\*swap\_count)++;**

**S\_swap(iter, iter + 1);**

**}**

**iter++;**

**}**

**end--;**

**}**

**}**

**void bubble\_rec\_info(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*), int\* swap\_count, int\* cmp\_count,**

**int current\_rec, int\* max\_rec)**

**{**

**if(begin >= end)**

**{**

**if(current\_rec > \*max\_rec) \*max\_rec = current\_rec;**

**return;**

**}**

**void\*\* iter = begin;**

**while(iter < end - 1)**

**{**

**(\*cmp\_count)++;**

**if(compare(\*iter, \*(iter + 1)) == 1)**

**{**

**(\*swap\_count)+=1;**

**S\_swap(iter, iter + 1);**

**}**

**iter++;**

**}**

**bubble\_rec\_info(begin, end - 1, compare, swap\_count, cmp\_count, current\_rec + 1, max\_rec);**

**}**

**void insertion\_iter\_info(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*), int\* swap\_count, int\* cmp\_count)**

**{**

**void\*\* iter\_1 = begin + 1;**

**while(iter\_1 < end)**

**{**

**void\* key = \*iter\_1;**

**void\*\* iter\_2 = iter\_1 - 1;**

**while (iter\_2 >= begin && ++(\*cmp\_count) && compare(\*iter\_2, key) == 1 )**

**{**

**\*(iter\_2 + 1) = \*iter\_2;**

**iter\_2--;**

**}**

**\*(iter\_2 + 1) = key;**

**iter\_1++;**

**}**

**}**

**void insertion\_rec\_info\_impl(void\*\* begin, void\*\* end, void\*\* iter\_end, int(\* compare)(void\*, void\*), int\* swap\_count, int\* cmp\_count,**

**int current\_rec, int\* max\_rec)**

**{**

**if(end <= iter\_end)**

**{**

**if(current\_rec > \*max\_rec) \*max\_rec = current\_rec;**

**return;**

**}**

**void\* key = \*iter\_end;**

**void\*\* iter\_2 = iter\_end - 1;**

**while (iter\_2 >= begin && ++(\*cmp\_count) && compare(\*iter\_2, key) == 1)**

**{**

**\*(iter\_2 + 1) = \*iter\_2;**

**iter\_2--;**

**}**

**\*(iter\_2 + 1) = key;**

**insertion\_rec\_info\_impl(begin, end, iter\_end + 1, compare, swap\_count, cmp\_count, current\_rec + 1, max\_rec);**

**}**

**void insertion\_rec\_info(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*), int\* swap\_count, int\* cmp\_count,**

**int current\_rec, int\* max\_rec)**

**{**

**insertion\_rec\_info\_impl(begin, end, begin + 1, compare, swap\_count, cmp\_count, current\_rec, max\_rec);**

**}**

**void\*\* quicksort\_info\_partition(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*), int\* swap\_count, int\* cmp\_count)**

**{**

**void\* pivot = \*end;**

**void\*\* pivot\_ptr = begin;**

**for(void\*\* iter\_i = begin; iter\_i < end; iter\_i++)**

**{**

**(\*cmp\_count)++;**

**if(compare(\*iter\_i, pivot) == -1)**

**{**

**(\*swap\_count)++;**

**S\_swap(pivot\_ptr, iter\_i);**

**pivot\_ptr++;**

**}**

**}**

**(\*swap\_count)++;**

**S\_swap(pivot\_ptr, end);**

**return pivot\_ptr;**

**}**

**void quicksort\_info\_impl(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*), int\* swap\_count, int\* cmp\_count,**

**int current\_rec, int\* max\_rec)**

**{**

**if(begin >= end)**

**{**

**if(current\_rec > \*max\_rec) \*max\_rec = current\_rec;**

**return;**

**}**

**void\*\* pivot\_ptr = quicksort\_info\_partition(begin, end, compare, swap\_count, cmp\_count);**

**quicksort\_info\_impl(begin, pivot\_ptr - 1, compare, swap\_count, cmp\_count, current\_rec + 1, max\_rec);**

**quicksort\_info\_impl(pivot\_ptr + 1, end, compare, swap\_count, cmp\_count, current\_rec + 1, max\_rec);**

**}**

**void quicksort\_info(void\*\* begin, void\*\* end, int(\* compare)(void\*, void\*), int\* swap\_count, int\* cmp\_count,**

