1. #include<stdio.h>
2. #include<stdlib.h>
3. struct node
4. {
5. **int** data;
6. struct node \*next;
7. };
8. struct node \*head;
10. **void** beginsert ();
11. **void** lastinsert ();
12. **void** randominsert();
13. **void** begin\_delete();
14. **void** last\_delete();
15. **void** random\_delete();
16. **void** display();
17. **void** search();
18. **void** main ()
19. {
20. **int** choice =0;
21. **while**(choice != 9)
22. {
23. printf("\n\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n");
24. printf("\nChoose one option from the following list ...\n");
25. printf("\n===============================================\n");
26. printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning\n
27. 5.Delete from last\n6.Delete node after specified location\n7.Search **for** an element\n8.Show\n9.Exit\n");
28. printf("\nEnter your choice?\n");
29. scanf("\n%d",&choice);
30. **switch**(choice)
31. {
32. **case** 1:
33. beginsert();
34. **break**;
35. **case** 2:
36. lastinsert();
37. **break**;
38. **case** 3:
39. randominsert();
40. **break**;
41. **case** 4:
42. begin\_delete();
43. **break**;
44. **case** 5:
45. last\_delete();
46. **break**;
47. **case** 6:
48. random\_delete();
49. **break**;
50. **case** 7:
51. search();
52. **break**;
53. **case** 8:
54. display();
55. **break**;
56. **case** 9:
57. exit(0);
58. **break**;
59. **default**:
60. printf("Please enter valid choice..");
61. }
62. }
63. }
64. **void** beginsert()
65. {
66. struct node \*ptr;
67. **int** item;
68. ptr = (struct node \*) malloc(sizeof(struct node \*));
69. **if**(ptr == NULL)
70. {
71. printf("\nOVERFLOW");
72. }
73. **else**
74. {
75. printf("\nEnter value\n");
76. scanf("%d",&item);
77. ptr->data = item;
78. ptr->next = head;
79. head = ptr;
80. printf("\nNode inserted");
81. }
83. }
84. **void** lastinsert()
85. {
86. struct node \*ptr,\*temp;
87. **int** item;
88. ptr = (struct node\*)malloc(sizeof(struct node));
89. **if**(ptr == NULL)
90. {
91. printf("\nOVERFLOW");
92. }
93. **else**
94. {
95. printf("\nEnter value?\n");
96. scanf("%d",&item);
97. ptr->data = item;
98. **if**(head == NULL)
99. {
100. ptr -> next = NULL;
101. head = ptr;
102. printf("\nNode inserted");
103. }
104. **else**
105. {
106. temp = head;
107. **while** (temp -> next != NULL)
108. {
109. temp = temp -> next;
110. }
111. temp->next = ptr;
112. ptr->next = NULL;
113. printf("\nNode inserted");
115. }
116. }
117. }
118. **void** randominsert()
119. {
120. **int** i,loc,item;
121. struct node \*ptr, \*temp;
122. ptr = (struct node \*) malloc (sizeof(struct node));
123. **if**(ptr == NULL)
124. {
125. printf("\nOVERFLOW");
126. }
127. **else**
128. {
129. printf("\nEnter element value");
130. scanf("%d",&item);
131. ptr->data = item;
132. printf("\nEnter the location after which you want to insert ");
133. scanf("\n%d",&loc);
134. temp=head;
135. **for**(i=0;i<loc;i++)
136. {
137. temp = temp->next;
138. **if**(temp == NULL)
139. {
140. printf("\ncan't insert\n");
141. **return**;
142. }
144. }
145. ptr ->next = temp ->next;
146. temp ->next = ptr;
147. printf("\nNode inserted");
148. }
149. }
150. **void** begin\_delete()
151. {
152. struct node \*ptr;
153. **if**(head == NULL)
154. {
155. printf("\nList is empty\n");
156. }
157. **else**
158. {
159. ptr = head;
160. head = ptr->next;
161. free(ptr);
162. printf("\nNode deleted from the begining ...\n");
163. }
164. }
165. **void** last\_delete()
166. {
167. struct node \*ptr,\*ptr1;
168. **if**(head == NULL)
169. {
170. printf("\nlist is empty");
171. }
172. **else** **if**(head -> next == NULL)
173. {
174. head = NULL;
175. free(head);
176. printf("\nOnly node of the list deleted ...\n");
177. }
179. **else**
180. {
181. ptr = head;
182. **while**(ptr->next != NULL)
183. {
184. ptr1 = ptr;
185. ptr = ptr ->next;
186. }
187. ptr1->next = NULL;
188. free(ptr);
189. printf("\nDeleted Node from the last ...\n");
190. }
191. }
192. **void** random\_delete()
193. {
194. struct node \*ptr,\*ptr1;
195. **int** loc,i;
196. printf("\n Enter the location of the node after which you want to perform deletion \n");
197. scanf("%d",&loc);
198. ptr=head;
199. **for**(i=0;i<loc;i++)
200. {
201. ptr1 = ptr;
202. ptr = ptr->next;
204. **if**(ptr == NULL)
205. {
206. printf("\nCan't delete");
207. **return**;
208. }
209. }
210. ptr1 ->next = ptr ->next;
211. free(ptr);
212. printf("\nDeleted node %d ",loc+1);
213. }
214. **void** search()
215. {
216. struct node \*ptr;
217. **int** item,i=0,flag;
218. ptr = head;
219. **if**(ptr == NULL)
220. {
221. printf("\nEmpty List\n");
222. }
223. **else**
224. {
225. printf("\nEnter item which you want to search?\n");
226. scanf("%d",&item);
227. **while** (ptr!=NULL)
228. {
229. **if**(ptr->data == item)
230. {
231. printf("item found at location %d ",i+1);
232. flag=0;
233. }
234. **else**
235. {
236. flag=1;
237. }
238. i++;
239. ptr = ptr -> next;
240. }
241. **if**(flag==1)
242. {
243. printf("Item not found\n");
244. }
245. }
247. }
248. **void** display()
249. {
250. struct node \*ptr;
251. ptr = head;
252. **if**(ptr == NULL)
253. {
254. printf("Nothing to print");
255. }
256. **else**
257. {
258. printf("\nprinting values . . . . .\n");
259. **while** (ptr!=NULL)
260. {
261. printf("\n%d",ptr->data);
262. ptr = ptr -> next;
263. }
264. }
265. }