# Практическая работа № 5

**ДЕКИ**

**Постановка задачи**

Составить программу создания дека на базе линейного двусвязного списка и реализовать основные алгоритмы работы с деком, обеспечивающие следующие действия:

**1 - Добавить элемент**

* 1. – Слева
  2. – Справа

**2 – Вывести элемент (с удалением)**

       2.1 – Слева

       2.2 – Справа

**3 – Вывести элемент (без удаления)**

       2.1 – Слева

       2.2 – Справа

**4 - Вычислить длину дека**

**5 - Вывести (распечатать) дек на экран**

Перечисленные действия оформить в виде самостоятельных режимов работы созданного дека. Выбор режимов производить с помощью пользовательского меню.

Провести полное тестирование (всех режимов работы) программы на стеке, сформированном вводом с клавиатуры. Тест-примеры определить самостоятельно. Результаты тестирования в виде скриншотов экранов включить в отчет по выполненной работе.

Оформить отчет с подробным описанием созданного стека, принципов программной реализации алгоритмов работы со стеком, описанием текста исходного кода и проведенного тестирования программы.

Сделать выводы о проделанной работе, основанные на полученных результатах.

1. **Описание алгоритма**

Алгоритм программы состоит из функции main и вызываемых в ней вспомогательных функций:

* **void add\_left** – функция добавления элемента в дек слева
* **void add\_right** – функция добавления элемента в дек справа
* **int pop\_left** – функция извлечения элемента слева (удаление)
* **int pop\_right** – функция извлечения элемента справа (удаление)
* **int peek\_left** – функция возврата элемента слева (без удаления)
* **int peek\_right** – функция возврата элемента справа (без удаления)
* **void show –** функция вывода дека на экран
* **int len –** функция вычисления длинны дека

Двусвязная очередь (жарг. дэк, дек от англ. deque — double ended queue; двусторонняя очередь, очередь с двумя концами) — абстрактный тип данных, в котором элементы можно добавлять и удалять как в начало, так и в конец. Может быть реализована при помощи двусвязного списка.

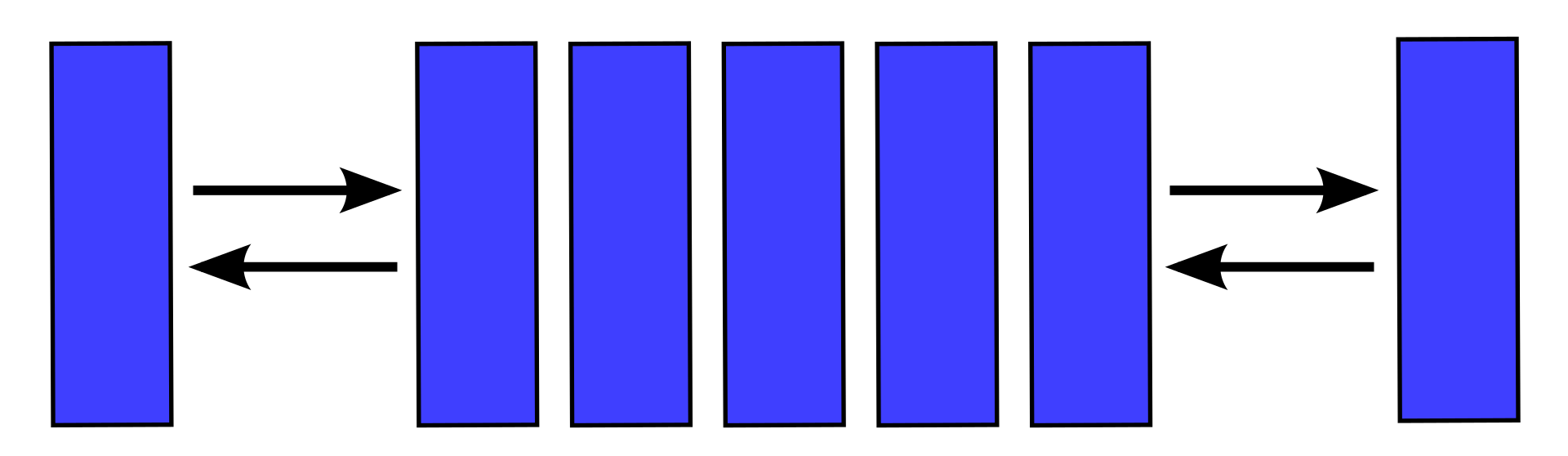


Рис.1 Принцип работы дека

dd

Функция main создает класс Stack и вызывает функцию menu.

