Отчёт по лабораторной работе №4 "Работа со стеком", Вариант 7

Выполнил: Калашков Павел ИУ7-36Б, Вариант 7 (по журналу)

Описание условия задачи

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.

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

Техническое задание

Исходные данные

Исходными данными для программы является данные, введённые с консоли (см. ниже);

Данные, введённые с консоли, связаны с меню программы:

```
Возможные действия:
 1 - Добавить элемент в стек
 2 - Достать элемент из стека
 3 - Вывести содержимое стека
 4 - Очистить стеки
 5 - Исследовать время работы программы
 0 - Выйти из программы
Введите номер действия:
```

Поэтому введённые данные являются **строкой**, которая может содержать в себе номер действия, а также входные данные для конкретных действий (см. ниже)

Результаты

Выходными данными являются:

- текущее состояние стека-массива
- текущее состояние стека-списка
- результаты измерений времени работы программы

Задача, реализуемая программой

Задачей программы является реализация операций использования стека (стека-массива и стека-списка): - вводить элементы; - удалять элементы; - просматривать текущее состояние стека; - исследовать время работы операций добавления и удаления элемента;

Способ обращения к программе

Обращение к программе происходит посредством запуска исполняемого файла app.exe

Возможные аварийные ситуации и ошибки пользователя

Аварийной ситуацией является:

- несоблюдение формата входных данных (см. исходные данные);

Описание внутренних структур данных

Для реализации деления и обработки больших чисел были реализованы четыре структуры данных:

```
typedef struct
{
    stack_array_t *stack_array;
    stack_node_t *stack_node;
} info;

typedef struct
{
    int *pointer_stack;
    int *beginning;
    int length;
} stack_array_t;

struct node_t
{
    struct node_t *value;
    struct node_t *next;
};

typedef struct
{
    struct node_t *pointer_stack;
    int length;
    struct node_t *free_space_list;
} stack_node_t;
```

Описанный алгоритм

Алгоритм делится на четыре главных части:

- 1 - вывод меню;
- 2 - запрос действия из меню;
- 3 - выполнение действия (если оно указано корректно);
- 4 - повтор, если действие - не выход из программы;

Вывод меню

Меню имеет формат:

```
Возможные действия:
 1 - Добавить элемент в стек
 2 - Достать элемент из стека
 3 - Вывести содержимое стека
 4 - Очистить стеки
 5 - Исследовать время работы программы
 0 - Выйти из программы
Введите номер действия:
```

Запрос действия из меню

Программа запрашивает номер действия как строку, считывает и проверяет на корректность.

Если корректно, то переводит строку в число, иначе выводит сообщение об ошибке.

Выполнение действия

- 1 - Уточнить у пользователя вид стека, над которым производится операция (стек-массив или стек-список), уточнить количество вводимых элементов, далее принять их значения. При этом так как значениями элементов являются адреса, то введённые данные должны являться неотрицательными целыми числами.
- 2 - Уточнить у пользователя вид стека, над которым производится операция (стек-массив или стек-список), уточнить количество доставаемых элементов, достать их и вывести их значения. Если речь идёт о стеке-списке, вывести ещё и адреса элементов (должны совпадать со значением).
- 3 - Уточнить у пользователя вид стека, над которым производится операция (стек-массив или стек-список), вывести содержимое стека. Если речь идёт о стеке-списке, вывести ещё и адреса элементов (должны совпадать со значением), а также содержание списка свободных областей.

Список свободных областей содержит адреса, использованные ранее в программе и освобождённые на данный момент. Просмотр адресов текущих элементов и списка свободных областей помогает лучше понять процесс выделения памяти в программе. Как результат выполнения лабораторной работы мы видим, что не на всех компьютерах и операционных системах выделение памяти затрагивает область свободных областей, например, в Linux Ubuntu (на котором производилось написание и выполнение данной лабораторной работы) адреса выделяются вне зависимости от содержания списка свободных областей. - 4 - Очистить стек-массив и стек-строку. - 5 - Уточнить у пользователя вид стека, операции над которым будут измеряться. Произвести замеры времени и используемой памяти, вывести результат; - 0 - Выйти из программы: вывести сообщение о завершении программы;

Сравнение операций

Сравним время работы и объём используемой памяти для используемых операций добавления и удаления элемента в стек-массив и стек-список, усреднённого по N - количеству повторений = максимальному количеству элементов в стеке.

