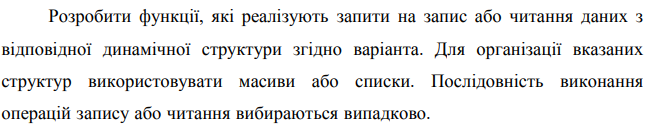
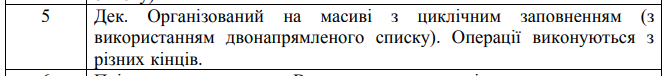
**Лабораторна робота №3**

Розробка та реалізація програм для роботи зі стеком, чергою, деком

**Мета роботи**: набуття практичних навичок опрацювання таких динамічних структур, як стек, черга, дек.

*Завдання*





**Хід роботи**

1. **Текст програми**

***main.cpp:***

#include <iostream>

#include <string>

#include "StaticDeque.hpp"

#include "DinamicDeque.hpp"

#include "Deque.hpp"

void printHello();

void createDeque();

void runCommands();

void printHr();

void cliHelp(), cliFront(), cliBack(), cliPushFront();

void cliPushBack(), cliPopFront(), cliPopBack();

dg::Deque\* deque;

int main() {

printHello();

createDeque();

runCommands();

return 0;

}

void runCommands() {

cliHelp();

while(true) {

std::string command = prompt<std::string>("> ");

if (command == "front") cliFront();

else if (command == "back") cliBack();

else if (command == "pushFront") cliPushFront();

else if (command == "pushBack") cliPushBack();

else if (command == "popFront") cliPopFront();

else if (command == "popBack") cliPopBack();

else if (command == "help") cliHelp();

else if (command == "exit") break;

else {

std::cout << "Wrong command. Try again" << std::endl;

}

}

}

void createDeque() {

std::string type = prompt<std::string>("Select Deque type\n(if 1 - static, otherwise - dinamic): ");

if (type == "1") {

int capacity = prompt<unsigned short>("Enter capacity: ");

deque = new dg::StaticDeque(capacity);

}

else {

deque = new dg::DinamicDeque;

}

}

void cliHelp() {

std::cout << "Commands:" << std::endl

<< "> front #print first item" << std::endl

<< "> back #pring last item" << std::endl

<< "> pushFront #add item to front" << std::endl

<< "> pushBach #add item to back" << std::endl

<< "> popFront #remove firstElement" << std::endl

<< "> popBack #remove lastElement" << std::endl

<< std::endl

<< "> help #print help" << std::endl

<< "> exit #close the program" << std::endl

<< std::endl;

}

void cliFront() {

std::cout << deque->front() << std::endl;

}

void cliBack() {

std::cout << deque->back() << std::endl;

}

void cliPushFront() {

int value = prompt<int>("Enter item value: ");

try {

deque->pushFront(value);

std::cout << "Successfully" << std::endl;

}

catch (int err) {

std::cout << "Can't add the item to the Deque" << std::endl;

}

}

void cliPushBack() {

int value = prompt<int>("Enter item value: ");

try {

deque->pushBack(value);

std::cout << "Successfully" << std::endl;

}

catch (std::runtime\_error err) {

std::cout << "Can't add the item to the Deque" << std::endl;

}

}

void cliPopFront() {

deque->popFront();

std::cout << "Successfully" << std::endl;

}

void cliPopBack() {

deque->popBack();

std::cout << "Successfully" << std::endl;

}

***Deque.hpp:***

#pragma once

namespace dg {

class Deque {

public:

virtual int front() = 0;

virtual int back() = 0;

virtual void pushFront(int value) = 0;

virtual void pushBack(int value) = 0;

virtual void popFront() = 0;

virtual void popBack() = 0;

virtual int size() = 0;

};

}

***DinamicDeque.hpp:***

#pragma once

#include "List.hpp"

#include "Deque.hpp"

namespace dg {

class DinamicDeque : public Deque {

public:

