МГТУ им. Н.Э. Баумана

**Дисциплина Типы и Структуры данных.**

**Лабораторный практикум №4**

**по теме:** «Работа со стеком»

Работу выполнил:

студент группы ИУ7-35Б

Прянишников Александр

**Цель работы**: реализовать операции работы со стеком, который представлен в виде массива (статического или динамического) и в виде односвязного линейного списка; оценить преимущества и недостатки каждой реализации: получить представление о механизмах выделения и освобождения памяти при работе со стеком.

# Условие задачи. Вариант 3

Создать программу работы со стеком, выполняющую операции добавление, удаления элементов и вывод текущего состояния стека. Реализовать стек: а) массивом; б) списком. Все стандартные операции со стеком должны быть оформлены подпрограммами. При реализации стека списком в вывод текущего состояния стека добавить просмотр адресов элементов стека и создать свой список или массив свободных областей (адресов освобождаемых элементов) с выводом его на экран.

Элементами стека являются слова. Распечатайте слова в обратном порядке.

# Требования к задаче

**Ввод данных**

Стек состоит из строк, которые пользователь последовательно вводит через соответствующий пункт меню. Длина строки ограничена – 30 символов, при этом сам стек также ограничен – до 100 элементов.

**Вывод данных**

К выходным данным относятся: сам стек, адреса, занимаемые в стеке, адреса, освободившиеся после удаления, время исполнения программы (касается только вывода слов в обратном порядке), а также непосредственно задание – вывод слов со стека в обратном порядке.

Также в программу был добавлен интерфейс, упрощающий работу для пользователей: есть пояснительные надписи, перед вводом данных написано, как правильно вводить данные.

**Описание задачи, реализуемой программой**

Программа предоставляет пользователю интерфейс для работы со стеком. Программа добавляет по команде пользователя очередную строку в стек, отслеживая правильность ввода. Также пользователь имеет возможность вытащить элемент со стека, вывести текущее состояние стека, отслеживать адреса элементов стека. Также программа умеет распечатывать последовательность слов в обратном порядке.

**Способ обращения к программе.**

Программу можно запустить через MSYS либо через QT Creator и любую среду разработки.

Выполняется следующая последовательность команд: *./main.exe*

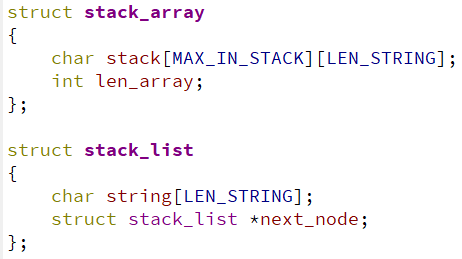
**Описание возможных аварийных ситуаций и ошибок пользователя**

* Выбор отсутствующего пункта меню (от 0 до 4)  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Ввод букв при вводе числа  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Пустой ввод  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Слишком длинная введённая строка  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести всё заново.*
* Добавление нового элемента в переполненный стек.  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и вернётся в главное меню.*
* Удаление элемента из пустого стека  
  *В этом случае программа сообщит пользователю, что удаление невозможно, и вернётся в главное меню.*

# Описание внутренних структур данных.

Для решения этой задачи я решил использовать две структуры: одна описывает стек через статический массив, другая реализует стек через связный список.

Вот так описаны структуры:



# Описание алгоритма

Программа работает для двух разных структур по разным алгоритмам, выполняя одно и то же действие:

**1 – Добавить элемент в стек.**Программа предлагает пользователю ввести строку для добавления в стек, при этом отслеживается корректность ввода.

Сначала добавление производится в стек, представленный массивов. Программа по текущей длине массива определяет индекс, куда нужно добавить строку, после чего её добавляет, длина массива увеличивается на 1. Стоит отметить, что в стеке есть доступ только к одному элементу, это гарантируется программистом.

Затем добавление происходит по стеку, представленным связным списком. Программа присваивает соответствующее значение узлу, после чего делает узел первым в связном списке.

**2 – Удалить элемент из стека**.  
Программа достаёт первый элемент из стека и печатает его. В случае стека–массива длина уменьшается на 1. В связном списке же удаляется последний узел.

