СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день любая организация сталкивается с постоянно растущим объёмом хранимой и обрабатываемой информации. Это может включать в себя различные типы данных. Например, сведения о клиентах компании, информация о предоставляемых услугах и т.д. В связи с этим возрастает необходимость внедрения автоматизированных систем в организации.

Трудно себе представить формирование базы данных о туристах без помощи компьютера. Сегодня все сведения о туристах и услугах хранятся в компьютере. Это естественно облегчает работу менеджера турагенства.

Разработанное ПО было запрограммировано для создания, изменения, удаления и обработки данных простейшим способом, что позволяет теперь быстро и эффективно использовать его. В программе предусмотрены: администратор и пользователи, имеющие разные права доступа. Администратор имеет все инструменты для работы с данными. Пользователи же ограничены в своих правах и могут лишь просматривать, искать и фильтровать данные.

Цель курсового проекта – разработка программы учета клиентов туристической компании.

Для реализации цели необходимо решить следующие задачи:

1. Провести обзор методов и алгоритмов решения поставленной задачи.
2. Определить структуру данных.
3. Разработать схему алгоритма и алгоритм по шагам.
4. Выполнить программную реализацию информационной системы учета клиентов туристической компании.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ОБ ИСПОЛЬЗУЕМЫХ АЛГОРИТМАХ

Одним из самых частых действий в программировании и повседневной жизни является поиск. Существует множество способов и методов поиска данных. Одним из таких является линейный поиск. При линейном поиске происходит сравнение каждого элемента списка с ключом поиска. Поскольку список не упорядочен особым образом, вероятность нахождения требуемого значения в первом и последнем элементах списка одинакова. Таким образом, в среднем программа должна сравнить ключ поиска с половиной элементов массива. Метод линейного поиска хорошо работает для небольших или неотсортированных списков [1]. Однако для больших списков он неэффективен.

Пример работы алгоритма:

```
void search()
{
 record_size();
 int n=cou;
 system("cls");
 char poisk[30], per;
 long int poisk_int;
 float poisk_float;
 int p=0,j=0;
 cout<<"\nПоиск по:";
 cout<<"\n 1.Имени\n 2.Фамилии\n 3.Номеру\n 4.Для
прекращения\n";
 per=_getch();
 read_file();
 system("cls");
 switch(per)
 {
  case '1':
  cout<<"Введите параметр поиска\n";
  cin >> poisk;
  for(int j=0;j<n;j++)
  {
   if(strcmp(poisk,file[j].incl_ns.name)==0)
```

```
    {

      for_search(j);
      printf("%d",j);
    }
    else p++;
  }
 if(p=n+1) printf("Таких записей нет\n");
 return ;
```

Ввод новой информации осуществляется с помощью стандартных функций языка C++ cout и cin, которые взаимодействуют со стандартным потоком ввода-вывода данных (клавиатура) [3].

Удаление данных осуществляется при помощи функции deleteData, которая удаляет необходимые данные, перемещая все последующие элементы на количество элементов, соответствующее количеству удалённых.

Для изменения уже существующих данных используется функция updateData, которая по заданному критерию предоставляет опции для редактирования необходимой информации.

Для сортировки информации используется алгоритм быстрой сортировки. Прежде всего при выполнении данного алгоритма выбирается опорный элемент (в данном случае из середины массива). Затем происходит разделение элементов на две части: все элементы, меньшие опорного, помещаются слева от него, а большие опорного, – справа от него. После этого происходит рекурсивный вызов этой же функции в зависимости от условия, таким образом происходит сортировка, если необходимо, соответственно левой части, а затем правой. Это происходит до тех пор, пока все элементы не будут отсортированы [4].

```
void quick_sort(int names[20], int left, int right)
{
int i, j;
int buf1, buf2;
i = left, j = right;
buf1 = names[(left + right) / 2];
do {
        while ((names[i] < buf1) && (i < right)) i++;
        while ((names[j] > buf1) && (j > left)) j--;
```

```
        if (i <= j) {
                buf2 = names[i];
                names[i] = names[j];
                names[j] = buf2;
                i++;
                j--;
        }
} while (i <= j);
if (left < j) quick_sort(names, left, j);
if (right > i) quick_sort(names, i, right);
}
```

2 ОПИСАНИЕ ОРГАНИЗАЦИИ СТРУКТУР ХРАНИМЫХ ДАННЫХ

Структура – это совокупность объектов, возможно, различных типов данных, объединенных одним именем, предоставляющая способ совместного хранения информации. В качестве объектов могут выступать переменные, массивы, указатели и другие структуры. Объекты, которые образуют структуру, называются членами структуры (элементами) [2].

