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
КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ЛАБОРАТОРНАЯ РАБОТА №4.

Студент Мицевич Максим Дмитриевич
фамилия, имя, отчество

Группа ИУ7-31Б

Студент _____ Мицевич М. Д.
подпись, дата *фамилия, и.о.*

Проверяющий _____ _____
подпись, дата *фамилия, и.о.*

Оценка _____

1. Описание условия 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.
Ввести целые числа в 2 стека. Используя третий стек отсортировать все введенные данные.

2. Техническое задание

1. Описание исходных данных
Все входные данные являются читаются в виде строк и переводятся в числа. В цикле вводятся с клавиатуру числа от 0 до 11, которые определяют дальнейшие поведение программы.

Формат ввода:
Возможные пункты меню, которые можно выбрать, введя число с клавиатуры:
0 - выход из программы
1 - чтение стеков в виде векторов
2 - чтение стеков в виде списков
3 - печать стеков виде векторов
4 - печать стеков в виде списков
5 - печать массива свободных областей
6 - добавление элемента в любой стек
7 - взятие элемента из любого стека
8 - сортировка двух стеков в виде векторов
9 - сортировка двух стеков в виде списков
10 - сравнение эффективности алгоритмов

При чтении стека в виде вектора требуется ввести его максимальный размер и элементы. Окончанием ввода является любая буква. При чтении стека в виде списка требуется ввести только его элементы, окончанием ввода является любая буква. При добавлении элемента в любой из стеков требуется ввести сам элемент и цифрами от одного до четырех выбрать стек, в который добавлять элемент. Аналогичный выбор требуется произвести при удалении элемента из стека.

2. Описание результата программы
При корректных входных данных будет выполнен один из пунктов меню, иначе выведено сообщение об ошибке.
Формат вывода:
При извлечении элемента с вершины стека печатается его значение. При печати стека в столбик печатаются его элементы и адреса, по котоым они хранятся:
<elem_value> <elem_addres>

Ограничения:
- Время работы сортировки содержит максимум 7 значащих цифр

3. Описание задачи, реализуемой программой
Программа выполняет сортировку информации из двух стеков в один, используя разные алгоритмы хранения стеков.

4. Способ обращения к программе
Обращение к программе происходит с помощью стандартных потоков ввода и вывода.

5. Описание возможных аварийных ситуаций и ошибок пользователя
Ошибки операционной системы при выделении динамической памяти, снятие элемента с вершины пустого стека, добавление в стек, хранящийся в виде вектора, элементов больше, чем максимальное допустимое число.

3. Описание внутренних структур данных

Для хранения стека в виде вектора была выбрана данная структура:

```
typedef struct
{
elem_t *aub; //addres up bound
elem_t *alb; //addres low bound
elem_t *ps; //pointer stack
}stack_vector_t;
```

aub — указатель на верхнюю вершину стека
alb — указатель на дно стека
ps — указатель на верхний элемент стека

Для хранения стека в виде списка были выбраны следующие структуры данных:

```
struct node
{
   elem_t value;
   struct node *next;
};

typedef struct node node_t;
```

node_t — узел списка с элементами стека,
value — значение элемента,
next — указатель на следующий элемент.

```
typedef struct
{
   node_t *top; //stack top
   node_t **free_spaces; //free elements
   size_t spaces_len; //length of free spaces array
   size_t spaces_alloc; //allocated lenght of spaces array
}stack_list_t;
```

top — указатель на вершину стека
free_spaces — указатель на вектор свободных областей
spaces_len — количество элементов в этом векторе
spaces_alloc — количество элементов, под которых выделена память в этом векторе.

4. Алгоритм

Алгоритм сортировки одного стека:
1. берем элемент из стека.
2. если дополнительный стек пустой или верхний элемент этого стека меньше или равен элементу, полученного в 1 пункте, то добавляем этот элемента в вершину дополнительного стека.
3. Иначе перемещаем элементы из дополнительного стека в стек до тех пор пока он не станет пустым или пока мы не встретим элемент меньше или равный элементу их первого пункта. Добавляем в вершину дополнительного стека элемент из 1 пункта и возвращаем на место все перемещенные раннее элементы.
4. Проделываем пункты 1-3, пока стек не станет пустым
5. Перекладываем все элементы из дополнительного стека в стек.

Алгоритм сортировки двух стеков в третий:
1. Сортируем оба стека по алгоритму, описанному выше.
2. Если один из стеков пустой, то добавляем в третий стек элемент с вершины не пустого.
3. Иначе берем элементы с вершины обоих стеков, сравниваем их, добавляем в вершину третьего стека меньшей из них и берем следующий элемент из стека, с которого добавили элемент.
4. Повторяем 1- 3, пока стеки не станут пустыми.

