УО «Белорусский государственный университет информатики и

радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4.1

по предмету

Основы Алгоритмизации и Программирования

Вариант 16

**Выполнил**

**Куфтырев А. В.**

**Проверила**

**Данилова Г.В.**

Группа:

**951007**

Минск 2020

**Задание**

Сведения о месячной заработной плате сотрудников отдела

- фамилия сотрудника;

- наименование отдела;

- размер заработной платы за месяц.

Вывести: общую сумму выплат за каждый месяц по каждому отделу в алфавитном порядке наименований отделов.

**Код программы**

**(С++)**

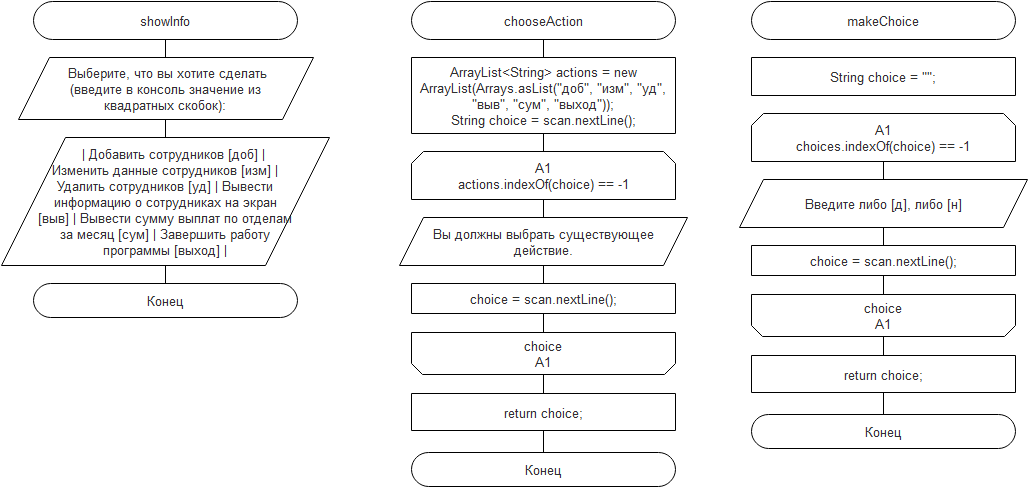
1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4. #include <fstream>
5. #include <cstring>
7. using namespace std;
9. struct Employee {
10. public:
11. char surname[20];
12. char department[20];
13. double salary;
15. Employee() {
16. strcpy(surname, "");
17. strcpy(department, "");
18. salary = 0;
19. }
21. Employee(char surname[20], char department[20], double salary) {
22. strcpy(this->surname, surname);
23. strcpy(this->department, department);
24. this->salary = salary;
25. }
26. };
28. struct DepartmentData {
29. public:
30. string name;
31. double sum = 0;
33. DepartmentData(string name, double sum) {
34. this->name = name;
35. this->sum = sum;
36. }
37. };
39. void executeAction(string action);
41. void showInfo() {
42. cout << "Выберите, что вы хотите сделать (введите в консоль значение из квадратных
43. скобок):" << endl;
44. cout << "| Добавить сотрудников [доб] | Изменить данные сотрудников [изм] | Удалить сотрудников
45. [уд] | Вывести информацию о сотрудниках на экран [выв] | Вывести сумму выплат по отделам за месяц
46. [сум] | Завершить работу программы [выход] |" << endl;
47. }
49. string chooseAction() {
50. vector<string> actions = {"доб", "изм", "уд", "выв", "сум", "выход"};
51. string choice;
52. getline(cin, choice);
53. while(find(actions.begin(), actions.end(), choice) == actions.end()) {
54. cout << "Вы должны выбрать существующее действие." << endl;
55. getline(cin, choice);
56. }
57. return choice;
58. }
60. void showArr(vector<Employee> data) {
61. for(int i = 0; i < data.size(); i++) {
62. Employee emp = data[i];
63. cout << "Сотрудник №" << (i+1) << endl;
64. cout << "Фамилия: " << emp.surname << endl;
65. cout << "Наименование отдела: " << emp.department << endl;
66. cout << "Заработная плата: " << emp.salary << '**\n**' << endl;
67. }
68. }
70. vector<Employee> grabData(bool show, string \*fp = nullptr) {
71. string filePath;
72. if(fp == nullptr) {
73. cout << "Введите имя файла" << endl;
74. getline(cin, filePath);
75. filePath += ".dat";
76. } else
77. filePath = \*fp;
78. vector<Employee> data;
79. ifstream is(filePath);
80. if(is.good()) {
81. Employee temp;
82. while(is.read((char\*)&temp, sizeof(Employee))) {
83. Employee emp(temp.surname, temp.department, temp.salary);
84. data.push\_back(emp);
85. }
86. if(show) {
87. if(data.size() > 0) {
88. cout << "Данные, записанные в файле:" << endl;
89. showArr(data);
90. } else
91. cout << "В файле не найдено ни одной записи" << endl;
92. }
93. } else
94. cout << "Произошла ошибка при чтении файла. Убедитесь, что такой файл существует." << endl;
95. is.close();
96. return data;
97. }
99. void createNewFile(vector<Employee> data, string filePath) {
100. ofstream os(filePath, ifstream::binary);
101. if(os.is\_open()) {
102. for(Employee emp : data)
103. os.write((char\*)&emp, sizeof(Employee));
104. cout << "Данные успешно сохранены в файл " << filePath << endl;
105. } else {
106. cout << "Не удалось сохранить данные в файл." << endl;
107. }
108. os.close();
109. }
111. void updateFile(vector<Employee> newData, string filePath) {
112. vector<Employee> data = grabData(false, &filePath);
113. data.insert(data.end(), newData.begin(), newData.end());
114. ofstream os(filePath, ifstream::binary);
115. if(os.is\_open()) {
116. for(Employee emp : data)
117. os.write((char\*)&emp, sizeof(Employee));
118. cout << "Данные успешно обновлены в файле " << filePath << endl;
119. } else {
120. cout << "Не удалось сохранить данные в файл." << endl;
121. }
122. os.close();
123. }
125. void saveDataToFile(vector<Employee> data, bool update) {
126. cout << "Введите [д], если хотите сохранить данные в файл." << endl;
127. string choice;
128. getline(cin, choice);
129. if(choice == "д") {
130. cout << "Введите имя файла" << endl;
131. string filePath;
132. getline(cin, filePath);
133. filePath += ".dat";
134. ifstream file (filePath);
135. if(update && file.good()) {
136. cout << "Файл с таким именем уже существует. Хотите его обновить? [д/н] (При выборе [н]
137. создастся новый файл)" << endl;
138. getline(cin, choice);
139. if(choice == "д")
140. updateFile(data, filePath);
141. else
142. createNewFile(data, filePath);
143. } else
144. createNewFile(data, filePath);
145. }
146. }
148. void addData() {
149. vector<Employee> data;
150. bool continueCycle = true;
151. bool notCorrect = true;
152. cout << "Если хотите прекратить ввод, введите [стоп] вместо фамилии сотрудника" << endl;
153. while(continueCycle) {
154. char surname[20];
155. char department[20];
156. double salary = 0;
157. while(notCorrect) {
158. cout << "Введите фамилию сотрудника" << endl;
159. cin.getline(surname, 20);
160. if(surname[0] != '**\0**') {
161. if(strcmp(surname, "стоп") == 0) {
162. continueCycle = false;
163. notCorrect = false;
164. } else {
165. cout << "Введите название отдела" << endl;
166. cin.getline(department, 20);
167. if(department[0] != '**\0**') {
168. cout << "Введите заработную плату сотрудника" << endl;
169. string str;
170. getline(cin, str);
171. try {
172. salary = stod(str);
173. if(salary > -1)
174. notCorrect = false;
175. else {
176. notCorrect = true;
177. cout << "Заработная плата не может быть отрицательной, повторите
178. ввод" << endl;
179. }
180. } catch(...) {
181. notCorrect = true;
182. cout << "Возникла ошибка при вводе данных, повторите ввод." << endl;
183. }
184. } else
185. cout << "Название отдела не может быть пустым, повторите ввод." << endl;
186. }
187. } else
188. cout << "Фамилия не может быть пустой, повторите ввод." << endl;
189. }
190. if(strcmp(surname, "стоп") != 0) {
191. notCorrect = true;
192. data.push\_back(\*new Employee(surname, department, salary));
193. }
194. }
195. saveDataToFile(data, true);
196. showInfo();
197. executeAction(chooseAction());
198. }
200. string makeChoice(vector<string> choices) {
201. string choice = "";
202. while(find(choices.begin(), choices.end(), choice) == choices.end()) {
203. cout << "Введите либо [д], либо [н]" << endl;
204. getline(cin, choice);
205. }
206. return choice;
207. }
209. int changeEmployeeData(Employee \*emp, int changes) {
210. cin.clear();
211. string choice = "";
212. bool notCorrect = true;
213. vector<string> choices = {"д", "н"};