Как и массив, структура представляет собой совокупность данных, но отличается от него тем, что к ее элементам необходимо обращаться по имени, и ее различные элементы не обязательно должны принадлежать одному типу.

Структуры удобно использовать там, где разнообразные данные, относящиеся к одному и тому же объекту, необходимо объединять. Например, ученика средней школы характеризуют следующие данные: фамилия, имя, дата рождения, класс, возраст.

В качестве выбора способа описания входных данных:

1. Структура *Authorisation* для учетных записей пользователя с полями login, password, role;

```
struct Authorisation   //структура для хранения аккаунтов
{
string login;  // логин пользователя
string password; //пароль пользователя
int role;  // приоритетность
};
```

2. Структура *trip* для информации о туристах с полями N,fio, end, time1,time2.

```
struct trip          // структура для хранения данных
{
int N;  // номер туриста
string type; //ФИО туриста
string end;  //пункт назначения
int time1;  //час отправления
int time2;  // час прибытия
};
```

3 СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ФУНКЦИЙ ПРИЛОЖЕНИЯ

Прототипы функций, использующихся в данном приложении, представлены в таблице 3.1.

Таблица 3.1 – Разработанные функции

Категория	Название функции	Описание
1	2	3
Меню	void menuAdminZapis(Authorisation *arr, int N_acc, trip *arr1, int n)	Меню для работы с уч. записями (администратор)
	void menuAdminFile(Authorisation *arr, int N_acc, trip *arr1, int n)	Меню для работы с файлом (администратор)
	void menuAdmin(Authorisation *arr, int N_acc, trip *arr1, int n)	Меню администратора
	void menuAdminData(Authorisation *arr, int N_acc, trip *arr1, int n);	Меню для работы с данными (администратор)
	void menuCorrectData(Authorisation *arr, int N_acc, trip *arr1, int n);	Меню для редактирования данных (администратор)
	void menuWorkData(Authorisation *arr, int N_acc, trip *arr1, int n);	Меню для обработки данных (администратор)
	void menuUser(Authorisation *arr, int N_acc, trip *arr1, int n);	Меню пользователя
	void menuPoisk(Authorisation *arr, int N_acc, trip *arr1, int n);	Меню для поиска
	void menuSort(Authorisation *arr, int N_acc, trip *arr1, int n);	Меню для сортировки

Продолжение таблицы 3.1

1	2	3
Работа с учетными записями	void showAccounts(Authorisation *arr_of_accounts, int N_acc);	Просмотр уч. записей
	void readFileAccounts(Authorisation *arr_of_accounts, int &N_acc);	Считывание уч. записей из файла в массив
	void deleteAccount(Authorisation *arr, int &N_acc,trip *arr1,int n);	Удаление уч. записи
	void addAccount(Authorisation *new_account, int &N_acc);	Добавление уч. записи
	void updateAcc(Authorisation *arr, int &n);	Редактирование уч. записи
Автори-зация	int Authoriz(Authorisation *arr_of_acc, int N_acc);	Авторизация пользователя: ввод логина и пароля. Передача значения роли в переменную.
	void SearchRole(Authorisation *arr_of_acc, int N_acc, int role, trip *arr1, int n);	В зависимости от роли запускает меню пользователя или администратора
	void addAdminFirst(Authorisation *arr, int *N_acc);	Добавление уч. записи администратора при создании файла с уч. записями
Работа с файлом	void deleteFile();	Удаление файла
	void createFile();	Создание файла
	void showFile();	Просмотр файла
Работа с данными	void readFileData(trip *arr,int &n);	Считывание данных из файла в массив
	void writeData(trip *arr,int &n);	Запись данных администратором
	void addData(trip *newdata, int &n);	Добавление туриста
	void showData(trip *arr, int n);	Просмотр туристов

	void updateData(Authorisation *arr1,int N_acc,trip *arr, int n);	Редактирование туристов
	void deleteData(Authorisation *arr1,int N_acc,trip *arr, int &n);	Удаление туриста

