6/24/24, 7:33 PM to do list.c

E:\to do list.c

```
1 // C program for the above approach
   #include <stdio.h>
 3
   #include <stdlib.h>
 4
 5
   // Renaming structure to avoid the
 6
   // repetitive use of struct keyword
 7
   typedef struct ToDo todo;
8
9
   // Declaration of structure
   struct ToDo {
10
11
        // char array as data part
        char buffer[101];
12
13
14
        // Pointer part to access addresses
15
        todo* next;
16
17
        // Count variable for counting
18
        // the number of nodes
19
        int count;
20
   };
21
   // Declare start pointer as null in
22
23
   // the beginning
24
   todo* start = NULL;
25
   // Driver Code
26
27
   int main()
28
    {
29
        int choice;
30
        interface();
31
32
        while (1) {
33
34
            // Change console color and
35
            // text color
            system("Color 3F");
36
37
            // Clear the console
38
39
            system("cls");
40
41
            printf("1. To see your ToDo list\n");
42
            printf("2. To create new ToDo\n");
            printf("3. To delete your ToDo\n");
43
            printf("4. Exit");
44
45
            printf("\n\n\nEnter your choice\t:\t");
46
47
            // Choice from the user
48
            scanf("%d", &choice);
49
50
            switch (choice) {
51
52
            // Calling functions defined
            // below as per the user input
53
54
            case 1:
```

```
6/24/24, 7:33 PM
 55
              seetodo();
 56
              break;
 57
           case 2:
              createtodo();
 58
 59
              break;
           case 3:
 60
              deletetodo();
 61
 62
              break;
 63
           case 4:
 64
              exit(1);
 65
              break;
           default:
 66
 67
              printf("\nInvalid Choice :-(\n");
              system("pause");
 68
 69
           }
 70
 71
        return 0;
 72
    }
 73
    // Code for Splash screen
 74
    void interface()
 75
 76
        system("color 4F");
 77
        printf("\n\n\n\n");
 78
        printf("\t~~~~~~~"
 79
           "~~~~~~"
 80
           "~~~~~~~"
 81
           "~~~~~~~"
 82
           83
        printf("\t~~~~~~~~~~~~~~"
 84
           "~~~~~~"
 85
           "~~~~~~~~~~~
 86
           "~~~~~~~"
 87
           "~~~~\n\n");
 88
 89
        printf("\t} : } : } : } : } "
           ": } : } : "
 90
           "WELCOME TO the TODO APP "
 91
             : { : { : { : { : { "
 92
 93
           ": { : { : {\n\n");
        printf("\t~~~~~~~~~"
 94
 95
           "~~~~~~"
           "~~~~~~~"
 96
           "~~~~~~
 97
           "~~~~\\n");
 98
        printf("\t~~~~~~~~~"
 99
           "~~~~~~"
100
           "~~~~~~~~
101
           "~~~~~~~"
102
           "~~~~\n");
103
        printf("\n\n\t\t\t\t\t\t\t\"
104
           "t\t\t
105
           "@Sushant_Gaurav\n\n\n"
106
107
           "\n\n\n\t");
108
109
        // Pausing screen until user
110
        // presses any key
```

```
111
         system("pause");
112
113
114
    // To view all the todos
115
    void seetodo()
116
117
         // Clearing the console
118
         system("cls");
119
         // Pointer to the node for traversal
120
         todo* temp;
121
122
123
         // temp is made to point the
124
         // start of linked list
125
         temp = start;
126
         // Condition for empty linked list
127
         if (start == NULL)
128
129
             printf("\n\nEmpty ToDo \n\n");
130
131
         // Traverse until last node
132
         while (temp != NULL) {
133
             // Print number of the node
134
             printf("%d.)", temp->count);
135
136
137
             // Print data of the node
138
             puts(temp->buffer);
139
140
             // Clear output console
141
             fflush(stdin);
142
             // Going to next node
143
144
             temp = temp->next;
145
         }
146
         printf("\n\n\n");
147
         system("pause");
148
149
     }
150
151
     // Function to insert a node todo
    void createtodo()
152
153
         // Choose choice from user
154
155
         char c;
156
157
         // Pointers to node
158
         todo *add, *temp;
         system("cls");
159
160
161
         // Infinite loop which will
         // break if "n" is pressed
162
163
         while (1) {
164
             printf("\nWant to add new ToDo ??"
165
                 + " Press 'y' for Yes and 'n' "
166
```

```
+ " for No :-)\n\t\t");
167
             fflush(stdin);
168
169
170
             // Input from user
171
             scanf("%c", &c);
172
173
             if (c == 'n')
174
                 break;
175
             else {
176
                 // If start node is NULL
177
                 if (start == NULL) {
178
179
180
                     // Dynamically allocating
181
                     // memory to the newly
182
                     // created node
                     add = (todo*)calloc(1, sizeof(todo));
183
184
                     // Using add pointer to
185
                     // create linked list
186
                     start = add;
187
188
                     printf("\nType it....\n");
189
                     // Input from user
190
191
                     fflush(stdin);
192
                     gets(add->buffer);
193
194
                     // As first input so
195
                     // count is 1
196
                     add->count = 1;
197
198
                     // As first node so
                     // start's next is NULL
199
200
                     start->next = NULL;
201
                 }
202
                 else {
203
                     temp = (todo*)calloc(1, sizeof(todo));
204
                     printf("\nType it....\n");
205
                     fflush(stdin);
206
                     gets(temp->buffer);
207
                     // Insertion is at last
208
209
                     // so pointer part is NULL
210
                     temp->next = NULL;
211
212
                     // add is now pointing
213
                     // newly created node
214
                     add->next = temp;
                     add = add->next;
215
216
                 }
217
218
                 // Using the concept of
219
                 // insertion at the end,
220
                 // adding a todo
221
                 // Calling function to adjust
222
```

6/24/24, 7:33 PM

```
6/24/24, 7:33 PM
 223
                   // the count variable
 224
                   adjustcount();
 225
              }
 226
          }
 227
      }
 228
 229
      // Function to delete the todo
 230
      void deletetodo()
 231
 232
          system("cls");
 233
          // To get the numbering of the
 234
 235
          // todo to be deleted
 236
          int x;
 237
 238
          todo *del, *temp;
          printf("\nEnter the ToDo's number"
 239
              + " that you want to remove.\n\t\t");
 240
 241
          // Checking empty condition
 242
          if (start == NULL)
 243
              printf("\n\nThere is no ToDo"
 244
 245
                  + " for today :-)\n\n\n");
 246
          else {
 247
              scanf("%d", &x);
 248
 249
              // del will point to start
 250
              del = start;
 251
 252
              // temp will point to start's
              // next so that traversal and
 253
              // deletion is achieved easily
 254
              temp = start->next;
 255
 256
 257
              // Running infinite loop so
 258
              // that user can delete and
 259
              // asked again for choice
 260
              while (1) {
 261
                  // When the values matches,
 262
 263
                  // delete the node
 264
                  if (del->count == x) {
 265
 266
                       // When the node to be
 267
                       // deleted is first node
 268
                       start = start->next;
 269
 270
                       // Deallocating the memory
                       // of the deleted node
 271
                       free(del);
 272
 273
                       // Adjusting the count when
 274
 275
                       // node is deleted
                       adjustcount();
 276
 277
                       break;
 278
                  }
```

```
279
280
                 if (temp->count == x) {
281
                      del->next = temp->next;
282
                      free(temp);
283
                      adjustcount();
                      break;
284
285
                 }
                 else {
286
287
                      del = temp;
                      temp = temp->next;
288
289
                 }
290
             }
291
         system("pause");
292
293
     }
294
295
     // Function to adjust the numbering
     // of the nodes
296
     void adjustcount()
297
298
     {
299
         // For traversal, using
300
         // a node pointer
         todo* temp;
301
302
         int i = 1;
303
         temp = start;
304
305
         // Running loop until last node
         // and numbering it one by one
306
         while (temp != NULL) {
307
             temp->count = i;
308
309
             i++;
310
             temp = temp->next;
311
         }
     }
312
313
```