Практическая работа №5

Структуры данных «стек» и «очередь»

Цель: Изучить сд «стек» и «очередь», а также программно реализовать их на языке С++

Стек характерен тем, что получить доступ к его элементам можно лишь с одного конца, называемого вершиной стека; иначе говоря: стек – структура данных типа «список», функционирующая по принципу LIFO (last in — first out, «последним пришёл — первым вышел»).

Стек, чаще всего, реализуется на основе обычных массивов, односвязных и двусвязных списков. В зависимости от конкретных условий, выбирается одна из этих структур данных.

Основными операциями над стеками являются:

-добавление элемента;

-удаление элемента;

-чтение верхнего элемента.

В языках программирования эти три операции, обычно дополняются и некоторыми другими.

Вот, например, список функций С++ для работы со стеком:

-push() – добавить элемент;

-pop() – удалить элемент;

-top() – получить верхний элемент;

-size() – размер стека;

-empty() – проверить стек на наличие элементов.

Код программы

```
#include "pch.h"
#include <iostream>
#include <stack>
using namespace std;

//главная функция
```

```
void main()
{
	setlocale(LC_ALL, "Russian");
	stack <int> S; //создание стека S типа int
	char number; int value;
	do
	{
		cout << "1. Добавить элемент" << endl;
		cout << "2. Удалить элемент" << endl;
		cout << "3. Получить верхний элемент" << endl;
		cout << "4. Узнать размер стека" << endl;
		cout << "0. Выйти" << endl;
		cout << "Номер команды > "; cin >> number;
		switch (number)
		{
		case '1': //добавление элемента
			cout << "Значение > "; cin >> value;
			S.push(value); cout << endl << "Элемент добавлен в стек\n\n";
			break;
			//-------------------------------------------
		case '2': //удаление элемента
			if (S.empty() == true) cout << "\nСтек пуст\n\n";
			else
			{
				S.pop(); cout << endl << "Элемент удален из стека\n\n";
			} break;
			//-------------------------------------------
		case '3': //вывод верхнего элемента
			if (S.empty() == true) cout << "\nСтек пуст\n\n";
			else cout << "\nВерхний элемент стека: " << S.top() << "\n\n";
			break;
			//-------------------------------------------
		case '4': //вывод размера стека
			if (S.empty() == true) cout << "\nСтек пуст\n\n";
			else cout << "\nРазмер стека: " << S.size() << "\n\n";
			break;
			//-------------------------------------------
		case '0': break; //выход
		default: cout << endl << "Команда не определенная\n\n";
			break;
		}
	} while (number != '0');

	system("pause");
}
```