Окончание таблицы 3.1

Сорти-ровка	void sortTime(trip *arr, int n);	Сортировка по возрастанию времени отправления	
	void sortAlf(trip *arr, int n);	ФИО туриста	Пункту назначения
		Сортировка по алфавиту	
	void sortATT(trip *arr, int n);	Сортировка по ФИО туриста	
Поиск	void PoiskFio(trip *arr, int n);	Поиск по ФИО туриста	
	void PoiskTime1(trip *arr, int n);	Поиск по времени отправления	
	void PoiskEnd(trip *arr, int n);	Поиск по пункту назначения	

На рисунке 3.1 представлена модульная структура программы.

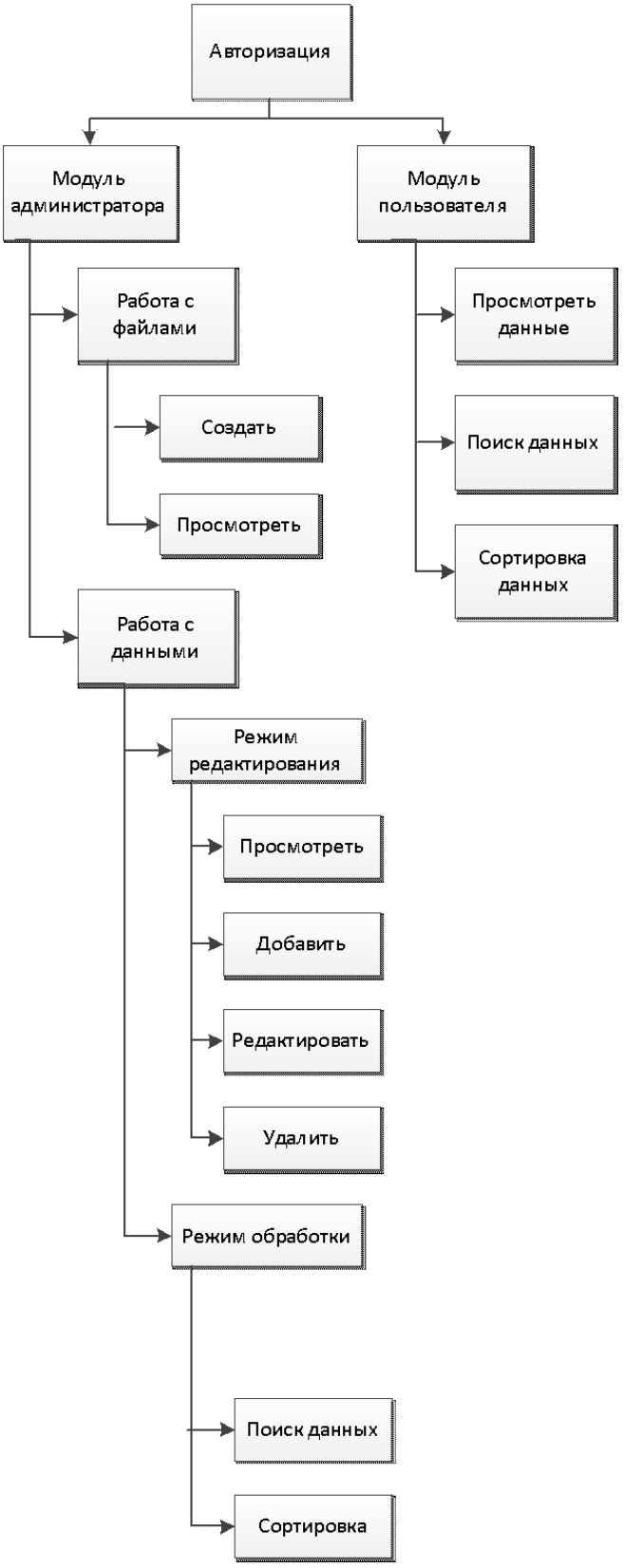

Рисунок 3.1 – Модульная структура программы

4 ФУНКЦИОНАЛЬНАЯ СХЕМА ЗАДАЧИ, СХЕМЫ АЛГОРИТМОВ

Функция организации главного меню представлена на рисунке 4.1.

