Subway Schedule Board

**Problem Statement:**

This program is made to facilitate passengers travelling through subway. This is done through making a schedule board for the passengers to see where they are headed and what subways they are boarding

**Brief:**

The map is called to visualize the passenger about the subway network. After the passenger enters their destination, the input from the passenger is first converted into integer form from character form. The program checks for available tracks at the current station. The information that the passenger will need are then displayed on the schedule board in a tabular form along with information required for other passengers

**Code:**

1. // Kamil Rasheed Siddiqui ME201024
2. // Syed Ali Abbas ME201018
3. // Farrukh Shahid ME201011
4. // Ubair Ahmed Khan ME201066
5. #include <iostream>
6. #include <ctime>
7. #include <iomanip>
8. using namespace std;
9. void subwayMap();
10. float table(int num, int destination, int platform, int arrival);
11. int main()
12. {
13. const int numPassengers = 10;
14. int track[5][2];
15. char pDepart, pArrive;
16. int platform;
17. int pDeparture[numPassengers];
18. int pArrival[numPassengers];
19. subwayMap();
20. cout << endl;
21. srand(time(0));
22. for(int i = 0; i < numPassengers; i++) {
23. pDeparture[i] = (rand() % 5) + 1;
24. }
25. for(int i = 0; i < numPassengers; i++) {
26. do {
27. pArrival[i] = (rand() % 5) + 1;
28. }
29. while(pDeparture[i] == pArrival[i]);
30. }
31. do {
32. cout << "Please select your departing station: ";
33. cin >> pDepart;
34. if(pDepart != 'A' && pDepart != 'B' && pDepart != 'C' && pDepart != 'D' && pDepart != 'E'){
35. cout << "Incorrect selection, please enter again" << endl;
36. }
37. }
38. while(pDepart != 'A' && pDepart != 'B' && pDepart != 'C' && pDepart != 'D' && pDepart != 'E');
39. cout << endl;
40. do {
41. cout << "Please select your arriving station: ";
42. cin >> pArrive;
43. if(pDepart != 'A' && pDepart != 'B' && pDepart != 'C' && pDepart != 'D' && pDepart != 'E'){
44. cout << "Incorrect selection, please enter again" << endl;
45. }
46. }
48. while(pArrive != 'A' && pArrive != 'B' && pArrive != 'C' && pArrive != 'D' && pArrive != 'E');
49. switch(pDepart){
50. case 'A':
51. pDeparture[0] = 1;
52. break;
53. case 'B':
54. pDeparture[0] = 2;
55. break;
56. case 'C':
57. pDeparture[0] = 3;
58. break;
59. case 'D':
60. pDeparture[0] = 4;
61. break;
62. case 'E':
63. pDeparture[0] = 5;
64. break;
65. }
66. switch(pArrive){
67. case 'A':
68. pArrival[0] = 1;
69. break;
70. case 'B':
71. pArrival[0] = 2;
72. break;
73. case 'C':
74. pArrival[0] = 3;
75. break;
76. case 'D':
77. pArrival[0] = 4;
78. break;
79. case 'E':
80. pArrival[0] = 5;
81. break;
82. }
83. for(int i = 0; i < 5; i++) {
84. track[i][0] = (rand() % 2) + 1;
85. if(track[i][0] == 1) {
86. track[i][0] = 0;
87. }
88. else if (track[i][0] == 2) {
89. track[i][0] = 1;
90. }
91. if(track[i][0] == 0) {
92. track[i][1] = 1;
93. }
94. else if(track[i][0] == 1) {
95. track[i][1] = 0;
96. }
97. }
98. cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ " << endl;
99. cout << setw(20) << left << " |Passenger no."
100. << setw(30) << left << " Departure from station"
101. << setw(26) << left << " Departure platform"
102. << setw(29) << left << " Arrival to destination|" << endl;
103. for(int i = 0; i < numPassengers; i++) {
104. if(pDeparture[i] == 1 && pArrival[i] == 2) {
105. if(track[i][0] == 1){
106. platform = 1;
107. }
108. else if(track[i][1] == 1){
109. platform = 2;
110. }
111. table(i, pDeparture[i], platform, pArrival[i]);
112. }
113. if(pDeparture[i] == 1 && pArrival[i] == 3) {
114. if(track[i][0] == 1){
115. platform = 1;
116. }
117. else if(track[i][1] == 1){
118. platform = 2;
119. }
120. table(i, pDeparture[i], platform, pArrival[i]);
121. }
122. if(pDeparture[i] == 1 && pArrival[i] == 4) {
123. if(track[i][0] == 1){
124. platform = 1;
125. }
126. else if(track[i][1] == 1){
127. platform = 2;
128. }
129. table(i, pDeparture[i], platform, pArrival[i]);
130. }
131. if(pDeparture[i] == 1 && pArrival[i] == 5) {
132. if(track[i][0] == 1){
133. platform = 1;
134. }
135. else if(track[i][1] == 1){
136. platform = 2;
137. }
138. table(i, pDeparture[i], platform, pArrival[i]);
139. }
140. if(pDeparture[i] == 2 && pArrival[i] == 1) {
141. if(track[i][0] == 1){
142. platform = 1;
143. }
144. else if(track[i][1] == 1){
145. platform = 2;
146. }
147. table(i, pDeparture[i], platform, pArrival[i]);
148. }
149. if(pDeparture[i] == 2 && pArrival[i] == 3) {
150. if(track[i][0] == 1){
151. platform = 1;
152. }
153. else if(track[i][1] == 1){
