ГУАП

КАФЕДРА № 14

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ст. преп |  |  |  | Т.Л. Прокофьева |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3 |
| МАССИВЫ И СВЯЗНЫЕ СПИСКИ |
| по курсу: ПРОГРАММИРОВАНИЕ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1043 |  |  |  | Д.Д. Шпагин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

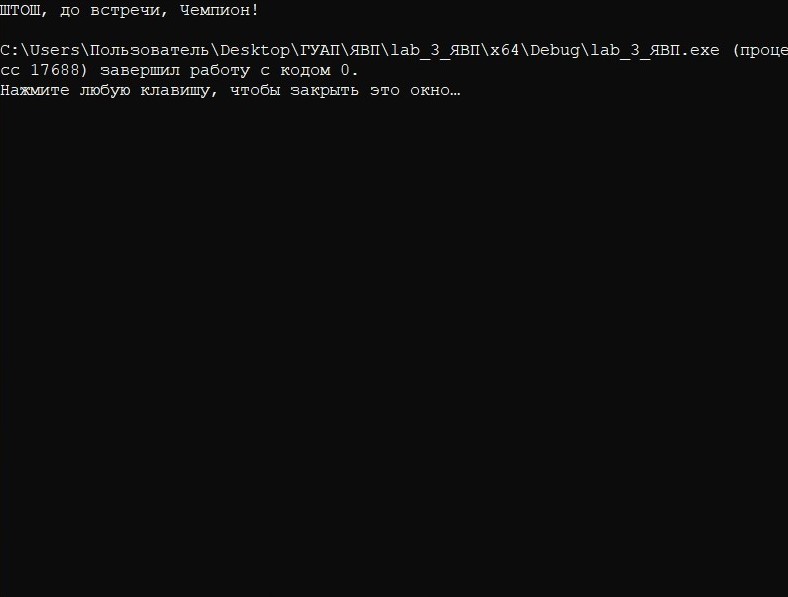
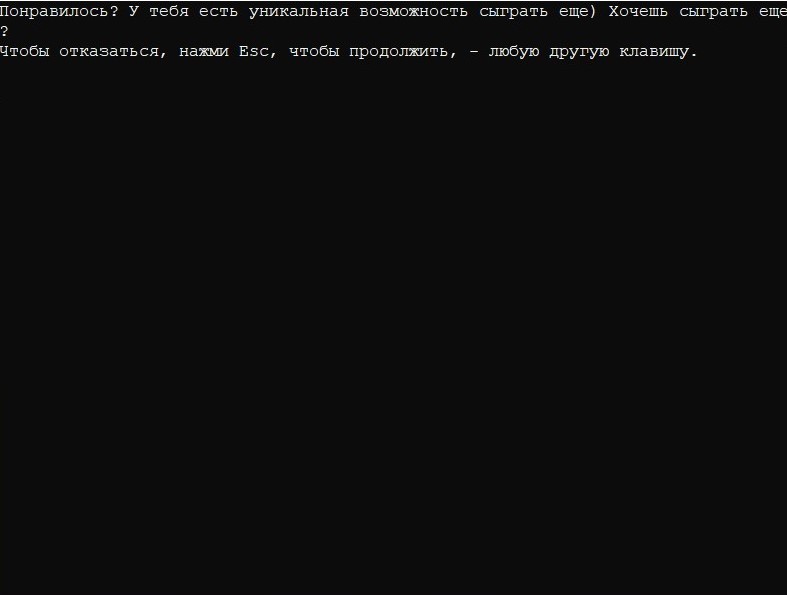
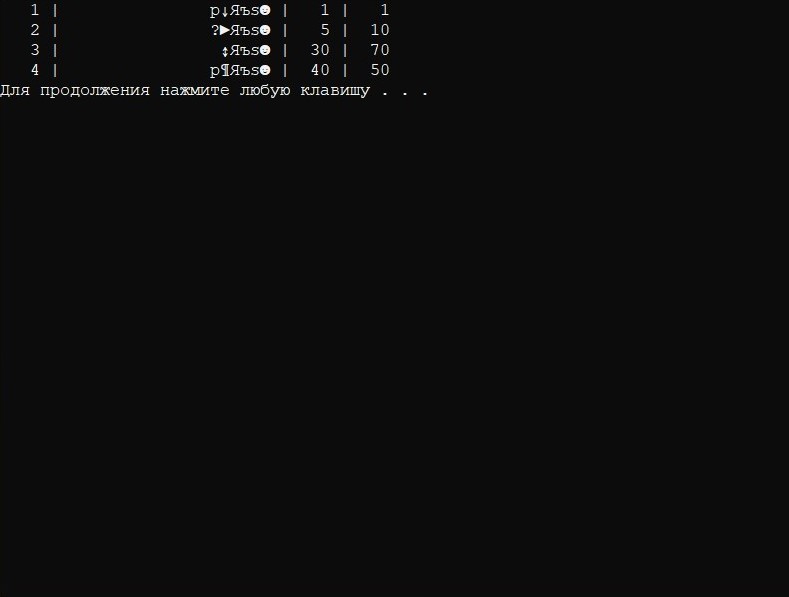
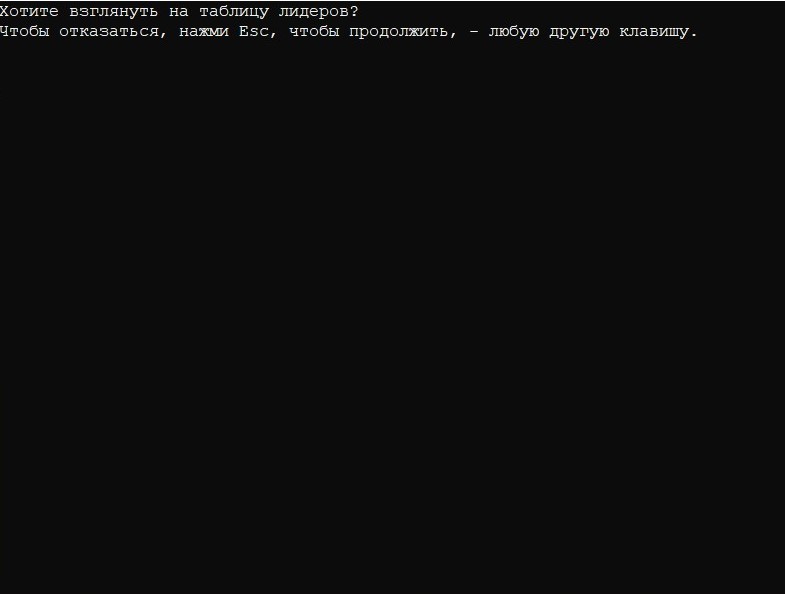
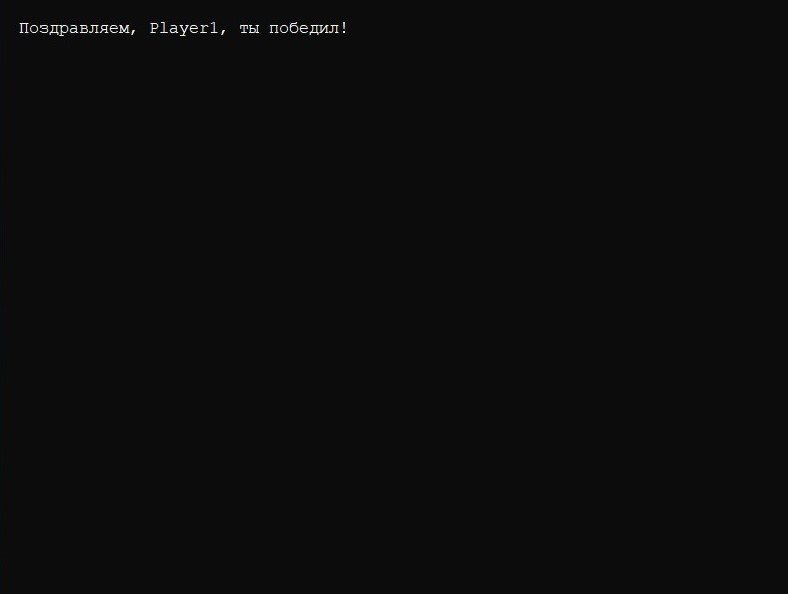
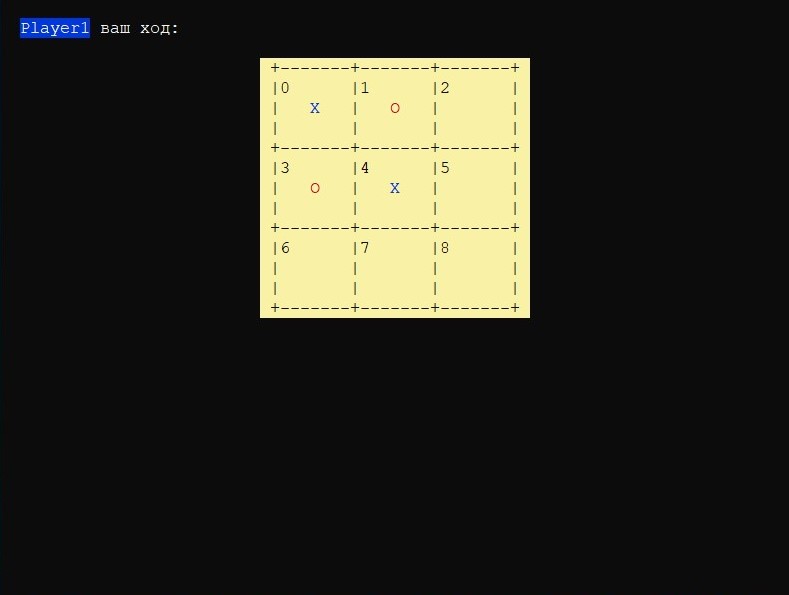
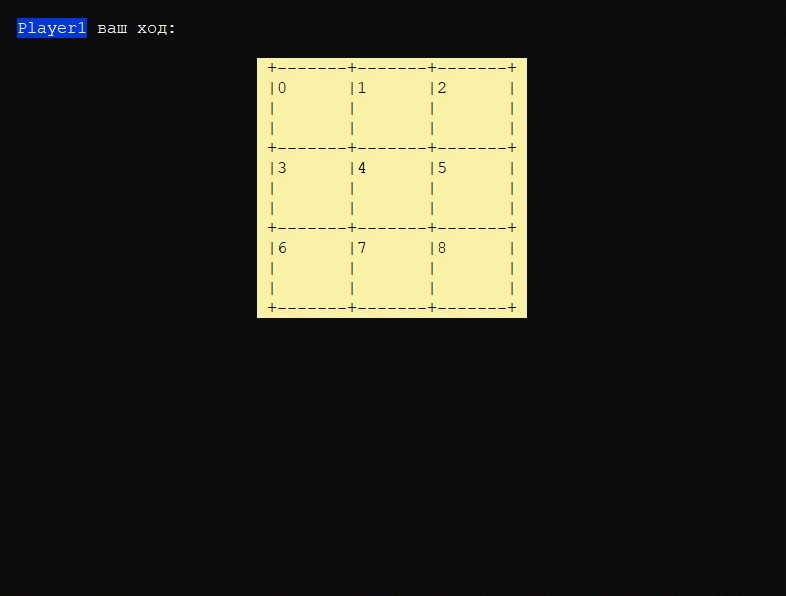
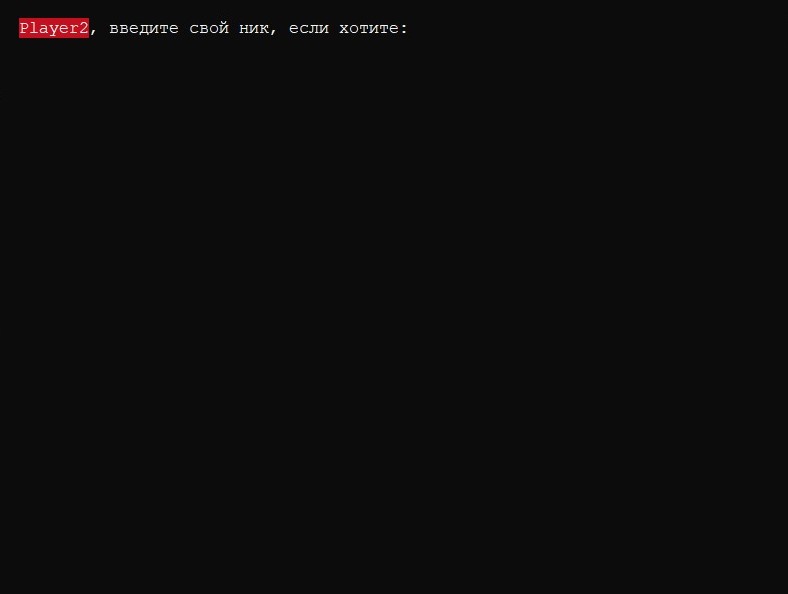
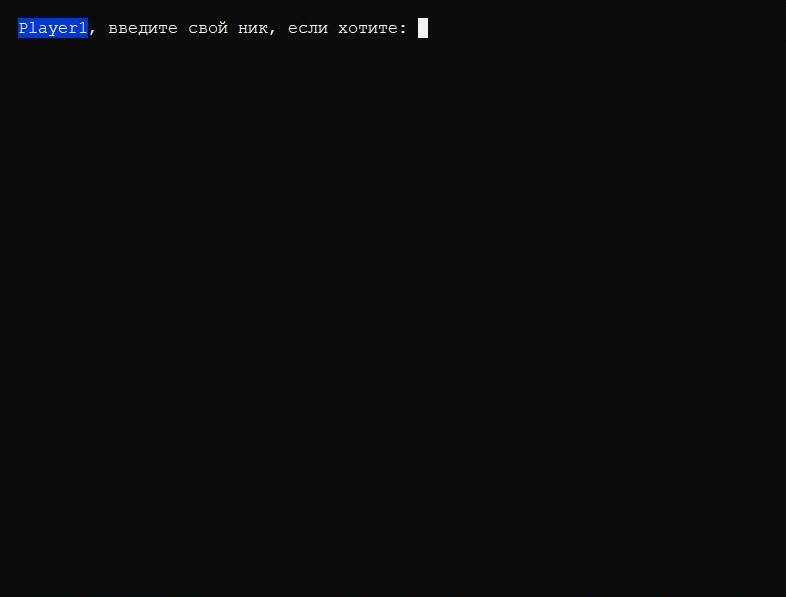
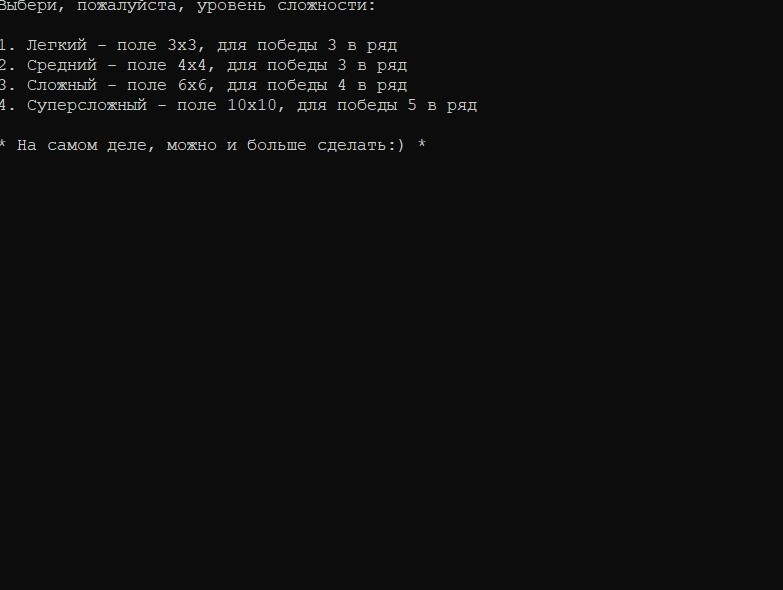
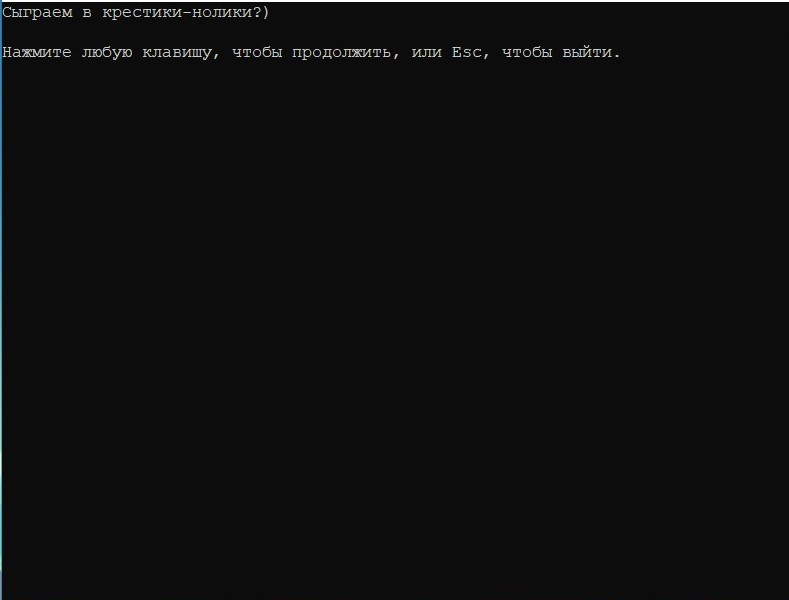
1. **Цель**

Получение навыков чтения и записи информации из файлов.

1. **Постановка задачи**

Игра в «крестики-нолики». 2 игрока, таблица рекордов. Три уровня сложности в зависимости от размера поля (3х3, 4х4, 6х6).

1. **Listing программы**
2. #include <iostream>
3. #include <vector>
4. #include <windows.h>
5. #include <string>
6. #include <fstream>
7. #include <locale>
8. #include <conio.h>
9. #include <algorithm>
11. **using** **namespace** std;
12. **static** HANDLE hndl = GetStdHandle(STD\_OUTPUT\_HANDLE);
13. template <typename T> **void** Print(T, **int**, **int**, **byte**);
14. template <typename T> **void** Print(T, **int**, **int**);
15. template <typename T> **void** Print(T, **byte**);
16. **void** Color(**byte**);
18. **class** TicTacToe {
