Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №15**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Методы внутренней сортировки массивов: быстрая сортировка, сортировка подсчётом

Вариант 19

Выполнила:

Студент группы ИВТ-20-2б

Ананина Арина Юрьевна

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

1. **Постановка задачи**

Дан список ручек и их цвета. Отсортировать список по цветам в алфавитном порядке.

1. **Анализ задачи**
2. Для решения задачи используются типы данных **string** и класс **vector**.
   1. Поля имени структуры и цвета имеют тип данных string.  
      string name, color;
   2. Для хранения данных объектов структуры используется vector. Функция заполнения элементов вектора имеет тип вектор.

vector<Pen> CreateArray(int size)

{

vector<Pen> res;

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

{

Pen newEl;

newEl.Get\_name();

res.push\_back(newEl);

}

return res;

}

* 1. Функция вывода элементов вектора не возвращает значений и принимает в качестве параметра вектор.

void PrintArray(vector<Pen> pens)

{

if (pens.size() < 1)

{

cout << "Список пуст" << endl;

return;

}

for (int i = 0; i < pens.size(); i++)

{

cout << endl << "Элемент " << i + 1 << endl;

pens[i].Print();

}

cout << "----" << endl;

}

* 1. Функция сортировки Шелла принимает в качестве параметра ссылку на первый элемент вектора и ничего не возвращает.

void Shell\_sort(vector<Pen>& list)

* 1. Функция сортировки Хоара принимает ссылку на первый элемент вектора, левую и правую границы массива.

void Quick\_sort(vector<Pen>& list, int left = NULL, int right = NULL)

1. Для решения задачи необходимо:
   1. Разработать методы структуры, которые будут вводить данные в поля объекта структуры.
   2. Реализовать функции ввода-вывода данных из списка (Fill).
   3. Разработать функцию сортировки Шелла (Shell\_sort).
   4. Разработать функцию сортировки Хоара (Quick\_sort).
2. Для решения задачи данные были представлены в следующем виде:
   1. Для сортировки списка с ручками данные представлены в виде vector’а с типом данных Pen.
   2. Для хранения информации о ручке реализована структура Pen.
3. Структура Pen имеет 2 поля: поля имени и цвета (string).

string name, color;

1. Для ввода вывода данных используются следующие операторы:
   1. cin и cout в функциях заполнения, вывода массива, так же, в функции main и методах структуры.
   2. getline используется в методе ввода поля строки.
2. Поставленные задача будут решены следующими действиями:
   1. Сортировка методом Шелла реализована с помощью вложенных циклов for. Первый цикл уменьшает так называемый шаг(step), второй отвечает за разбиение массива на списки, отстающих на step, 3 цикл, вложенный во второй, отвечает за саму сортировку.

void Shell\_sort(vector<Pen>& list)

{

int size = list.size();

for (int step = size / 2; step > 0; step /= 2)

{

for (int i = step; i < size; i++)

{

for (int j = i; j >= step; j -= step)

{

if (list[j].color < list[j - step].color)

{

Pen temp = list[j];

list[j] = list[j - step];

list[j - step] = temp;

}

}

}

}

}

* 1. Сортировка методом Хоара реализована с помощью цикла do while, в который вложено еще два цикла while, которые отвечают за сдвиг индексов, указывающих на текущие элементы. В случае выхода из вложенных индексов, i и j элемент массива меняются. Цикл do while работает до тех пор, пока i-ый индекс <= j-ому индексу. В случае, если j-ый индекс больше левой границы, то функция сортировки вызывается рекурсивно, с новым индексом правой границы (j). Если же i < right, то функция вызывается рекурсивно с новым значением левой границы (i). Для деления массива на двое каждый раз используется переменная pivot, в которой хранится значение элемента с индексом (left+right)/2.

void Quick\_sort(vector<Pen>& list, int left = NULL, int right = NULL)