214. while(notCorrect) {
215. cout << "Хотите изменить фамилию?" << endl;
216. choice = makeChoice(choices);
217. if (choice == "д") {
218. cout << "Введите новую фамилию" << endl;
219. char surname[20];
220. cin.getline(surname, 20);
221. if(surname[0] != '**\0**') {
222. strcpy(emp->surname, surname);
223. cout << "Фамилия успешно изменена, хотите продолжить изменение данных?" << endl;
224. notCorrect = false;
225. changes++;
226. choice = makeChoice(choices);
227. if (choice == "д")
228. changeEmployeeData(emp, changes);
229. } else
230. cout << "Фамилия не может быть пустой, повторите ввод." << endl;
231. } else {
232. cout << "Хотите изменить наименование отдела?" << endl;
233. choice = makeChoice(choices);
234. if (choice == "д") {
235. cout << "Введите наименование отдела" << endl;
236. char department[20];
237. cin.getline(department, 20);
238. if(department[0] != '**\0**') {
239. strcpy(emp->department, department);
240. cout << "Наименование отдела успешно изменено, хотите продолжить изменение
241. данных?" << endl;
242. notCorrect = false;
243. changes++;
244. choice = makeChoice(choices);
245. if (choice == "д")
246. changeEmployeeData(emp, changes);
247. } else
248. cout << "Название отдела не может быть пустым, повторите ввод." << endl;
249. } else {
250. cout << "Хотите изменить размер заработной платы?" << endl;
251. choice = makeChoice(choices);
252. if (choice == "д") {
253. cout << "Введите размер заработной платы" << endl;
254. double salary;
255. string str;
256. try {
257. getline(cin, str);
258. salary = stod(str);
259. if (salary > -1) {
260. notCorrect = false;
261. emp->salary = salary;
262. cout << "Заработная плата успешно изменена" << endl;
263. changes++;
264. } else {
265. notCorrect = true;
266. cout << "Заработная плата не может быть отрицательной, повторите
267. ввод" << endl;
268. }
269. } catch (...) {
270. cout << "Заработная плата должна быть числом" << endl;
271. }
272. }
273. }
274. }
275. }
276. return changes;
277. }
279. void changeData() {
280. vector<Employee> data = grabData(true);
281. if(data.size() > 0) {
282. bool continueCycle = true;
283. bool notCorrect = true;
284. int changedCount = 0;
285. while(continueCycle) {
286. cout << "Введите номер сотрудника, данные которого хотите изменить, либо [стоп], чтобы
287. вернуться в меню" << endl;
288. while(notCorrect) {
289. string input;
290. getline(cin, input);
291. string choice = "";
292. if(input == "стоп") {
293. notCorrect = false;
294. continueCycle = false;
295. } else {
296. try {
297. int changeNum = stoi(input);
298. if(changeNum <= data.size() && changeNum > 0) {
299. Employee \*emp = &data[changeNum-1];
300. changedCount = changeEmployeeData(emp, 0);
301. notCorrect = false;
302. } else
303. cout << "Сотрудник с таким номером не найден, повторите ввод." << endl;
304. } catch (...) {
305. cout << "Номер сотрудника должен быть числом, повторите ввод." << endl;
306. }
307. }
308. }
309. notCorrect = true;
310. }
311. if(changedCount > 0)
312. saveDataToFile(data, false);
313. }
314. showInfo();
315. executeAction(chooseAction());
316. }
318. void deleteData() {
319. vector<Employee> data = grabData(true);
320. if(data.size() > 0) {
321. bool continueCycle = true;
322. bool notCorrect = true;
323. int deletedCount = 0;
324. while(continueCycle) {
325. cout << "Введите номер сотрудника, которого хотите удалить, либо [стоп], чтобы вернуться
326. в меню" << endl;
327. while(notCorrect) {
328. string input;
329. getline(cin, input);
330. if(input == "стоп") {
331. notCorrect = false;
332. continueCycle = false;
333. } else {
334. try {
335. int deleteNum = stoi(input);
336. if(deleteNum > 0 && deleteNum <= data.size()) {
337. data.erase(data.begin()+deleteNum-1);
338. deletedCount++;
339. cout << "Сотрудник № " << deleteNum << " успешно удален." << endl;
340. notCorrect = false;
341. } else {
342. cout << "Сотрудник с таким номером не найден, повторите ввод." << endl;
343. notCorrect = true;
344. }
345. } catch (...) {
346. cout << "Номер сотрудника должен быть числом." << endl;
347. notCorrect = true;
348. }
349. }
350. }
351. notCorrect = true;
352. }
353. if(deletedCount > 0)
354. saveDataToFile(data, false);
355. }
356. showInfo();
357. executeAction(chooseAction());
358. }
360. void showData() {
361. grabData(true);
362. showInfo();
363. executeAction(chooseAction());
364. }
366. DepartmentData\* grabCurrentDepartment(string departmentName, vector<DepartmentData> &departments) {
367. bool isInArr = false;
368. int i = 0;
369. DepartmentData \*department = nullptr;
370. while(!isInArr && i < departments.size()) {
371. DepartmentData \*temp = &departments[i];
372. if(temp->name == departmentName) {
373. isInArr = true;
374. department = temp;
375. }
376. i++;
377. }
378. return department;
379. }
381. vector<DepartmentData> grabDepartments(vector<Employee> data) {
382. vector<DepartmentData> departments;
383. for(Employee emp : data) {
384. DepartmentData \*department = grabCurrentDepartment(emp.department, departments);
385. if(department == nullptr)
386. departments.push\_back(\*new DepartmentData(emp.department, emp.salary));
387. else
388. department->sum += emp.salary;
389. }
390. return departments;
391. }
393. void sortDepartments(vector<DepartmentData> &departments) {
394. for(int i = 0; i < departments.size(); i++) {
395. for(int j = 1; j < departments.size()-i; j++) {
396. DepartmentData \*prev = &departments[j-1];
397. DepartmentData \*curr = &departments[j];
398. char currName[curr->name.size()];
399. char prevName[prev->name.size()];
400. strcpy(currName, curr->name.c\_str());
401. strcpy(prevName, prev->name.c\_str());
402. if(strcmp(currName, prevName) < 0)
403. swap(departments[j], departments[j-1]);
404. }
405. }
406. }
408. void showDepartmentSum() {
409. vector<Employee> data = grabData(false);
410. vector<DepartmentData> departments = grabDepartments(data);
411. sortDepartments(departments);
412. for(DepartmentData department : departments) {
413. cout << department.name << endl;
414. cout << "Общая сумма выплат за месяц по отделу = " << department.sum << '**\n**' << endl;
415. }
416. showInfo();
417. executeAction(chooseAction());
418. }
420. void executeAction(string action) {
421. if(action == "доб")
422. addData();
423. else if(action == "изм")
424. changeData();
425. else if(action == "уд")
426. deleteData();
427. else if(action == "выв")
428. showData();
429. else if(action == "сум")
430. showDepartmentSum();
431. else if(action == "выход")
432. exit(0);
433. }
435. int main() {
436. setlocale(LC\_ALL,".1251");
437. cout << "Данная программа позволяет работать со сведениями о месячной заработной плате
438. сотрудников отдела.**\n**" << endl;
439. showInfo();
440. string action = chooseAction();
441. executeAction(action);
442. return 0;
443. }

