МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ»

(МТУСИ)

Кафедра «Информационная безопасность»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9

по дисциплине

«Программирование в системах информационной безопасности»

на тему

«Классы с динамическими структурами данных»

Вариант 15

Выполнил:

студент группы БСУ1801

Матяева В.И.

Проверил:

старший преподаватель кафедры ИБ

Барков В.В.

Москва, 2021

# Цель работы

Изучить динамические структуры данных, овладеть навыками создания конструкторов копирования, перемещения, деструкторов, перегрузки операций копирования и перемещения.

# Задание

Для типа динамической структуры данных, указанного в индивидуальном задании, разработать соответствующий класс, предусмотрев в нем конструкторы инициализации, копирования, перемещения, деструктор, функции вставки и удаления элемента, просмотра доступного элемента и функцию, проверяющую наличие элементов.

Перегрузить операции присваивания, присваивание с перемещением и потокового вывода для вывода содержимого динамической структуры на экран.

Обязательные функции-члены:

|  |  |  |
| --- | --- | --- |
| Стек | Очередь | Дек |
| int GetSize() const;  bool Push(const T &element);  bool Pop(T &element);  bool Peek(T &element); | int GetSize();  bool Push(const T &element);  bool Pop(T &element);  bool Peek(T &element); | int GetSize() const;  bool PushFront(const T &element);  bool PopFront(T &element);  bool PushBack(const T &element);  bool PopBack(T &element);  bool PeekFront(T &element) const;  bool PeekBack(T &element) const; |

Функции Push, PushFront и PushBack должны возвращать true в случае успешного добавления и false в случае нехватки места.

Функции Pop, PopBack, PopFront, Peek, PeekBack, PeekFront, должны возвращать true и копию элемента через параметр element, передаваемый по ссылке, в случае если в контейнере есть элементы и false, в случае если в контейнере нет элементов.

**Индивидуальное задание**

**Вариант 15**

|  |  |  |
| --- | --- | --- |
| 15 | Дек на основе двунаправленного циклического списка | DequeBasedOnBidirectionalCyclicLinkedList |

# Выполнение

Листинг 1 – программный код файла Deque.cpp

#include "Header.h"

Deque::Deque()

{

head = tail = nullptr;

count = 0;

}

Deque::Deque(int a)

{

head = tail = new Node();

head->prev = tail;

head->next = tail;

head->data = a;

}

Deque& Deque::operator=(Deque &&other)

{

if (this == &other)

{

return \*this;

}

delete head;

head = other.head;

other.head = nullptr;

}

int Deque::GetSize()

{

return count;

}

bool Deque::pushFront(const int a)

{

Node\* newNode = new Node();

newNode->data = a;

if (head == nullptr)

{

head = tail = newNode;

}

else

{

newNode->next = head;

head->prev = tail->next = newNode;

head = newNode;

}

count++;

return true;

};

bool Deque::pushBack(const int a)

{

Node\* newNode = new Node();

newNode->data = a;

if (tail == nullptr)

{

head = tail = newNode;

}

else

{

newNode->prev = tail;

tail->next = head->prev = newNode;

tail = newNode;

}

count++;

return true;

};

bool Deque::popFront(int& a)

{

if (head == nullptr)

{

return false;

}

else

{

a = head->data;

Node\* temp = head;

head = head->next;

if (head == nullptr) {

tail = nullptr;

}

else

{

head->prev = tail;

tail->next = head;

}

count--;

return true;

}

};

bool Deque::popBack(int& a)

{

if (head == nullptr)

{

return false;

}

else

{

a = tail->data;

Node\* temp = tail;

tail = tail->prev;

if (tail == nullptr) {

head = nullptr;

}

else

{

tail->next = head;

head->prev = tail;

}

count--;

return true;

}

};

bool Deque::peekFront(int& a)

{

if (head == nullptr)

{

return false;

}

else

{

a = head->data;

return true;

}

}

bool Deque::peekBack(int& a)

