Aashita desai
SE 13
15

EXP_7

AIM : Write a program to build packages and modules for stack and queue data structures.

THEORY :

A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop

The functions associated with stack are:

- empty() – Returns whether the stack is empty – Time Complexity: O(1)
- size() – Returns the size of the stack – Time Complexity: O(1)
- top() – Returns a reference to the topmost element of the stack – Time Complexity: O(1)
- push(a) – Inserts the element 'a' at the top of the stack – Time Complexity: O(1)
- pop() – Deletes the topmost element of the stack – Time Complexity: O(1)

Python Stack Implementation

Stack in Python can be implemented using the following ways:

- list
- Collections.deque
- queue.LifoQueue

- As a stack, the queue is a linear data structure that stores items in a First In First Out (FIFO) manner. With a queue, the least recently added item is removed first. A good example of the queue is any queue of consumers for a resource where the consumer that came first is served first. Operations associated with queue are:

- Enqueue: Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition – Time Complexity: O(1)
- Dequeue: Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition – Time Complexity: O(1)
- Front: Get the front item from queue – Time Complexity: O(1)
- Rear: Get the last item from queue – Time Complexity: O(1)

Python queue Implementation

Queue in Python can be implemented in the following ways:

- list
- collections.deque
- queue.Queue

Aashita desai
SE 13
15

CODE :

```
# stack using queue module

from queue import LifoQueue
stack = LifoQueue(maxsize=3)
print(stack.qsize())
stack.put('a')
stack.put('b')
stack.put('c')
print("Full: ", stack.full())
print("Size: ", stack.qsize())
print('\nElements popped from the stack')
print(stack.get())
print(stack.get())
print(stack.get())

print("\nEmpty: ", stack.empty())

# queue using queue module

from queue import Queue
q = Queue(maxsize = 3)
print(q.qsize())
q.put('a')
q.put('b')
q.put('c')
print("\nFull: ", q.full())

print("\nElements dequeued from the queue")
print(q.get())
print(q.get())
print(q.get())


print("\nEmpty: ", q.empty())

q.put(1)
print("\nEmpty: ", q.empty())
print("Full: ", q.full())
```

Aashita desai
SE 13
15

OUTPUT :

```
========================================= RESTART: C:/Users/Aashita Desai/Downloads/2_Engineering/4_SEM/4_Python/Q-mod.py ========
0
Full:  True
Size:  3

Elements popped from the stack
c
b
a

Empty:  True
0

Full:  True

Elements dequeued from the queue
a
b
c

Empty:  True

Empty:  False
Full:  False
```

CONCLUSION : Program to build packages and modules for stack and queue data structures was implemented.