МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ ИМЕНИ

ПАТРИСА ЛУМУМБЫ»

(РУДН)

Факультет Искусственного Интеллекта

факультет

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №**

Практикум по программированию

наименование дисциплины

Вариант №8

тема (вариант)

Преподаватель \_\_\_\_\_\_\_\_\_\_ В.С. Тынченко

подпись, дата инициалы, фамилия

Студент ЗФИмд01-24 1132249437 \_\_\_\_\_\_\_\_\_\_ А.В. Клюкин

номер группы номер студ. билета подпись, дата инициалы, фамилия

Москва 2025

**Цель работы**

- Приобретение навыков работы с текстовыми и двоичными файлами

**Задачи работы:**Информация о студентах размещается в двух двоичных файлах. В первом

файле: фамилия, имя, отчество, курс, оценки по физике, математике, программированию. Во втором файле для тех же студентов в том же порядке:

фамилия, имя, отчество, пол, год рождения. Отсортировать фамилии в

обоих файлах по алфавиту. Определить, по какому предмету больше всего

не успевают юноши.

**Основные этапы**

1. Ознакомиться с постановкой задачи (вариант задания соответствует

вашему номеру в списке группы).

1. Внимательно изучить пример программы, выполняющей обработку

информации из текстовых и двоичных файлов.

1. Составить программу, решающую поставленную задачу. Общие требования к программе:

двоичные файлы создаются в программе из текстовых файлов.

4. Отладить программу на подготовленных наборах тестовых данных.

5. Подготовить отчет по лабораторной работе.

**Листинг**

1. #include <iostream>

2. #include <fstream>

3. #include <sstream>

4. #include <vector>

5. #include <string>

6. #include <locale>

7. #include <codecvt>

8. #include <windows.h>

9. #include <algorithm>

10.

11. using namespace std;

12.

13. struct StudentAcademics {

14. wstring lastName;

15. wstring firstName;

16. wstring patronymic;

17. int course;

18. int physics;

19. int math;

20. int programming;

21. };

22.

23. struct StudentPersonal {

24. wstring lastName;

25. wstring firstName;

26. wstring patronymic;

27. wchar\_t gender;

28. int birthYear;

29. };

30.

31. struct StudentFull {

32. StudentAcademics academics;

33. StudentPersonal personal;

34.

35. StudentFull(StudentAcademics a, StudentPersonal p) : academics(a), personal(p) {}

36. };

37.

38. wstring utf8\_to\_wstring(const string& str) {

39. wstring\_convert<codecvt\_utf8<wchar\_t>> converter;

40. return converter.from\_bytes(str);

41. }

42.

43. string removeCR(const string& str) {

44. string result;

45. for (char c : str) {

46. if (c != '\r') {

47. result += c;

48. }

49. }

50. return result;

51. }

52.

53. vector<StudentAcademics> readScores(const string& filename) {

54. vector<StudentAcademics> students;

55. ifstream file(filename);

56.

57. if (!file.is\_open()) {

58. cerr << "Ошибка открытия файла: " << filename << endl;

59. return students;

60. }

61.

62. string line;

63. while (getline(file, line)) {

64. line = removeCR(line);

65. stringstream ss(line);

66. StudentAcademics s;

67. string token;

68.

69. getline(ss, token, ','); s.lastName = utf8\_to\_wstring(token);

70. getline(ss, token, ','); s.firstName = utf8\_to\_wstring(token);

71. getline(ss, token, ','); s.patronymic = utf8\_to\_wstring(token);

72. getline(ss, token, ','); s.course = stoi(token);

73. getline(ss, token, ','); s.physics = stoi(token);

74. getline(ss, token, ','); s.math = stoi(token);

75. getline(ss, token, ','); s.programming = stoi(token);

76.

77. students.push\_back(s);

78. }

79.

80. return students;

81. }

82.

83. vector<StudentPersonal> readPersonal(const string& filename) {

84. vector<StudentPersonal> students;

85. ifstream file(filename);

86.

87. if (!file.is\_open()) {

88. cerr << "Ошибка открытия файла: " << filename << endl;

89. return students;

90. }

91.

92. string line;

93. while (getline(file, line)) {

94. line = removeCR(line);

95. stringstream ss(line);

96. StudentPersonal s;

97. string token;

98.

