|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

**«Работа с стеком»**

Преподаватель: Силантьева А. В.

Студент: Динь Вьет Ань, ИУ7И - 34Б

**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.

**Оисание условие задачи:**

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.

Используя стек, определить, является ли строка палиндромом.

**Техническое задание**

**Входные данные:**

Целое число — номер пункта меню, который вызывает описанное в

пункте действие.

Строка символов — последовательность элементов стека при его создании или добавлении в него.

Максимальный размер стека - 100

Символ переноса строки - признак окончания ввода последовательности

**Входные данные:**

В зависимости от пункта меню.

**Возможные аварийные ситуации и ошибки пользователя**

Аварийные ситуации и ошибки:

- некорректный выбор пункта меню.

- переполнение стека.

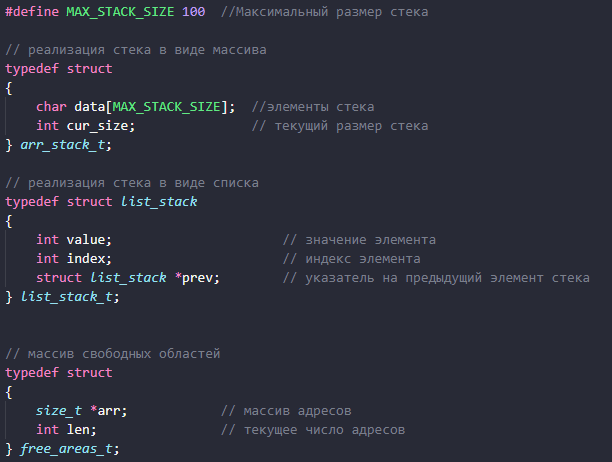
- удаление элемента из пустого стека.

- вывод пустого стека.

- проверка на палиндром пустого стека.

В случае аварийной ситуации пользователю выдается сообщение об ошибке и происходит возвращение в меню.

**Структуры данных**



**Описание алгоритма**

Работа всей программы основана на взаимодействии пользователя с меню.

Каждый пункт меню (от 0 до 6) вызывает описанное в пункте действие.

Алгоритм добавления элементов в стек реализуется при помощи операции push (вставки элемента) в соответствии с каждой структурой данных.

Алгоритмы удаления элемента и вывода текущего состояния стека и массива свободных областей реализуются при помощи операции pop (извлечения последнего элемента стека) в соответствии с каждой структурой данных.

Алгоритм проверки строки на палиндром реализован так: из стека удаляется size / 2 элементов и записывается в дополнительный стек. Затем извлекается один элемент, если size - нечетный. Далее элементы одновременно извлекаются из каждого стека и сравниваются: если все элементы совпали, строка является палиндромом, иначе - не является. Данный алгоритм работает с копией стека при реализации его в виде массива и с самим стеком (удаляя все элементы) при реализации его в виде списка.

**Основные функции**

Выбор действия в меню:

int choose\_action(void);

Ввод элементов с добавлением их в стек (1 - массив, 2 - список)

int array\_input\_element (arr\_stack\_t \*stack);

int list\_input\_element (list\_stack\_t \*\*stack, free\_areas\_t \*areas);

Добавление одного элемента в стек (1 - массив, 2 - список)

int array\_push (arr\_stack\_t \*stack, char element);

int list\_push (list\_stack\_t \*\*head, char element, int first\_node);

Удаление последнего элемента из стека (1 - массив, 2 - список)

int array\_delete\_element (arr\_stack\_t \*stack);

int list\_delete\_element (list\_stack\_t \*\*stack, free\_areas\_t \*areas);

Извлечение последнего элемента из стека (1 - массив, 2 - список)

int array\_pop (arr\_stack\_t \*stack, char \*element);

int list\_pop (list\_stack\_t \*\*stack, char \*element, size\_t \*address);

Вывод текущего состояния стека (1 - массив, 2 - список)

void array\_print (arr\_stack\_t \*stack);

void list\_print (list\_stack\_t \*stack, free\_areas\_t \*areas);

Проверка на палиндром (1 - массив, 2 - список)

int array\_is\_palindrome (arr\_stack\_t \*word);

int list\_is\_palindrome (list\_stack\_t \*\*head, free\_areas\_t \*areas);

Работа с освобождаемыми областями памяти

void create\_areas (free\_areas\_t \*areas);

void print\_free\_areas (free\_areas\_t \*areas);

Освобождение списка

void free\_list (list\_stack\_t \*\*head);

**Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Ввод | Действие | Вывод |
| 1 | 7 | Ввод неверного пункта меню | Ошибка ввода: необходимо  ввести номер одного из  предложенных вариантов |
| 2 | 1  строка более чем из 100 элементов | Ввод больше 100 элементов (в стек добавляются 100 элементов, для остальных выводится сообщение) | Элемент 'a' успешно добавлен  …  Элемент 'a' успешно добавлен  Невозможно добавить элемент 'a': стек заполнен |
| 3 | 1  a | Добавление в стек, в котором уже 100 элементов | Невозможно добавить элемент 'a': стек заполнен |
| 4 | 2 | Удаление элемента из пустого стека | Невозможно удалить элемент: стек пуст |
| 5 | 3 | Вывод пустого стека | Текущее состояние стека,  реализованного в виде списка: Стек пуст |
| 6 | 4 | Проверка на палиндром пустого стека | Стек пуст |
| 7 | 1  abc | Валидное добавление  элементов в стек | Элемент 'a' успешно добавлен  Элемент 'b' успешно добавлен  Элемент 'c' успешно добавлен |
| 8 | 2 | Валидное удаление элемента | Элемент 'c' успешно удален |
| 9 | 4  abcba | Проверка строки, которая является палиндромом, на палиндром | Строка является палиндромом |
| 10 | 4  abc | Проверка строки, которая не является палиндромом, на палиндром | Строка не является  палиндромом |
| 11 | 3 | Вывод стека в виде массива | Текущее состояние стека,  реализованного в виде массива: ->  c  b  a |
| 12 | 3 | Вывод стека в виде  односвязного списка | Текущее состояние стека,  реализованного в виде списка:  ->  c : 000002bd569580a0  b : 000002bd56958060  a : 000002bd569581c0  Массив освободившихся областей:  Массив освободившихся областей пуст |
| 13 | 3 | Вывод стека в виде  односвязного списка с  наличием освободившихся областей | Текущее состояние стека,  реализованного в виде списка:  ->  c : 000002705b658120  b : 000002705b6581e0  a : 000002705b6580c0  Массив освободившихся  областей:  2705b658220 |

**Анализ эффективности**

Время добавления элемента в стек (такты):

|  |  |  |
| --- | --- | --- |
| Количество элементов | Массив | Список |
| 10 | 519 | 8218 |
| 50 | 1465 | 28084 |
| 100 | 3657 | 49591 |

Время удаления элемента из стека (такты):

|  |  |  |
| --- | --- | --- |
| Количество элементов | Массив | Список |
| 10 | 334 | 1398 |
| 50 | 1592 | 12861 |
| 100 | 2833 | 26299 |

Время проверки на палиндром (такты):

|  |  |  |
| --- | --- | --- |
| Количество элементов | Массив | Список |
| 10 | 1377 | 28662 |
| 50 | 3743 | 55375 |
| 100 | 5027 | 76599 |

Объем памяти (байты)

|  |  |  |
| --- | --- | --- |
| Количество элементов | Массив | Список |
| 10 | 104 | 160 |
| 50 | 104 | 800 |
| 100 | 104 | 1600 |

**Выводы**

Стек, составленный из связанного списка, требует как памяти, так и времени обработки. Следовательно, мы можем сделать вывод, что если вам нужно реализовать структуру данных, такую ​​как стек, лучше использовать массив, а не связанный список. Но если максимальное количество элементов заранее не известно, рекомендуется использовать стек из связного списка.

Время работать с массивом меньше на (около) 10 раз чем с списком.

**Контрольные вопросы**

1. 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).

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

При реализации стека при помощи связанного списка память выделяется по мере необходимости: (sizeof(type) + sizeof(type\_t\*)) \* count,

где count — число элементов, type — тип элементов, type\_t — тип узла.

При реализации стека при помощи массива память выделяется в начале

работы программы: sizeof(type) \* count, где count — число элементов, type — тип элементов.

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

При реализации стека при помощи связанного списка: освобождается

память, ранее занимаемая верхним элементом (по мере необходимости), и смещается указатель, который указывает на начало стека. При реализации стека при помощи статического массива: смещается указатель, который указывает на вершину стека. Память освобождается при завершении работы программы.

4. Что происходит с элементами стека при его просмотре?

Элементы извлекаются из стека (уничтожаются).

5. Каким образом эффективнее реализовывать стек? От чего это зависит?

Реализовывать стек при помощи списка эффективнее тем, что память для него ограничена размером оперативной памяти, в то время как для статического массива память ограничена размером стека. По времени работы реализация стека при помощи массива эффективнее.