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

Вариант 1

Студент **Абдуллаев Ш. В.**

Группа **ИУ7-34Б**

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

Студент _____ **Абдуллаев Ш. В.**

Преподаватель _____ **Силантьева А. В.**

2024

Описание условия задачи

Создать программу работы со стеком, выполняющую операции добавления, удаления элементов и вывод текущего состояния стека. Реализовать стек: а) статическим массивом (дополнительно можно реализовать динамическим 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. Распечатайте убывающие серии последовательности целых чисел в обратном порядке.

Описание исходных данных

Исходные данные представляют собой стек в виде статического массива и в виде односвязного линейного списка. Программа поддерживает несколько способов ввода стеков:

- Добавить один элемент (ввод поэлементно):
  Пользователь вводит элемент, который добавляется в стек.
- Ввод данных вручную:
  Пользователь вводит количество элементов и сами элементы, которые добавляются в стек.

Имеются следующие ограничения:

- Ограничение по формату ввода данных: программа ожидает ввод данных в формате целых чисел. Это означает, что пользователь должен вводить только целочисленные значения при выполнении операций, связанных с добавлением элементов в стек. Ввод данных в других форматах может привести к некорректным результатам или ошибкам выполнения программы.
- Ограничение по размеру стека: программа может иметь ограничение на максимальный размер стека, как это определено в реализации стека (в случае стека на основе массива). Пользователь должен быть осторожен при

добавлении элементов в стек, чтобы избежать переполнения стека и потенциальных ошибок программы.

- Ограничение по возможным операциям: Некоторые операции могут быть ограничены и доступны только при определенных условиях. Например, удаление элемента из пустого стека может быть недопустимой операцией.

Дополнительные ошибки:

Несоответствие типа данных при вводе (например, ввод букв вместо чисел) приводит к завершению программы с сообщением о некорректных данных.

Допустимый ввод	**Недопустимый ввод**
Выберите количество элементов: 3	Выберите количество элементов: -1
Выберите команду: 1	Выберите команду: q
Введите элементы: 3 3 1	Введите элементы: 1 q 1
Введите через пробел индексы строки, столбца и значение: 0 0 1	Введите через пробел индексы строки, столбца и значение: -1 a 99

Таблица 1. Примеры ввода

Описание результатов

1. Выйти из программы: пользователь завершает выполнение программы.
2. Добавить элемент в стек (с использованием массива): пользователь может добавить новый элемент в стек, который реализован на основе массива.
3. Удалить элемент из стека (с использованием массива): программа удаляет верхний элемент из стека, реализованного с использованием массива.
4. Вывести стек (с использованием массива): программа выводит все элементы стека, реализованного с использованием массива, на экран.
5. Добавить элемент в стек (с использованием списка): пользователь добавляет новый элемент в стек, который реализован на основе связного списка.
6. Удалить элемент из стека (с использованием списка): программа удаляет верхний элемент из стека, реализованного с использованием списка.

7. Вывести стек (с использованием списка): программа выводит все элементы стека, реализованного с использованием списка, на экран.
8. Ввод элементов в стек (с использованием массива): пользователь может ввести несколько элементов сразу в стек, реализованный на основе массива.
9. Распечатать убывающие серии последовательности целых чисел в обратном порядке (с использованием массива): программа находит и выводит серии убывающих чисел, введенных пользователем, в обратном порядке, используя стек на основе массива.
10. Ввод элементов в стек (с использованием списка): пользователь вводит несколько элементов сразу в стек, реализованный на основе связного списка.
11. Распечатать убывающие серии последовательности целых чисел в обратном порядке (с использованием списка): программа находит и выводит серии убывающих чисел, введенных пользователем, в обратном порядке, используя стек на основе списка.
12. Сравнить производительность стеков с использованием массива и списка: программа измеряет и сравнивает производительность операций стека на основе массива и стека на основе списка.
13. Вывести массив свободных областей памяти: программа выводит информацию о свободных областях памяти, доступных для операций стека.

Описание задачи, реализуемой программой

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.

Способ обращения к программе

Обращения к программе пользователем происходит с помощью вызова исполняемого файла (app.exe).

Описание возможных аварийных ситуаций и ошибок пользователя

1. Ошибка ввода команды (INPUT_COMMAND_ERR): программа обнаружила ошибку в формате или содержании введенной команды.
2. Ошибка значения при вводе (INPUT_VALUE_ERR): программа получила некорректное значение от пользователя и не может продолжить выполнение.
3. Ошибка выделения памяти для инициализации списка (INIT_LIST_ALLOC_ERR): программа не смогла выделить память для создания нового списка.
4. Ошибка длины ввода (LEN_INPUT_ERR): введенная длина данных не соответствует ожидаемому формату или допустимому диапазону.
5. Ошибка ввода элемента (EL_INPUT_ERR): программа обнаружила ошибку при вводе значения элемента, которое не соответствует ожидаемому типу или формату.