154. platform = 2;
155. }
156. table(i, pDeparture[i], platform, pArrival[i]);
157. }
158. if(pDeparture[i] == 2 && pArrival[i] == 4) {
159. if(track[i][0] == 1){
160. platform = 1;
161. }
162. else if(track[i][1] == 1){
163. platform = 2;
164. }
165. table(i, pDeparture[i], platform, pArrival[i]);
166. }
167. if(pDeparture[i] == 2 && pArrival[i] == 5) {
168. if(track[i][0] == 1){
169. platform = 1;
170. }
171. else if(track[i][1] == 1){
172. platform = 2;
173. }
174. table(i, pDeparture[i], platform, pArrival[i]);
175. }
176. if(pDeparture[i] == 3 && pArrival[i] == 1) {
177. if(track[i][0] == 1){
178. platform = 1;
179. }
180. else if(track[i][1] == 1){
181. platform = 2;
182. }
183. table(i, pDeparture[i], platform, pArrival[i]);
184. }
185. if(pDeparture[i] == 3 && pArrival[i] == 2) {
186. if(track[i][0] == 1){
187. platform = 1;
188. }
189. else if(track[i][1] == 1){
190. platform = 2;
191. }
192. table(i, pDeparture[i], platform, pArrival[i]);
193. }
194. if(pDeparture[i] == 3 && pArrival[i] == 4) {
195. if(track[i][0] == 1){
196. platform = 1;
197. }
198. else if(track[i][1] == 1){
199. platform = 2;
200. }
201. table(i, pDeparture[i], platform, pArrival[i]);
202. }
203. if(pDeparture[i] == 3 && pArrival[i] == 5) {
204. if(track[i][0] == 1){
205. platform = 1;
206. }
207. else if(track[i][1] == 1){
208. platform = 2;
209. }
210. table(i, pDeparture[i], platform, pArrival[i]);
211. }
212. if(pDeparture[i] == 4 && pArrival[i] == 1) {
213. if(track[i][0] == 1){
214. platform = 1;
215. }
216. else if(track[i][1] == 1){
217. platform = 2;
218. }
219. table(i, pDeparture[i], platform, pArrival[i]);
220. }
221. if(pDeparture[i] == 4 && pArrival[i] == 2) {
222. if(track[i][0] == 1){
223. platform = 1;
224. }
225. else if(track[i][1] == 1){
226. platform = 2;
227. }
228. table(i, pDeparture[i], platform, pArrival[i]);
229. }
230. if(pDeparture[i] == 4 && pArrival[i] == 3) {
231. if(track[i][0] == 1){
232. platform = 1;
233. }
234. else if(track[i][1] == 1){
235. platform = 2;
236. }
237. table(i, pDeparture[i], platform, pArrival[i]);
238. }
239. if(pDeparture[i] == 4 && pArrival[i] == 5) {
240. if(track[i][0] == 1){
241. platform = 1;
242. }
243. else if(track[i][1] == 1){
244. platform = 2;
245. }
246. table(i, pDeparture[i], platform, pArrival[i]);
247. }
248. if(pDeparture[i] == 5 && pArrival[i] == 1) {
249. if(track[i][0] == 1){
250. platform = 1;
251. }
252. else if(track[i][1] == 1){
253. platform = 2;
254. }
255. table(i, pDeparture[i], platform, pArrival[i]);
256. }
257. if(pDeparture[i] == 5 && pArrival[i] == 2) {
258. if(track[i][0] == 1){
259. platform = 1;
260. }
261. else if(track[i][1] == 1){
262. platform = 2;
263. }
264. table(i, pDeparture[i], platform, pArrival[i]);
265. }
266. if(pDeparture[i] == 5 && pArrival[i] == 3) {
267. if(track[i][0] == 1){
268. platform = 1;
269. }
270. else if(track[i][1] == 1){
271. platform = 2;
272. }
273. table(i, pDeparture[i], platform, pArrival[i]);
274. }
275. if(pDeparture[i] == 5 && pArrival[i] == 4) {
276. if(track[i][0] == 1){
277. platform = 1;
278. }
279. else if(track[i][1] == 1){
280. platform = 2;
281. }
282. table(i, pDeparture[i], platform, pArrival[i]);
283. }
284. }
285. cout << " |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;
286. return 0;
287. }
288. void subwayMap()
289. {
290. cout << "This is the map of the an underground subway network" << endl << endl;
291. cout << " D" << endl;
292. cout << " /" << endl;
293. cout << " /" << endl;
294. cout << " /" << endl;
295. cout << " A ------------------------ B ---------------- C" << endl;
296. cout << " \\ " << endl;
297. cout << " \\ " << endl;
298. cout << " E " << endl;
299. }
300. float table(int num, int destination, int platform, int arrival)
301. {
302. char station1, station2;
303. switch(destination){
304. case 1:
305. station1 = 'A';
306. break;
307. case 2:
308. station1 = 'B';
309. break;
310. case 3:
311. station1 = 'C';
312. break;
313. case 4:
314. station1 = 'D';
315. break;
316. case 5:
317. station1 = 'E';
318. break;
319. }
320. switch(arrival){
321. case 1:
322. station2 = 'A';
323. break;
324. case 2:
325. station2 = 'B';
326. break;
327. case 3:
328. station2 = 'C';
329. break;
330. case 4:
331. station2 = 'D';
332. break;
333. case 5:
334. station2 = 'E';
335. break;
336. }
337. cout << " |-------------------------------------------------------------------------------------------------|" << endl;
338. cout << " |" << setw(19) << num + 1 << setw(30) << station1 << setw(26) << platform << setw(22) << station2 << "|" << endl;
339. return 0;
340. }

**Sample Run Output:**