При этом программа печатает на экран адрес, в котором лежала строка.

**3 –**  **Посмотреть текущее состояние стека.**Программа делает обход по обоим стекам и выводит на экран стек для каждого из вида представлений. Тоже самое происходит и для массива свободных адресов и адресов, которые сейчас используются связным списком для хранения строк.

**4 – Вывести последовательность слов в стеке в обратном порядке**Программа для обоих видов стеков создаёт временный стек и добавляет туда элементы стека, делая прямой обход. После этого программа выводит на экран все элементы временного стека, то есть начальный стек выведен наоборот.

**0 – Выход.**

# Тесты

Тестирование происходило по такому принципу: каждый пункт тестировался отдельно, после чего контролировалась работа программы целиком. Поэтому здесь указаны только негативные тесты, которые в целом проверяли работоспособность модулей программы.

|  |  |
| --- | --- |
| **123345fd** | **Буквы при задании числа** |
|  | **Пустой ввод** |
| **–1** | **Выбор неправильного пункта меню** |
|  | **Переполнение стека** |
|  | **Удаление из пустого стека** |

# Оценка эффективности

Для начала нужно сравнить по эффективности функции pop и push, которые есть в каждой реализации стека.

**Pop**

|  |  |  |
| --- | --- | --- |
|  | Массив | Связный список |
| N = 5 | 0.000000012 | 0.000000664 |
| N = 10 | 0.000000005 | 0.000000304 |
| N = 50 | 0.000000009 | 0.000000621 |
| N = 100 | 0.000000006 | 0.000000488 |
| N = 500 | 0.000000010 | 0.000000229 |
| N = 1000 | 0.000000004 | 0.000000381 |

Можно сделать вывод, что время действия pop не зависит от N для обоих реализаций, то есть теоретическая зависимость подтвердилась.

Теперь проверим время работы для различных реализаций Push

**Push**

|  |  |  |
| --- | --- | --- |
|  | Массив | Связный список |
| N = 5 | 0.000000021 | 0.000000450 |
| N = 10 | 0.000000022 | 0.000000370 |
| N = 50 | 0.000000016 | 0.000000613 |
| N = 100 | 0.000000019 | 0.000000501 |
| N = 500 | 0.000000023 | 0.000000263 |
| N = 1000 | 0.000000022 | 0.000000402 |

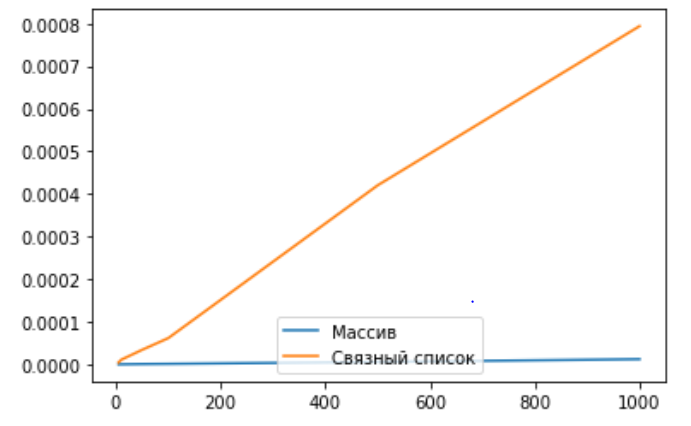
Как видно, массивная реализация в несколько десятков раз эффективнее, чем связный список.

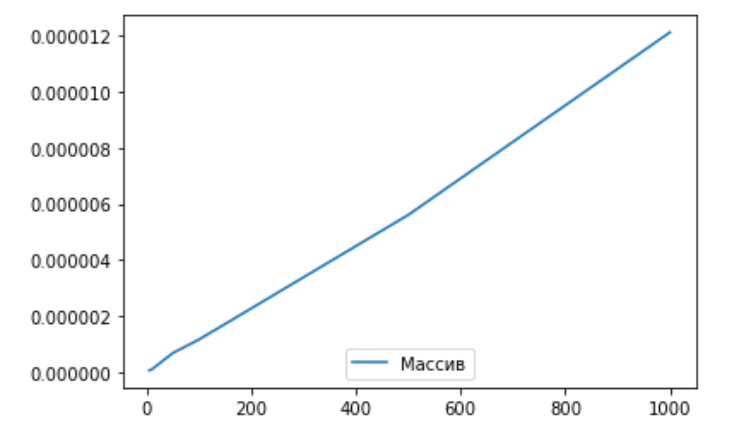
Теперь сравним время работы вывода при выводе последовательности наоборот.

