Лабораторная работа №2

Вариант 16

Салихов А.Г. гр. Б04-191-1з

1.Постановка задачи.

Очередь

Структуры на базе массива должны содержать следующие операции:

**1)** Инициализация **2)** Вставка элемента (целочисленное значение):

2.1) Очередь – вставка в конец очереди

В случае переполнения внутреннего буфера, структура данных должна увеличить размер буфера в 2 раза (создать новый буфер, скопировать существующие элементы и удалить старый буфер)

1. Извлечение элемента

2.1) Очередь – извлечение из начала очереди

Операция должна возвращать код ошибки, в случае если в структуре данных нет элементов (например, функция возвращает false в случае, если нет элементов, и true - если элемент был извлечен, сам элемент в таком случае возвращается через параметр по ссылке)

1. Клонирование (создание копии структуры данных со всеми заполненным на текущий момент элементами)
2. Освобождение памяти от структуры данных

2.Исходный код решения

#include <iostream>

using namespace std;

struct Queue {

int\* arr = NULL; int size = 0; int start = 0;

int end = 0;

bool init(int bufferSize)

{

if (arr) return false; if (bufferSize < 1)

return false;

arr = new int[(unsigned int)bufferSize + 1]; size = bufferSize + 1;

return true;

}

bool enqueue(int val)

{

if (((end + 1) % size) == start) return false; arr[end] = val; end++; end = end % size;

return true;

}

bool dequeue(int\* arg) { arg = new int; if (start == end)

{

\*arg = -999; return false;

}

\*arg = arr[start]; start++;

start = start % size;

return true;

}

Queue\* clone() {

Queue\* queue = new Queue(); queue->arr = new int[size]; queue->size = size; queue->start = start;

queue->end = end;

for (int i = 0; i < size; i++)

{

queue->arr[i] = arr[i];

}

return queue;

}

void removeQueueFromMemory() { size = 0; end = 0;

start = 0; delete[] arr;

arr = NULL;

}

void print() {

int s = start; int e = end; if (s == e) {

cout << "Queue is empty." << endl;

}

while (s != e) { cout << arr[s] << " ";

s++;

s = s % size;

}

cout << endl;

}

};

int main() {

Queue q;

q.init(10);

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

q.enqueue(4);

q.print();

int\* arg = new int; q.dequeue(arg);

q.print();

Queue\* clone = q.clone();

q.removeQueueFromMemory();

q.print(); clone->print();

}

# 3.Контрольный пример

