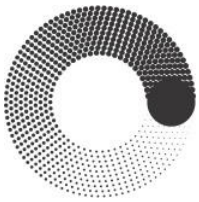
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 1

Дисциплина: «Основы современных алгоритмов»

Тема: «Программное моделирование стеков и очередей»

Выполнил: студент группы 211-723

Сергеев Станислав Олегович

Дата, подпись_____ _____
(Дата) (Подпись)

Проверил:_____ _____
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись_____ _____
(Дата) (Подпись)

Замечания: _____

_____

_____

Москва

2022

Программное моделирование стеков и очередей

Цель:

Получить практические навыки в использовании стеков и очередей в языке С.

Постановка задачи:

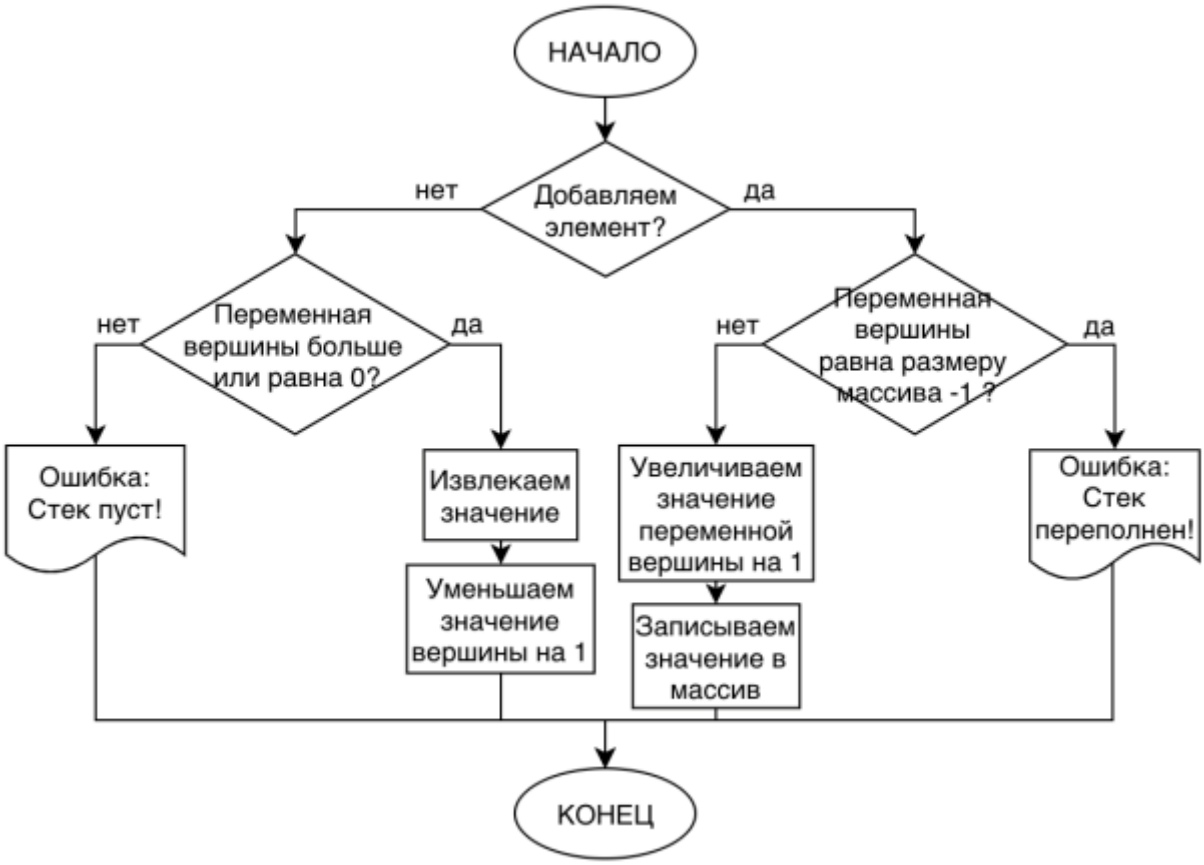
Необходимо разработать программу, реализующую работу со стеком очередью.
Программа должна содержать функции вставки, извлечения, контроля на переполнение, контроля на отсутствие элементов (пустой стек или очередь), отображения на экране значения извлекаемого элемента.

