Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №15**

Дисциплина: основы алгоритмизации и программирования

Тема: “Быстрые сортировки”

Вариант 19

Выполнил:

Студент группы РИС-20-2Б

Пономарёв Артём Викторович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Цель задачи**

Научиться программировать различные методы сортировки одномерных массивов.

**Постановка задачи**

Дан список языков программирования и фамилии их авторов. Упорядочить в алфавитном порядке фамилии авторов.

2

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать структуру, поля которой будут использоваться для ввода языка программирования (ЯП) и фамилии его автора.

struct Spisok

{

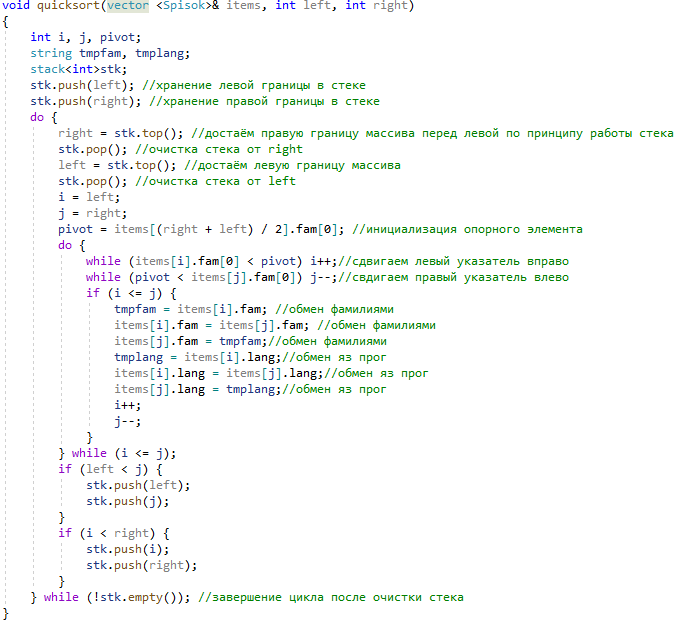
string lang, fam;

};

**1.2.** Организовать массив, который будет хранить введённые элементы. В данном случае удобно использовать vector, чтобы экономить память.

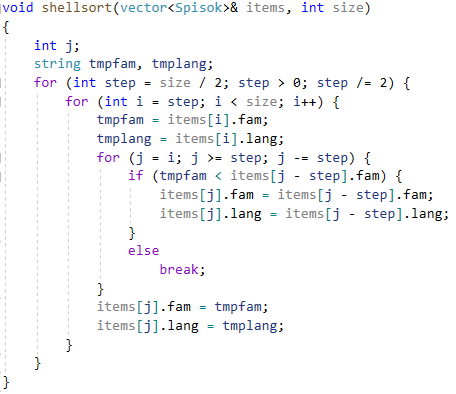
vector <Spisok> array(n);

**1.3.** Организовать функцию quicksort, которая будет сортировать фамилии создателей ЯП по алфавитному порядку, методом Хоара.



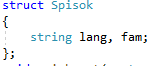
3

**1.4.** Организовать функцию shellsort(), которая будет сортировать фамилии создателей ЯП по алфавитному порядку, методом Хоара.



**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Структура Spisok для запоминания в списке ЯП и фамилий.



**2.2.** Массив vector для хранения данных о нескольких людях.

vector <Spisok> array(n);

**2.3.** Для функции quicksort() используются следующие аргументы:

1. Ссылка на массив, чтобы мы могли переставлять в нём элементы.

2. int left – индекс первого элемента, int right – индекс последнего элемента

Сама функция имеет тип void, так как никаких значений возвращать не нужно.

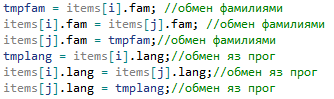
void quicksort(vector <Spisok>& items, int left, int right)

4

**2.4.** В функции quicksort() используем стек для хранения границ подмассивов.

stack<int>stk;

**2.5.** В quicksort() также используем переменные tmplang и tmpfam для последующего обмена значениями.



**2.6.** Переменная pivot хранит в себе первую букву опорного элемента для сортировки Хоара.

pivot = items[(right + left) / 2].fam[0];

**2.7.** Переменные i и j будут указателями, которые изначально будут хранить крайние индексы массива в стеке.

stk.push(left); //хранение левой границы в стеке

stk.push(right); //хранение правой границы в стеке

...

right = stk.top(); //достаём правую границу массива перед левой по принципу работы стека

stk.pop(); //очистка стека от right

left = stk.top(); //достаём левую границу массива

stk.pop(); //очистка стека от left

i = left;

j = right;

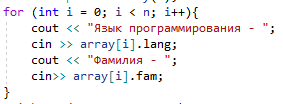
**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Данные вводятся через консоль в виде массива структур (тип Spisok и полями типа string).

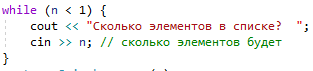
**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Ввод данных на консоль реализован функцией cin в цикле for

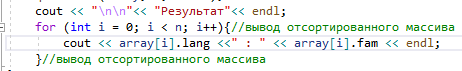
5



**4.2.** Переменная n проверяется на отрицательность, так как количество людей не может быть меньше 0.

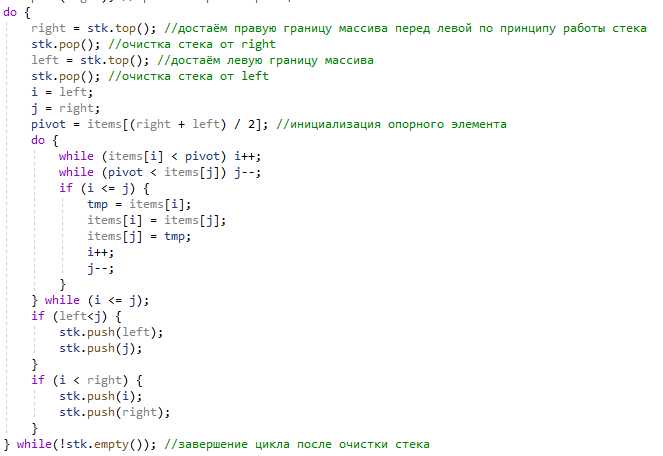


**4.3.** Вывод на консоль реализован через команду cout в цикле for



**5.** Поставленные задачи будут решены следующими действиями:

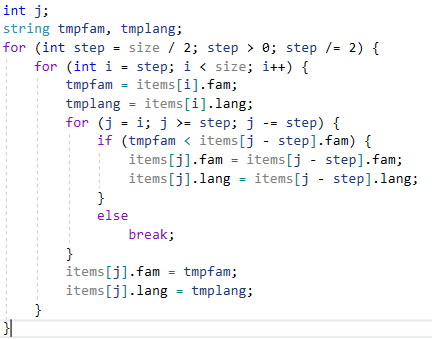
**5.1.** В функции quicksort() будет организован цикл do while, который будет работать, пока стек не очистится. Внутри цикла использовать ещё один цикл do while, в котором будут выполняться все основные действия для сортировки.



6

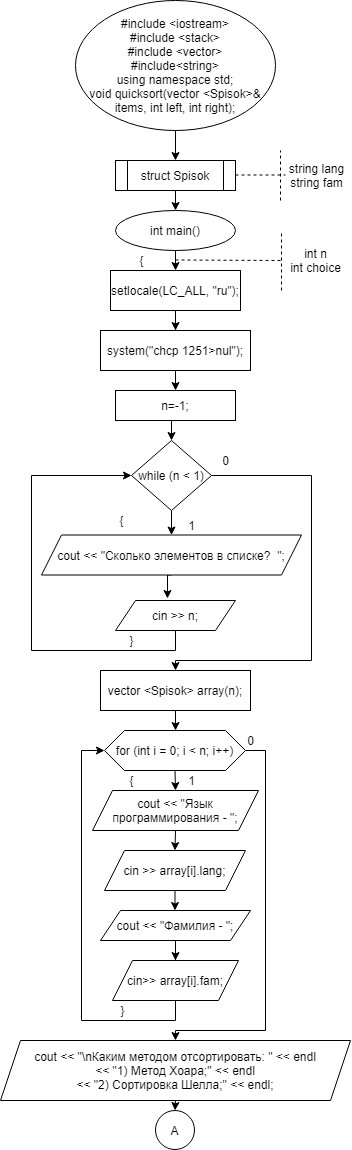
Функция реализует быструю сортировку Хоара, для которой используется стек в явном виде, чтобы сохранять границы подмассивов. В функции происходит сравнение первых символов фамилий через таблицу ASCII с дальнейшей сортировкой.

**5.2.** В функции shellsort() будет выполнятся сортировка методом Шелла: будет организован цикл for, который будет выполняться, пока, так называемый, шаг (итератор цикла) не достигнет 0. Внутри цикла будет выполнятся другой цикл for, в котором будут сортироваться группы элементов, кратные шагу. Внутри данного цикла будет реализован ещё один цикл for, который по принципу метода вставки будет сортировать элементы.

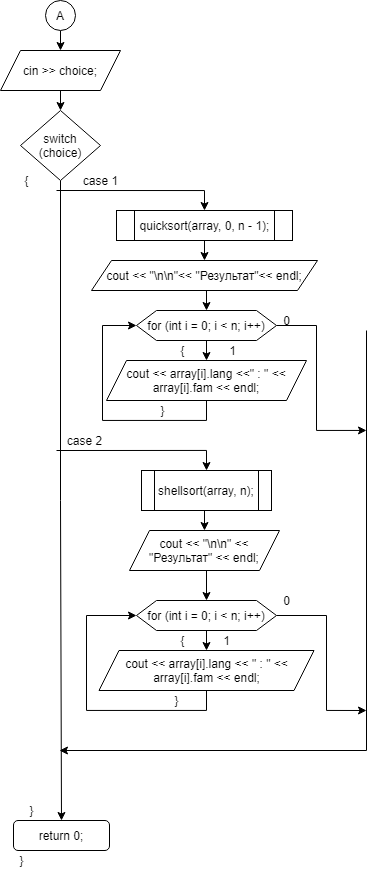


7

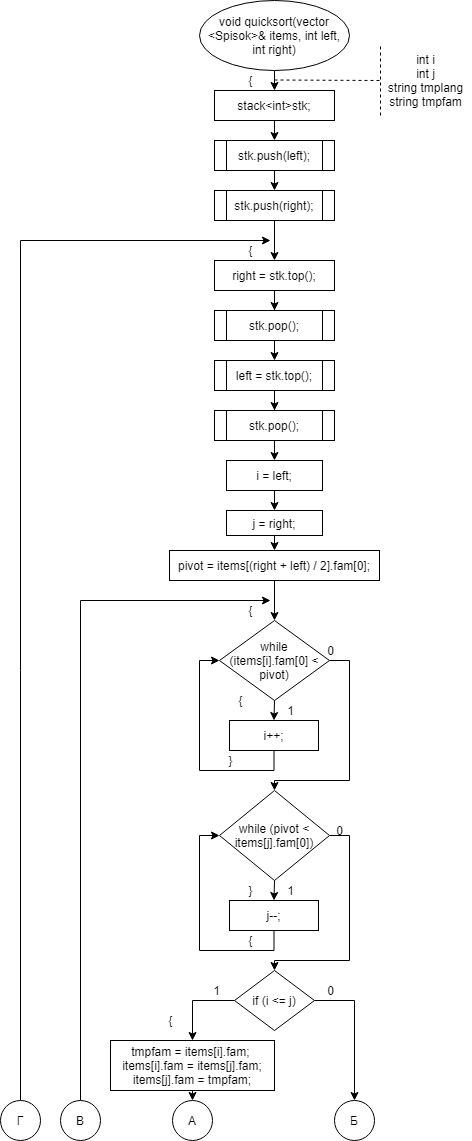
**Блок-схема**



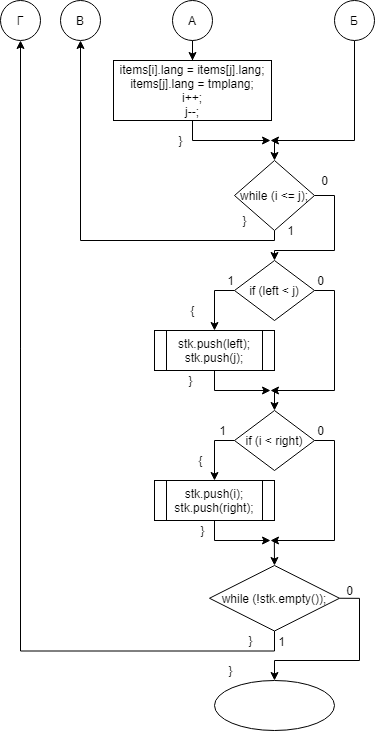
8



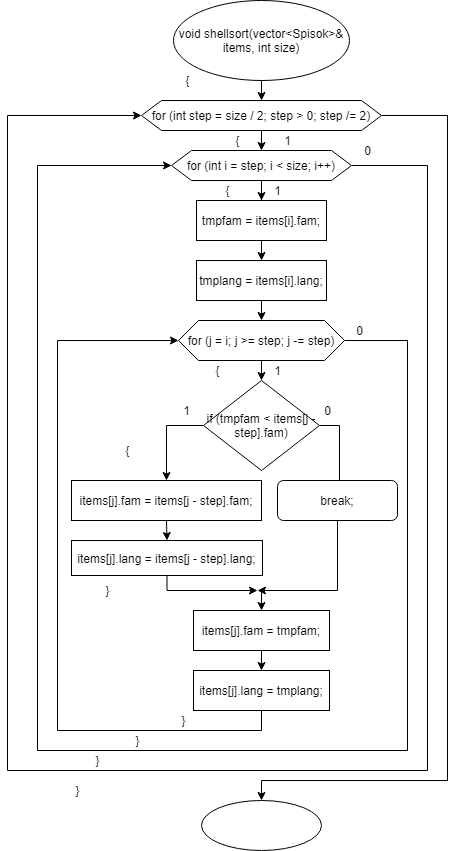
9



10



11



12

**Код**

#include <iostream>

#include <stack>

#include <vector>

#include<string>

using namespace std;

struct Spisok

{

string lang, fam;

};

void shellsort(vector<Spisok>& items, int size)

{

int j;

string tmpfam, tmplang;

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

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

tmpfam = items[i].fam;

tmplang = items[i].lang;

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

if (tmpfam < items[j - step].fam) {

items[j].fam = items[j - step].fam;

items[j].lang = items[j - step].lang;

}

else

break;

}

items[j].fam = tmpfam;

items[j].lang = tmplang;

}

}

}

void quicksort(vector <Spisok>& items, int left, int right)

{

int i, j, pivot;

string tmpfam, tmplang;

stack<int>stk;

stk.push(left); //хранение левой границы в стеке

stk.push(right); //хранение правой границы в стеке

do {

right = stk.top(); //достаём правую границу массива перед левой по принципу работы стека

stk.pop(); //очистка стека от right

left = stk.top(); //достаём левую границу массива

stk.pop(); //очистка стека от left

i = left;

j = right;

pivot = items[(right + left) / 2].fam[0]; //инициализация опорного элемента

do {

while (items[i].fam[0] < pivot) i++;//сдвигаем левый указатель вправо

while (pivot < items[j].fam[0]) j--;//свдигаем правый указатель влево

if (i <= j) {

tmpfam = items[i].fam; //обмен фамилиями

items[i].fam = items[j].fam; //обмен фамилиями

items[j].fam = tmpfam;//обмен фамилиями

tmplang = items[i].lang;//обмен яз прог

items[i].lang = items[j].lang;//обмен яз прог

items[j].lang = tmplang;//обмен яз прог

i++;

j--;

}

} while (i <= j);

13

if (left < j) {

stk.push(left);

stk.push(j);

}

if (i < right) {

stk.push(i);

stk.push(right);

}

} while (!stk.empty()); //завершение цикла после очистки стека

}

int main()

{

setlocale(LC\_ALL, "ru");

system("chcp 1251>nul");

int n=-1;

while (n < 1) {

cout << "Сколько элементов в списке? ";

cin >> n; // сколько элементов будет

}

vector <Spisok> array(n);

for (int i = 0; i < n; i++){

cout << "Язык программирования - ";

cin >> array[i].lang;

cout << "Фамилия - ";

cin>> array[i].fam;

}

cout << "\nКаким методом отсортировать: " << endl

<< "1) Метод Хоара;" << endl

<< "2) Сортировка Шелла;" << endl;

int choice;

cin >> choice;

switch (choice) {

case 1:

quicksort(array, 0, n - 1);

cout << "\n\n" << "Результат" << endl;

for (int i = 0; i < n; i++) {//вывод отсортированного массива

cout << array[i].lang << " : " << array[i].fam << endl;

}//вывод отсортированного массива

break;

case 2:

shellsort(array, n);

for (int i = 0; i < n; i++) {//вывод отсортированного массива

cout << array[i].lang << " : " << array[i].fam << endl;

}//вывод отсортированного массива

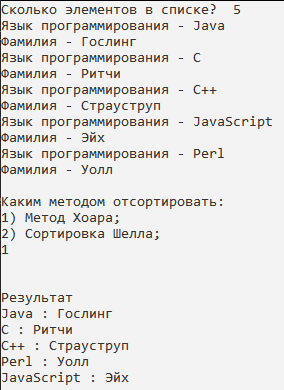
break;

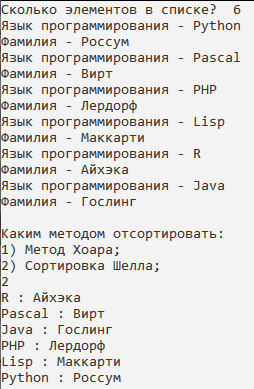
}

}

14

**Результат работы программы**





15