19. **private**:
20. **string** nick\_player1\_, nick\_player2\_, winner\_;
21. **int** size\_, block\_;
22. **char** symbol\_ = 'X';
23. vector <vector <**char**>> map\_;
25. **bool** checkDiagonal(**char**) **const**;
26. **bool** checkLanes(**char**) **const**;
27. **public**:
28. TicTacToe() : size\_(3), block\_(3) {
29. nick\_player1\_ = "Player1";
30. nick\_player2\_ = "Player2";
31. map\_.resize(3, vector<**char**>(3));
32. }
33. TicTacToe(**int** size) : size\_(size) {
34. nick\_player1\_ = "Player1", nick\_player2\_ = "Player2";
35. (size\_ == 3 || size\_ == 4) ? block\_ = 3 : block\_ = 4;
36. map\_.resize(size\_, vector<**char**>(size\_));
37. }
38. TicTacToe(**int** size, **int** block) : size\_(size), block\_(block) {
39. nick\_player1\_ = "Player1", nick\_player2\_ = "Player2";
40. map\_.resize(size\_, vector<**char**>(size\_));
41. }
42. **void** sizeAndBlock(**int** size, **int** block) {
43. size\_ = size, block\_ = block;
44. map\_.clear();
45. map\_.resize(size, vector<**char**>(size));
46. }
48. **void** addPlayers();
49. **void** drawMap() **const**;
50. **bool** checkWin();
51. **bool** isFull() **const**;
52. **void** makeMove();
53. **void** play();
55. **string** nickPlayer1() { **return** nick\_player1\_; }
56. **string** nickPlayer2() { **return** nick\_player1\_; }
57. **string** nickWinner() { **return** winner\_; }
58. };
60. typedef **struct** {
61. **string** nick;
62. unsigned wins, games;
63. } Player;
65. **int** main() {
66. setlocale(LC\_ALL, "ru");