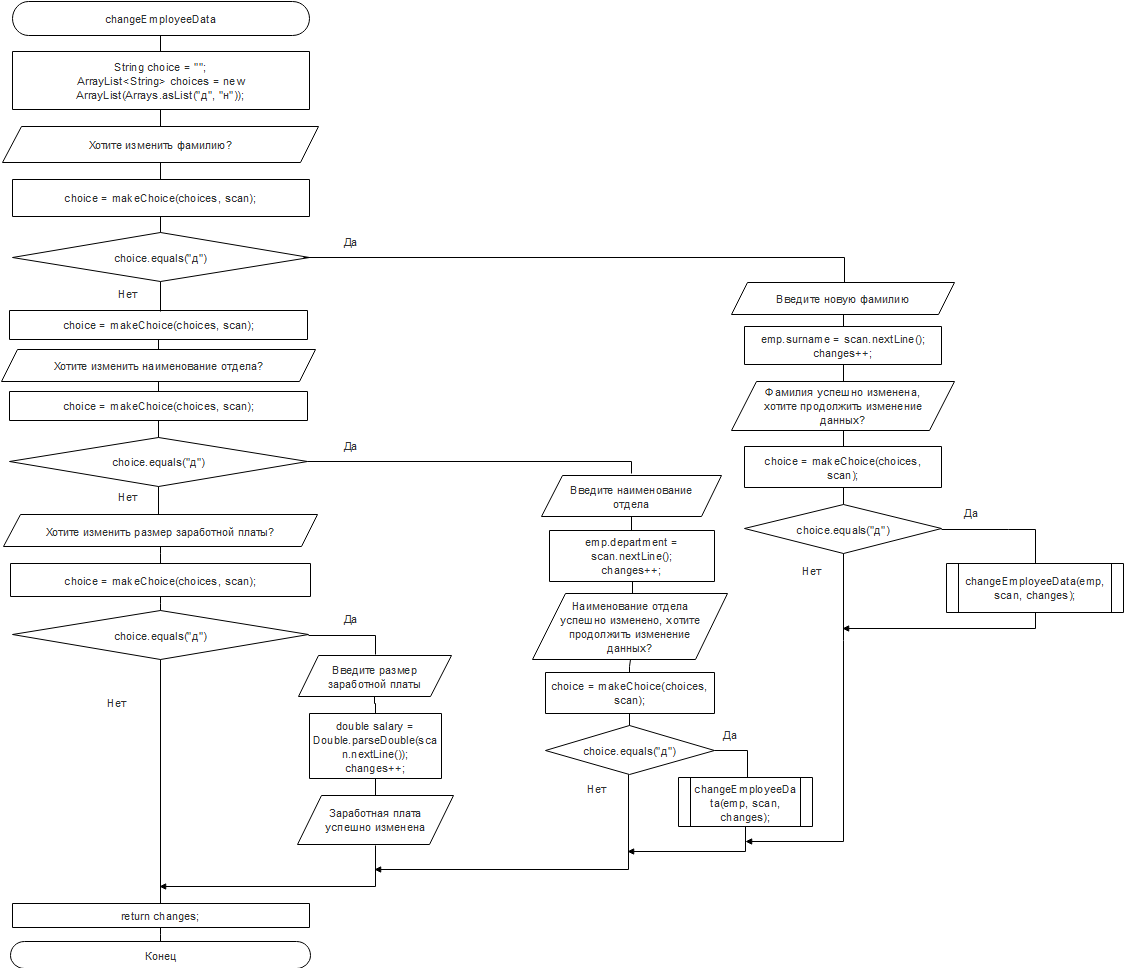
**Код программы**

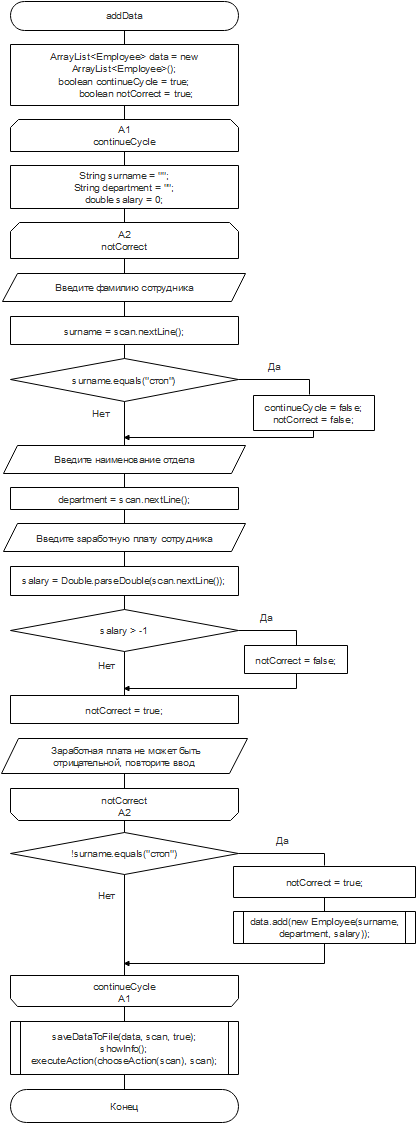