|  |  |  |
| --- | --- | --- |
|  | Массив | Связный список |
| N = 5 | 0.00000008 | 0.00000439 |
| N = 10 | 0.00000012 | 0.00001080 |
| N = 50 | 0.00000070 | 0.00003410 |
| N = 100 | 0.00000118 | 0.00006161 |
| N = 500 | 0.00000561 | 0.00042042 |
| N = 1000 | 0.00001210 | 0.00079400 |

Построим аналогичную таблицу для памяти, размер стека – 1000 элементов.

|  |  |  |
| --- | --- | --- |
|  | Массив (Память) | Связный список (Память) |
| N = 5 | 30004 | 200 |
| N = 10 | 30004 | 400 |
| N = 50 | 30004 | 2000 |
| N = 100 | 30004 | 4000 |
| N = 500 | 30004 | 20000 |
| N = 1000 | 30004 | 40000 |





Заметно, что время работы пропорционально N, и при этом массив в сотни раз эффективнее по времени, чем связный список.

Теперь разберёмся с памятью. Статический массив всегда занимает одинаковое число байт – 30 \* Max\_Stack + 4, где Len – максимальная длина строки, Max\_Stack – максимальное количество элементов в стеке. Связный список же занимает 40 \* N байт, где N – количество элементов.

То есть видно, что по памяти связный список в большинстве случаев эффективнее, но при устремлении количества элементов к максимальному числу элементов в стеке, массив становится лучше и тут. Если мы знаем, сколько в стеке будет эффективнее, и знаем, что статической памяти хватит, то стоит выбрать массив, так как он намного быстрее.

Но если стек настолько большой, что не хватает места в статической памяти (а так как хранятся строки, то это вполне реально), то лучше использовать связный список.

# Выводы по проделанной работе

Сегодня я познакомился с двумя способами хранения стека: в виде статического массива, и в виде связного списка. Первый работает гораздо быстрее, но при этом требует больше памяти, но при больших N массив нивелирует и это отставание. Связный список работает намного медленнее, но зато есть выигрыш по памяти для N при не слишком большом числе элементов.

# Ответы на контрольные вопросы

Что такое стек?

Стек – это последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины. Стек функционирует по принципу: последним пришел – первым ушел, Last In – First Out (LIFO).

Каким образом и сколько памяти выделяется под хранение стека при различной его реализации?

До начала работы указатель стека показывает на нулевой, физически отсутствующий адрес (т. е. указатель - пустой). При включении элемента в стек сначала происходит выделение области памяти, адрес которой записывается в указатель стека, а затем по значению этого указателя в стек помещается информация.

Если стек реализован в виде статического или динамического массива (вектора), то для его хранения обычно отводится непрерывная область памяти ограниченного размера, имеющая нижнюю и верхнюю границу.

Каким образом освобождается память при удалении элемента стека при различной реализации стека?

Узел, соответствующий последнему элементу, освобождается функцией free, и последним элементом стека становится предпоследний узел.

В стеке–массиве указатель просто смещается на –1, и элемент становится недоступным.

Что происходит с элементами стека при его просмотре?

При просмотре стека с элементами указатель сдвигается, и элемент уничтожается.

Каким образом эффективнее реализовывать стек? От чего это зависит?

Стек значительно эффективнее работает в массивном варианте, при этом его лучше использовать, если мы знаем, сколько точно элементов будет в стеке. Связный стек при небольших размерах будет эффективнее по памяти, но при этом если стек планируется слишком большой, то для статического массива может не хватить памяти, и тогда связный список – единственный способ реализации стека.