Министерство науки и высшего образования 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»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4 ПО ДИСЦИПЛИНЕ: ТИПЫ И СТРУКТУРЫ ДАННЫХ

“Работа со стеком”

Студент **Зернов Георгий Павлович**

Группа **ИУ7-34Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Зернов Г.П.**

Преподаватель _____ **Никульшина Т.А.**

2024

Условие задачи

Реализовать операции работы со стеком, который представлен в виде статического массива и в виде односвязного линейного списка; оценить преимущества и недостатки каждой реализации; получить представление о механизмах выделения и освобождения памяти при работе со стеком. При реализации стека массивом располагать два стека в одном массиве. Один стек располагается в начале массива и растет к концу, а другой располагается в конце массива и растет к началу. Заполнять и освобождать стеки произвольным образом с экрана. Элементами стека являются вещественные числа. Списком реализовать один стек.

Техническое задание

Реализуемая в программе задача

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

Входные данные

В случае выбора опции в меню целое число, соответствующее одной из указанных опций:

- 1 - Добавить число в стек слева
- 2 - Удалить число из стека слева
- 3 - Вывести вершину левого стека
- 4 - Вывести левый стек
- 5 - Добавить число в стек справа
- 6 - Удалить число из стека справа
- 7 - Вывести вершину правого стека
- 8 - Вывести правый стек
- 9 - Добавить число в стек-список
- 10 - Удалить число из стека-списка
- 11 - Ввести вершину стека-списка
- 12 - Вывести стек-список
- 13 - Освобожденные адреса
- 14 - Анализ эффективности
- 15 - Выход

В случае добавления элемента в стек:

- Добавляемый элемент – вещественное число.

Выходные данные

1. Вывести вершину левого стека – выведенная на экран вершина левого стека.
2. Вывести левый стек – выведенный на экран левый стек.
3. Вывести вершину правого стека – выведенная на экран вершина правого стека.
4. Вывести правый стек – выведенный на экран правый стек.
5. Ввести вершину стека-списка – выведенная на экран вершина стека.
6. Вывести стек-список – выведенный на экран стек.
7. Освобожденные адреса – выведенные на экран адреса освобожденных элементов.
8. Анализ эффективности – выведенные на экран результаты измерения времени работы операций push/pop для разных реализаций стека.

Способ обращения к программе

Чтобы обратиться к программе, пользователю необходимо запустить файл app.exe.

Возможные аварийные ситуации или ошибки пользователя

- Введено невалидное значение опции – вывод на экран информирующего сообщения о неверном вводе и предоставление пользователю возможности повторно ввести значение.

- Введено невалидное значение добавляемого элемента – вывод на экран информирующего сообщения о неверном вводе и предоставление пользователю возможности повторно ввести значение.
- Попытка добавления в стек максимальной длины элемента – сообщение об ошибке.
- Попытка удалить элемент из пустого стека или вывести его вершину – сообщение об ошибке.
- Нехватка памяти при её выделении – соответствующее сообщение об ошибке и остановка текущего процесса с возвратом программы в состояние до его начала.

Внутренние структуры и типы данных

Структура, представляющее тип данных “стек на массиве”.

- double *topl_p – указатель на вершину левого стека.
- double *topr_p – указатель на вершину правого стека.
- double data[MAX_ARR_STACK_SIZE] – массив значений.

Листинг структуры:

```
typedef struct
{
   double *topl_p;
   double *topr_p;
   double data[MAX_ARR_STACK_SIZE];
} array_stack_t;
```

Структура, представляющее тип данных “узел списка”.

- double data – значение узла.
- struct node_t* next – указатель на следующий узел.

Листинг структуры:

```
typedef struct node_t
{
   double data;
   struct node_t* next;
} node_t;
```

Структура, представляющее тип данных “стек на списке”.

- node_t *head – указатель на начало списка.
- size_t size – текущая длина списка.

Листинг структуры:

```
typedef struct
{
   node_t *head;
   size_t size;
} list_stack_t;
```

Алгоритм программы

- Общий алгоритм программы:

1. Запуск основного цикла программы.
2. Вывод меню.
3. Запрос у пользователя опции для выполнения.
4. Выполнение выбранной опции.
5. Переход к новой итерации цикла или завершение основного цикла при соответствующем выборе пользователя.