Описание внутренних структур данных

Программа содержит в себе структуру, которая используется для хранения стека в виде статического массива, которая представлена в Листинге 1.

```
typedef struct
{
   int data[MAX_SIZE];
   int ps;
} arr_stack;
```

Листинг 1. Структура arr_stack

Рассмотрим каждое поле структуры:

1. data: массив целых чисел типа int[MAX_SIZE], который хранит элементы стека.
2. ps: целочисленное значение типа int, который указывает индекс верхнего элемента стека. Это значение показывает, какой элемент в массиве data является текущим верхом стека.

Также программа содержит в себе структуру list_stack, которая используется для хранения стека в виде списка, которая представлена в Листинге 2.

```
typedef struct Node
{
    int value;
    struct Node *next;
} Node;

typedef struct
{
    Node *ps;
    int len;
} list_stack;
```

Листинг 2. Структура list_stack

Рассмотрим каждое поле структур:

1. value: целочисленное значение типа int, которое содержится в текущем узле списка.
2. next: указатель на предыдущий узел типа struct Node *. Указатель на предыдущий элемент (или узел) в связном списке. Если это первый элемент в списке, next будет указывать на NULL.
3. ps: указатель на узел типа Node *. Указатель на верхний элемент стека. Этот указатель указывает на первый узел в связном списке (верх стека).
4. len: целочисленное значение типа int. Хранит текущую длину стека, т.е. количество элементов в стеке.

Описание алгоритма

1. Инициализация системы и выделение памяти.
2. Основной цикл программы начинается. Пользователю предоставляется меню с различными опциями, и программа ожидает ввода выбора пользователя.

3. В зависимости от выбора пользователя, программа выполняет следующие действия:
   - Выйти из программы
   - Добавить элемент в стек (с использованием массива)

     Алгоритм функции заключается в том, чтобы добавить новый элемент в структуру данных, представляющую стек, увеличив указатель на текущую позицию и присвоив элемент по новой позиции в массиве. Сначала функция использует текущий индекс для записи нового значения в массив, а затем увеличивает индекс для следующей вставки.
   - Удалить элемент из стека (с использованием массива)

     Алгоритм функции заключается в извлечении элемента из стека. Сначала происходит уменьшение указателя на текущую позицию, чтобы получить индекс последнего добавленного элемента. Затем значение этого элемента возвращается. Это действие также уменьшает размер стека, указывая на новый верхний элемент.
   - Вывести стек (с использованием массива)
   - Добавить элемент в стек (с использованием списка)

     Алгоритм функции заключается в добавлении нового элемента в стек, реализованный с использованием связанного списка. Сначала создается новый узел с переданным значением. Если создание узла не удалось, функция возвращает ошибку. Далее указатель на предыдущий узел нового элемента устанавливается на текущий верхний элемент стека, после чего новый узел становится верхом стека. В конце увеличивается длина стека.
   - Удалить элемент из стека (с использованием списка)

     Алгоритм функции заключается в извлечении элемента из стека, реализованного с использованием связанного списка. Сначала проверяется, не пуст ли стек (если стек пуст, функция возвращает ошибку). Затем сохраняется указатель на верхний элемент стека, и верхним элементом становится предыдущий узел. Значение извлеченного элемента

сохраняется, узел добавляется в список для освобождения памяти, и память за него освобождается. В конце уменьшается длина стека, и возвращается извлеченное значение.

- Вывести стек (с использованием списка)
- Ввод элементов в стек (с использованием массива)
- Распечатать убывающие серии последовательности целых чисел в обратном порядке (с использованием массива)

  Алгоритм функции заключается в том, чтобы вывести убывающие серии чисел из стека. Функция начинается с извлечения верхнего элемента стека. Затем она поочередно извлекает остальные элементы и проверяет, является ли текущий элемент меньше или равным предыдущему. Если это так, серия заканчивается, и начинается новая строка (если включен флаг `v`). Если текущий элемент больше, то продолжается вывод чисел в текущей серии. В конце выводится последний элемент серии и, при необходимости, завершается вывод пустой строки.
- Ввод элементов в стек (с использованием списка)
- Распечатать убывающие серии последовательности целых чисел в обратном порядке (с использованием списка)

  Алгоритм функции аналогичен предыдущему, но для стека, реализованного с использованием связанного списка. Функция начинается с извлечения верхнего элемента стека. Затем она поочередно извлекает элементы из стека, сравнивая их с предыдущим значением. Если текущий элемент не больше предыдущего, серия чисел завершается, и начинается новая строка (если включен флаг `v`). В противном случае продолжается вывод чисел в текущей серии. В конце выводится последний элемент серии и, при необходимости, завершается вывод пустой строки.
- Сравнить производительность стеков с использованием массива и списка
- Вывести массив свободных областей памяти

4. После выполнения каждой операции программа возвращает пользователя в главное меню, где он может выбрать следующее действие.