{

left = left != NULL ? left : 0;

right = right != NULL ? right : list.size() - 1;

int i = left;

int j = right;

string pivot = list[(i + j) / 2].color;

do

{

while (list[i].color < pivot)

{

i++;

}

while (list[j].color > pivot)

{

j--;

}

if (i <= j)

{

Pen tmp = list[i];

list[i] = list[j];

list[j] = tmp;

i++;

j--;

}

} while (i <= j);

if (j > left)

{

Quick\_sort(list, left, j);

}

if (i < right)

{

Quick\_sort(list, i, right);

}

}

1. **Код программы**

#include <iostream>

#include <vector>

#include <string>

#include <windows.h>

using namespace std;

struct Pen

{

string name, color;

void Get\_name()

{

string na, col;

string names[] = { "Stabila", "EricCrauser" };

string colors[] = { "Синий" , "Желтый", "Розовый", "Черный", "Белый" };

name = names[rand() % 2];

color = colors[rand() % 5];

}

void Print()

{

cout << "Название ручки: " << name << endl;

cout << "Цвет ручки: " << color << endl;

}

};

vector<Pen> CreateArray(int size)

{

vector<Pen> res;

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

{

Pen newEl;

newEl.Get\_name();

res.push\_back(newEl);

}

return res;

}

void PrintArray(vector<Pen> pens)

{

if (pens.size() < 1)

{

cout << "Список пуст" << endl;

return;

}

for (int i = 0; i < pens.size(); i++)

{

cout << endl << "Элемент " << i + 1 << endl;

pens[i].Print();

}

cout << "----" << endl;

}

void Shell\_sort(vector<Pen>& list)

{

int size = list.size();

for (int step = size / 2; step > 0; step /= 2)

{

for (int i = step; i < size; i++)

{

for (int j = i; j >= step; j -= step)

{

if (list[j].color < list[j - step].color)

{

Pen temp = list[j];

list[j] = list[j - step];

list[j - step] = temp;

}

}

}

}

}

void Quick\_sort(vector<Pen>& list, int left = NULL, int right = NULL)

{

left = left != NULL ? left : 0;

right = right != NULL ? right : list.size() - 1;

int i = left;

int j = right;

string pivot = list[(i + j) / 2].color;

do

{

while (list[i].color < pivot)

{

i++;

}

while (list[j].color > pivot)

{

j--;

}

if (i <= j)

{

Pen tmp = list[i];

list[i] = list[j];

list[j] = tmp;

i++;

j--;

}

} while (i <= j);

if (j > left)

{

Quick\_sort(list, left, j);

}

if (i < right)

{

Quick\_sort(list, i, right);

}

}

int main()

{

system("color F0");

setlocale(LC\_ALL, "RUS");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int size;

do

{

cout << "Введите кол-во ручек в списке: ";

cin >> size;

} while (size <= 0);

vector<Pen> pensMas;

pensMas = CreateArray(size);

PrintArray(pensMas);

int vvod;

cout << "1 - Сортировка методом Шелла " << endl;

cout << "2 - Сортировка методом Хоара " << endl;

cin >> vvod;

if (vvod == 1)

{

cout << "Отсортированный список: " << endl;

Shell\_sort(pensMas);

PrintArray(pensMas);

}

else if (vvod == 2)

{

cout << "Отсортированный список: " << endl;

Quick\_sort(pensMas);

PrintArray(pensMas);

}

else

{

cout << "Введено некорректное значение:(" << endl;