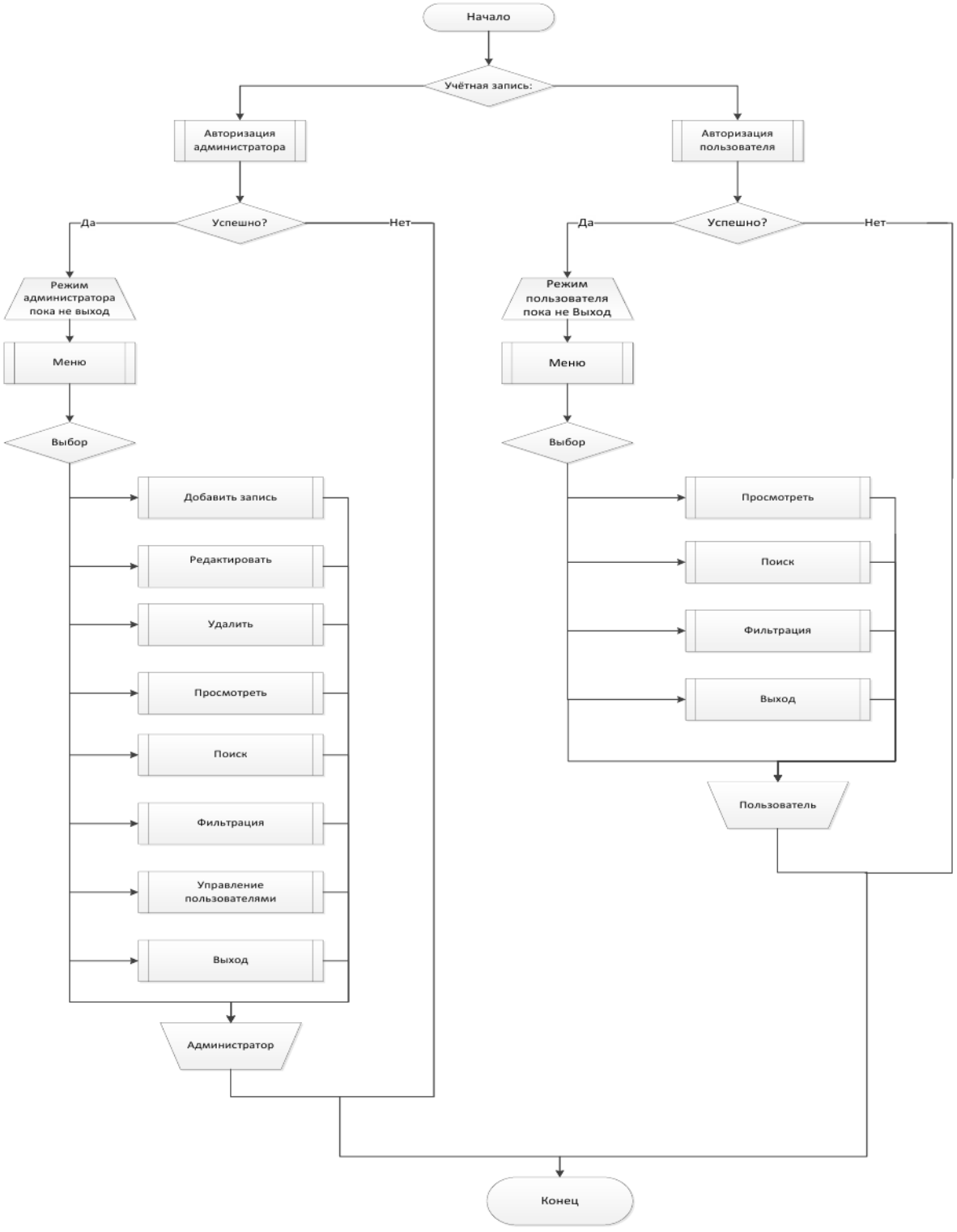

Рисунок 4.1 – Алгоритм функции main

Алгоритм поиска по заданному критерию представлен на рисунке 4.2.

