Министерство образования Республики Беларусь
Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет инфомационных технологий и управления
Кафедра интеллектуальных информационных технологий
Основы алгоритмизации и программирования

Отчёт по лабораторной работе №3

ДИНАМИЧЕСКАЯ СТРУКТУРА СТЕК

Студент гр. 321701 В. В. Перминова
Преподаватель С. И. Матюшкин

Минск 2023

Цель работы: изучить алгоритмы работы с динамическими структурами данных в виде стека.

Написать программу по созданию, добавлению, просмотру и решению приведенных далее задач (в рассмотренных примерах это действие отсутствует) для однонаправленного линейного списка типа Stack. Решение поставленной задачи описать в виде блок-схемы.

8. Перенести из созданного списка в новый список все элементы, находящиеся между вершиной и элементом с минимальным значением.

Код программы:

```
#include <iostream>
#include <ctime>

using namespace std;

struct Stack {
	int info;
	Stack* next;
} *top, *t;

struct N_Stack {
	int n_info;
	N_Stack* n_next;
} *k;

Stack* InStack(Stack* p, int in) {
	Stack* t = new Stack;
	t->info = in;
	t->next = p;
	return t;
}

void View(Stack* p) {
	Stack* t = p;
	while (t != NULL) {
		cout << " " << t->info << endl;
		t = t->next;
```

```
	}
}

void Del_All(Stack** p) {
	while (*p != NULL) {
		Stack* t = *p;
		*p = (*p)->next;
		delete t;
	}
}

N_Stack* New_Stack(Stack* p) {
	Stack* t = p->next;
	Stack* min = p;
	N_Stack* k = new N_Stack;
	while (t) {
		if (t->info < min->info) {
			min = t;
		}
		t = t->next;
	}
	if ((p == min) || (p->next == min)) {
		cout << "Элементов между вершиной и минимальным элементом -
		нет" << endl;
	}
	else {
		Stack* i = p->next;
		while (i->info != min->info) {
			k->n_info = i->info;
			cout << " " << k->n_info;
			i = i->next;
			k->n_next = new N_Stack;
		}
		cout << endl;
	}
	return k;
}

void main()
```

```
{
	setlocale(LC_ALL, "Russian");
	int i, in, n, kod;
	while (true) {
		cout << "\n\tСоздание - 1.\n\tДобавление - 2.\n\tПросмотр -
		3.\n\tУдаление - 4.\n\tВыполнение задания - 5.\n\tExit - 0. : ";
		cin >> kod;
		switch (kod) {
		case 1: case 2:
			if (kod == 1 && top != NULL) {
				cout << "Очистите память!" << endl;
				break;
			}
			cout << "Введите количество = ";
			cin >> n;
			srand(time(NULL));
			for (i = 1; i <= n; i++) {
				in = rand() % 41 - 20;
				top = InStack(top, in);
			}
			if (kod == 1)
				cout << "Создано " << n << " элементов" << endl;
			else
				cout << "Добавлено " << n << " элементов" << endl;
			break;
		case 3:
			if (!top) {
				cout << "Стек пуст!" << endl;
				break;
			}
			cout << "--- Стек ---" << endl;
			View(top);
			break;
		case 4:
			Del_All(&top);
			cout << "Память освобождена!" << endl;
			break;
		case 5:
```

```
                New_Stack(top);
                cout << "Данные элементы перенесены" << endl;
                break;
        case 0:
                if (top != NULL)
                        Del_All(&top);
                while (k != NULL) {
                        N_Stack* d = k;
                        k = k->n_next;
                        delete d;
                        }
                return;
        }
    }
}
```

Результат:

```
        Создание - 1.
        Добавление - 2.
        Просмотр - 3.
        Удаление - 4.
        Выполнение задания - 5.
        Exit - 0. : 1
Введите количество = 7
Создано 7 элементов

        Создание - 1.
        Добавление - 2.
        Просмотр - 3.
        Удаление - 4.
        Выполнение задания - 5.
        Exit - 0. : 3
--- Стек ---
 -2
 19
 -18
 -16
 13
 -9
 3

        Создание - 1.
        Добавление - 2.
        Просмотр - 3.
        Удаление - 4.
        Выполнение задания - 5.
        Exit - 0. : 5
 19
Данные элементы перенесены

        Создание - 1.
        Добавление - 2.
        Просмотр - 3.
        Удаление - 4.
        Выполнение задания - 5.
        Exit - 0. :
```

Вывод: в ходе данной лабораторной работы мы изучили алгоритмы работы с динамическими структурами данных в виде стека.