/// Andrew Souza

/// Comp 210 -- Spring 2024

/// Stacks & Queues Assignment -- Problem 1

#include <iostream>

using namespace std;

// 1) Implement the Deque ADT using an array based implementation.

class DequeArray {

private:

unsigned int data;

unsigned int size;

unsigned int allocationSize;

unsigned int\* array;

unsigned int firstIndex;

void resize() {

allocationSize \*= 2;

unsigned int\* tempArr = new unsigned int[allocationSize];

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

tempArr[i] = array[i + firstIndex];

}

delete[] array;

array = tempArr;

firstIndex = 0;

}

public:

DequeArray() {

data = 0;

size = 0;

allocationSize = 1;

array = new unsigned int[allocationSize];

firstIndex = 0;

};

~DequeArray() {

delete[] array;

}

void PushBack(int x) {

if (size == allocationSize) {

resize();

}

for (int i = size; i >= 1; i--) {

array[i] = array[i - 1];

}

array[firstIndex] = x;

size += 1;

} // Inserts x at the front of the deque

void PushFront(int x) {

if (size == allocationSize) {

resize();

}

array[size] = x;

size++;

} // Inserts x at the back of the deque

int PopBack() {

unsigned int poppedVal = array[firstIndex];

firstIndex++;

size--;

return poppedVal;

} // Returns and removes item at front of deque

int PopFront() {

size--;

return array[size];

} // Returns and removes item at back of deque

int PeekFront() {

return array[size - 1];

} // Returns but does not remove the item at the front of deque

int PeekBack() {

return array[firstIndex];

} // Returns but does not remove the item at the back of deque

bool isEmpty() {

return (size == 0);

} // Returns true if the deque is empty

int GetLength() {

return size;

} // Returns the number of items in the deque

};

void PrintMenu() {

cout << "Peek the Front: A";

cout << "\nPeek the Back: B";

cout << "\nPop the Front: C";

cout << "\nPop the Back: D";

cout << "\nPush to the Front: E";

cout << "\nPush to the Back: F";

cout << "\nReturn the length: G";

cout << "\nInput: ";

}

int main() {

DequeArray userDeque;

unsigned int userSize;

cout << "Enter a deque size: ";

cin >> userSize;

for (int i = 0; i < userSize; i++) {

int input;

cout << "\nEnter a digit: ";

cin >> input;

userDeque.PushBack(input);

}

char input;

PrintMenu();

cin >> input;

while (input == 'A' || input == 'B' ||

input == 'C' || input == 'D' ||

input == 'E' || input == 'F' ||

input == 'G') {

int tempInt;

switch (input) {

case 'A':

cout << "\nFront: " << userDeque.PeekFront();

break;

case 'B':

cout << "\nBack: " << userDeque.PeekBack();

break;

case 'C':

cout << "\nPop Front: " << userDeque.PopFront();

break;

case 'D':

cout << "\nPop Back: " << userDeque.PopBack();

break;

case 'E':

cout << "\nInput digit: ";

cin >> tempInt;

userDeque.PushFront(tempInt);

break;

case 'F':

cout << "\nInput digit: ";

cin >> tempInt;

userDeque.PushBack(tempInt);

break;

case 'G':

cout << "\nLength: ";

if (!userDeque.isEmpty()) {

cout << userDeque.GetLength();

} else {

cout << "\nEmpty";

}

default:

break;

}

cout << "\nPress M to view the menu or press any command: ";

cin >> input;

if (input == 'M') {

PrintMenu();

cin >> input;

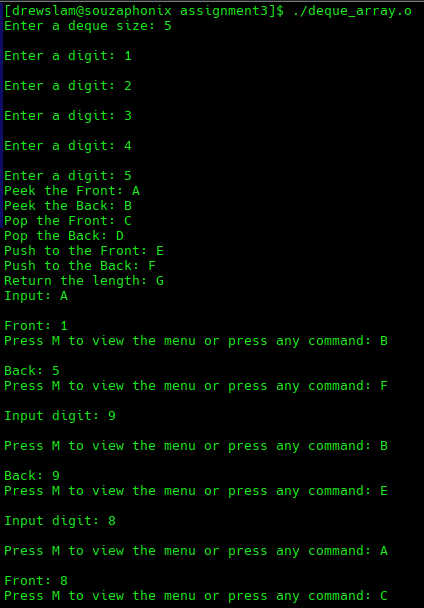
}

}

cout << endl;

return 0;

