Отчёт по лабораторной работе №4

“Работа со стеком”

Вариант 7

Выполнил: Соколов Ефим Маркович (группа ИУ7-33Б)

### ***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.

Техническое задание

Элементами стека являются адреса памяти. При реализации массивами - их вводить, при реализации списком – брать адрес выделенной памяти под элемент.

Типы входных и выходных данных

***Входные данные:***

* Ключ меню
* Адрес памяти, который будет являться элементом стека
* Адрес памяти, под которым хранится элемент стека

**Выходные данные:**

* Текущее состояние стека
* Замеры времени на добавление и удаление элементов в стек
* Замеры затраченной памяти
* Список освобожденных адресов памяти

***Возможные аварийные ситуации:***

* Некорректный ввод данных
* Переполнение стека
* Попытка вывода содержимого пустого стека

Структуры данных

Для представления стека в статическом массиве и односвязном списке используются следующие структуры данных:

#define STACK\_SIZE 1000 // Максимальная размер стека

***Реализация стека статическим массивом:***

typedef struct

{

int \*\*start; // Границы стека

int \*\*end; //

int \*\*head; // “голова” стека

} stackArray\_t;

int \*array[STACK\_SIZE];

stackArray\_t stack\_array; // стек

stack\_array.start = array;

stack\_array.end = array + STACK\_SIZE - 1;

stack\_array.head = array - 1;

***Реализация линейным односвязным списком:***

struct stackList\_t

{

size\_t \*value;

struct stackList\_t \*next;

};

typedef struct stackList\_t stackList\_t;

stackList\_t \*stack\_list = NULL; // стек

stackList\_t \*freed\_list = NULL; // список освобожденной памяти

size\_t \*memory = NULL; // освобожденная память

Алгоритм

***Статический массив:***  
Для контроля изменений в стеке из статического массива введена структура *stack\_array* с полями:

1. *start* — начало области памяти под стек
2. *end* — конец области памяти под стек
3. *head* — текущее расположение головы

С помощью данной структуры делается вывод о пустоте стека и его переполнении.

***Связанный список:***

В стеке из списка при удалении элемента с головы возвращается участок памяти *memory* удаленной ячейки и добавляется в список *freed\_list*, где хранится вся освобожденная память.

Оценка эффективности

***Количество элементов в стеке: 10***

|  |  |  |
| --- | --- | --- |
| **Реализация** | **Добавление (c)** | **Удаление (c)** |
| **Массив** | **0.000006** | **0.000003** |
| **Список** | **0.000006** | **0.000004** |

|  |  |
| --- | --- |
| **Реализация** | **Память (байт)** |
| **Массив** | **80** |
| **Список** | **160** |

***Количество элементов в стеке: 100***

|  |  |  |
| --- | --- | --- |
| **Реализация** | **Добавление (c)** | **Удаление (c)** |
| **Массив** | **0.000007** | **0.000004** |
| **Список** | **0.000010** | **0.000007** |

|  |  |
| --- | --- |
| **Реализация** | **Память (байт)** |
| **Массив** | **800** |
| **Список** | **1600** |

***Количество элементов в стеке: 500***

|  |  |  |
| --- | --- | --- |
| **Реализация** | **Добавление (c)** | **Удаление (c)** |
| **Массив** | **0.000021** | **0.000010** |
| **Список** | **0.000038** | **0.000037** |

|  |  |
| --- | --- |
| **Реализация** | **Память(байт)** |
| **Массив** | **4000** |
| **Список** | **8000** |

***Количество элементов в стеке: 1000***

|  |  |  |
| --- | --- | --- |
| **Реализация** | **Добавление (c)** | **Удаление (c)** |
| **Массив** | **0.000029** | **0.000014** |
| **Список** | **0.000056** | **0.000050** |

|  |  |
| --- | --- |
| **Реализация** | **Память (байт)** |
| **Массив** | **8000** |
| **Список** | **16000** |

Вывод

Из полученных данных видно, что при увеличении размера стека при реализации списком время добавления и удаления растет быстрее. Памяти при реализации массивом требуется в 2 раза меньше чем списком. При этом у статического массива есть недостаток — максимальный объем памяти ограничен размером стека операционной системы, к тому в таком случае память располагается линейно – сложнее выделить непрерывный большой участок памяти. Эти недостатки нивелируются использованием списка, так как его узлы расположены в памяти не друг за другом (фрагментация памяти) и возможно хранить больше данных.

Ответы на контрольные вопросы

1. ***Что такое стек?***

Стек — структура данных организованная по принципу ***LIFO*** (последним пришел — первым вышел), поддерживающая операции **push** - добавить элемент в вершину, **pop** — удалить элемент из вершины, **peek** — просмотреть элемент из вершины стека.

1. **Каким образом и сколько памяти выделяется под хранение стека при**

**различной его реализации?**

Согласно полученным данным при реализации списком требуется в 2 раза больше памяти.

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

При реализации массивом просто сдвигается указатель на головной элемент, если списком, то необходимо освободить память под узел списка, где хранится головной элемент и далее присвоить **NULL** ссылке предыдущего элемента списка на головной.

1. **Что происходит с элементами стека при его просмотре?**

Чтобы просмотреть второй с головы элемент стека - надо удалить головной и так далее.

1. **Каким образом эффективнее реализовывать стек? От чего это зависит?**

В зависимости от нужд, если нужно хранить большие объемы данных, то тут пригодится список, если важна скорость обработки, то следует использовать статический или динамический массив.