Набор тестов

Описание	**Результат**
Добавить элемент в стек Элемент: 3	Успешное считывание добавления элемента в стек Добавленный элемент: 3
Удалить элемент из стека Стек: 1 2 3	Успешное удаление элемента из стека Удалённый элемент: 3
Вывести стек Стек: 1 4 5 3	Успешный вывод стека: 1 4 5 3
Распечатать убывающие серии последовательности целых чисел в обратном порядке Стек: 9 10 5 4 0 9 8 0 1 1	Успешный вывод убывающей серии: 1 1 0 8 9 0 4 5 10 9
Вывести массив свободных областей памяти	Успешный вывод свободных областей памяти. Список освобожденных узлов: { 0x4a96910 } [0]: 1

Таблица 2. Набор позитивных тестов

Описание	**Результат**
Попытка ввести символы вместо команды	Возврат ошибки INPUT_COMMAND_ERR
Попытка ввести не целое число	Возврат ошибки INPUT_VALUE_ERR
Неудачная аллокация памяти	Возврат ошибки INIT_LIST_ALLOC_ERR
Попытка ввести не длину массива	Возврат ошибки LEN_INPUT_ERR

Попытка ввести символы вместо числа	Возврат ошибки EL_INPUT_ERR

Таблица 3. Набор негативных тестов

Оценка эффективности

При запуске программы N = 100 раз, были получены следующие данные:

Размер стека	Время, нс		Объем памяти, байт	
	Массив	Список	Массив	Список
10	11434	52357	44	132
50	43269	244234	204	612
100	83834	558996	404	1212
500	460660	2454889	2004	6012
1000	881804	6294540	4004	12012

Таблица 4. Эффективность стеков в разных представлениях

Выводы

В данной лабораторной работе было проведено равнение производительности стеков на основе массива и связанного списка. Были рассмотрены операции на разных размера стека: 10, 50, 100, 500, 1000 элементов.

Результаты сравнения были представлены в виде таблицы, в которой указаны размер стека, объем занимаемой памяти для массива и связанного списка, а также время выполнения операций для обоих типов стеков. В результате анализа, мы видим, что стек на основе массива показывает лучшую эффективность во всех случаях, а именно в ~6 раз лучше по времени и 3 раз лучше по памяти. Стек на основе массива выигрывает как в скорости, так и в памяти.

В итоге, стек на основе массива чаще всего более эффективен с точки зрения времени выполнения и использования памяти, особенно при больших размерах стека.

Контрольные вопросы

1. Что такое стек?

   Стек – это последовательный список с переменной длиной, в котором включение исключение элементов происходит только с одной стороны – с его вершины. Стек функционирует по принципу: последним пришел – первым ушел, Last In – First Out (LIFO).

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

   Реализации стека могут быть разными, и количество выделяемой памяти зависит от конкретной реализации. Вот несколько примеров:

   - Стек на основе массива:

     Память выделяется под массив фиксированного размера, который используется для хранения элементов стека. Размер массива определяется заранее, и вся память для стека выделяется одновременно.

   - Стек на основе связанного списка:

     Здесь память выделяется динамически при добавлении каждого элемента. Каждый элемент (узел) стека содержит указатель на предыдущий элемент и значение. Таким образом, память выделяется по мере необходимости, и стек может расти динамически.

   Количество памяти, выделенной для стека, зависит от его размера и структуры данных, используемых для его реализации.

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

   При удалении элемента из стека, освобождение памяти происходит по-разному в зависимости от реализации стека:

   - Стек на основе массива:

     При удалении элемента из массива, память не освобождается явным образом. Просто уменьшается указатель (индекс) на вершину стека. Это позволяет заменить старое значение новым при следующем добавлении элемента.

- Стек на основе связанного списка:

  При удалении элемента из стека на основе связанного списка, удаляется соответствующий узел, и память, выделенная под этот узел, освобождается с помощью функции free.

4. Что происходит с элементами стека при его просмотре?

   При просмотре элементов стека, они остаются в стеке в неизменном виде. Просто читается значение элемента с вершины стека, но этот элемент не удаляется. Таким образом, элементы стека не изменяются при его просмотре.

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

   Эффективность реализации стека зависит от конкретных требований и ограничений приложения:

   - Стек на основе массива более эффективен по памяти, если известен максимальный размер стека заранее. Он может работать быстрее, чем связанный список, при доступе к элементам по индексу. Однако его размер ограничен.
   - Стек на основе связанного списка более гибок и способен динамически расти и сжиматься в зависимости от потребности. Это подходит для случаев, когда размер стека заранее неизвестен или может изменяться.

   Эффективность также зависит от операций, которые выполняются чаще: добавление, удаление или просмотр элементов, а также от общей структуры программы