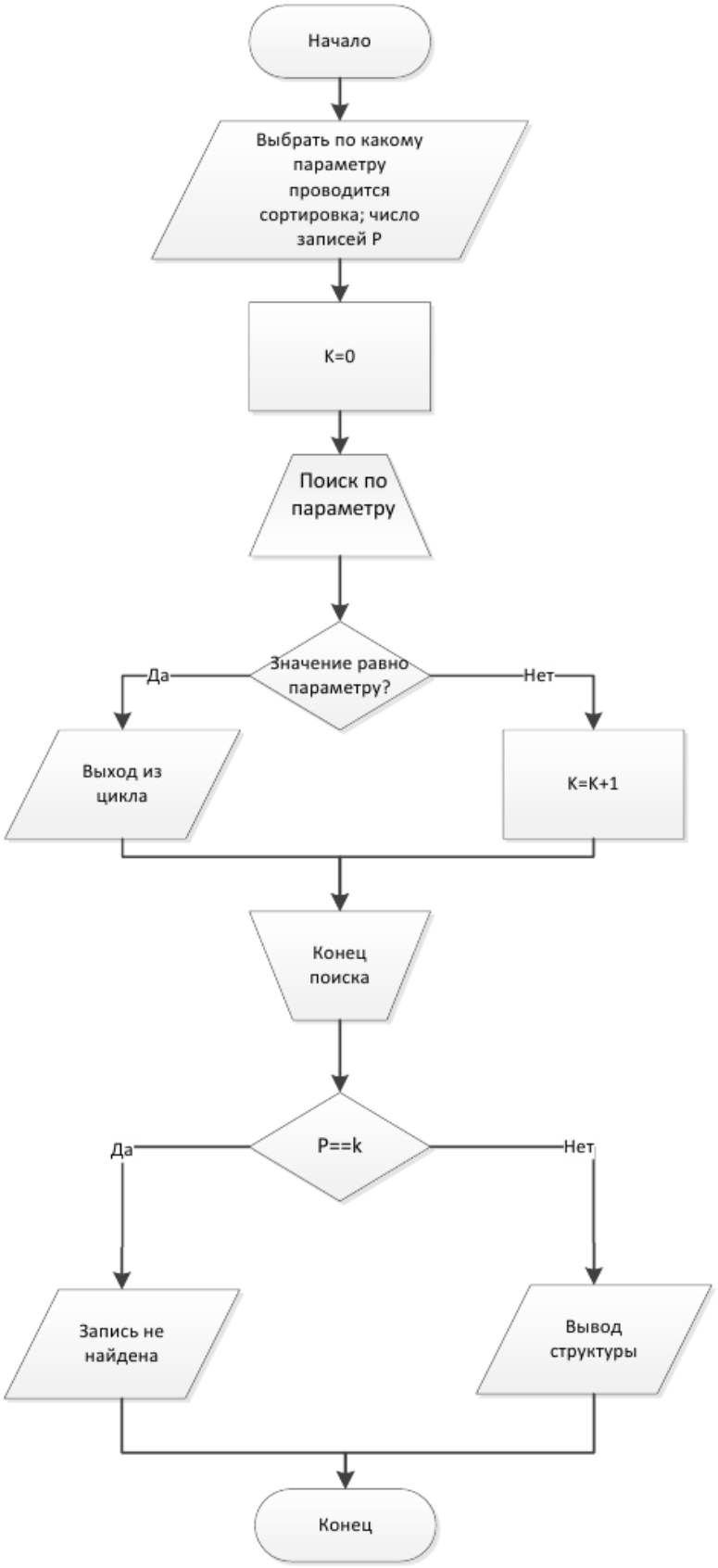

Рисунок 4.2 – Схема алгоритма поиска записи

Функция поиска записей по одному из критериев в данном приложении представлена следующим образом:

```
void Poiskfio(trip *arr, int p) {
int f;
cout << "Поиск по направлению: \n\t1-Турция \n\t2-Египет  \n\t3-Крым";
while(true) {
cin >> f;
if (isNumberNumeric() && f > 0 && f < 4) break;
else cout << "\n uncorrect input!\n";
}
int k=1;
switch (f) {
cout << "\n    №" << setw(15) << "ФИО туриста" << setw(15) << "Пункт назнач." << setw(15) << "Час отпр." << setw(15) << "Час приб." << endl;
case 1: {       cout << "\n    №" << setw(15) << "ФИО туриста" << setw(15) << "Пункт назнач." << setw(15) << "Час отпр." << setw(15) << "Час приб." << endl;
for (int i = 0;i < p;i++) {
if (arr[i].end=="Турция")
cout << k<< "." << arr[i].N << setw(15) <<
arr[i].fio << setw(15) << arr[i].end << setw(15) << arr[i].time1 << setw(15) << arr[i].time2 << endl;
k++;cout << "--------------------------------------------------------------------------\n";
}break;
}
case 2: {
cout << "\n    №" << setw(15) << "ФИО туриста" << setw(15) << "Пункт назнач." << setw(15) << "Час отпр." << setw(15) << "Час приб." << endl;
for (int i = 0;i < n;i++) {
if (arr[i].end=="Египет")
cout << k<< "." << arr[i].N << setw(15) <<
arr[i].fio << setw(15) << arr[i].end << setw(15) << arr[i].time1 << setw(15) << arr[i].time2 << endl;
k++;cout << "--------------------------------------------------------------------------\n";
} break;}
```

```