68. cout << "Сыграем в крестики-нолики?)\n\nНажмите любую клавишу, чтобы продолжить, или Esc, чтобы выйти." << endl;
69. **if** (\_getch() == 27) {
70. system("cls");
71. cout << "ШТОШ, до встречи, Чемпион!" << endl;
72. **return** 0;
73. }
75. vector <Player> players;
76. fstream fpin;
77. fpin.open("LiderBoard.ttt");
79. **while** (**true**) {
80. system("cls");
81. TicTacToe game;
83. cout << "Выбери, пожалуйста, уровень сложности:" << endl << endl;
84. cout << "1. Легкий - поле 3х3, для победы 3 в ряд" << endl;
85. cout << "2. Средний - поле 4х4, для победы 3 в ряд" << endl;
86. cout << "3. Сложный - поле 6х6, для победы 4 в ряд" << endl;
87. cout << "4. Суперсложный - поле 10х10, для победы 5 в ряд" << endl << endl;
88. cout << "\* На самом деле, можно и больше сделать:) \*" << endl;
89. **string** level;
90. **while** (level.empty()) {
91. getline(cin, level);
92. **switch** (level[0]) {
93. **case** '1': game.sizeAndBlock(3, 3); **break**;
94. **case** '2': game.sizeAndBlock(4, 3); **break**;
95. **case** '3': game.sizeAndBlock(6, 4); **break**;
96. **case** '4': game.sizeAndBlock(10, 5); **break**;
97. **default**:
98. cout << "Хм, что-то не то... Попробуй еще)\n";
99. level.clear();
100. **break**;
101. }
102. }
103. system("cls");
105. game.addPlayers();
106. game.play();
108. **if** (!players.empty()) {
109. **if** (fpin.is\_open()) {
110. fpin.clear();
111. **for** (auto i = 0; i < players.size(); i++)
112. fpin << players[i].games << players[i].wins << players[i].nick;
113. }
114. }
116. **if** (fpin.is\_open()) {
117. **while** (!fpin.eof()) {
118. Player temp;