Функция peek\_left возвращает значение левого элемента.

Функция peek\_right возвращает значение правого элемента.

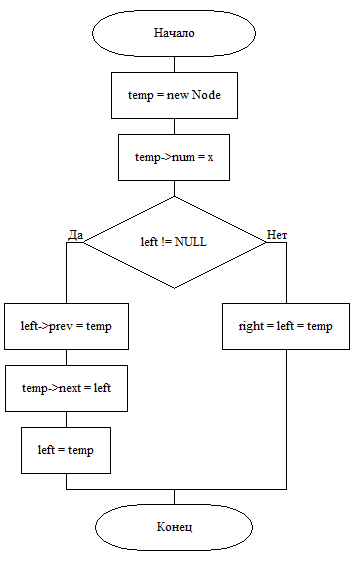


Рис.2 Схема алгоритма функции add\_left

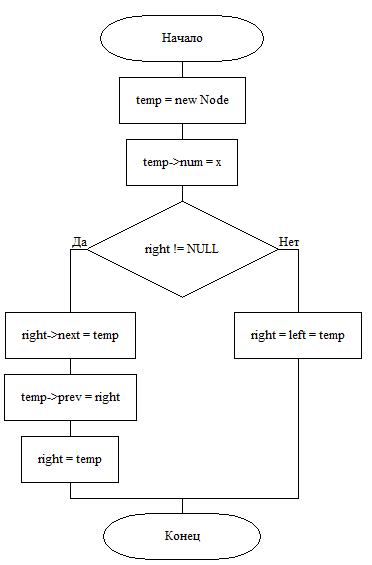


Рис.3 Схема алгоритма функции add\_right

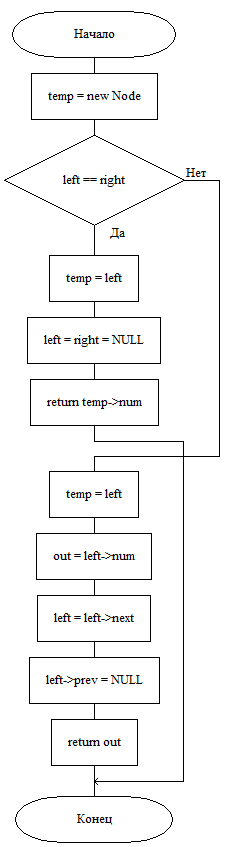


Рис.4 Схема алгоритма функции pop\_left

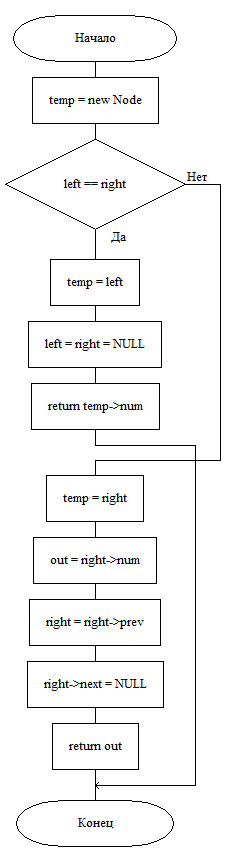


Рис.5 Схема алгоритма функции pop\_right

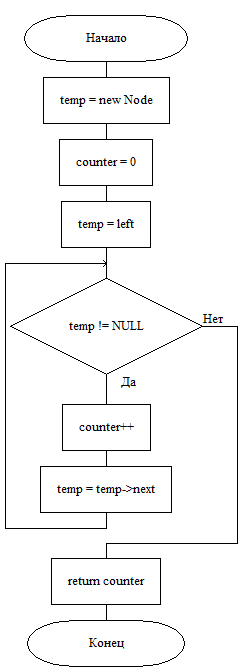


Рис.6 Схема алгоритма функции len

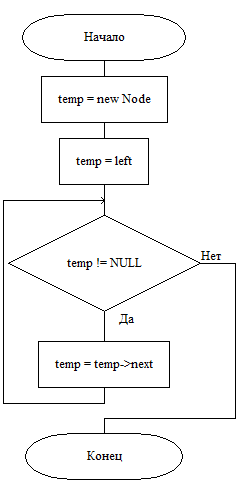


Рис.7 Схема алгоритма функции show

**Реализация алгоритма**

**Текст исходного кода программы**

**main.cpp**

#include "Deque.h"