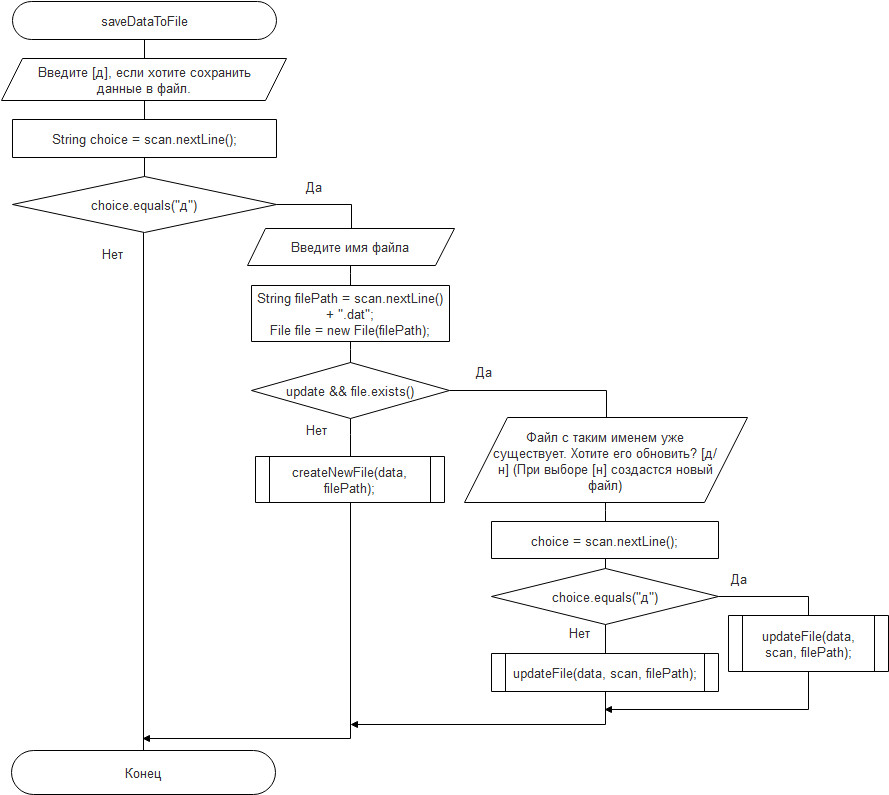
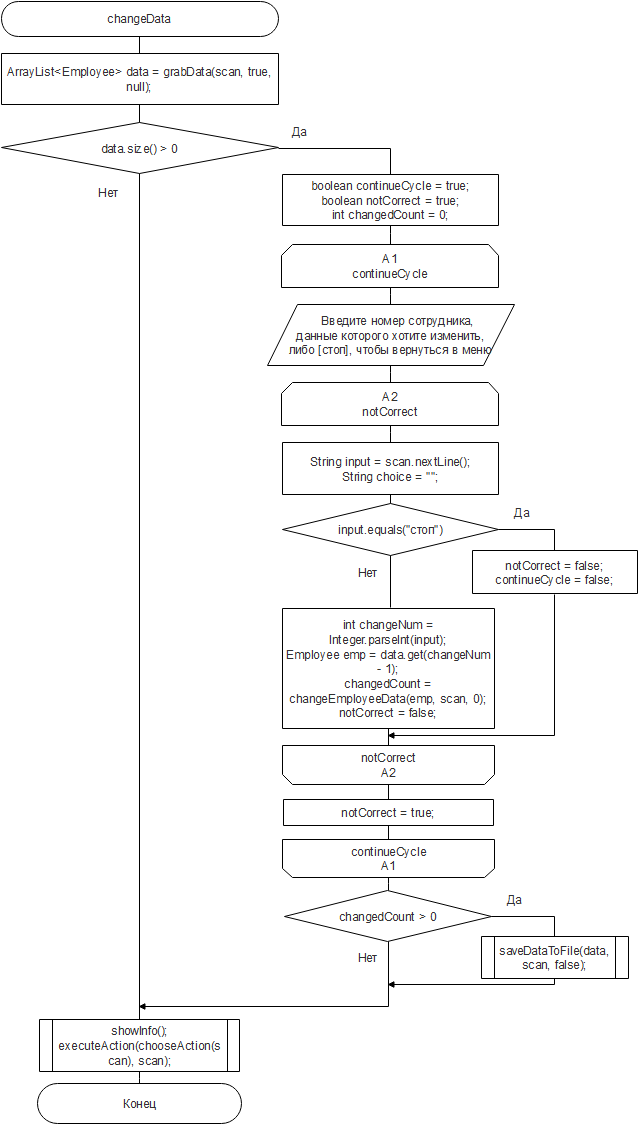
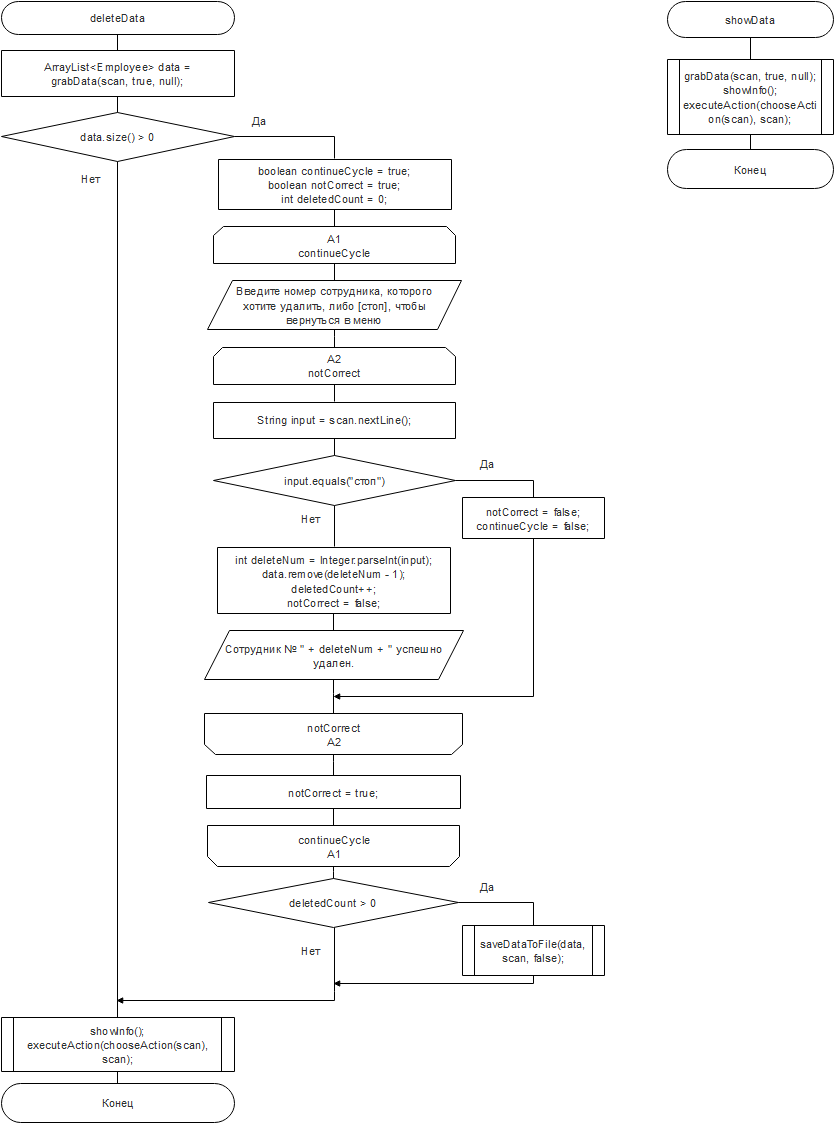