119. fpin >> temp.games >> temp.wins >> temp.nick;
120. players.push\_back(temp);
121. }
122. **if** (!players.empty()) {
123. **if** (game.nickPlayer1() != "Player1") {
124. **bool** flag = **false**;
125. **for** (auto i = 0; i < players.size(); i++)
126. **if** (game.nickPlayer1() == players[i].nick) {
127. players[i].games++;
128. **if** (game.nickWinner() == players[i].nick)
129. players[i].wins++;
130. flag = **true**;
131. exit;
132. }
133. **if** (!flag) {
134. Player temp; temp.nick = game.nickPlayer1(), temp.games = 1;
135. **if** (game.nickWinner() == temp.nick)
136. temp.wins = 1;
137. players.push\_back(temp);
138. }
139. }
140. **if** (game.nickPlayer2() != "Player2") {
141. **bool** flag = **false**;
142. **for** (auto i = 0; i < players.size(); i++)
143. **if** (game.nickPlayer2() == players[i].nick) {
144. players[i].games++;
145. **if** (game.nickWinner() == players[i].nick)
146. players[i].wins++;
147. flag = **true**;
148. exit;
149. }
150. **if** (!flag) {
151. Player temp; temp.nick = game.nickPlayer2(), temp.games = 1;
152. **if** (game.nickWinner() == temp.nick)
153. temp.wins = 1;
154. players.push\_back(temp);
155. }
156. }
157. sort(players.begin(), players.end(), [](**const** Player& player1, **const** Player& player2) -> **bool**
158. {
159. **return** player1.wins < player2.wins;
160. }
161. );
162. }
163. }
165. cout << "Хотите взглянуть на таблицу лидеров?\nЧтобы отказаться, нажми Esc, чтобы продолжить, - любую другую клавишу.\n";
166. **if** (\_getch() != 27) {
167. system("cls");
168. **if** (!players.empty()) {