void menu(Deque dec) {

cin.ignore();

cout << "Выберите команду:" << endl;

cout << "[1] - Добавить элемент." << endl;

cout << "[2] - Вывести элемент. (с удалением)" << endl;

cout << "[3] - Вывести элемент. (без удаления)" << endl;

cout << "[4] - Вывести длинну дека." << endl;

cout << "[5] - Вывести дек." << endl;

cout << "[6] - Завершить программу." << endl;

cout << "---->";

int ch = 0;

cin >> ch;

if (ch == 1) {

cout << "\_\_\_\_[1] - Добавить слева." << endl;

cout << "\_\_\_\_[2] - Добавить справа." << endl;

cout << "\_\_\_\_[3] - Назад." << endl;

cout << "-------->";

int ch2 = 0;

cin >> ch2;

if (ch2 == 1) {

int x = 0;

cout << "Введите число." << endl;

cout << "-------->";

cin >> x;

dec.add\_left(x);

menu(dec);

}

else if (ch2 == 2) {

int x = 0;

cout << "Введите число." << endl;

cout << "-------->";

cin >> x;

dec.add\_right(x);

menu(dec);

}

else if (ch2 == 3) {

menu(dec);

}

}

else if (ch == 2)

{

cout << "\_\_\_\_[1] - Вывести слева." << endl;

cout << "\_\_\_\_[2] - Вывести справа." << endl;

cout << "\_\_\_\_[3] - Назад." << endl;

cout << "-------->";

int ch2 = 0;

cin >> ch2;

if (ch2 == 1) {

cout << dec.pop\_left()<<endl;

menu(dec);

}

else if (ch2 == 2) {

cout << dec.pop\_right() << endl;

menu(dec);

}

else if (ch2 == 3) {

menu(dec);

}

}

else if (ch == 3)

{

cout << "\_\_\_\_[1] - Вывести слева." << endl;

cout << "\_\_\_\_[2] - Вывести справа." << endl;

cout << "\_\_\_\_[3] - Назад." << endl;

cout << "-------->";

int ch2 = 0;

cin >> ch2;

if (ch2 == 1) {

cout << dec.peek\_left() << endl;

menu(dec);

}

else if (ch2 == 2) {

cout << dec.peek\_right() << endl;

menu(dec);

}

else if (ch2 == 3) {

menu(dec);

}

}

else if (ch == 4)

{

cout << dec.len() << endl;

menu(dec);

}

else if (ch == 5)

{

dec.show();

menu(dec);

}

else if (ch == 6)

{

return;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

Deque deque;

menu(deque);

}

**Deque.h**

#include <iostream>

#pragma once

using namespace std;

struct Node {

int num = 0;

Node\* prev = NULL;

Node\* next = NULL;

};

class Deque

{

private:

Node \* left, \* right;

public:

Deque() : left(NULL), right(NULL) {}

void add\_left(int x);

void add\_right(int x);

int pop\_left();

int pop\_right();

int peek\_left();

int peek\_right();

int len();

void show();

};

**Deque.cpp**

#include "Deque.h"

void Deque::add\_left(int x) {

Node\* temp = new Node;

temp->num = x;

if (left != NULL) {

left->prev = temp;

temp->next = left;

left = temp;

}

else {

right = left = temp;

}

}

void Deque::add\_right(int x) {

Node\* temp = new Node;

temp->num = x;

if (right != NULL) {

right->next = temp;

temp->prev = right;

right = temp;

}

else {

right = left = temp;

}

}

int Deque::pop\_left() {

Node\* temp = new Node;

if (left == right) {

temp = left;

left = right = NULL;

return temp->num;

}

temp = left;

int out = left->num;

left = left->next;

left->prev = NULL;

delete temp;

return out;

}

int Deque::pop\_right() {

Node\* temp = new Node;

if (left == right) {

temp = left;

left = right = NULL;

return temp->num;

}

temp = right;

int out = right->num;

right = right->prev;

right->next = NULL;

delete temp;

return out;