Добавление элемента

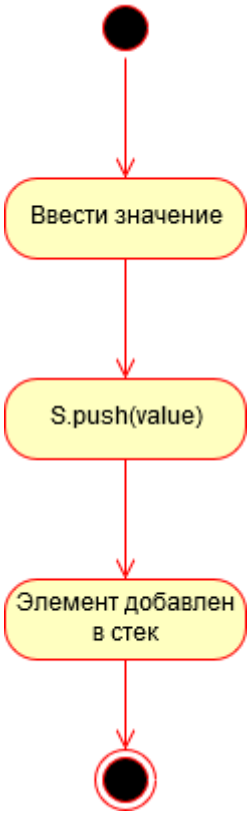

Рисунок 1 - Диаграмма деятельности добавление элемента в стек

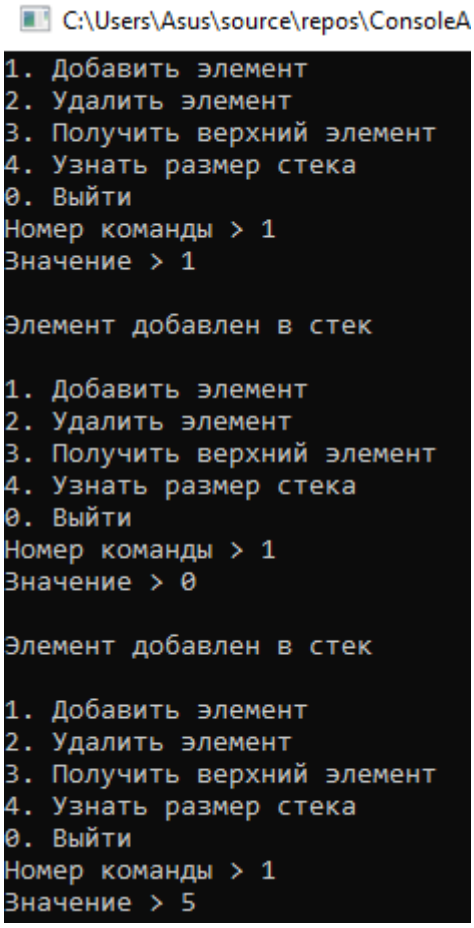

Рисунок 2 - Результат

Получить верхний элемент

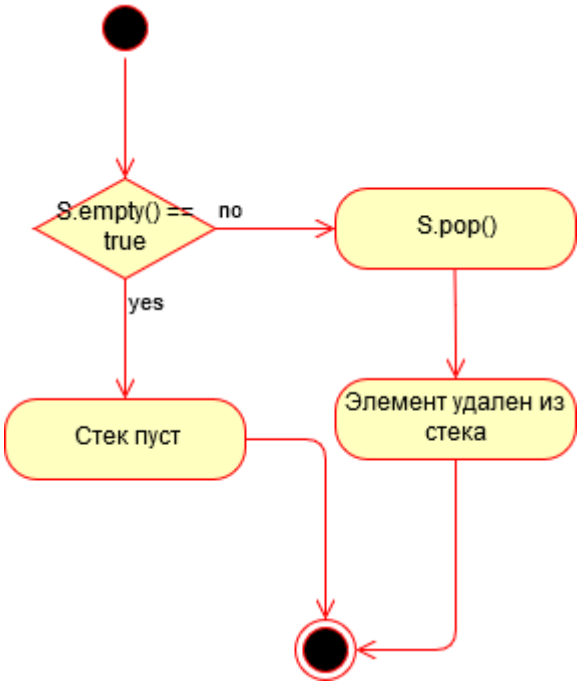

Рисунок 3 - Диаграмма деятельности удаление элемента из стека

```
1. Добавить элемент
2. Удалить элемент
3. Получить верхний элемент
4. Узнать размер стека
0. Выйти
Номер команды > 3

Верхний элемент стека: 5
```

Рисунок 4 - Результат

Узнать размер стека

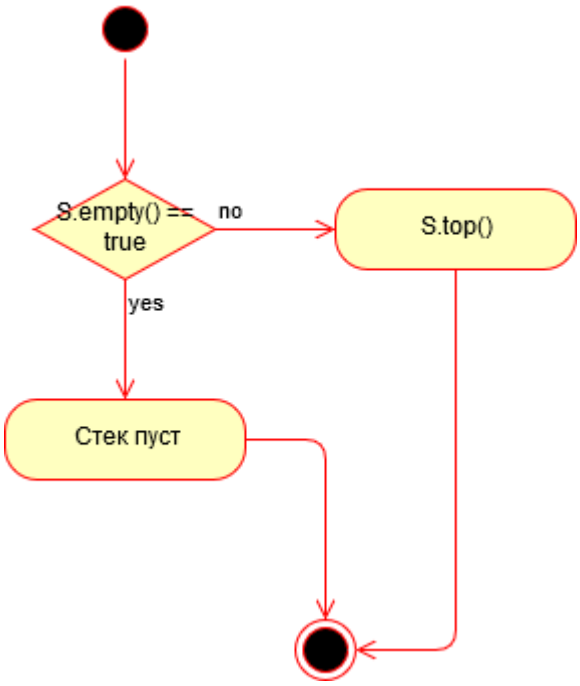

Рисунок 5 - Диаграмма деятельности получить верхний элемент стека

```
1. Добавить элемент
2. Удалить элемент
3. Получить верхний элемент
4. Узнать размер стека
0. Выйти
Номер команды > 4

Размер стека: 3
```