- Алгоритм добавления элемента в стек на массиве:

1. Проверка наложения стеков
2. Запись элемента по указателю на вершину и его увеличение

- Алгоритм удаления элемента из стека на массиве:

1. Проверка выхода за границы массива

2. Уменьшение указателя на текущий элемент

- Алгоритм добавления элемента в стек на списке:

1. Создание узла списка с указанным значением
2. Запись элемента по указателю на вершину и её изменение

- Алгоритм удаления элемента из стека на списке:

1. Замена указателя на начало списка на указатель на следующий узел
2. Разрушение текущего узла

Функции программы

- void create_arr_stack(array_stack_t *stack) – инициализирует стек на массиве.
- int pushl(array_stack_t *stack, double value) – добавляет элемент в левый стек, возвращает код ошибки.
- int pushr(array_stack_t *stack, double value) – добавляет элемент в правый стек, возвращает код ошибки.
- int popl(array_stack_t *stack) – удаляет элемент из левого стека, возвращает код ошибки.
- int popr(array_stack_t *stack) – удаляет элемент из правого стека, возвращает код ошибки.
- int topl(array_stack_t *stack, double *value) – получает элемент с вершины левого стека, возвращает код ошибки.
- int topr(array_stack_t *stack, double *value) – получает элемент с вершины правого стека, возвращает код ошибки.
- void create_stack(list_stack_t *stack) – инициализирует стек на списке.
- void destroy_stack(list_stack_t *stack) – разрушает стек на списке.
- int push(list_stack_t *stack, double value) – добавляет элемент в стек, возвращает код ошибки.

- int pop(list_stack_t *stack) – удаляет элемент из стека, возвращает код ошибки.
- int top(list_stack_t *stack, double *value) – получает элемент с вершины стека, возвращает код ошибки.
- int time_measurement(void) – приводит анализ времени работы стеков.
- void print_menu(void) – выводит меню.
- int main(void) – выполняет основной алгоритм программы.

Тесты

Тестовый случай	Ожидаемый результат
Введена невалидная опция	Соответствующее сообщение об ошибке, возможность снова ввести требуемое значение.
Попытка добавить элемент в стек при достижении максимальной глубины	Сообщение о соответствующей ошибке
Попытка удалить элемент из стека, когда он пуст	Сообщение о соответствующей ошибке
Попытка вывести стек, когда он пуст	Сообщение о соответствующей ошибке
Попытка вывести вершину стека, когда он пуст	Сообщение о соответствующей ошибке
Попытка добавить элемент в правый или левый стек, когда суммарно стеки занимают весь массив	Сообщение о соответствующей ошибке
Добавление элемента в стек Стек: 1 2 3 4 Добавляемый элемент: 4	Полученный стек: 4 1 2 3 4
Удаление элемента из стека Стек: 6 7 8 9	Полученный стек: 7 8 9

Просмотр вершины стека Стек: 99 1 2 4	Вершина стека: 99
Вывод стека Стек: 42 43 44 45 46 47	Выведенный стек: 42 43 44 45 46 47

Анализ эффективности

Производится 10^6 операций добавления элемента, потом производится столько же удалений элемента и считается среднее для соотвествующей операции. Элементами стека являются вещественные числа.

Стек на массиве:

- push — 13(нс)
- pop — 5(нс)
- Размер элемента 8(байт)

Стек на списке:

- push — 81(нс)
- pop — 21(нс)
- Размер элемента 16(байт)

Таким образом стек на массиве в более чем 4 раза эффективнее по времени. Но при этом при размере стека на списке до 50% от размера стека на массиве стек на списке будет эффективнее по памяти.

Вывод

Стек на массиве значительно эффективнее по времени, но при этом требует изначально большого количества выделенной памяти и до определённого процента заполнения стека на массиве стек на списке будет эффективнее, затем стек на массиве будет эффективнее как по времени так и по памяти.

Контрольные вопросы:

Что такое 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?

При реализации стека на массиве память выделяется сразу под весь массив. При реализации стека на списке память выделяется под каждый элемент по отдельности и освобождается по мере удалении элементов из стека.

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

При реализации стека на массиве память не освободжается, меняется лишь указатель на вершину стека. При реализации стека на списке при удалении разрушается узел списка.

Что происходит с элементами стека при его просмотре?

Значение считывается только с вершины стека. Для получения значения следующего элемента надо извлечь предыдущий.

Каким образом эффективнее реализовывать стек? От чего это зависит?

Эффективность реализации стека зависит от решаемой задачи, так стек на массиве эффективнее по времени и по памяти, но его размер ограничен, и максимальная необходимая глубина должна быть известна заранее.

Стек на списке менее эффективен как по времени так и по памяти в общем случае, но может применяться в случаях когда начальная глубина стека неизвестна или большую часть времени стек имеет малую заполненность.