Стек

Идея алгоритма:

Стек - своего рода запоминающее устройство, из которого элементы извлекаются в порядке, обратном их добавлению. Это как бы неправильная очередь, в которой первым обслуживают того, кто встал в нее последним по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»).

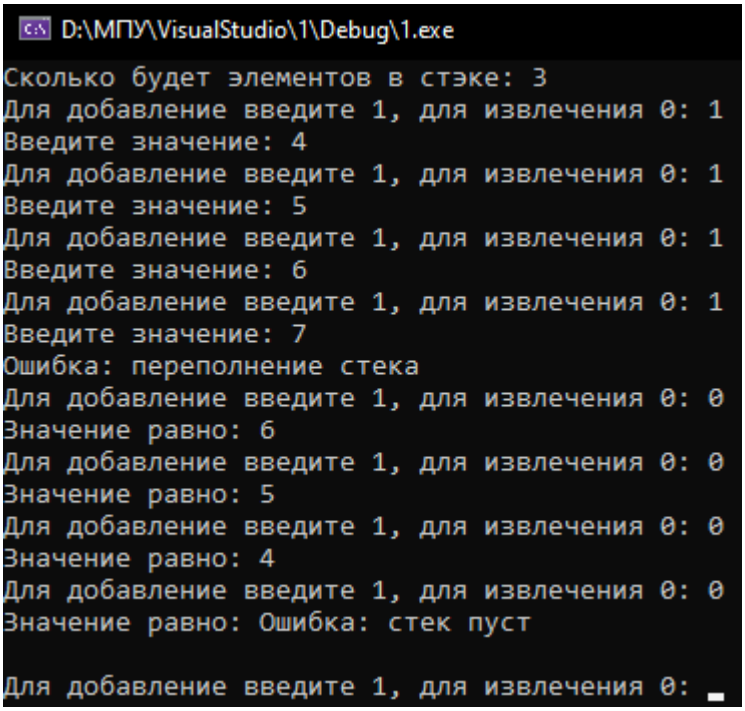
Блок схема:



Программная реализация стека на основе массива фиксированной длины:

```
#include <iostream>
using namespace std;
class stack
{
private:
	int* arr;
	int top;
	int N = 10;
public:
	class Full {};
	class Empty {};
	stack() //конструктор без параметров
	{
		top = -1;
		arr = new int[N];
	}
	stack(int m) //конструктор с одним параметром
	{
		top = -1;
		arr = new int[m];
		N = m;
	}
	~stack() //деструктор
	{
		delete[] arr;
	}
	void push(int a) //функция добавления элемента
	{
		if (top == N - 1) //значение вершины равно размеру массива(стека)?
			throw Full();
		else
		{
			top++;
			arr[top] = a; //увеличиваем значение вершины и кладем это значение в массив
		}
	}
	int pop() //функция извлечения элемента
	{
		if (top < 0) //переменная вершины больше либо равна 0
			throw Empty();
		return arr[top--];
	}
};
int main()
{
	setlocale(LC_ALL, "Russian");
	int znach, o;
	bool r; //укажем логический тип данных для проверки логических условий
	cout << "Сколько будет элементов в стэке: ";
	cin >> o;
	stack one(o);
	do {
		cout << "Для добавление введите 1, для извлечения 0: ";
		cin >> r;
		if (r)
		{
			cout << "Введите значение: ";
			cin >> znach;
			try
			{
				one.push(znach);
			}
			catch (stack::Full)
			{
				cout << "Ошибка: переполнение стека" << endl;
```

```
                }
        }
        else
        {
                cout << "Значение равно: ";
                try
                {
                        cout << one.pop();
                }
                catch (stack::Empty)
                {
                        cout << "Ошибка: стек пуст" << endl;
                }
                cout << endl;
        }
   } while (true);
   return 0;
}
```