**int current\_rec, int\* max\_rec)**

**{ quicksort\_info\_impl(begin, end - 1, compare, swap\_count, cmp\_count, current\_rec, max\_rec); }**

**testing.h** (Функции тестирования)  **#pragma once**

**#include "book.h"**

**void print\_books(book\*\* books, int count);**

**void copy\_ptr\_arr(void\*\* to, int count, void\*\* from);**

**void test\_sort(book\*\* books, int count,**

**void (\*sorting\_func)(void\*\*, void\*\*, int(\*)(void\*, void\*)),**

**int(\*compare\_func)(void\*, void\*),**

**char\* msg);**

**void test\_info\_sort(book\*\* books,**

**int count,**

**void (\*sorting\_func)(void\*\*, void\*\*, int(\*)(void\*, void\*), int\*, int\*),**

**int(\*compare\_func)(void\*, void\*),**

**char\* msg);**

**void test\_info\_sort\_rec(book\*\* books,**

**int count,**

**void (\*sorting\_func)(void\*\*, void\*\*, int(\*)(void\*, void\*), int\*, int\*, int, int\*),**

**int(\*compare\_func)(void\*, void\*),**

**char\* msg);**

**testing.c**

**#include "testing.h"**

**#include <stdio.h>**

**#include <stdlib.h>**

**void print\_books(book\*\* books, int count)**

**{**

**for(int i = 0; i < count; i++)**

**{**

**printf("%s %s %d %d\n",**

**books[i]->surname,**

**books[i]->theme,**

**books[i]->year,**

**books[i]->page\_count**

**);**

**}**

**}**

**void copy\_ptr\_arr(void\*\* to, int count, void\*\* from)**

**{**

**void\*\* end\_ptr = to + count;**

**while(to < end\_ptr)\*(to++) = \*(from++);**

**}**

**void test\_sort(book\*\* books,**

**int count,**

**void (\*sorting\_func)(void\*\*, void\*\*, int(\*)(void\*, void\*)),**

**int(\*compare\_func)(void\*, void\*),**

**char\* msg)**

**{**

**book\*\* arr\_for\_sorting = malloc(sizeof(book\*) \* count);**

**copy\_ptr\_arr(arr\_for\_sorting, count, books);**

**printf(msg);**

**sorting\_func(arr\_for\_sorting, arr\_for\_sorting + count, compare\_func);**

**printf("\nSorted array:\n\n");**

**print\_books(arr\_for\_sorting, count);**

**free(arr\_for\_sorting);**

**}**

**void test\_info\_sort(book\*\* books,**

**int count,**

**void (\*sorting\_func)(void\*\*, void\*\*, int(\*)(void\*, void\*), int\*, int\*),**

**int(\*compare\_func)(void\*, void\*),**

**char\* msg)**

**{**

**book\*\* arr\_for\_sorting = malloc(sizeof(book\*) \* count);**

**copy\_ptr\_arr(arr\_for\_sorting, count, books);**

**printf(msg);**

**int swap\_count = 0;**

**int cmp\_count = 0;**

**sorting\_func(arr\_for\_sorting, arr\_for\_sorting + count, compare\_func, &swap\_count, &cmp\_count);**

**printf("%d swaps, %d compares\n\nSorted array:\n\n", swap\_count, cmp\_count);**

**print\_books(arr\_for\_sorting, count);**

**}**

**void test\_info\_sort\_rec(book\*\* books,**

**int count,**

**void (\*sorting\_func)(void\*\*, void\*\*, int(\*)(void\*, void\*), int\*, int\*, int, int\*),**

**int(\*compare\_func)(void\*, void\*),**

**char\* msg)**

**{**

**book\*\* arr\_for\_sorting = malloc(sizeof(book\*) \* count);**

**copy\_ptr\_arr(arr\_for\_sorting, count, books);**

**printf(msg);**

**int swap\_count = 0;**

**int cmp\_count = 0;**

**int max\_rec = 0;**

**sorting\_func(arr\_for\_sorting, arr\_for\_sorting + count, compare\_func, &swap\_count, &cmp\_count, 0, &max\_rec);**

**printf("%d swaps, %d compares, %d max rec\n\nSorted array:\n\n", swap\_count, cmp\_count, max\_rec);**

**print\_books(arr\_for\_sorting, count);**

**}**

**Отладка приложения**

Далее представлен вывод консоли, поскольку он не помещается в скриншоты. Вы можете проверить вывод консоли самостоятельно с помощью exe файла.

Before sort:

Martin Clean Code Piter 2021 464

Richter CLR via C# 2012 896

Shuuichi Saiki Kusuo no PSI Nan vol. 1 2012 193

Prata C Primer Plus 5th Edition 2004 1202

Marx Das Capital 1867 200

Gyasi Transcendent Kingdom 2020 288

Fujio Doraemon Vol 1 1969 657

Matthes Python Crash Course, 3rd Edition 2023 552

Heisig Remembering the Kanji vol. I 2001 522

Yong An Immense World 2022 464

Privalov Entrance to CVFT 1999 431

Yolen Dragon's Blood: The Pit Dragon Chronicles, Vol. 1 2021 320

Ulrickson A Brief Quadrivium 2023 302

Tokuno New Game vol. 01 2013 126

O'Farrell Hamnet 2020 320

Golden World of Warcraft: Arthas: Rise of the Lich King 2010 416

Nosonov Socio-economic geography 2nd Edition 2019 476

Bubble sort.