Рисунок 6 - Результат

Удалить элемент

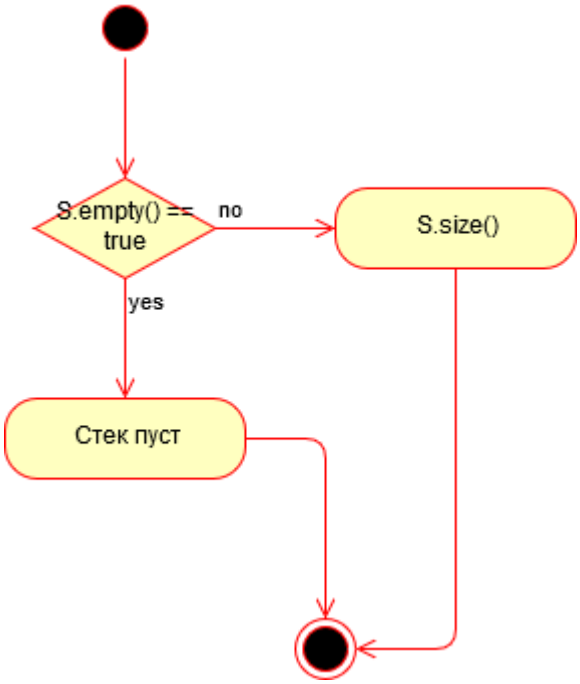

Рисунок 7 - Диаграмма деятельности узнать размер стека

```
1. Добавить элемент
2. Удалить элемент
3. Получить верхний элемент
4. Узнать размер стека
0. Выйти
Номер команды > 2

Элемент удален из стека
```

Рисунок 8 - Результат

Очередь

Очереди очень похожи на стеки. Они также не дают доступа к произвольному элементу, но, в отличие от стека, элементы кладутся *(enqueue)* и забираются *(dequeue)* с разных концов. Такой метод называется «первый вошел, первый вышел» *(First-In-First-Out или FIFO)*.

То есть забирать элементы из очереди мы будем в том же порядке, что и клали. Как реальная очередь или конвейер.

Очереди часто используются в программах для реализации буфера, в который можно положить элемент для последующей обработки, сохраняя порядок поступления. Например, если база данных поддерживает только одно соединение, можно использовать очередь потоков, которые будут, как ни странно, ждать своей очереди на доступ к БД.

Класс Queue

Класс Queue, как и стек, будет реализован с помощью связного списка. Он будет предоставлять методы Enqueue для добавления элемента, Dequeue для удаления, Peek и Count. Как и класс Stack, он не будет реализовывать интерфейс ICollection<T>, поскольку это коллекции специального назначения.

Код программы

```
#include "pch.h"
#include <iostream>
using namespace std;
struct Node //описание узла списка
{
	int data; //информационное поле
	Node *next; //указатель на следующий элемент
};
struct Queue //описание очереди
{
	int size; //счетчик размера очереди
	Node *first; //указатель на начало очереди
	Node *last; //указатель на конец очереди
};
void Creation(Queue *Q) //создание очереди
{
	Q->first = new Node;
	Q->first->next = NULL;
	Q->last = Q->first;
	Q->size = 0;
}
bool Full(Queue *Q) //проверка очереди на пустоту
{
	if (Q->first == Q->last) return true;
	else return false;
}
int Top(Queue *Q) //вывод начального элемента
{
	return Q->first->next->data;
}
void Add(Queue *Q) //добавление элемента
{
	int value;
	cout << "\nЗначение > "; cin >> value;
```

```
	Q->last->next = new Node;
	Q->last = Q->last->next;
	Q->last->data = value; //добавление элемента в конец
	Q->last->next = NULL; //обнуление указателя на следующий элемент
	Q->size++;
	cout << "\nЭлемент добавлен\n\n";
}
void Delete(Queue *Q) //удаление элемента
{
	Q->first = Q->first->next; //смещение указателя
	Q->size--;
	cout << "\nЭлемент удален\n\n";
}
int Size(Queue *Q) //размер очереди
{
	return Q->size;
}
void main() //главная функция
{
	setlocale(LC_ALL, "Rus");
	Queue Q;
	Creation(&Q);
	char number;
	do
	{
		cout << "1. Добавить элемент" << endl;
		cout << "2. Удалить элемент" << endl;
		cout << "3. Вывести верхний элемент" << endl;
		cout << "4. Узнать размер очереди" << endl;
		cout << "0. Выйти\n\n";
		cout << "Номер команды > "; cin >> number;
		switch (number)
		{
		case '1': Add(&Q);
			break;
			//---------------------------------------------
		case '2':
			if (Full(&Q)) cout << endl << "Очередь пуста\n\n";
			else Delete(&Q);
			break;
			//---------------------------------------------
		case '3':
			if (Full(&Q)) cout << endl << "Очередь пуста\n\n";
			else { cout << "\nНачальный элемент: " << Top(&Q) << "\n\n"; }
			break;
			//---------------------------------------------
		case '4':
			if (Full(&Q)) cout << endl << "Очередь пуста\n\n";
			else cout << "\nРазмер очереди: " << Size(&Q) << "\n\n";
			break;
			//---------------------------------------------
		case '0': break;
		default: cout << endl << "Команда не определена\n\n";
```

```
                    break;
                }
            } while (number != '0');
            system("pause");
}
```

