

~~1~~ queue.cpp n/s p 662

~~1~~ #include <stdio.h>

~~2~~ #include <stdlib.h>

~~3~~ typedef struct node

~~4~~ {

~~5~~ void * dataPtr;

~~6~~ struct node * next;

~~7~~ } QUEUE_NODE;

~~8~~ typedef struct

~~9~~ {

~~10~~ QUEUE_NODE * front;

~~11~~ QUEUE_NODE * rear;

~~12~~ int count;

~~13~~ } QUEUE;

~~14~~ QUEUE * createQueue(void);

~~15~~ bool enqueue(QUEUE * queue, void * itemPtr);

~~16~~ void printQueue(QUEUE * stack);

~~17~~ int main(void)

~~18~~ {

~~19~~ QUEUE * queue1;

~~20~~ QUEUE * queue2;

~~21~~ QUEUE * queue3;

~~22~~ int * numPtr;

~~23~~ int ** itemPtr;

~~24~~ queue1 = createQueue();

~~25~~ queue2 = createQueue();

~~26~~ queue3 = createQueue();

~~27~~ int i = 11;

~~28~~ numPtr = (int *) malloc(sizeof(i));

~~29~~ *numPtr = i;

~~30~~ enqueue(queue1, numPtr);

```
317 i = 22;
327 numPtr = (int*) malloc(sizeof(i));
337 *numPtr = i;
347 enqueue(queue1, numPtr);
357 i = 33;
367 numPtr = (int*) malloc(sizeof(i));
377 *numPtr = i;
387 enqueue(queue1, numPtr);
397 i = 44;
407 numPtr = (int*) malloc(sizeof(i));
417 *numPtr = i;
427 enqueue(queue2, numPtr);
437 i = 55;
447 numPtr = (int*) malloc(sizeof(i));
457 *numPtr = i;
467 enqueue(queue3, numPtr);
477 i = 66;
487 numPtr = (int*) malloc(sizeof(i));
497 *numPtr = i;
507 enqueue(queue2, numPtr);
517 i = 77;
527 numPtr = (int*) malloc(sizeof(i));
537 *numPtr = i;
547 enqueue(queue3, numPtr);
557 i = 88;
567 numPtr = (int*) malloc(sizeof(i));
577 *numPtr = i;
587 enqueue(queue3, numPtr);
597 i = 99;
607 numPtr = (int*) malloc(sizeof(i));
617 *numPtr = i;
```



```
69* enqueue(queue3, newPtr);
```

```
69* printf("Queue 1:\n");
```

```
64* printfQueue(queue1);
```

```
66* printf("Queue 2:\n");
```

```
66* printfQueue(queue2);
```

```
67* printf("Queue 3:\n");
```

```
68* printfQueue(queue3);
```

```
69* return 0;
```

```
70* }
```

```
71* QUEUE* createQueue(void)
```

```
72* {
```

```
73* QUEUE* queue;
```

```
74* queue = (QUEUE*) malloc(sizeof(QUEUE));
```

```
75* if(queue)
```

```
76* {
```

```
77* queue->front = NULL;
```

```
78* queue->rear = NULL;
```

```
79* queue->count = 0;
```

```
80* }
```

```
81* return queue;
```

```
82* }
```

```
83* bool enqueue(QUEUE* queue, void* itemPtr)
```

```
84* {
```

```
85* QUEUE_NODE* newPtr = (QUEUE_NODE*) malloc(sizeof(QUEUE_NODE));
```

```
86* newPtr->dataPtr = itemPtr;
```

```
87* newPtr->next = NULL;
```

```
88* if(queue->count == 0)
```

```
89* queue->front = newPtr;
```

```
90* else
```

```

91* queue->rear->next = newPtr;
92* (queue->count)++;
93* queue->rear = newPtr;
94* return true;
95* }
96* QUEUE* destroyQueue(QUEUE* queue)
97* {
98*     QUEUE_NODE* deletePtr;
99*     if (queue)
100*     {
101*         while (queue->front != NULL)
102*         {
103*             free(queue->front->dataPtr);
104*             deletePtr = queue->front;
105*             queue->front = queue->front->next;
106*             free(deletePtr);
107*         }
108*         free(queue);
109*     }
110*     return NULL;
111* }
112* void printQueue(QUEUE* queue)
113* {
114*     QUEUE_NODE* node = queue->front;
115*     printf("Front = ");
116*     while (node)
117*     {
118*         printf("%d ", *(int*)node->dataPtr);
119*         node = node->next;
120*     }

```

12 ~~1~~ Print($\ell \leq \text{Rear}$);

12 ~~2~~ return;

12 ~~3~~ }