}

int Deque::peek\_left() {

return left->num;

}

int Deque::peek\_right() {

return right->num;

}

int Deque::len() {

Node\* temp = new Node;

int counter = 0;

temp = left;

while (temp != NULL) {

counter++;

temp = temp->next;

}

return counter;

}

void Deque::show() {

Node\* temp = new Node;

temp = left;

while (temp != NULL) {

cout << temp->num<<endl;

temp = temp->next;

}

}

1. **Тестирование программы**

Ниже представлен результат работы программы с введённым деком

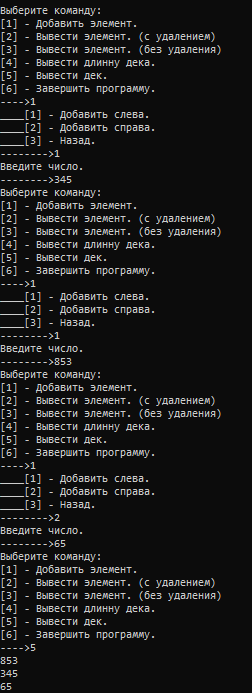


Рис.8 Скриншот добавления элементов в дек и вывод дека

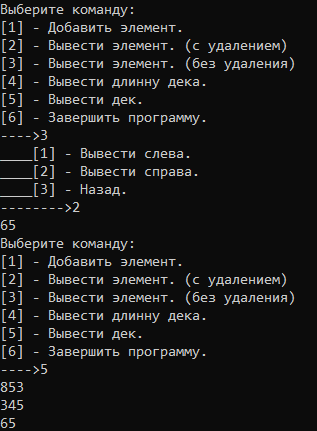


Рис.9 Скриншот вывода элемента справа без удаления и вывода дека

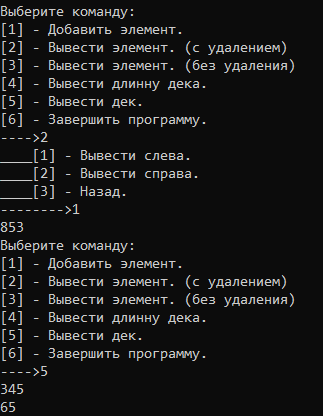


Рис.10 Скриншот вывода элемента справа с удалением и вывода дека

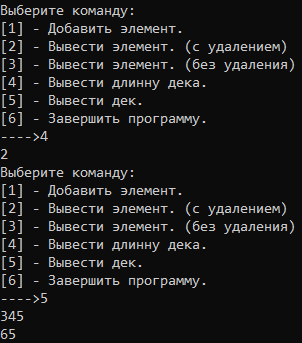


Рис.11 Скриншот вывода длинны дека и вывода дека

1. **Выводы**
2. В ходе работы был создан дек на основе двусвязного списка, ссылки между элементами которого осуществляются при помощи указателей на объекты класса Node.
3. Также были реализованы функции работы с деком: добавление элементов, удаление, вывод всех элементов списка, вывод длины.
4. Были изучены положительные и негативные стороны стека:
5. Преимущества: дек может хранить неограниченное количество данных любого типа.
6. Недостатки: элементы можно извлекать и вставлять только с концов дека, что усложняет взаимодействие с элементами.
7. Таким образом, была изучена работа дека на базе линейного двусвязного списка и функций работы над ними и их реализация.

**Список используемых информационных источников**

1. Сыромятников В.П. Структуры и алгоритмы обработки данных, лекции, РТУ МИРЭА, Москва, 2020/2021 уч./год.
2. Документация по языку программирования С++, интернет-ресурс: <https://en.cppreference.com/w/> (Дата обращения – 02.11.2020)
3. Интегрированная среда разработки для языков программирования C и C++, разработанная компанией JetBrains - CLion / Copyright © 2000-2020 JetBrains s.r.o., интернет-ресурс: <https://www.jetbrains.com/clion/learning-center/> (Дата обращения – 02.11.2020).
4. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. Интернет-ресурс: <http://docs.cntd.ru/document/gost-19-701-90-espd> (Дата обращения – 02.11.2020).
5. Описание Дека. интернет-ресурс: https://ru.wikipedia.org/wiki/Двухсторонняя\_очередь (Дата обращения – 02.11.2020).