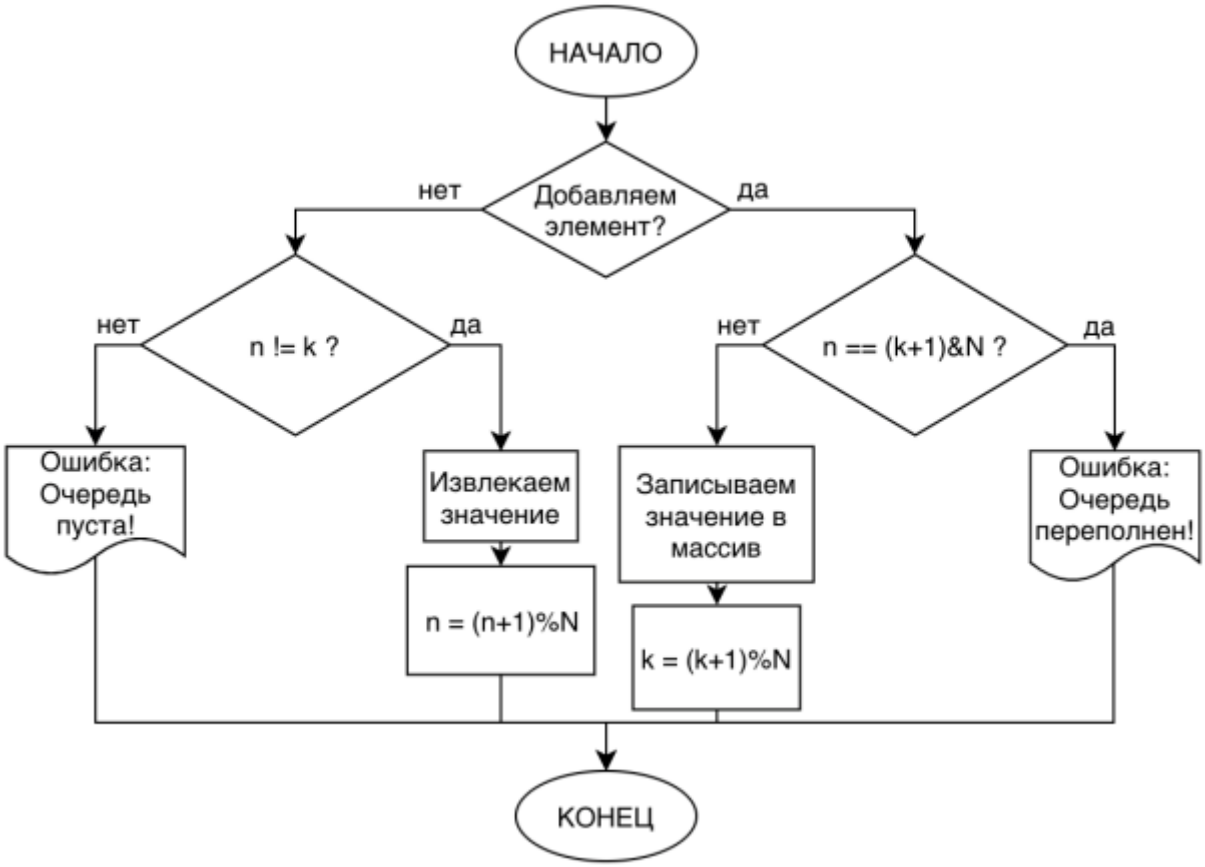


Очередь

Идея алгоритма:

Очередь - это структура данных, представляющая собой последовательность элементов, включение в которую производится с одного конца, а исключение – с другого конца (т.е. образованная в порядке их поступления).