}



/// Andrew Souza

/// Stacks & Queues Assignment

/// LinkedList.h

#ifndef DOUBLY\_LINKED\_LIST

#define DOUBLY\_LINKED\_LIST

class Node {

public:

int data;

Node\* next = nullptr;

Node\* previous = nullptr;

Node(int data) {

this->data = data;

}

};

class LinkedList {

public:

Node\* head = nullptr;

Node\* tail = nullptr;

void Append(Node\* newNode) { // Checks if head is null

if (head == nullptr) { // If head is null then head

head = newNode; // and tail are both assigned

tail = newNode; // to newNode

head->next = tail;

tail->previous = head;

tail->next = nullptr;

} else {

tail->next = newNode; // If head is not null then

newNode->previous = tail;

tail = newNode; // newNode is appended to

tail->next = nullptr; // the end of the list

}

}

void Prepend(Node\* newNode) {

if (head != nullptr) { // If head is not null then

newNode->next = head; // newNode is pushed to the front

head->previous = newNode; // of the list.

head = newNode;

} else {

head = newNode; // If head is null then head

tail = newNode; // is assigned to newNode

head->next = tail;

tail->previous = head;

tail->next = nullptr;

}

}

};

#endif

/// Andrew Souza

/// Comp 210 -- Spring 2024

/// Assignment 3 -- Problem 2

#include <iostream>

#include "LinkedList.h"

using namespace std;

// 2) Implement the Deque ADT using a Linked List based implementation.

class DequeLinkedList : public LinkedList {

private:

unsigned int length;

public:

DequeLinkedList() {

head = nullptr;

tail = nullptr;

length = 0;

}

~DequeLinkedList() {

}

void PushFront(int x) {

Node\* temp = new Node(x);

Append(temp);

length++;

}

void PushBack(int x) {

Node\* temp = new Node(x);

Prepend(temp);

length++;

}

int PopFront() {

int poppedVal = tail->data;

tail = tail->previous;

tail->next = nullptr;

length--;

return poppedVal;

}

int PopBack() {

int poppedVal = head->data;

if (head->next) {

head->next = (head->next->next);

head->next->previous = head;

}

head = head->next;

length--;

return poppedVal;

}

int PeekFront() {

return tail->data;

}

int PeekBack() {

return head->data;

}

bool isEmpty() {

return (length == 0);

}

int GetLength() {

return length;

}

};

void PrintMenu() {

cout << "Peek the Front: A";

cout << "\nPeek the Back: B";

cout << "\nPop the Front: C";

cout << "\nPop the Back: D";

cout << "\nPush to the Front: E";

cout << "\nPush to the Back: F";

cout << "\nReturn the length: G";

cout << "\nInput: ";

}

int main() {

DequeLinkedList userDeque;

unsigned int userSize;

cout << "Enter a deque size: ";

cin >> userSize;

for (int i = 0; i < userSize; i++) {

int input;

cout << "\nEnter a digit: ";

cin >> input;

userDeque.PushBack(input);

}

char input;

PrintMenu();

cin >> input;

while (input == 'A' || input == 'B' ||

input == 'C' || input == 'D' ||

input == 'E' || input == 'F' ||

input == 'G') {

int tempInt;

switch (input) {

case 'A':

cout << "\nFront: " << userDeque.PeekFront();

break;

case 'B':

cout << "\nBack: " << userDeque.PeekBack();

break;

case 'C':

cout << "\nPop Front: " << userDeque.PopFront();

break;

case 'D':

cout << "\nPop Back: " << userDeque.PopBack();

break;

case 'E':

cout << "\nInput digit: ";

cin >> tempInt;

userDeque.PushFront(tempInt);

break;

case 'F':

cout << "\nInput digit: ";

cin >> tempInt;

userDeque.PushBack(tempInt);

break;

case 'G':

cout << "\nLength: ";

if (!userDeque.isEmpty()) {

cout << userDeque.GetLength();

} else {

cout << "\nEmpty";

}

default:

break;

}

cout << "\nPress M to view the menu or press any command: ";

cin >> input;

if (input == 'M') {

PrintMenu();

cin >> input;

}

}

cout << endl;

return 0;

}

