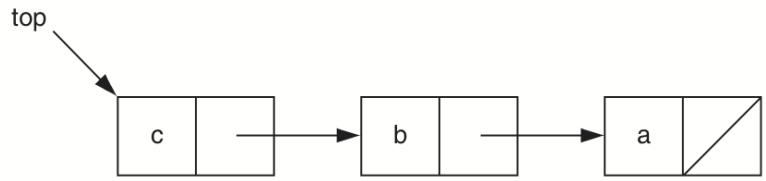
Лабораторная работа № 8

Абстрактный тип данных «Стек» версия 2

При реализации класса Stack на предыдущей лабораторной работе мы выделяли память для стека в момент объявления, при этом сразу резервировалась память под максимально возможный размер стека и часть выделенной памяти никогда не использовалась.

Альтернативный подход состоит в том, чтобы выделять память только тогда, когда она фактически необходима. Для этого необходимо формировать представление стека в виде связанного списка (как показано на следующем рисунке):



Реализация стека при помощи связанного списка представляет собой более сложную задачу по сравнению с реализацией стека с использованием массива, поэтому для ее упрощения разделим задачу на реализацию двух шаблонов классов: в одном шаблоне класса сфокусируемся на структуре стека (класс StackL), в другом шаблоне класса сосредоточимся на отдельных узлах связанного списка (класс StackNode).

Начнем с класса StackNode. Каждый узел в связанном списке содержит элемент данных стека и указатель на узел, содержащий следующий элемент списка. Единственный метод в классе StackNode – это конструктор, который создает необходимый узел. Доступ к классу StackNode ограничен методами класса StackL. Другим классам доступ к связанному списку заблокирован, так как элементы класса StackNode объявлены закрытыми (private). Методы класса StackL имеют доступ к классу StackNode, так как класс StackL объявлен дружественным классом для класса StackNode.

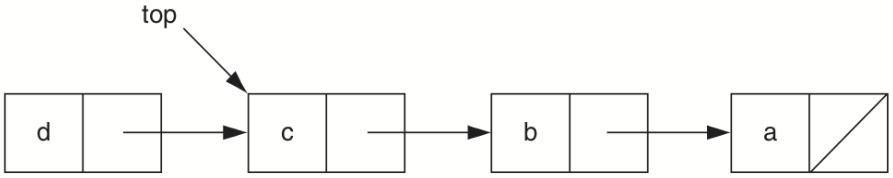
```
template < class DT >    // форвардное объявление класса StackL
class StackL;
template < class DT >    // вспомогательный класс для класса StackL
class StackNode
{
private:
    // Constructor
    StackNode ( const DT &nodeData, StackNode *nextPtr );
    // Data members
    DT dataItem;    // Stack data item
    StackNode* next;    // Pointer to the next data item
    friend class StackL<DT>;
};
template < class DT >
class StackL
{
...
private:
```

```
    StackNode<DT>* top;    // Pointer to the top data item
};
```

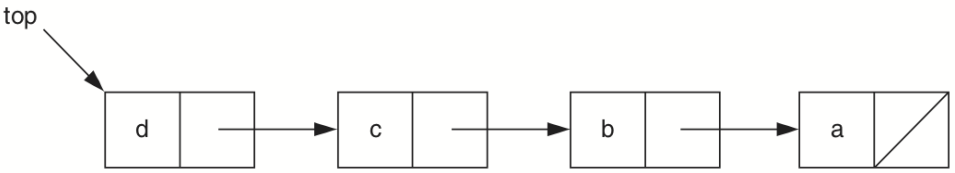
Конструктор класса StackNode нужен, чтобы добавлять узлы в стек. Например, следующее утверждение добавляет узел, содержащий символ 'd' в стек из символов (параметр DT шаблона класса должен быть эквивалентен типу char и переменная top имеет тип StackNode*):

```
top = new StackNode<DT>('d',top);
```

Здесь оператор new выделяет память для узла связанного списка и вызывает конструктор класса StackNode, передавая в конструктор как элемент данных ('d'), так и указатель на следующий узел в списке (top).



Наконец, оператор присваивания присваивает переменной top указатель на новый узел, завершая тем самым добавление узла в связанный список.



Методы класса StackL реализуют операции со стеком. В классе StackL поддерживается указатель на начало связанного списка или, другими словами, на вершину стека. Объявление класса StackL имеет вид:

```
template < class DT >
class Stack
{
public:
    // Constructor
    Stack ();
    // Destructor
    ~Stack ();
    // Stack manipulation operations
    void push ( const DT &newDataItem ); // Push data item
    DT pop (); // Pop data item
    void clear (); // Clear stack
    // Stack status operations
    bool isEmpty () const; // Is stack empty?
    bool isFull () const; // Is stack full?
    void show () const; // Output the stack structure
private:
    // Data member
    StackNode<DT> *top;                  // Pointer to the top data item
};
```

Особенности реализации класса StackL:

- В пустом стеке указатель top является нулевым указателем (top = 0)
- При написании кода деструктора класса StackL и метода clear проходим по связанному списку и освобождаем память для всех объектов класса StackNode, созданных при помощи команды new
- Метод isFull всегда возвращает false
- Итерация по связанному списку может осуществляться при помощи цикла:

```
for (StackNode* temp = top; temp != 0; temp = temp->next) {
    ...
}
```

Задание 8.1. (5 баллов) Разработайте конструктор и методы **класса** StackL с использованием шаблонов классов и с учетом указанных выше ограничений. Разработайте интерактивную программу для работы со стеком чисел с плавающей точкой, поддерживающую следующие команды с консоли:

Команда	Действия
+ x	Поместить элемент x на вершину стека
-	Извлечь элемент с вершины стека и вывести его на консоль
*	Извлечь элемент с вершины стека и вывести его на консоль его квадрат
=	Извлечь из стека все элементы и вывести на консоль их сумму
E	Вывести, является ли стек пустым
F	Вывести, является ли стек заполненным
C	Очистить стек
Q	Выйти из программы

После выполнения каждой команды выводите на консоль состояние стека при помощи метода show.

При тестировании разработанного кода исполните следующие команды:

- + 1.1 + 2.2 + 3.3 + 4.4 (добавление элементов в стек)
- - (извлечение элемента из стека)
- * (извлечение элемента из стека с возведением в квадрат)
- + 5.5 (добавление элемента в стек)
- = (суммирование элементов в стеке)
- Q (выход)