5. Тесты

Позитивные тесты:

Входные данные	Что проверяем	Ожидаемый выходной результат
Выбор: Сортировка стеков векторов 5 1 4 -1 a 5 2 -9 7 a	Проверка сортировки	-9 -1 1 2 4 7
Выбор: сортировка стеков списков 1 4 -1 a 2 -9 7 a	Проверка сортировки	-9 -1 1 2 4 7
Выбор: добалвление элемента Стек 1 2 3 Элемент 4	Проверка на добавление элемента в стек	Стек: 4 1 2 3
Выбор: удаление элемента Стек 1 2 3	Проверка на удаление из стека	Стек: 2 3 Элемент: 1

Негативные тесты:

Входные данные	Что проверяем	Ожидаемый выходной результат
Выбор: Удаление элемента из стека Стек: пустой стек	Проверка на взятие элемента из пустого стека	Empty stack
Выбор: Добавление элемента Стек размером 5 элементов: 1 2 3 4 5	Проверка на переполнение стека	Stack overflow

6. Функции

Чтение и печать стека в виде списка
int read_lstack(stack_list_t *stc)
int write_lstack(stack_list_t *stc)

Добавление и удаление элемента в стек список
int popl(void *stc, elem_t *elem)
int pushl(void *stc, elem_t elem)

Аналогичные функции для стека в виде вектора
int vpop(void *stc, elem_t *elem)
int vpush(void *stc, elem_t elem)
int read_vstack(stack_vector_t *stc)
void write_vstack(stack_vector_t *stc)

Функции для сортировки стеков
int sort_1_stack(void *stc, void *temp, int (*push)(void *, elem_t), int (*pop)(void *, elem_t *))
int sort_2_stacks(void *stc1, void *stc2, void *stc3, int (*push)(void *, elem_t), int (*pop)(void *, elem_t *))

7. Оценка эффективности программы.

Произведем замер по времени и памяти для разного количество элментов в стеке и разного максимального размера стека на основе вектора.

Максимальный размер стека вектора — количество элементов, умноженное на 1

```
stack sizes 100
| list time | vector time | list memory | vector memory |
| 0.000414s |   0.000148s |       1608b |          824b |
stack sizes 200
| list time | vector time | list memory | vector memory |
| 0.001514s |   0.000386s |       3208b |         1624b |
stack sizes 400
| list time | vector time | list memory | vector memory |
| 0.005699s |   0.001459s |       6408b |         3224b |
```

Максимальный размер стека вектора — количество элементов, умноженное на 2

```
stack sizes 100
| list time | vector time | list memory | vector memory |
| 0.001619s |   0.000396s |       1608b |         1624b |
stack sizes 200
| list time | vector time | list memory | vector memory |
| 0.005913s |   0.001486s |       3208b |         3224b |
stack sizes 400
| list time | vector time | list memory | vector memory |
| 0.023756s |   0.006085s |       6408b |         6424b |
```

Максимальный размер стека вектора — количество элементов, умноженное на 4

```
stack sizes 100
| list time | vector time | list memory | vector memory |
| 0.007450s |   0.001850s |       1608b |         3224b |
stack sizes 200
| list time | vector time | list memory | vector memory |
| 0.025166s |   0.006364s |       3208b |         6424b |
stack sizes 400
| list time | vector time | list memory | vector memory |
| 0.093564s |   0.023685s |       6408b |        12824b |
```

Максимальный размер стека вектора — количество элементов, умноженное на 8

```
stack sizes 100
| list time | vector time | list memory | vector memory |
| 0.024033s |   0.005998s |       1608b |         6424b |
stack sizes 200
```

```
| list time | vector time | list memory | vector memory |
| 0.093401s |   0.022835s |       3208b |        12824b |
stack sizes 400
| list time | vector time | list memory | vector memory |
| 0.405991s |   0.094514s |       6408b |        25624b |
```

Стек на основе списка всегда проигрывает стеку на основе вектора по времени, так как на каждую операцию push и pop в списке происходит выделение или очистка динамической памяти, в то время как в стеке на основе вектора происходит просто изменение адреса и запись элемента.
Касательно памяти, стек на основе списка начинает выигрывать, если стек на основе вектора заполнен меньше, чем на половину от своего максимального размера.

8. Выводы

Если при решении задачи основная цель — улучшение времени работы программы, то стоит использовать стек на основе вектора. Если точное количество элементов в стеке заранее не известно, а его размер большой, то в целях экономии памяти может быть целесообразнее использовать стек на основе списка. Также стоит отметить, что количество элементов в стеке-списке ограничена только количеством динамической памяти, которую может выделить операционнная система под нашу программу, а не каким-то конкретным числом, как в случае стека-вектора.

Ответы на контрольные вопросы:

1. Что такое стек?
   Стек — это последовательный список с переменной длинной, добавление элементов в который и исключение элементов из которого происходит с одной строны — его вершины. Стек действует по принципу Last in first out.
2. Каким образом и сколько памяти выделяется под хранение стека при различной его реализации?
   При реализации стека на основе вектора память под него выделяется при его создании. Память выделяется сплошным куском, размер выделяемой памяти равен sizeof(elem_t) * <максимальное количество элементов в стеке>
   При реализации стека на основе списка память выделяется в процессе его обработки, то есть при каждом вызове push создается элемент узла списка, а при каждом pop он удаляется. Размер памяти пропорционален количеству элементов в стеке — sizeof(node_t) * <количество элементов в стеке>
3. Каким образом освобождается память при удалении элемента стека при различной его реализации?
   В стеке на основе вектора просто перемещается указатель в векторе, а память очищается, когда стек уже не нужен.
   В стеке на основе списка память под элементом каждый раз очищается при снятии этого элемента со стека
4. Что происходит с элементами стека при его просмотре?
   Для того чтобы получить элемент, его надо снять с вершины стека, то есть после просмотра стека, он становится пустым
5. Каким образом эффективнее реализовать стек? От чего это зависит?
   См. «Вывод»