80 swaps, 136 compares

Sorted array:

Tokuno New Game vol. 01 2013 126

Shuuichi Saiki Kusuo no PSI Nan vol. 1 2012 193

Marx Das Capital 1867 200

Gyasi Transcendent Kingdom 2020 288

Ulrickson A Brief Quadrivium 2023 302

Yolen Dragon's Blood: The Pit Dragon Chronicles, Vol. 1 2021 320

O'Farrell Hamnet 2020 320

Golden World of Warcraft: Arthas: Rise of the Lich King 2010 416

Privalov Entrance to CVFT 1999 431

Martin Clean Code Piter 2021 464

Yong An Immense World 2022 464

Nosonov Socio-economic geography 2nd Edition 2019 476

Heisig Remembering the Kanji vol. I 2001 522

Matthes Python Crash Course, 3rd Edition 2023 552

Fujio Doraemon Vol 1 1969 657

Richter CLR via C# 2012 896

Prata C Primer Plus 5th Edition 2004 1202

Insertion sort.

0 swaps, 79 compares

Sorted array:

Fujio Doraemon Vol 1 1969 657

Golden World of Warcraft: Arthas: Rise of the Lich King 2010 416

Gyasi Transcendent Kingdom 2020 288

Heisig Remembering the Kanji vol. I 2001 522

Martin Clean Code Piter 2021 464

Marx Das Capital 1867 200

Matthes Python Crash Course, 3rd Edition 2023 552

Nosonov Socio-economic geography 2nd Edition 2019 476

O'Farrell Hamnet 2020 320

Prata C Primer Plus 5th Edition 2004 1202

Privalov Entrance to CVFT 1999 431

Richter CLR via C# 2012 896

Shuuichi Saiki Kusuo no PSI Nan vol. 1 2012 193

Tokuno New Game vol. 01 2013 126

Ulrickson A Brief Quadrivium 2023 302

Yolen Dragon's Blood: The Pit Dragon Chronicles, Vol. 1 2021 320

Yong An Immense World 2022 464

Bubble sort recursive.

60 swaps, 136 compares, 17 max rec

Sorted array:

Marx Das Capital 1867 200

Fujio Doraemon Vol 1 1969 657

Privalov Entrance to CVFT 1999 431

Heisig Remembering the Kanji vol. I 2001 522

Prata C Primer Plus 5th Edition 2004 1202

Golden World of Warcraft: Arthas: Rise of the Lich King 2010 416

Richter CLR via C# 2012 896

Shuuichi Saiki Kusuo no PSI Nan vol. 1 2012 193

Tokuno New Game vol. 01 2013 126

Nosonov Socio-economic geography 2nd Edition 2019 476

Gyasi Transcendent Kingdom 2020 288

O'Farrell Hamnet 2020 320

Martin Clean Code Piter 2021 464

Yolen Dragon's Blood: The Pit Dragon Chronicles, Vol. 1 2021 320

Yong An Immense World 2022 464

Matthes Python Crash Course, 3rd Edition 2023 552

Ulrickson A Brief Quadrivium 2023 302

Insertion sort recursive.

0 swaps, 73 compares, 16 max rec

Sorted array:

Marx Das Capital 1867 200

Fujio Doraemon Vol 1 1969 657

Privalov Entrance to CVFT 1999 431

Heisig Remembering the Kanji vol. I 2001 522

Prata C Primer Plus 5th Edition 2004 1202

Golden World of Warcraft: Arthas: Rise of the Lich King 2010 416

Richter CLR via C# 2012 896

Shuuichi Saiki Kusuo no PSI Nan vol. 1 2012 193

Tokuno New Game vol. 01 2013 126

Nosonov Socio-economic geography 2nd Edition 2019 476

Gyasi Transcendent Kingdom 2020 288

O'Farrell Hamnet 2020 320

Martin Clean Code Piter 2021 464

Yolen Dragon's Blood: The Pit Dragon Chronicles, Vol. 1 2021 320

Yong An Immense World 2022 464

Matthes Python Crash Course, 3rd Edition 2023 552

Ulrickson A Brief Quadrivium 2023 302

Quick sort.

34 swaps, 45 compares, 4 max rec

Sorted array:

Marx Das Capital 1867 200

Fujio Doraemon Vol 1 1969 657

Privalov Entrance to CVFT 1999 431

Heisig Remembering the Kanji vol. I 2001 522

Prata C Primer Plus 5th Edition 2004 1202

Golden World of Warcraft: Arthas: Rise of the Lich King 2010 416

Shuuichi Saiki Kusuo no PSI Nan vol. 1 2012 193

Richter CLR via C# 2012 896

Tokuno New Game vol. 01 2013 126

Nosonov Socio-economic geography 2nd Edition 2019 476

Gyasi Transcendent Kingdom 2020 288

O'Farrell Hamnet 2020 320

Martin Clean Code Piter 2021 464

Yolen Dragon's Blood: The Pit Dragon Chronicles, Vol. 1 2021 320

Yong An Immense World 2022 464

Ulrickson A Brief Quadrivium 2023 302

Matthes Python Crash Course, 3rd Edition 2023 552

Press any key to continue . . .

**Содержимое файлов**

books.txt – исходный файл, из которого считываются книги.

