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(С)»

# О Т Ч Е Т
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.c**

```c
#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.c**

```c
#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**

```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.

```
17
        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
```