case 3: {      cout << "\n   №" << setw(15) << "ФИО туриста" << setw(15) << "Пункт назнач." << setw(15) << "Час отпр." << setw(15) << "Час приб." << endl;
for (int i = 0;i < p;i++) {
if (arr[i].end=="Крым")
cout << k << "." << arr[i].N << setw(15) <<
arr[i].fio << setw(15) << arr[i].end << setw(15) << arr[i].time1 << setw(15) << arr[i].time2 << endl;
k++;                                    cout                                    << "--------------------------------------------------------------------------------\n";
}}
}
}
```

Блок-схема функции сортировки элементов по заданному критерию представлена на рисунке 4.3.

Функция сортировки записей по заданному критерию имеет вид:

```
void sortAlf(trip *a, int n) {
cout << "Сортировка по алфавиту \n\t1-ФИО туриста \n\t2-Пункт назначения    -- ";
int choice;
while (true) {
cin >> choice;
if (isNumberNumeric() && choice > 0 && choice < 3) break;
else cout << "\n uncorrect input!\n";
}
switch (choice) {
case 1: {for (int j = 0; j < n - 1;j++) // начинается сама перестановка
{
for (int k = j + 1; k < n;k++)
{
if (a[j].fio > a[k].fio)
{
swap(a[j].fio, a[k].fio);
swap(a[j].N, a[k].N);
swap(a[j].time1, a[k].time1);
swap(a[j].end, a[k].end);
```

```
swap(a[j].time2, a[k].time2);
}
}
}
break;}
case 2: {
for (int j = 0; j < n - 1;j++) // начинается сама перестановка
{
for (int k = j + 1; k < n;k++)
{
if (a[j].end > a[k].end)
{
swap(a[j].fio, a[k].fio);
swap(a[j].N, a[k].N);
swap(a[j].time1, a[k].time1);
swap(a[j].end, a[k].end);
swap(a[j].time2, a[k].time2);
}
}
}break;}
}
showData(a, n);
}
```