169. **for** (auto i = 0; i < players.size(); i++)
170. printf(" %3i | %20s | %3i | %3i\n", i + 1, players[i].nick, players[i].wins, players[i].games);
171. }
172. **else**
173. cout << "Или что-то пошло не так, и файл с данными оказался пуст, или вы наш первый игрок!)" << endl;
174. system("pause");
175. }
177. system("cls");
178. cout << "Понравилось? У тебя есть уникальная возможность сыграть еще) Хочешь сыграть еще?\nЧтобы отказаться, нажми Esc, чтобы продолжить, - любую другую клавишу.";
179. **if** (\_getch() == 27) {
180. system("cls");
181. cout << "ШТОШ, до встречи, Чемпион!" << endl;
182. **return** 0;
183. }
184. }
185. }
187. template <typename T> **void** Print(T buf, **int** x, **int** y, **byte** color) {
188. SetConsoleTextAttribute(hndl, (WORD)((0 << 4) | color));
189. COORD p = { (SHORT)x, (SHORT)y };
190. SetConsoleCursorPosition(hndl, p);
191. cout << buf;
192. SetConsoleTextAttribute(hndl, (WORD)((0 << 4) | 15));
193. }
194. template <typename T> **void** Print(T buf, **int** x, **int** y) {
195. SetConsoleTextAttribute(hndl, (WORD)((0 << 4) | 15));
196. COORD p = { (SHORT)x, (SHORT)y };
197. SetConsoleCursorPosition(hndl, p);
198. cout << buf;
199. }
200. template <typename T> **void** Print(T buf, **byte** color) {
201. SetConsoleTextAttribute(hndl, (WORD)((0 << 4) | color));
202. cout << buf;
203. SetConsoleTextAttribute(hndl, (WORD)((0 << 4) | 15));
204. }
206. **void** Color(**byte** color) {
207. SetConsoleTextAttribute(hndl, (WORD)((0 << 4) | color));
208. }
210. **void** TicTacToe::addPlayers() {