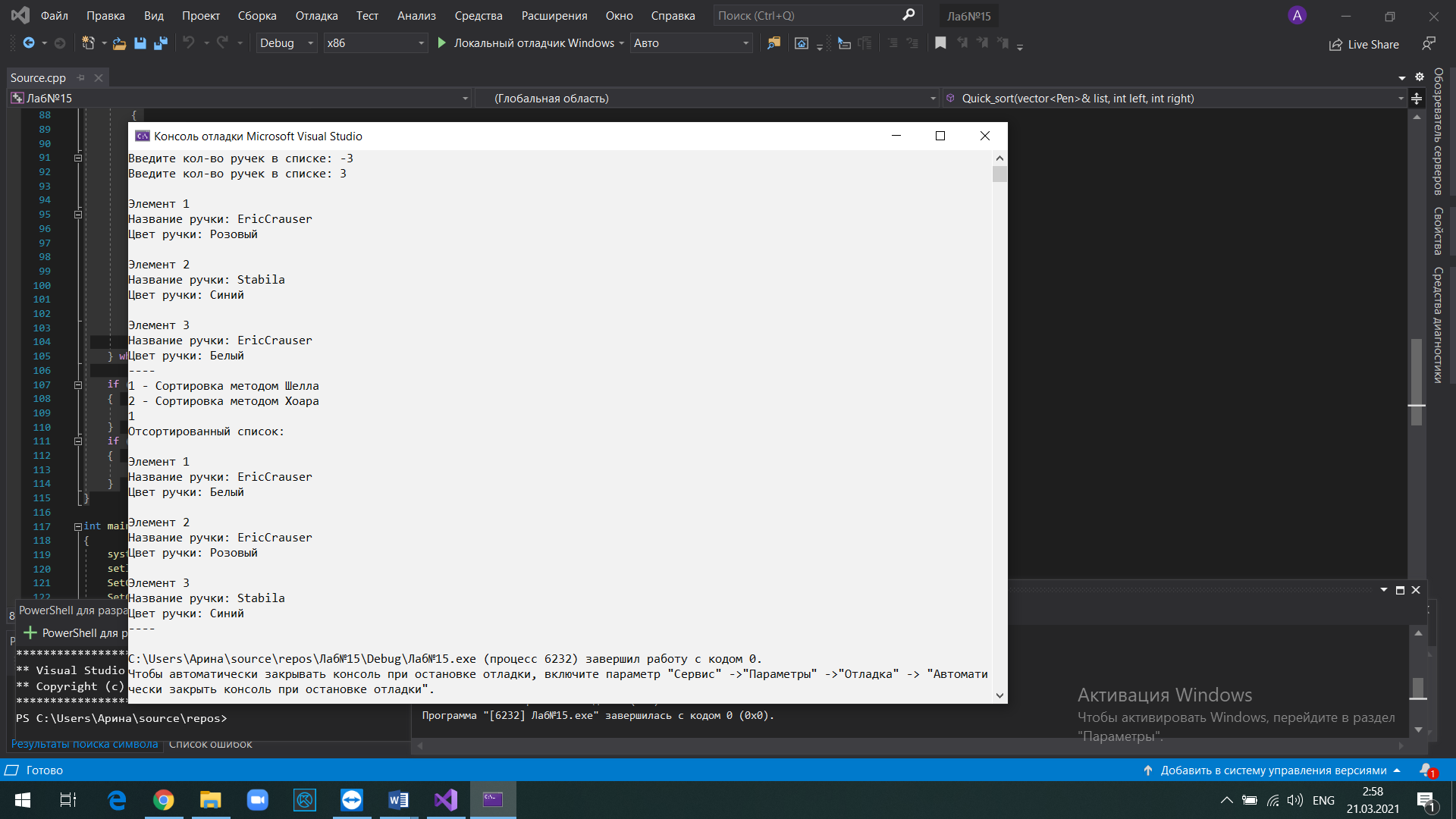
}

return 0;