{

if (head == nullptr)

{

return false;

}

else

{

a = tail->data;

return true;

}

}

Листинг 2 – программный код файла Header.h

struct Node

{

int data;

Node\* prev, \* next;

};

class Deque

{

public:

Deque();

Deque(int count);

Deque& operator=(Deque&& other);

int GetSize();

bool pushFront(const int a);

bool pushBack(const int a);

bool popFront(int& a);

bool popBack(int& a);

bool peekFront(int& a);

bool peekBack(int& a);

private:

Node\* head;

Node\* tail;

int count;

};

Листинг 3 – программный код файла main.cpp

#include "Header.h"

#include <conio.h>

#include<iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "RUS");

int b, dsize;

bool flag = true;

Deque a;

while (flag)

{

cout << "1. Добавить элемент в начало дека\n";

cout << "2. Получить элемент из начала дека\n";

cout << "3. Добавить элемент в конец дека\n";

cout << "4. Получить элемент из конца дека\n";

cout << "5. Посмотреть элемент в конец дека\n";

cout << "6. Посмотреть элемент из конца дека\n";

cout << "7. Выход\n";

switch (\_getch())

{

case '1':

cout << "Добавить элемент: ";

cin >> b;

a.pushFront(b) ? cout << "Элемент добавлен в начало" << endl : cout << "Нет места\n";

break;

case '2':

a.popFront(b) ? cout << "Полученный элемент " << b << endl : cout << "Дек пуст\n";

break;

case '3':

cout << "Добавить элемент: ";

cin >> b;

a.pushBack(b) ? cout << "Элемент добавлен в конец " << endl : cout << "Нет места\n";

break;

case '4':

a.popBack(b) ? cout << "Полученный элемент " << b << endl : cout << "Дек пуст\n";

break;

case '5':

a.peekFront(b) ? cout << "Полученный элемент " << b << endl : cout << "Дек пуст\n";

break;

case '6':

a.peekBack(b) ? cout << "Полученный элемент " << b << endl : cout << "Дек пуст\n";

break;

case '7':

flag = false;

break;

}

}

}

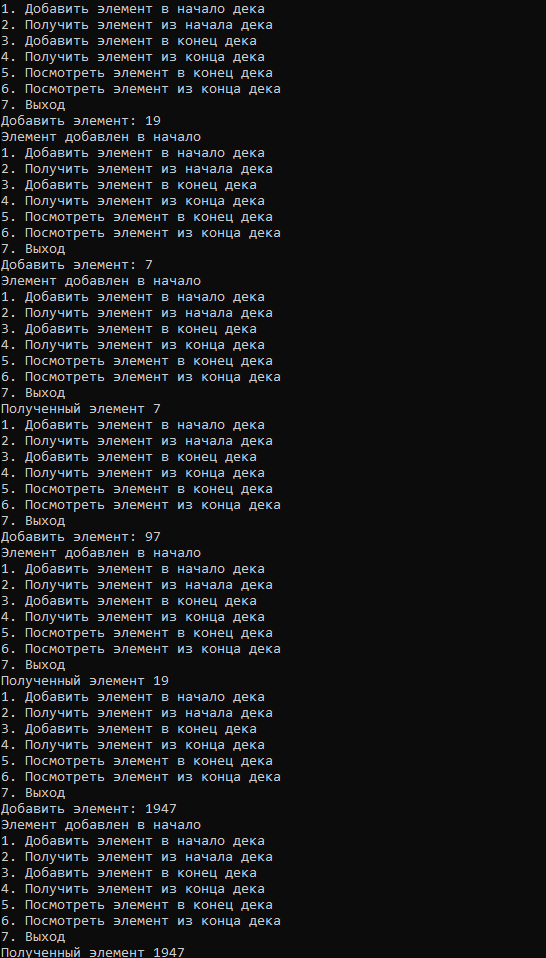


Рисунок 1 – Результат работы с программой.