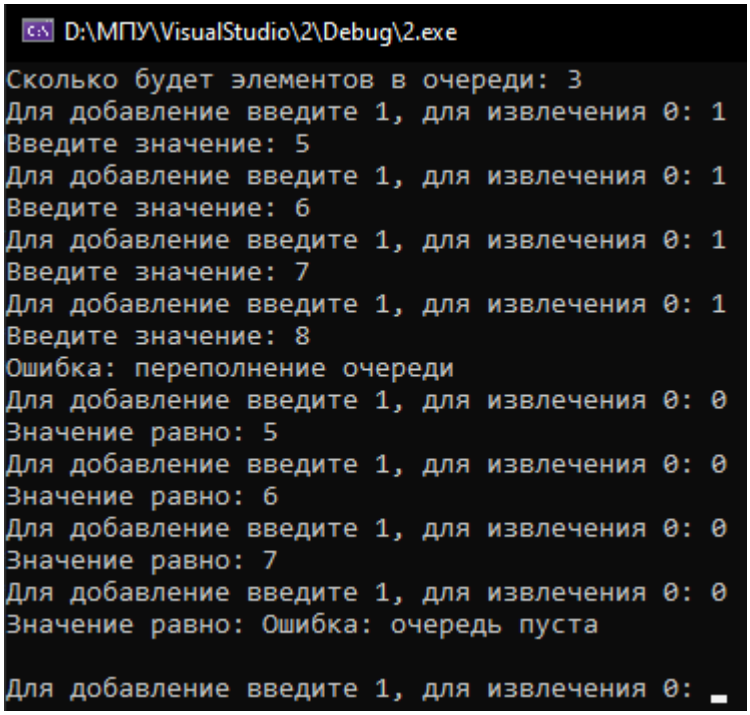
Блок схема:



Программная реализация стека на основе массива фиксированной длины:

```
#include <iostream>
using namespace std;
class queue
{
private:
	int* arr; //сделаем указатель на массив
	int head; //переменная для начала очереди(головы)
	int tail; //переменная для конца очереди(хвоста)
	int N = 10;
public:
	class Full {};
	class Empty {};
	queue()
	{
		head = 0;
		tail = 0;
		arr = new int[N];
	}
	queue(int m)
	{
		N = m + 1;
		head = 0;
		tail = 0;
```

```
		arr = new int[N];
	}
	~queue() //деструктор для освобождения памяти
	{
		delete[] arr;
	}
	void add_item(int a) //функция добавления элемента
	{
		if (head == (tail + 1) % N) //проверяем наличие места в очереди
			throw Full();
		else
		{
			arr[tail] = a; //записываем значение в массив
			tail = (tail + 1) % N; //присваиваем нашему “хвосту” номер последнего эл-та
		}
	}
	int extract_item() //функция извлечения элемента
	{
		if (head != tail) //проверяем наличие элементов в очереди
		{
			int a = arr[head]; //выводим на экран последний в очереди элемент
			head = (head + 1) % N; //присваиваем нашей “голове” номер следующего эл-та
			return a;
		}
		else
			throw Empty();
	}
};
int main()
{
	setlocale(LC_ALL, "Russian");
	int znach, o;
	bool r; //укажем логический тип данных для проверки логических условий
	cout << "Сколько будет элементов в очереди: ";
	cin >> o;
	queue one(o);
	do
	{
		cout << "Для добавление введите 1, для извлечения 0: ";
		cin >> r;
		if (r)
		{
			cout << "Введите значение: ";
			cin >> znach;
			try
			{
				one.add_item(znach);
			}
			catch (queue::Full)
			{
				cout << "Ошибка: переполнение очереди" << endl;
			}
		}
		else
		{
			cout << "Значение равно: ";
			try
			{
				cout << one.extract_item();;
			}
			catch (queue::Empty)
			{
				cout << "Ошибка: очередь пуста" << endl;
			}
			cout << endl;
		}
	} while (true);
	return 0;
}
```


D:\МПУ\VisualStudio\2\Debug\2.exe
Сколько будет элементов в очереди: 3
Для добавление введите 1, для извлечения 0: 1
Введите значение: 5
Для добавление введите 1, для извлечения 0: 1
Введите значение: 6
Для добавление введите 1, для извлечения 0: 1
Введите значение: 7
Для добавление введите 1, для извлечения 0: 1
Введите значение: 8
Ошибка: переполнение очереди
Для добавление введите 1, для извлечения 0: 0
Значение равно: 5
Для добавление введите 1, для извлечения 0: 0
Значение равно: 6
Для добавление введите 1, для извлечения 0: 0
Значение равно: 7
Для добавление введите 1, для извлечения 0: 0
Значение равно: Ошибка: очередь пуста
Для добавление введите 1, для извлечения 0: