



## Outline

fig12\_13.c (Part 1 of 7)

```
1 /* Fig. 12.13: fig12_13.c
2    operating and maintaining a queue */
3
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 /* self-referential structure */
8 struct queueNode {
9     char data;           /* define data as a char */
10    struct queueNode *nextPtr; /* queueNode pointer */
11 }; /* end structure queueNode */
12
13 typedef struct queueNode QueueNode;
14 typedef QueueNode *QueueNodePtr;
15
16 /* function prototypes */
17 void printQueue( QueueNodePtr currentPtr );
18 int isEmpty( QueueNodePtr headPtr );
19 char dequeue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr );
20 void enqueue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr,
21              char value );
22 void instructions( void );
23
24 /* function main begins program execution */
25 int main()
26 {
```



## Outline

fig12\_13.c (Part 2 of 7)

```
27 QueueNodePtr headPtr = NULL; /* initialize headPtr */
28 QueueNodePtr tailPtr = NULL; /* initialize tailPtr */
29 int choice; /* user's menu choice */
30 char item; /* char input by user */
31
32 instructions(); /* display the menu */
33 printf( "? " );
34 scanf( "%d", &choice );
35
36 /* while user does not enter 3 */
37 while ( choice != 3 ) {
38
39     switch( choice ) {
40
41         /* enqueue value */
42         case 1:
43             printf( "Enter a character: " );
44             scanf( "\\n%c", &item );
45             enqueue( &headPtr, &tailPtr, item );
46             printQueue( headPtr );
47             break;
48
49             /* dequeue value */
50             case 2:
51
```



## Outline

fig12\_13.c (Part 3 of 7)

```
52  /* if queue is not empty */
53  if ( !isEmpty( headPtr ) ) {
54      item = dequeue( &headPtr, &tailPtr );
55      printf( "%c has been dequeued.\n", item );
56  } /* end if */
57
58  printQueue( headPtr );
59  break;
60
61  default:
62      printf( "Invalid choice.\n\n" );
63      instructions();
64      break;
65
66  } /* end switch */
67
68  printf( "? " );
69  scanf( "%d", &choice );
70  } /* end while */
71
72  printf( "End of run.\n" );
73
74  return 0; /* indicates successful termination */
75
76  } /* end main */
77
```



## Outline

fig12\_13.c (Part 4 of 7)

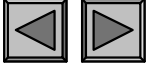
```
78 /* display program instructions to user */
79 void instructions( void )
80 {
81     printf ( "Enter your choice:\n"
82             "    1 to add an item to the queue\n"
83             "    2 to remove an item from the queue\n"
84             "    3 to end\n" );
85 } /* end function instructions */
86
87 /* insert a node a queue tail */
88 void enqueue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr,
89              char value )
90 {
91     QueueNodePtr newPtr; /* pointer to new node */
92
93     newPtr = malloc( sizeof( QueueNode ) );
94
95     if ( newPtr != NULL ) { /* is space available */
96         newPtr->data = value;
97         newPtr->nextPtr = NULL;
98
99         /* if empty, insert node at head */
100         if ( isEmpty( *headPtr ) ) {
101             *headPtr = newPtr;
102         } /* end if */
```



## Outline

### fig12\_13.c (Part 5 of 7)

```
103 else {
104     ( *tailPtr )->nextPtr = newPtr;
105 } /* end else */
106
107 *tailPtr = newPtr;
108 } /* end if */
109 else {
110     printf( "%c not inserted. No memory available.\n", value );
111 } /* end else */
112
113 } /* end function enqueue */
114
115 /* remove node from queue head */
116 char dequeue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr )
117 {
118     char value;          /* node value */
119     QueueNodePtr tempPtr; /* temporary node pointer */
120
121     value = ( *headPtr )->data;
122     tempPtr = *headPtr;
123     *headPtr = ( *headPtr )->nextPtr;
124
125     /* if queue is empty */
126     if ( *headPtr == NULL ) {
127         *tailPtr = NULL;
128     } /* end if */
129 }
```



## Outline

fig12\_13.c (Part 6 of 7)

```
130 free( tempPtr );
131
132 return value;
133
134 } /* end function dequeue */
135
136 /* Return 1 if the list is empty, 0 otherwise */
137 int isEmpty( QueueNodePtr headPtr )
138 {
139     return headPtr == NULL;
140 }
141 /* end function isEmpty */
142
143 /* Print the queue */
144 void printQueue( QueueNodePtr currentPtr )
145 {
146
147     /* if queue is empty */
148     if ( currentPtr == NULL ) {
149         printf( "Queue is empty.\n\n" );
150     } /* end if */
151     else {
152         printf( "The queue is:\n" );
153     }
```



## Outline

fig12\_13.c (Part 7 of 7)

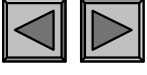
## Program Output (Part 1 of 2)

```
154  /* while not end of queue */
155  while ( currentPtr != NULL ) {
156      printf( "%c --> ", currentPtr->data );
157      currentPtr = currentPtr->nextPtr;
158  } /* end while */
159
160      printf( "NULL\n\n" );
161  } /* end else */
162
163 } /* end function printQueue */
```

```
Enter your choice:
  1 to add an item to the queue
  2 to remove an item from the queue
  3 to end
? 1
Enter a character: A
The queue is:
A --> NULL

? 1
Enter a character: B
The queue is:
A --> B --> NULL

? 1
Enter a character: C
The queue is:
A --> B --> C --> NULL
```



## Outline

### Program Output (Part 2 of 2)

```
? 2
A has been dequeued.
The queue is:
B --> C --> NULL

? 2
B has been dequeued.
The queue is:
C --> NULL

? 2
C has been dequeued.
Queue is empty.

? 2
Queue is empty.

? 4
Invalid choice.

Enter your choice:
  1 to add an item to the queue
  2 to remove an item from the queue
  3 to end
? 3
End of run.
```