МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

### федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент Л.Н. Бариков

Отчёт

по лабораторной работе №15

по дисциплине ОСНОВЫ ПРОГРАММИРОВАНИЯ

на тему: «Стек, дек, очередь»

Работу выполнил

студент гр. 4941 А.Ю. Дыкань

Санкт-Петербург

2020

**Цель лабораторной работы:** изучение способов создания и принципов использования динамических структур данных типа стек, дек, очередь; изучение стандартных средств языка C/C++ для работы с динамической памятью; совершенствование навыков модульного программирования на языке C/C++ при решении задач обработки динамических структур данных; совершенствование способов разработки многофайловых проектов.

**Задание на программирование:** используя технологию процедурного программирования разработать программу обработки данных, содержащихся в заранее подготовленном файле, в соответствии с индивидуальным заданием. Применить динамическую структуру указанного в задании вида: стек, очередь или дек.

**Порядок выполнения работы:**

1. Получить у преподавателя индивидуальное задание.

2. Разработать математическую модель: описать с помощью формул и рисунков вид используемой динамической структуры и процессы её создания и использования.

3. Построить схему алгоритма решения задачи.

4. Использовать функции, реализующие полный набор операций для этой структуры:

- допустимые операции для стека: инициализация, проверка на пустоту, добавление нового элемента в начало, извлечение элемента из начала;

- допустимые операции для очереди: инициализация, проверка на пустоту, добавление нового элемента в конец, извлечение элемента из начала;

- допустимые операции для дека: инициализация, проверка на пустоту, добавление нового элемента в начало, добавление нового элемента в конец, извлечение элемента из начала, извлечение элемента из конца.

5. Создать многофайловый проект на языке *C*/*C*++.

6. Проверить и продемонстрировать преподавателю работу программы на полном наборе тестов. Обеспечить одновременный показ на экране входной и выходной информации.

7. Выходные данные должны выводиться на экран с пояснениями. Операторы вывода результатов работы должны находиться либо в функции *main*(), либо в специальной функции вывода результата, вызов которой осуществляется из функции *main*().

8. Оформить отчет о лабораторной работе в составе: постановка задачи, математическая модель, схема алгоритма решения основной (основных) задач, текст программы, контрольные примеры.

**Вариант 35**

Дана произвольная строка. Под *словом* понимается последовательность букв алфавита. Между соседними словами - не менее одного пробела. Перед первым и за последним словом строки произвольное число пробелов.

Используя ***дек***, построить новую строку, превратив все слова исходной строки в палиндромы добавлением в начало каждого слова всех исходных символов этого слова в обратном порядке. Например:

- исходная строка: “Ehal Greka Cherez River”;

- результат: “lahEEhal akerGGreka zerehCCherez reviRRiver”.

**Текст программы**

Файл source.cpp:

#include <iostream>

#include <sstream>

#include "DEK\_fun.h"

#include <locale>

using namespace std;

int main()

{

dek\* head = NULL,

\* end = NULL;

string res;

string vvod;

setlocale(LC\_ALL, "Russian");

cout << "\n Введите исходную строку;"

"\n в конце - энтер:" << endl;

getline(cin, vvod);

cout << endl;

cout << " Исходное содержимое:" << endl << vvod << endl;

res = obrdek(vvod, &head, &end);

cout << "Результат работы программы:" << endl << res << endl;

}

Файл Dek\_fun.h:

#include <iostream>

#include <sstream>

using namespace std;

typedef char t\_elem; //тип информационной части элемента стека

struct dek //тип элемента стека

{

t\_elem data; //поле данных

dek\* linknext; //поле адреса

dek\* linkprev; //поле адреса

};

//Объявления (прототипы) функций

int pu\_dek(dek\*); //проверка на пустоту

void push\_beg(t\_elem ch, dek\*\* end, dek\*\* beg); //добавление элемента в начало

void push\_end(t\_elem ch, dek\*\* end, dek\*\* beg); //добавление элемента в конец

string obrdek(string s, dek\*\* beg, dek\*\* end); //обработка

void clear\_dek(dek\*\* beg, dek\*\* end);

Файл DEK\_fun.cpp

//Содержит определения функций для работы с деком

//

#include <iostream>

#include <sstream>

#include "DEK\_fun.H"

using namespace std;

//push\_beg\_\_Begin\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Занесение нового элемента в дек

void push\_beg(t\_elem ch, dek\*\* end, dek\*\* beg)

{

dek\* nov = new(dek); //выделяем память под элемент дека

nov->data = ch; //заполняем поля

nov->linkprev = NULL;

if (pu\_dek(\*end) && pu\_dek(\*beg)) //если дек пуст

{

nov->linknext = NULL;

\*beg = nov;

\*end = nov;

}

else

{

nov->linknext = \*beg;

(\*beg)->linkprev = nov; //подсоединяем новый элемент

\*beg = nov; //указатель на первый элемент дека

}

}

//push\_beg\_\_End\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

void push\_end(t\_elem ch, dek\*\* end, dek\*\* beg)

{

dek\* nov = new(dek); //выделяем память под элемент дека

nov->data = ch; //заполняем поля

nov->linknext = NULL;

if (pu\_dek(\*end) && pu\_dek(\*beg))//если дек пуст

{

nov->linkprev = NULL;

\*beg = nov;

\*end = nov;

}

else

{

nov->linkprev = \*end;

(\*end)->linknext = nov; //подсоединяем новый элемент

\*end = nov; //указатель на последний элемент дека

}

}

//pu\_dek\_\_Begin\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//Проверка дека на пустоту

int pu\_dek(dek\* head)

{

return(head == NULL);

}

//pu\_dek\_\_End\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

void clear\_dek(dek\*\* beg, dek\*\* end)

{

dek\* tec = (\*beg)->linknext;

while (tec != \*end)

{

dek\* buf = tec;

tec = tec->linknext;

delete buf;

}

\*beg = NULL;

\*end = NULL;

}

string obrdek(string s, dek\*\* beg, dek\*\* end)

{

string result;

string ch;

stringstream buf, res;

buf << s; //Ведденую стоку вносим в поток

while (buf >> ch)

{

for (int i = 0; i < ch.length(); i++)

{

push\_beg(ch[i], end, beg);

push\_end(ch[i], end, beg);

}

dek\* tec = \*beg;

while (tec != \*end) //пока дек не пуст

{

res << tec->data; //вводим в поток элементы стека

tec = tec->linknext;

}

res << (\*end)->data;

res << " ";

clear\_dek(beg, end);

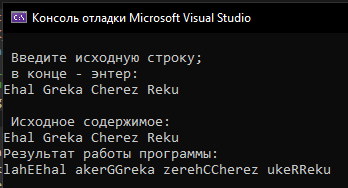
}

result = res.str(); //из потока в строку

return result;

}

**Скриншоты работы программы**



Вывод: используя технологию процедурного программирования разработал программу обработки данных, в соответствии с индивидуальным заданием. Применил динамическую структуру вида дек.