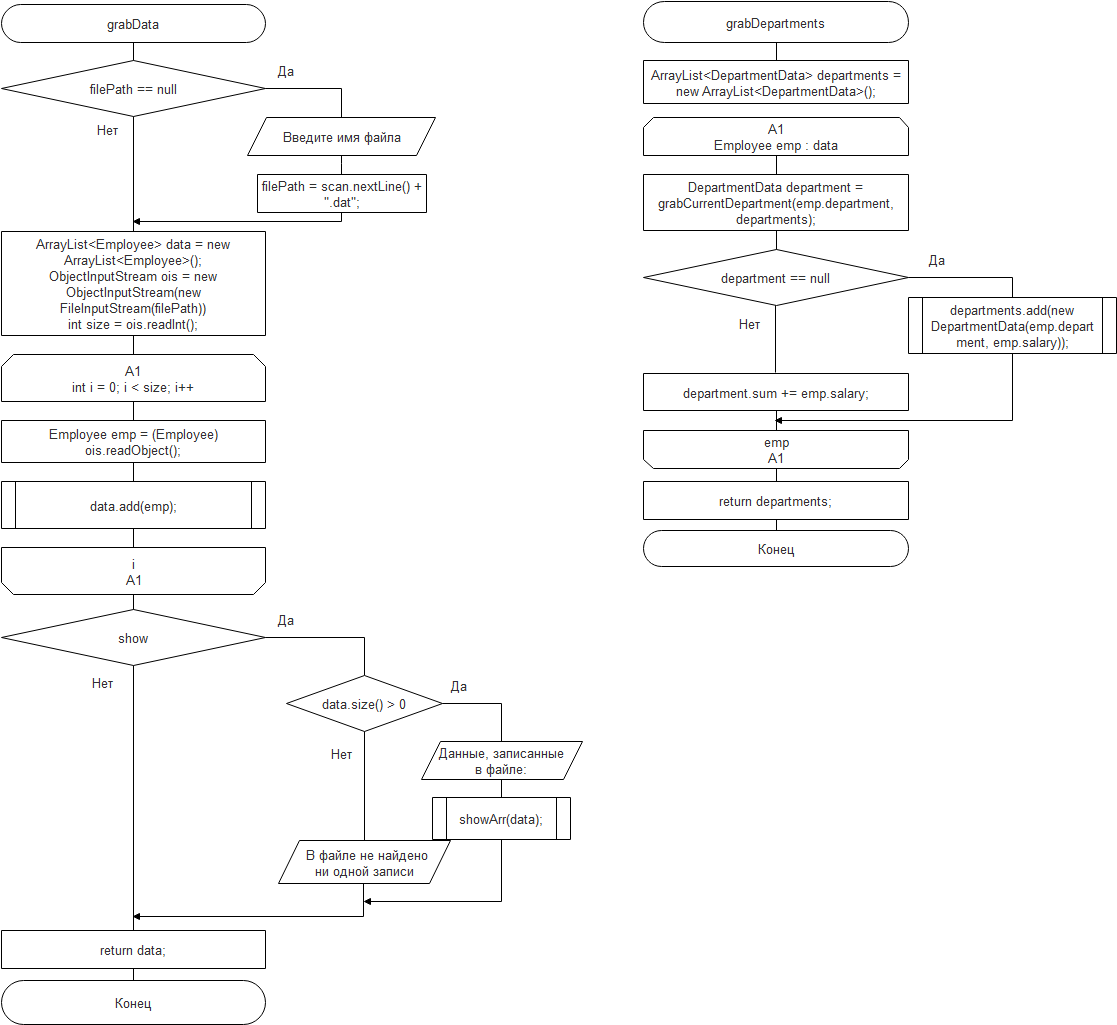
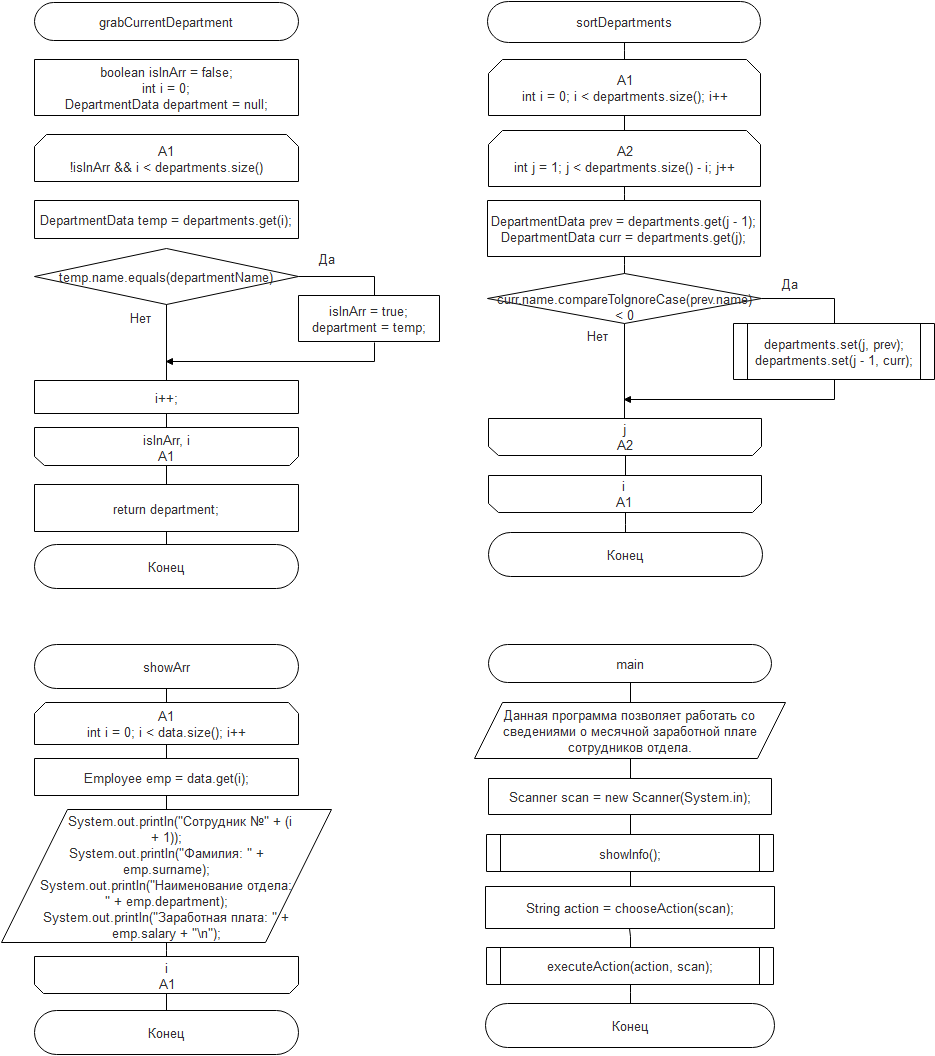
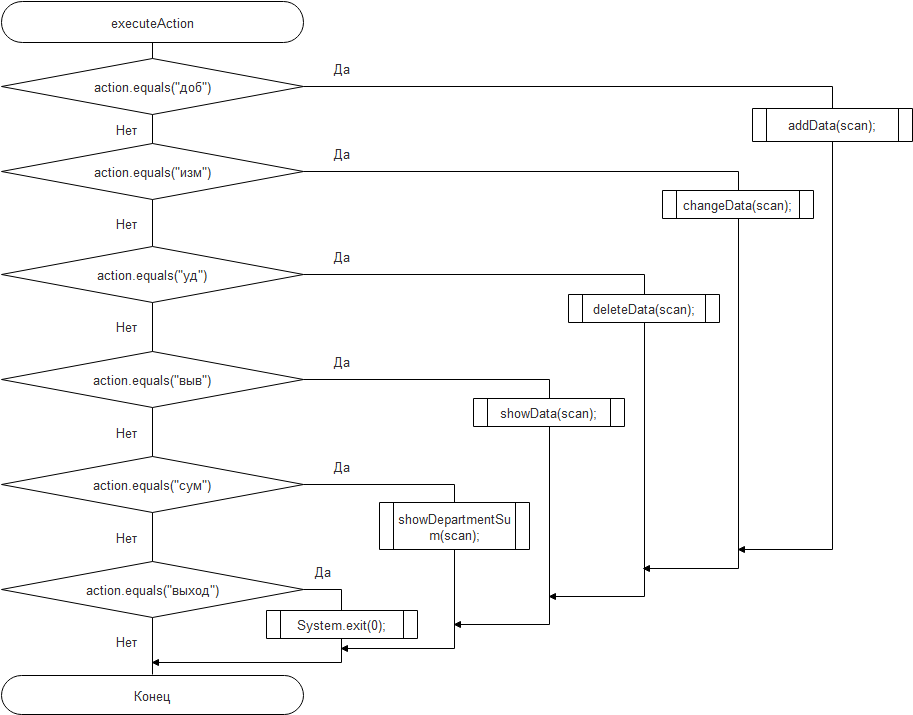
**(Java)**