99. getline(ss, token, ','); s.lastName = utf8\_to\_wstring(token);

100. getline(ss, token, ','); s.firstName = utf8\_to\_wstring(token);

101. getline(ss, token, ','); s.patronymic = utf8\_to\_wstring(token);

102. getline(ss, token, ','); s.gender = utf8\_to\_wstring(token)[0];

103. getline(ss, token, ','); s.birthYear = stoi(token);

104.

105. students.push\_back(s);

106. }

107.

108. return students;

109. }

110.

111.

112. void printStudent(const StudentFull& s) {

113. wcout << s.academics.lastName << L" "

114. << s.academics.firstName << L" "

115. << s.academics.patronymic

116. // Дополнительно можно вывести другие поля

117.

118. << L" (Курс: " << s.academics.course << L")"

119. << endl;

120. }

121.

122.

123.

124. int main() {

125. // Настройка кодировок

126. SetConsoleOutputCP(CP\_UTF8);

127. setlocale(LC\_ALL, "Russian");

128.

129. // Чтение данных

130. auto academics = readScores("C:/Users/bardr/source/repos/Programming workshop/Лабы\_по\_cpp/лаб10/scores.csv");

131. auto personal = readPersonal("C:/Users/bardr/source/repos/Programming workshop/Лабы\_по\_cpp/лаб10/students.csv");

132.

133. // Проверка чтения

134. if (academics.empty() || personal.empty()) {

135. cerr << "Ошибка: один из файлов не прочитан!" << endl;

136. return 1;

137. }

138.

139. // Объединение данных

140. vector<StudentFull> students;

141. size\_t minSize = min(academics.size(), personal.size());

142.

143. for (size\_t i = 0; i < minSize; ++i) {

144. students.push\_back({ academics[i], personal[i] });

145. }

146.

147. // Вывод информации

148. wcout << L"Список студентов:\n";

149. for (const auto& s : students) {

150. wcout

151. << L"-------------------------\n"

152. << L"ФИО: " << s.academics.lastName << L" "

153. << s.academics.firstName << L" "

154. << s.academics.patronymic << L"\n"

155. << L"Курс: " << s.academics.course << L"\n"

156. << L"Пол: " << s.personal.gender << L"\n"

157. << L"Год рождения: " << s.personal.birthYear << L"\n"

158. << L"Оценки:\n"

159. << L" Физика: " << s.academics.physics << L"\n"

160. << L" Математика: " << s.academics.math << L"\n"

161. << L" Программирование: " << s.academics.programming << L"\n";

162. }

163.

164.

165.

166.

167.

168. wcout << L"--- До сортировки ---" << endl;

169. for (const auto& s : students) {

170. printStudent(s);

171. }

172.

173.

174.

175. // --- Сортировка с помощью лямбда-функции ---

176.

177. sort(students.begin(), students.end(), [](const StudentFull& a, const StudentFull& b) {

178. // Сравниваем фамилии

179. if (a.academics.lastName != b.academics.lastName) {

180. return a.academics.lastName < b.academics.lastName;

181. }

182. // Если фамилии одинаковые, сравниваем имена

183. if (a.academics.firstName != b.academics.firstName) {

184. return a.academics.firstName < b.academics.firstName;

185. }

186. // Если и имена одинаковые, сравниваем отчества

187. return a.academics.patronymic < b.academics.patronymic;

188. });

189.

190.

191.

192.

193.

194.

195. wcout << L"\n--- После сортировки (по ФИО) ---" << endl;

196. for (const auto& s : students) {

197. printStudent(s);

198. }

199. int sum\_phy = 0, sum\_math = 0, sum\_prog = 0;

200. int n = 0;

201. for (const auto& s : students) {

202. sum\_phy += s.academics.physics;

203. sum\_math += s.academics.math;

204. sum\_prog += s.academics.programming;

205.

206. n++;

207. }

208.

209. double m = min(min(sum\_phy, sum\_math), sum\_prog);

210. wcout << '\n' << endl << endl << endl;

211. if (m == sum\_phy) {

212. wcout << L"Наихудший средний балл по физике: " << m / n;

213. }

214. else if (m == sum\_math) {

215. wcout << L"Наихудший средний балл по математике: " << m / n;

216. }

217. else {

218. wcout << L"Наихудший средний балл по программированию: " << m / n;

219. }

220.

221. wcout << '\n' << endl << endl << endl;

222.

223.

224.

225.

226.

227.

228.

229.

230.

231.

232.

233. Return 0

234. }

235.