Лабораторная работа №2 по дисциплине «Типы и структуры данных »

Работа со стеком

Условие задачи:

Реализовать операции работы со стеком, который представлен в виде массива (статического или динамического) и в виде односвязного списка, оценить преимущества и недостатки каждой реализации, получить представление о механизмах выделения и освобождения памяти при работе с динамическими структурами данных.

Исходные данные:

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.

Проверить правильность расстановки скобок трех типов (круглых, квадратных и фигурных) в выражении.

Ввод данных:

Пользователь для работы с программой использует консольное меню.

```
Для редактирования стека нажмите 1 и введите дополнительные элементы последовательности
Для печати стека нажмите 2
Для добавления элемента в стек нажмите 3 и введите символ (только скобки)
Для удаления элемента из стека нажмите 4
Для вывода адресов нажмите 5
Для выхода нажмите -1 или любое другое число
```

Вывод:

В зависимости от выбранного пункта меню, вывод программы будет отличаться.

Возможные ошибки:

1) Во время работы программы произошла попытка чтения из пустого стека
   Текст ошибки: "Error: Empty stack."

2) Превышен максимальный размер стека

Текст ошибки: "Error: Full stack"

Структура данных для стека на основе списка

```
struct stack
{
  int value;
  struct stack *next;
};
```

Структура данных для стека на основе массива

```
StackArray()
{
  pos=-1;
  maxlen=10;
  array=new int[maxlen];
}
```

Функции

int Pop() //возвращает элемент стека
int Push(int val) //добавляет в стек элемент и возвращает код успеха
int Size() //размер занимаемый стеком в памяти
void Show() //показывает на экран результат содержимое стека
int IsEmpty() //свойство позволяющее узнать свободен ли стек

Тесты

1) Стандартные входные значения

Проверяемая строка	Результат проверки:
{(){}[]}	Правильная последовательность скобок
)()(	Error: Empty stack Неправильная последовательность скобок
(()(	Неправильная последовательность скобок

Сравнение эффективности реализаций списков:
Время выполнения программы:

Кол-во эл-тов	Время-работы стека-массива, мс	Время работы стека-списка, мс	Список медленнее массива (в процентах)
10	3	8	267%
50	6	14	233%
100	19	25	132%
500	25	70	280%
1 000	41	117	285%
5 000	313	1039	332%
10 000	310	1332	430%
50 000	1526	5786	379%
100 000	2984	9963	334%
500 000	15437	51122	331%
1 000 000	29186	108542	372%

Реализация стека на массиве более эффективна по времени более чем в 2,5 раза.

Оценка памяти:

Для хранения каждого нового элемента при реализации через список требуется на 8 байт больше места чем при добавление его реализацию на массиве.

Фрагментация памяти не происходит, так как в ходе ряда тестов было обнаружено, что после освобождения из-под элемента списка память может быть выделена еще раз для нового элемента.

Вывод

Эффективность реализации стека на списке проявляется, только при значительном количестве элементов и при хранение в каждом его элементе значительного объема информации, так как значительная часть память выделяется под хранение указателей, что порой больше размера хранимых

данных в несколько раз. К недостаткам реализации стека на списка можно отнести неравномерность распределения памяти, что иногда приводит к фрагментации. При переполнении стека-массива, необходимо совершать дополнительно копирование стека и расширять размер массива, что достаточно трудоемко на значительном количестве элементов.

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.

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

Если стек основан на массиве, то для его хранения отводится непрерывный ограниченный блок памяти, характеризуемый заранее указанным количеством элементов.

В случае реализации стека на списке, память доступная для хранения элементов ограничена только объемом свободной RAM. Память выделяется каждый раз при добавлении нового эл-та.

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

При реализации в виде списка, то при удалении очередного эл-та происходит освобождение области памяти, которую он занимал.

Если стек реализован в виде массива, то память освобождается после окончания работы со стеком: освобождается весь массив.

4. Что происходит с элементами стека при его просмотре?

Операция чтения производится с помощью операции извлечения эл-та из стека.

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

Эффективность определяется типом хранимых элементов и их количеством. При хранении в стеке основанном на списке легких элементов (например int), происходит неэффективное использование памяти, так как

значительная доля памяти выделяется не под значение элемента, а указатель на него. При значительном объеме выделяемой памяти для хранения элемента, данные накладные расходы уже не оказывают особого влияния. При использование стека-списка, происходит выделение памяти в разных областях оперативной памяти. Это может привести к возникновению пробелов в памяти, что не позволит эффективно использовать предоставленную программе память. Также имеет значение, известно ли заранее максимальное кол-во эл-тов в стеке. Стек-список ограничен лишь объемом оперативной памяти компьютера.