1. **package** com.company;
3. **import** java.io.\*;
4. **import** java.util.\*;
6. **class** Employee **implements** Serializable {
7. String surname;
8. String department;
9. **double** salary;
11. Employee(String surname, String department, **double** salary) {
12. **this**.surname = surname;
13. **this**.department = department;
14. **this**.salary = salary;
15. }
16. }
18. **class** DepartmentData {
19. String name;
20. **double** sum = 0;
22. DepartmentData(String name, **double** sum) {
23. **this**.name = name;
24. **this**.sum = sum;
25. }
26. }
28. **public** **class** Main {
29. **static** **void** showInfo() {
30. System.out.println("Выберите, что вы хотите сделать (введите в консоль значение из квадратных
31. скобок):");
32. System.out.println("| Добавить сотрудников [доб] | Изменить данные сотрудников [изм] |
33. Удалить сотрудников [уд] | Вывести информацию о сотрудниках на экран [выв] | Вывести сумму выплат по
34. отделам за месяц [сум] | Завершить работу программы [выход] |");
35. }
37. **static** String chooseAction(Scanner scan) {
38. ArrayList<String> actions = **new** ArrayList(Arrays.asList("доб", "изм", "уд", "выв", "сум", "вы
39. ход"));
40. String choice = scan.nextLine();
41. **while** (actions.indexOf(choice) == -1) {
42. System.out.println("Вы должны выбрать существующее действие.");
43. choice = scan.nextLine();
44. }
45. **return** choice;
46. }
48. **static** String makeChoice(ArrayList<String> choices, Scanner scan) {
49. String choice = "";
50. **while** (choices.indexOf(choice) == -1) {
51. System.out.println("Введите либо [д], либо [н]");
52. choice = scan.nextLine();
53. }
54. **return** choice;
55. }
57. **static** **int** changeEmployeeData(Employee emp, Scanner scan, **int** changes) {
58. String choice = "";
59. ArrayList<String> choices = **new** ArrayList(Arrays.asList("д", "н"));
60. **boolean** notCorrect = **true**;
61. **while** (notCorrect) {
62. System.out.println("Хотите изменить фамилию?");
63. choice = makeChoice(choices, scan);
64. **if** (choice.equals("д")) {
65. System.out.println("Введите новую фамилию");
66. String surname = scan.nextLine();
67. **if** (surname.length() > 0) {
68. emp.surname = surname;
69. System.out.println("Фамилия успешно изменена, хотите продолжить изменение
70. данных?");
71. notCorrect = **false**;
72. changes++;
73. choice = makeChoice(choices, scan);
74. **if** (choice.equals("д"))
75. changeEmployeeData(emp, scan, changes);
76. } **else**
77. System.out.println("Фамилия не может быть пустой, повторите ввод.");
78. } **else** {
79. System.out.println("Хотите изменить наименование отдела?");
80. choice = makeChoice(choices, scan);
81. **if** (choice.equals("д")) {
82. System.out.println("Введите наименование отдела");
83. String department = scan.nextLine();
84. **if** (department.length() > 0) {
85. emp.department = department;
86. System.out.println("Наименование отдела успешно изменено, хотите продолжить
87. изменение данных?");
88. notCorrect = **false**;
89. changes++;
90. choice = makeChoice(choices, scan);
91. **if** (choice.equals("д"))
92. changeEmployeeData(emp, scan, changes);
93. } **else**
94. System.out.println("Название отдела не может быть пустым, повторите ввод.");
95. } **else** {
96. System.out.println("Хотите изменить размер заработной платы?");
97. choice = makeChoice(choices, scan);
98. **if** (choice.equals("д")) {
99. System.out.println("Введите размер заработной платы");
100. **try** {
101. **double** salary = Double.parseDouble(scan.nextLine());
102. **if** (salary > -1) {
103. notCorrect = **false**;
104. emp.salary = salary;
105. System.out.println("Заработная плата успешно изменена");
106. changes++;
107. } **else** {
108. notCorrect = **true**;
109. System.out.println("Заработная плата не может быть отрицательной,
110. повторите ввод");
111. }
112. } **catch** (Exception err) {
113. System.out.println("Заработная плата должна быть числом");
114. }
115. }
116. }
117. }
118. }
119. **return** changes;
120. }
122. **static** **void** addData(Scanner scan) {
123. ArrayList<Employee> data = **new** ArrayList<Employee>();
124. **boolean** continueCycle = **true**;
125. **boolean** notCorrect = **true**;
126. System.out.println("Если хотите прекратить ввод, введите [стоп] вместо фамилии сотрудника");
127. **while** (continueCycle) {
128. String surname = "";
129. String department = "";
130. **double** salary = 0;
131. **while** (notCorrect) {
132. System.out.println("Введите фамилию сотрудника");
133. surname = scan.nextLine();
134. **if**(surname.length() > 0) {
135. **if** (surname.equals("стоп")) {
136. continueCycle = **false**;
137. notCorrect = **false**;
138. } **else** {
139. System.out.println("Введите наименование отдела");
140. department = scan.nextLine();
141. **if**(department.length() > 0) {
142. System.out.println("Введите заработную плату сотрудника");
143. **try** {
144. salary = Double.parseDouble(scan.nextLine());
145. **if** (salary > -1)
146. notCorrect = **false**;
147. **else** {
148. notCorrect = **true**;
149. System.out.println("Заработная плата не может быть отрицательной,
150. повторите ввод");
151. }
152. } **catch** (Exception err) {
153. notCorrect = **true**;
154. System.out.println("Возникла ошибка при вводе данных, повторите
155. ввод.");
156. }
157. } **else**
158. System.out.println("Название отдела не может быть пустым, повторите
159. ввод.");
160. }
161. } **else**
162. System.out.println("Фамилия не может быть пустой, повторите ввод.");
163. }
164. **if** (!surname.equals("стоп")) {
165. notCorrect = **true**;
166. data.add(**new** Employee(surname, department, salary));
167. }
168. }
169. saveDataToFile(data, scan, **true**);
170. showInfo();
171. executeAction(chooseAction(scan), scan);
172. }
174. **static** **void** createNewFile(ArrayList<Employee> data, String filePath) {
175. **try** (ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream(filePath))) {
176. oos.writeInt(data.size());
177. **for** (Employee emp : data)
178. oos.writeObject(emp);
179. System.out.println("Данные успешно сохранены в файл " + filePath);
180. } **catch** (Exception err) {
181. System.out.println("Не удалось сохранить данные в файл.");
182. }
183. }
185. **static** **void** updateFile(ArrayList<Employee> newData, Scanner scan, String filePath) {
186. ArrayList<Employee> data = grabData(scan, **false**, filePath);
187. data.addAll(newData);
188. **try** (ObjectOutputStream oos = **new** ObjectOutputStream(**new** FileOutputStream(filePath))) {
189. oos.writeInt(data.size());
190. **for** (Employee emp : data)
191. oos.writeObject(emp);
192. System.out.println("Данные успешно обновлены в файле " + filePath);
193. } **catch** (Exception err) {
194. System.out.println("Не удалось обновить данные в файле.");
195. }
196. }
198. **static** **void** saveDataToFile(ArrayList<Employee> data, Scanner scan, **boolean** update) {
199. System.out.println("Введите [д], если хотите сохранить данные в файл.");
200. String choice = scan.nextLine();
201. **if** (choice.equals("д")) {
202. System.out.println("Введите имя файла");
203. String filePath = scan.nextLine() + ".dat";
204. File file = **new** File(filePath);
205. **if** (update && file.exists()) {
206. System.out.println("Файл с таким именем уже существует. Хотите его обновить? [д/н]
207. (При выборе [н] создастся новый файл)");
208. choice = scan.nextLine();
209. **if** (choice.equals("д"))
210. updateFile(data, scan, filePath);
211. **else**
212. createNewFile(data, filePath);
213. } **else**
214. createNewFile(data, filePath);
215. }
216. }
218. **static** **void** changeData(Scanner scan) {
219. ArrayList<Employee> data = grabData(scan, **true**, **null**);
220. **if** (data.size() > 0) {
221. **boolean** continueCycle = **true**;
222. **boolean** notCorrect = **true**;
223. **int** changedCount = 0;
224. **while** (continueCycle) {
225. System.out.println("Введите номер сотрудника, данные которого хотите изменить, либо
226. [стоп], чтобы вернуться в меню");
227. **while** (notCorrect) {
228. String input = scan.nextLine();
229. String choice = "";
230. **if** (input.equals("стоп")) {
231. notCorrect = **false**;
232. continueCycle = **false**;
233. } **else** {
234. **try** {
235. **int** changeNum = Integer.parseInt(input);
236. Employee emp = data.get(changeNum - 1);
237. changedCount = changeEmployeeData(emp, scan, 0);
238. notCorrect = **false**;
239. } **catch** (Exception err) {
240. System.out.println("Сотрудник с таким номером не найден, повторите
241. ввод.");
242. }
243. }
244. }
245. notCorrect = **true**;
246. }
247. **if** (changedCount > 0)
248. saveDataToFile(data, scan, **false**);
249. }
250. showInfo();
251. executeAction(chooseAction(scan), scan);
252. }
254. **static** **void** deleteData(Scanner scan) {
255. ArrayList<Employee> data = grabData(scan, **true**, **null**);
256. **if** (data.size() > 0) {
257. **boolean** continueCycle = **true**;
258. **boolean** notCorrect = **true**;
259. **int** deletedCount = 0;
260. **while** (continueCycle) {
261. System.out.println("Введите номер сотрудника, которого хотите удалить, либо [стоп],
262. чтобы вернуться в меню");
263. **while** (notCorrect) {
264. String input = scan.nextLine();
265. **if** (input.equals("стоп")) {
266. notCorrect = **false**;
267. continueCycle = **false**;
268. } **else** {
269. **try** {
270. **int** deleteNum = Integer.parseInt(input);
271. data.remove(deleteNum - 1);
272. deletedCount++;
273. System.out.println("Сотрудник № " + deleteNum + " успешно удален.");
274. notCorrect = **false**;
275. } **catch** (Exception err) {
276. System.out.println("Сотрудник с таким номером не найден, повторите
277. ввод.");
278. notCorrect = **true**;
279. }
280. }
281. }
282. notCorrect = **true**;
283. }
284. **if** (deletedCount > 0)
285. saveDataToFile(data, scan, **false**);
286. }
287. showInfo();
288. executeAction(chooseAction(scan), scan);
289. }
291. **static** **void** showData(Scanner scan) {
292. grabData(scan, **true**, **null**);
293. showInfo();
294. executeAction(chooseAction(scan), scan);
295. }
297. **static** ArrayList<Employee> grabData(Scanner scan, **boolean** show, String filePath) {
298. **if** (filePath == **null**) {
299. System.out.println("Введите имя файла");
300. filePath = scan.nextLine() + ".dat";
301. }
302. ArrayList<Employee> data = **new** ArrayList<Employee>();
303. **try** (ObjectInputStream ois = **new** ObjectInputStream(**new** FileInputStream(filePath))) {
304. **int** size = ois.readInt();
305. **for** (**int** i = 0; i < size; i++) {
306. Employee emp = (Employee) ois.readObject();
307. data.add(emp);
308. }
309. **if** (show) {
310. **if**(data.size() > 0) {
311. System.out.println("Данные, записанные в файле:");
312. showArr(data);
313. } **else**
314. System.out.println("В файле не найдено ни одной записи");
315. }
316. } **catch** (Exception err) {
317. System.out.println("Произошла ошибка при чтении файла. Убедитесь, что такой файл
318. существует.");
319. }
320. **return** data;
321. }
323. **static** ArrayList<DepartmentData> grabDepartments(ArrayList<Employee> data) {
324. ArrayList<DepartmentData> departments = **new** ArrayList<DepartmentData>();
325. **for** (Employee emp : data) {
326. DepartmentData department = grabCurrentDepartment(emp.department, departments);
327. **if** (department == **null**)
328. departments.add(**new** DepartmentData(emp.department, emp.salary));
329. **else**
330. department.sum += emp.salary;
331. }
332. **return** departments;
333. }
335. **static** DepartmentData grabCurrentDepartment(String departmentName,
336. ArrayList<DepartmentData> departments) {
337. **boolean** isInArr = **false**;
338. **int** i = 0;
339. DepartmentData department = **null**;
340. **while** (!isInArr && i < departments.size()) {
341. DepartmentData temp = departments.get(i);
342. **if** (temp.name.equals(departmentName)) {
343. isInArr = **true**;
344. department = temp;
345. }
346. i++;
347. }
348. **return** department;
349. }
351. **static** **void** sortDepartments(ArrayList<DepartmentData> departments) {
352. **for** (**int** i = 0; i < departments.size(); i++) {
353. **for** (**int** j = 1; j < departments.size() - i; j++) {
354. DepartmentData prev = departments.get(j - 1);
355. DepartmentData curr = departments.get(j);
356. **if** (curr.name.compareToIgnoreCase(prev.name) < 0) {
357. departments.set(j, prev);
358. departments.set(j - 1, curr);
359. }
360. }
361. }
362. }
364. **static** **void** showDepartmentSum(Scanner scan) {
365. ArrayList<Employee> data = grabData(scan, **false**, **null**);
366. ArrayList<DepartmentData> departments = grabDepartments(data);
367. sortDepartments(departments);
368. **for** (DepartmentData department : departments) {
369. System.out.println(department.name);
370. System.out.println("Общая сумма выплат за месяц по отделу = " + department.sum + "**\n**");
371. }
372. showInfo();
373. executeAction(chooseAction(scan), scan);
374. }
376. **static** **void** showArr(ArrayList<Employee> data) {
377. **for** (**int** i = 0; i < data.size(); i++) {
378. Employee emp = data.get(i);
379. System.out.println("Сотрудник №" + (i + 1));
380. System.out.println("Фамилия: " + emp.surname);
381. System.out.println("Наименование отдела: " + emp.department);
382. System.out.println("Заработная плата: " + emp.salary + "**\n**");
383. }
384. }
386. **static** **void** executeAction(String action, Scanner scan) {
387. **if** (action.equals("доб"))
388. addData(scan);
389. **else** **if** (action.equals("изм"))
390. changeData(scan);
391. **else** **if** (action.equals("уд"))
392. deleteData(scan);
393. **else** **if** (action.equals("выв"))
394. showData(scan);
395. **else** **if** (action.equals("сум"))
396. showDepartmentSum(scan);
397. **else** **if** (action.equals("выход"))
398. System.exit(0);
399. }
401. **public** **static** **void** main(String[] args) {
402. System.out.println("Данная программа позволяет работать со сведениями о месячной заработной
403. плате сотрудников отдела.**\n**");
404. Scanner scan = **new** Scanner(System.in);
405. showInfo();
406. String action = chooseAction(scan);
407. executeAction(action, scan);
408. }
409. }

**Схема алгоритма**

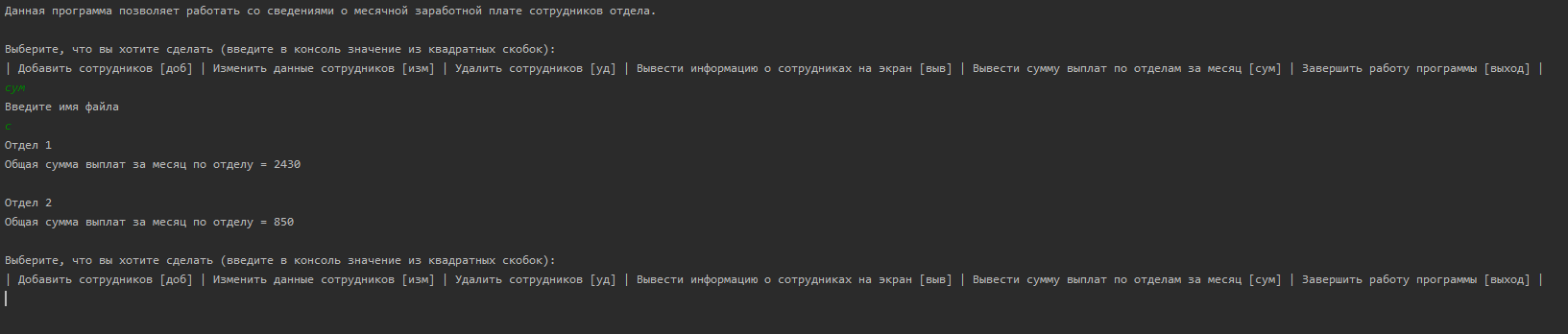




**Скриншоты**

**C++**



**Java**