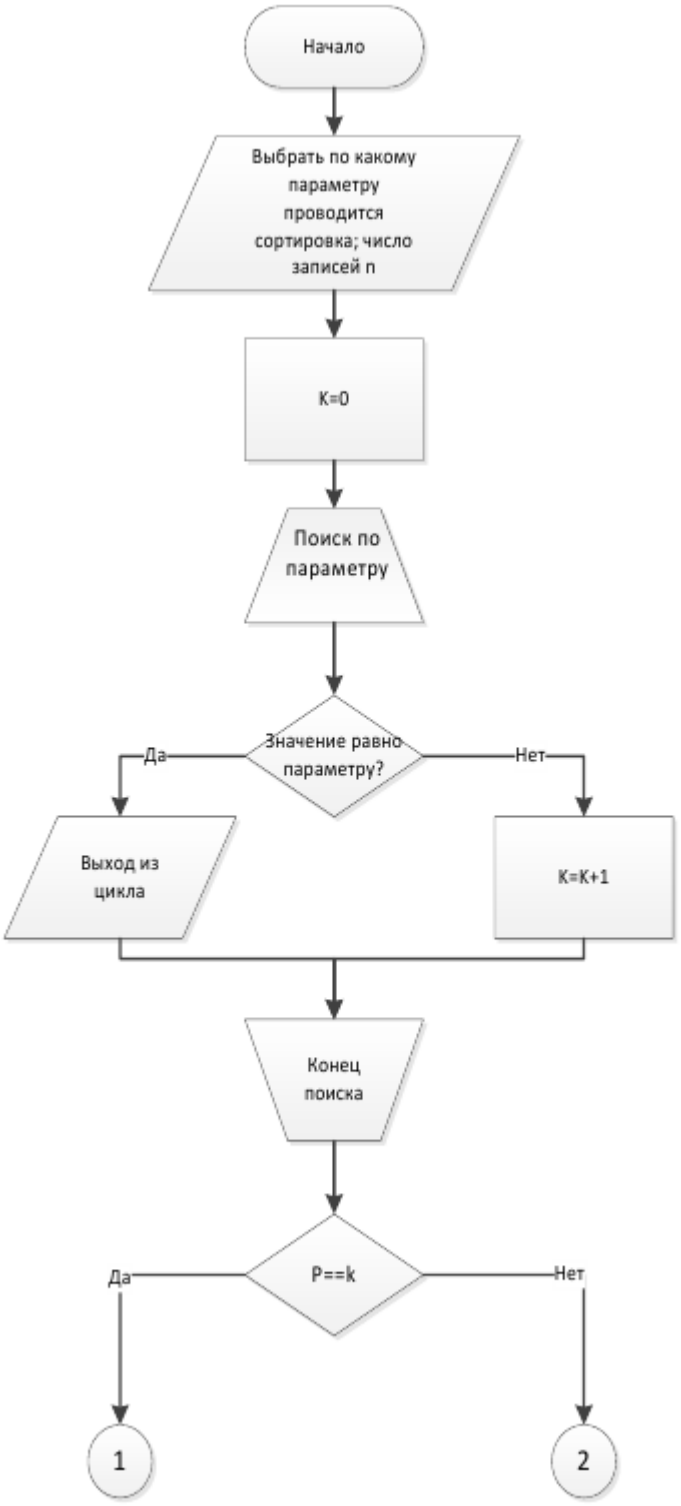

Рисунок 4.3 – Схема алгоритма функции сортировки

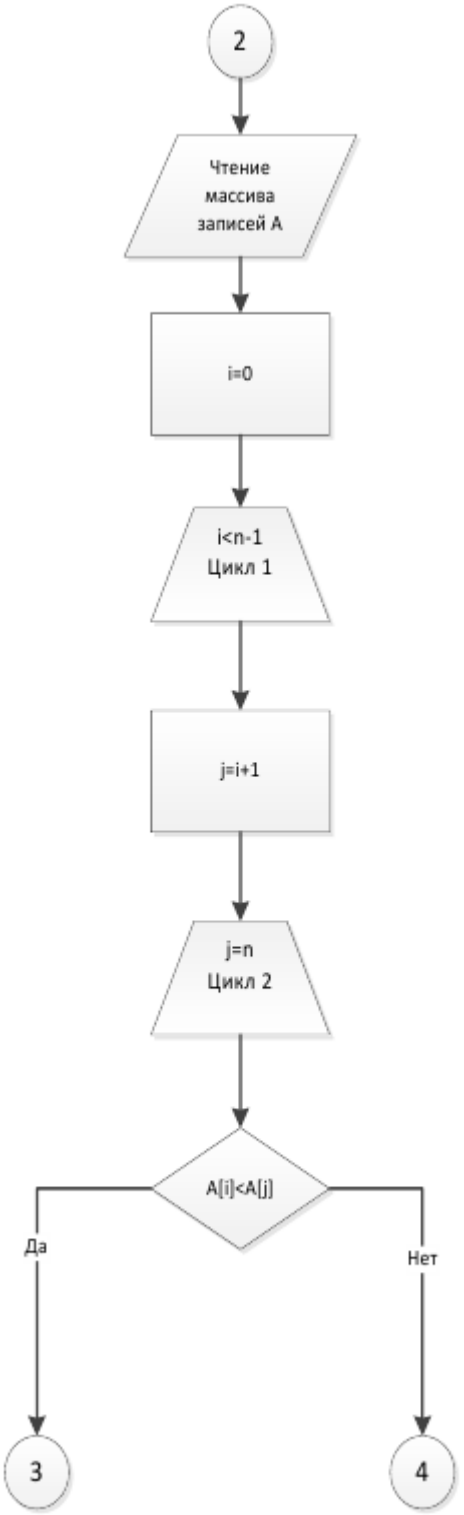