Добавление элемента

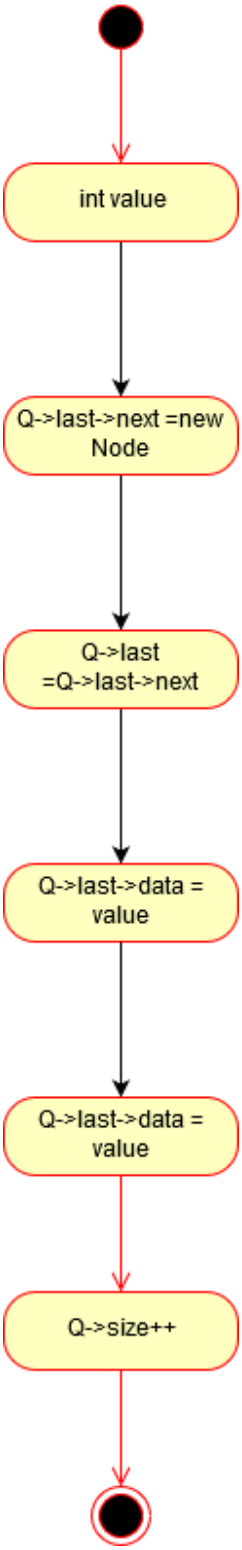

Рисунок 9 - Диаграмма деятельности добавление элемента в очередь

```
C:\Users\Asus\source\repos\ConsoleA
1. Добавить элемент
2. Удалить элемент
3. Вывести верхний элемент
4. Узнать размер очереди
0. Выйти

Номер команды > 1

Значение > 2

Элемент добавлен

1. Добавить элемент
2. Удалить элемент
3. Вывести верхний элемент
4. Узнать размер очереди
0. Выйти

Номер команды > 1

Значение > 0

Элемент добавлен
```

Рисунок 10 - Результат

Вывод верхнего элемента

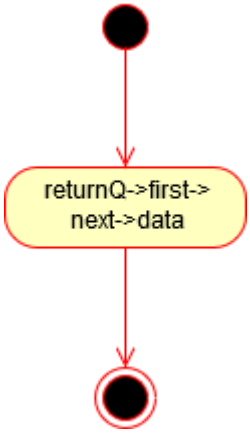

Рисунок 11 - Диаграмма деятельности вывод верхнего элемента

```
1. Добавить элемент
2. Удалить элемент
3. Вывести верхний элемент
4. Узнать размер очереди
0. Выйти

Номер команды > 3

Начальный элемент: 2
```

Рисунок 12 - Результат

Удаление элемента

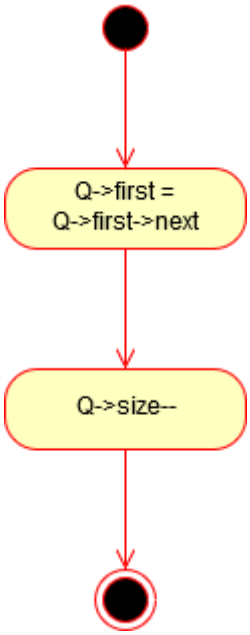

Рисунок 13 - Диаграмма деятельности удаление элемента

```
1. Добавить элемент
2. Удалить элемент
3. Вывести верхний элемент
4. Узнать размер очереди
0. Выйти

Номер команды > 2

Элемент удален
```

Рисунок 14 - Результат

Размер очереди

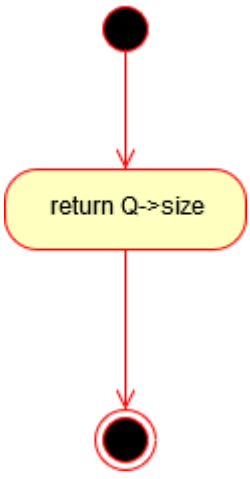

Рисунок 15 - Диаграмма деятельности размер очереди

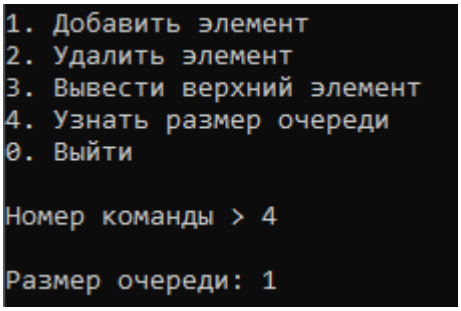

Рисунок 16 – Результат

Вывод: в ходе выполнения лабораторной работы изучила сд «стек» и «очередь», а также программно реализовала их на языке C++.