211. Print(symbol\_ == 'X' ? nick\_player1\_ : nick\_player2\_, 2, 1, symbol\_ == 'X' ? 31 : 79);
212. cout << ", введите свой ник, если хотите: ";
213. **string** nick; getline(cin, nick);
214. **if** (!nick.empty()) nick\_player1\_ = nick;
215. system("cls");
216. Print(symbol\_ == 'O' ? nick\_player2\_ : nick\_player2\_, 2, 1, symbol\_ == 'O' ? 31 : 79);
217. cout << ", введите свой ник, если хотите: ";
218. getline(cin, nick);
219. **if** (!nick.empty()) nick\_player2\_ = nick;
220. system("cls");
221. }
223. **void** TicTacToe::drawMap() **const** {
224. CONSOLE\_SCREEN\_BUFFER\_INFO csbiData;
225. GetConsoleScreenBufferInfo(hndl, &csbiData);
226. unsigned cell\_size\_x = 9, cell\_size\_y = 5;
227. unsigned offset\_x = (csbiData.dwSize.X - (cell\_size\_x - 1) \* size\_) / 2, offset\_y = 3;
228. **byte** color = 224;
229. **string** horizontal = " ", vertical = " ";
230. **for** (unsigned i = 0; i < size\_ \* (cell\_size\_x - 1) + 1; i++) {
231. **if** (i % (cell\_size\_x - 1) == 0)
232. horizontal += "+", vertical += "|";
233. **else**
234. horizontal += "-", vertical += " ";
235. }
236. horizontal += " ", vertical += " ";
237. **for** (unsigned i = 0; i < size\_ \* (cell\_size\_y - 1) + 1; i++) {
238. **if** (i % (cell\_size\_y - 1) == 0)
239. Print(horizontal, offset\_x - 1, i + offset\_y, color);
240. **else**
241. Print(vertical, offset\_x - 1, i + offset\_y, color);
242. }
243. **for** (auto i = 0; i < size\_ \* size\_; i++) {
244. Print(i, (i % size\_) \* (cell\_size\_x - 1) + offset\_x + 1,
245. (i / size\_) \* (cell\_size\_y - 1) + offset\_y + 1, color);
246. **if** (map\_[i % size\_][i / size\_])
247. Print(map\_[i % size\_][i / size\_],
248. (i % size\_) \* (cell\_size\_x - 1) + offset\_x + cell\_size\_x / 2,
249. (i / size\_) \* (cell\_size\_y - 1) + offset\_y + cell\_size\_y / 2,
250. map\_[i % size\_][i / size\_] == 'X' ? color + 1: color + 4);
251. }
252. }
254. **void** TicTacToe::makeMove() {
255. Print(symbol\_ == 'X' ? nick\_player1\_ : nick\_player2\_, 2, 1, symbol\_ == 'X' ? 31 : 79);
256. cout << +" ваш ход: ";
257. **bool** flag = **true**;
258. **while** (flag) {
259. **string** str;
260. getline(cin, str);
261. **if** (!str.empty()) {
262. **int** num = stoi(str);
263. **if** (num >= 0 && num < size\_ \* size\_)
264. **if** (map\_[num % size\_][num / size\_] == '\0') {
265. map\_[num % size\_][num / size\_] = symbol\_;
266. symbol\_ == 'X' ? symbol\_ = symbol\_ = 'O' : symbol\_ = 'X';
267. flag = **false**;
268. }
269. }
270. }
271. }
273. **bool** TicTacToe::isFull() **const** {
274. **for** (auto col = 0; col < size\_; col++)
275. **for** (auto row = 0; row < size\_; row++)
276. **if** (map\_[col][row] == NULL)
277. **return** **false**;
278. **return** **true**;
279. }
281. **bool** TicTacToe::checkDiagonal(**char** symbol) **const** {
282. **for** (auto row = 0; row <= size\_ - block\_; row++)
283. **for** (auto col = 0; col <= size\_ - block\_; col++) {
284. **bool** toright = **true**, toleft = **true**;
285. **for** (auto i = 0; i < block\_; i++) {
286. toright &= (map\_[(size\_t)i + row][(size\_t)i + col] == symbol);
287. toleft &= (map\_[(size\_t)block\_ - 1 - i + row][(size\_t)i + col] == symbol);
288. }
289. **if** (toright || toleft) **return** **true**;
290. }
291. **return** **false**;
292. }
294. **bool** TicTacToe::checkLanes(**char** symbol) **const** {
295. **for** (auto offset = 0; offset <= size\_ - block\_; offset++)
296. **for** (auto row = 0; row < size\_; row++) {
297. **bool** cols = **true**, rows = **true**;
298. **for** (auto col = offset; col < offset + block\_; col++) {
299. cols &= (map\_[(size\_t)row][(size\_t)col] == symbol);
300. rows &= (map\_[(size\_t)col][(size\_t)row] == symbol);
301. }
302. **if** (cols || rows) **return** **true**;
303. }
304. **return** **false**;
305. }
307. **bool** TicTacToe::checkWin() {
308. **bool** player1 = checkLanes('X') || checkDiagonal('X');
309. **bool** player2 = checkLanes('O') || checkDiagonal('O');
310. **if** (player1)
311. winner\_ = nick\_player1\_;
312. **if** (player2)
313. winner\_ = nick\_player2\_;
314. **return** (player1 || player2);
315. }
317. **void** TicTacToe::play() {
318. **while** (!isFull() && !checkWin()) {
319. drawMap();
320. makeMove();
321. system("cls");
322. }
323. system("cls");
324. **if** (!winner\_.empty()) Print("Поздравляем, " + winner\_ + ", ты победил!", 2, 1);
325. **else** Print("Сегодня победила дружба!", 2, 1);
326. \_getch();
327. system("cls");
328. }
329. **Тестовые примеры**



1. **Вывод**

В результате выполнения лабораторной работы была создана игра в «крестики-нолики» на 2 игрока, с таблицей рекордов и тремя уровнями сложности в зависимости от размера поля (3х3, 4х4, 6х6).