Рисунок 4.4 – Схема алгоритма функции сортировки (продолжение)

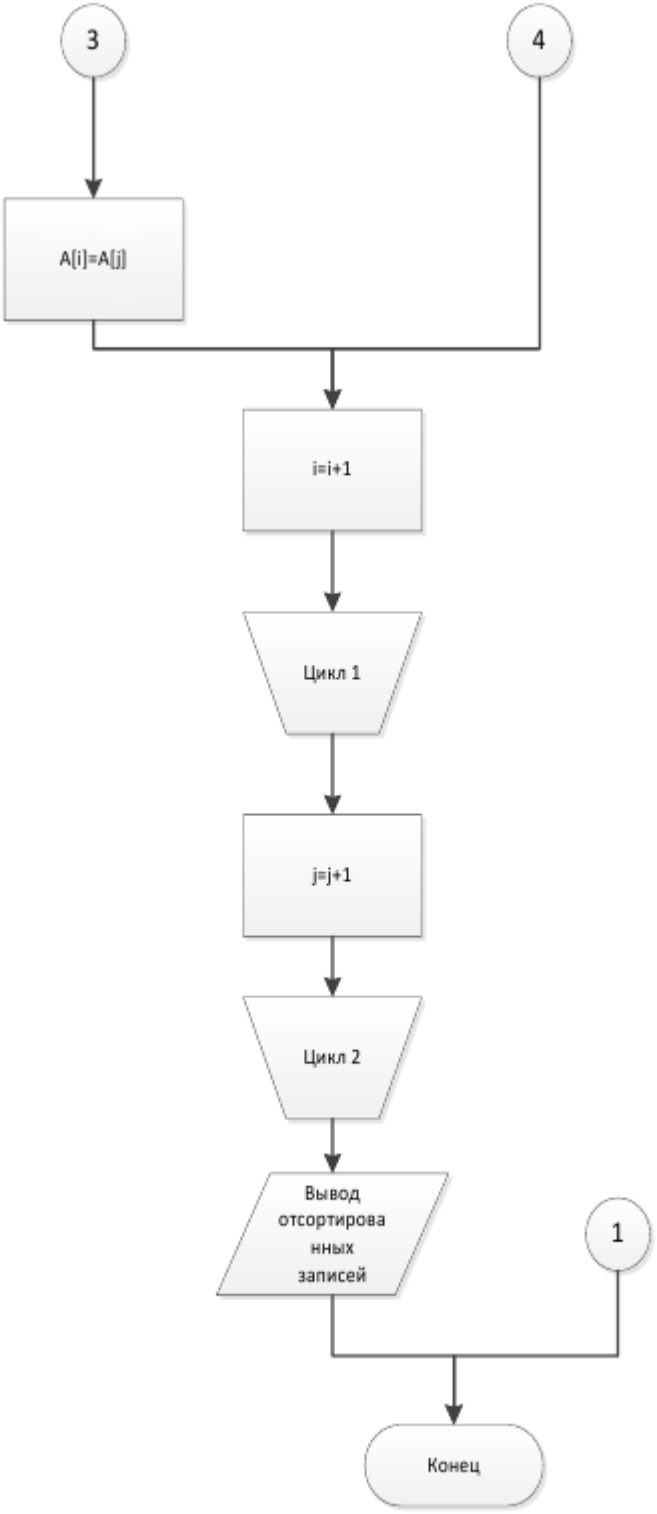

Рисунок 4.5 – Схема алгоритма функции сортировки (окончание)