Тестирование программного обеспечения

Лабораторная работа №1

Тестирование класса очереди

В приложении 1 приведена реализация класса очереди целых чисел фиксированного размера согласно следующей спецификации:

* Конструктор принимает один аргумент: целое число > 0, которое соответствует максимальному размеру очереди;
* Empty() – метод возвращает True, если очередь не содержит ни одного элемента, иначе False;
* Full() – возвращает True если очередь содержит максимально возможное число элементов, иначе False;
* Enqueue(i) – добавляет целое число i в очередь, если число успешно добавлено возвращает True, если очередь содержит максимальное число элементов возвращает False;
* Dequeue() – удаляет элемент из очереди и возвращает его, если очередь пуста возвращает None.

Требуется написать модульные тесты для тестирования очереди согласно описанной спецификации.

Требуется написать рандомизированный тест для тестирования очереди согласно описанной спецификации. Можно использовать следующий шаблон:

for t in range(100):

random.seed(t)

q = Queue(random.randint(1, 1000))

for i in range(100):

r = random.randint(1, 1000)

method = r % 4

q.checkrep()

if method == 0:

q.empty()

elif method == 1:

q.full()

elif method == 2:

q.enqueue(r)

elif method == 3:

q.dequeue()

Приложение 1

# CORRECT SPECIFICATION:

#

# the Queue class provides a fixed-size FIFO queue of integers

#

# the constructor takes a single parameter: an integer > 0 that

# is the maximum number of elements the queue can hold.

#

# empty() returns True if and only if the queue currently

# holds no elements, and False otherwise.

#

# full() returns True if and only if the queue cannot hold

# any more elements, and False otherwise.

#

# enqueue(i) attempts to put the integer i into the queue; it returns

# True if successful and False if the queue is full.

#

# dequeue() removes an integer from the queue and returns it,

# or else returns None if the queue is empty.

#

# Example:

# q = Queue(1)

# is\_empty = q.empty()

# succeeded = q.enqueue(10)

# is\_full = q.full()

# value = q.dequeue()

#

# 1. Should create a Queue q that can only hold 1 element

# 2. Should then check whether q is empty, which should return True

# 3. Should attempt to put 10 into the queue, and return True

# 4. Should check whether q is now full, which should return True

# 5. Should attempt to dequeue and put the result into value, which

# should be 10

#

# Your test function should run assertion checks and throw an

# AssertionError for each of the 5 incorrect Queues. Pressing

# run will tell you how many you successfully catch so far.

import array

class Queue:

def \_\_init\_\_(self, size\_max):

assert size\_max > 0

self.max = size\_max

self.head = 0

self.tail = 0

self.size = 0

self.data = array.array('i', range(size\_max))

def empty(self):

return self.size == 0

def full(self):

return self.size == self.max

def enqueue(self, x):

if self.size == self.max:

return False

self.data[self.tail] = x

self.size += 1

self.tail += 1

if (self.tail == self.max):

self.tail = 0

return True

def dequeue(self):

if (self.size == 0):

return None

x = self.data[self.head]

self.head += 1

self.size -= 1

if (self.head == self.max):

self.head = 0

return x

def checkrep(self):

assert self.size >= 0 and self.size <= self.max

if self.tail > self.head:

assert (self.tail - self.head) == self.size

if self.tail < self.head:

assert (sefl.max - self.head + self.tail) == self.size

if (self.head == self.tail):

assert self.size == 0 or self.size == self.max

def test1():

q = Queue(3)

if not q.empty():

print "test1 FAILED"

return

print "test1 PASSED"

def test2():

print "test2 PASSED"

def test3():

print "test3 PASSED"

def test4():

print "test4 PASSED"

def test5():

print "test5 PASSED"

def main():

test1()

test2()

test3()

test4()

test5()

if \_\_name\_\_ == '\_\_main\_\_':

main()