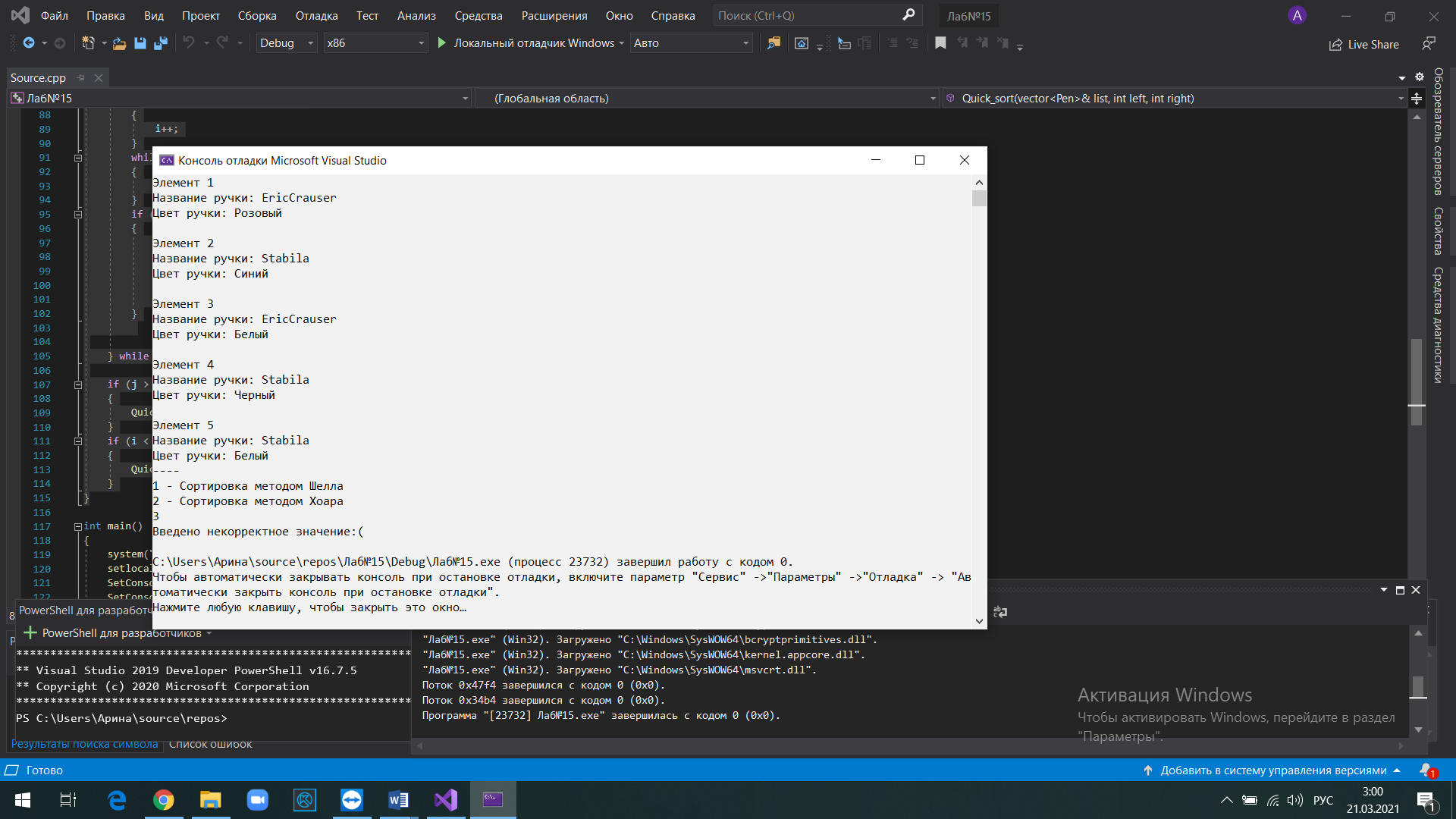
}

1. **Результат выполнения**

При вводе пользователем отрицательного значения кол-ва ручек, программа будет запрашивать повторный ввод, пока не будет введено корректное значение.



После создания списка ручек, пользователю будет предложен выбор одной сортировки из двух, при вводе несуществующей команды программа выведет сообщение об этом.



1. **Блок-схема**