DinamicDeque();

int front();

int back();

void pushFront(int value);

void pushBack(int value);

void popFront();

void popBack();

int size();

void clear();

~DinamicDeque();

private:

List m\_list;

};

}; // namespace dg

***DinamicDeque.cpp:***

#include "DinamicDeque.hpp"

namespace dg {

DinamicDeque::DinamicDeque() {

}

int DinamicDeque::front() {

auto item = m\_list.front();

return (item) ? item->value : 0;

}

int DinamicDeque::back() {

auto item = m\_list.back();

return (item) ? item->value : 0;

}

void DinamicDeque::pushFront(int value) {

m\_list.pushFront(value);

}

void DinamicDeque::pushBack(int value) {

m\_list.pushBack(value);

}

void DinamicDeque::popFront() {

m\_list.popFront();

}

void DinamicDeque::popBack() {

m\_list.popBack();

}

int DinamicDeque::size() {

return m\_list.size();

}

void DinamicDeque::clear() {

m\_list.clear();

}

DinamicDeque::~DinamicDeque() {

}

}; // namespace dg

***StaticDeque.hpp:***

#pragma once

#include <iostream>

#include <vector>

#include "Deque.hpp"

namespace dg {

class StaticDeque: public Deque {

public:

StaticDeque(unsigned int capacity = 16);

int front();

int back();

void pushFront(int value);

void pushBack(int value);

void popFront();

void popBack();

int size();

int capacity();

~StaticDeque();

private:

int m\_size = 0;

int m\_capacity;

std::vector<int> m\_items;

int m\_front = 0;

int m\_back = 0;

int \_toIndex(int index);

void \_checkCanAdd();

};

}; // namespace dg

***StaticDeque.cpp:***

#include "StaticDeque.hpp"

namespace dg {

StaticDeque::StaticDeque(unsigned int capacity):

m\_capacity(capacity),

m\_items(capacity)

{

}

int StaticDeque::front() {

return m\_items[m\_front];

}

int StaticDeque::back() {

return m\_items[m\_back];

}

void StaticDeque::pushFront(int value) {

\_checkCanAdd();

if (m\_size != 0) {

m\_front = \_toIndex(m\_front - 1);

}

m\_size++;

m\_items[m\_front] = value;

}

void StaticDeque::pushBack(int value) {

\_checkCanAdd();

if (m\_size != 0) {

m\_back = \_toIndex(m\_back + 1);

}

m\_size++;

m\_items[m\_back] = value;

}

void StaticDeque::popFront() {

if (m\_size == 0) return;

m\_size--;

if (m\_size == 0) return;

m\_front = \_toIndex(m\_front + 1);

}

void StaticDeque::popBack() {

if (m\_size == 0) return;

m\_size--;

if (m\_size == 0) return;

m\_back = \_toIndex(m\_back - 1);

}

int StaticDeque::size() {

return m\_size;

}

int StaticDeque::capacity() {

return m\_capacity;

}

inline

int StaticDeque::\_toIndex(int index) {

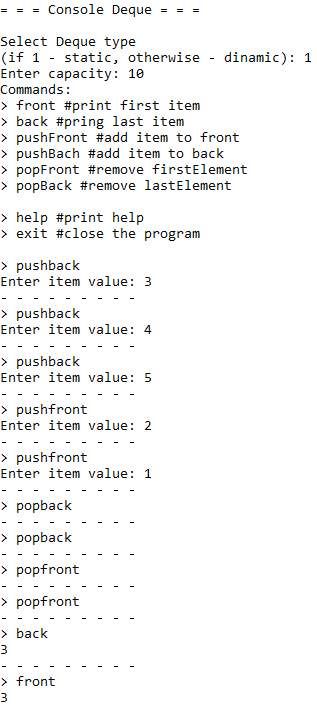
if (index < 0) index = m\_capacity + index;

return index % m\_capacity;

}

}

1. **Результат виконання програми**



**Висновок:** на цій лабораторній роботі я набув практичні навички опрацювання таких динамічних структур, як стек, черга, дек.