Для стека-массива:

N	Время добавления, стек-массив, нс	Время удаления, стек-массив, нс	Память на N элементов, стек-массив, байты
50	545	500	256
100	567	534	512
200	612	517	1024
300	621	594	2048
400	547	532	2048
500	724	653	2048
600	636	520	4096
700	582	530	4096
800	539	533	4096
900	581	547	4096
1000	557	528	4096
5000	499	476	32768

Для стека-списка:

N	Время добавления, стек-список, нс	Время удаления, стек-список, нс	Память на N элементов, стек-список, байты
50	780	980	400
100	720	590	800
200	690	535	1600
300	740	610	2400
400	717	547	3200

N	Время добавления, стек-список, нс	Время удаления, стек-список, нс	Память на N элементов, стек-список, байты
500	646	532	4000
600	630	538	4800
700	672	561	5600
800	653	541	6400
900	660	534	7200
1000	650	529	8000
5000	612	414	40000

Видим, что и реализация стека при помощи массива, и реализация при помощи списка обеспечивают выполнение операций добавления и удаления элементов примерно за 0.5 мкс, однако в целом реализация при помощи массива является быстрее на 0.1 мкс (20%), в то время как использование массива вместо списка эффективнее при реальной заполненности массива больше, чем на 50% (т.к. в ином случае будет выделяться лишнее количество памяти).

Теоретическая часть

1. Что такое стек?

Стек - структура данных, представляющая последовательный список с переменной длиной, в котором добавление и удаление элементов происходит по принципу Last In - First Out (LIFO), т.е. последним пришёл - первым ушёл.

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

При хранении стека в виде массива под него выделяется определённый объём памяти - этот объём ограничен объёмом доступной оперативной памяти (но может быть ограничен искусственно).

При хранении стека в виде списка выделяется объём памяти, равный length * sizeof(<элемент>). Память выделяется не только на сам элемент, но и на указатель на следующий, поэтому хранение стека в виде списка использует больше памяти, чем хранение стека в виде массива.

Для стека-массива память выделяется в начале использования и далее перевыделяется по мере необходимости, в то время как для стека-списка память выделяется каждый раз, когда добавляется элемент.

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

При хранении стека в виде массива память освобождается в конце [области видимости массива] при уничтожении соответствующей переменной.

При хранении стека в виде списка память освобождается при удалении каждого элемента.

4. Что происходит с элементами стека при его просмотре?

Стек очищается, т.к. чтобы просмотреть элемент, его нужно достать из списка.

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

   Рализовывать стек при помощи массива эффективнее при заполненности этого массива больше, чем не 50%, поскольку благодаря произвольному доступу доступ происходит быстрее, память выделяется меньшее количество раз (только когда нужно увеличить размер массива) и удаляется один раз, в то время как при реализации при помощи списка память на каждый элемент выделяется и удаляется индивидуально. Получается, что эффективность того или другого метода зависит от размера самого стека, а также типа доступа к элементам (произвольному или послеловательному) и от метода выделения памяти.

Выводы по проделанной работе

В ходе лабораторной работы я научился создавать, использовать и удалять стек, а также освоил два способа реализации стека: при помощи массива и при помощи списка. Также я выяснил, что зачастую хранение стека в виде массива является более эффективным способом, чем хранение в виде списка (при заполненности массива более, чем на 50%), однако бывает и наоборот. Однако ффективность использования динамического массива можно повысить, зная пределы возможных размеров или оптимизируя алгоритм выделения памяти (например, не увеличивать в два раза, а добавлять 100 элементов).