

ФАКУЛЬТЕТ: Информатика и системы управления

КАФЕДРА: Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

**«Работа со стеком»**

Студент: Зайцева Алена Андреевна

Группа ИУ7 – 32Б

*2020 г.*

Оглавление

Оглавление

[1. Описание условия задачи 3](#_Toc54285848)

[2. Описание ТЗ 4](#_Toc54285849)

[1) Описание исходных данных: 4](#_Toc54285850)

[2) Описание результатов: 4](#_Toc54285851)

[3) Описание задачи, реализуемой программой: 4](#_Toc54285852)

[4) Способ обращения к программе: 5](#_Toc54285853)

[5) Описание возможных аварийных ситуаций и ошибок пользователя. 5](#_Toc54285854)

[3. Описание внутренних структур данных. 6](#_Toc54285855)

[4. Описание алгоритма 6](#_Toc54285856)

[5. Набор тестов с указанием, что проверяется 7](#_Toc54285857)

[6. Оценка эффективности 9](#_Toc54285858)

[7. Выводы по проделанной работе 11](#_Toc54285859)

[8. Ответы на вопросы 12](#_Toc54285860)

# Описание условия задачи

Создать программу работы со стеком, выполняющую операции

1. добавления элементов,
2. удаления элементов,
3. вывода текущего состояния стека,
4. печати убывающих серий последовательности целых чисел в обратном порядке.

Реализовать стек:

1. массивом;
2. 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.

При разработке интерфейса программы следует предусмотреть:

* указание формата и диапазона вводимых данных,
* блокирование ввода данных, неверных по типу,
* указание операций, производимых программой:
* наличие пояснений при выводе результата.

Кроме того, нужно вывести на экран время выполнения программы при реализации стека списком и массивом, а также указать требуемый объем памяти. Необходимо так же выдать на экран список адресов освобождаемых элементов при удалении элементов стека.

При тестировании программы необходимо:

* проверить правильность ввода и вывода данных (в том числе, отследить попытки ввода данных, неверных по типу);
* обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);
* проверить правильность выполнения операций;
* обеспечить вывод соответствующих сообщений при попытке удаления элемента из пустого стека;
* отследить переполнение стека.

При реализации стека в виде списка необходимо:

* ограничить доступный объем оперативной памяти путем указания:
  + максимального количества элементов в стеке;
  + максимального адреса памяти, превышение которого будет свидетельствовать о переполнении стека;
* следить за освобождением памяти при удалении элемента из стека.

# Описание ТЗ

Внешняя спецификация:

## Описание исходных данных:

* 1. Номер команды: целое число в диапазоне от 0до 12.
  2. Данные, необходимые для определенной команды
* Максимальное количество элементов в стеке (целое, >=0)
* Начальное количество элементов в стеке (целое, >=0 и <= максимального)
* Значение элемента стека (целое)

## Описание результатов:

1. Результат выполнения команды
2. Время выполнения программы при данной реализации стека списком и массивом, а также требуемый объем памяти.

## Описание задачи, реализуемой программой:

Программа выполняет следующие команды, указанные при её первом запуске.

0) Выход из программы

Операции над стеком, реализованным в виде массива:

* 1. Создать стек-массив и ввести начальные значения.
  2. Добавить элемент в стек-массив.
  3. Удалить элемент из стека-массива (реализуется как pop – при удалении выведется значение удаляемого элемента).
  4. Вывести убывающие серии последовательности целых чисел из стека-массива в обратном порядке и количественную характеристику обработки. (при этом стек-массив полностью опустошается)
  5. Вывести текущее состояние стека-массива (при этом сам стек-массив не меняется)

Операции над стеком, реализованным в виде линейного односвязного списка:

* 1. Создать стек-список и ввести начальные значения.
  2. Добавить элемент в стек-список.
  3. Удалить элемент из стека-списка (реализуется как pop – при удалении выведется значение удаляемого элемента) и вывести массив освободившихся адресов
  4. Вывести массив освободившихся адресов
  5. Вывести убывающие серии последовательности целых чисел из стека-списка в обратном порядке и количественную характеристику обработки. (при этом стек-список полностью опустошается и удаляется)
  6. Вывести текущее состояние стека-списка с указанием адресов элементов. (при этом сам стек-список не меняется)
  7. Вывести информацию об объеме занимаемой памяти в каждой реализации

## Способ обращения к программе:

Программа запускается из папки, содержащей исполняемый файл, через терминал.

## Описание возможных аварийных ситуаций и ошибок пользователя.

Возможные ошибки пользователя (для обеих реализаций стека):

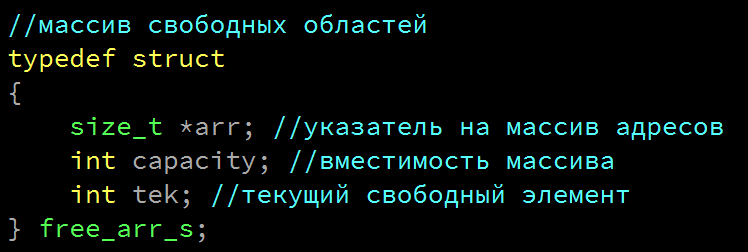
1. Ввод несуществующей команды: нецелочисленное значение или целое, меньшее 0 или большее 12
2. Обработка стека до его создания
3. Создание стека, когда в программе уже существует стек (того же типа)
4. Ввод неверного максимального количества элементов в стеке (нецелочисленное значение или целое, меньшее 0)
5. Ввод неверного начального количества элементов в стеке (нецелочисленное значение или целое, меньшее 0 или большее максимального)
6. Удаление элемента из пустого стека
7. Добавление элемента в уже полный стек
8. Ввод нецелочисленного значения элемента стека

Возможные аварийные ситуации:

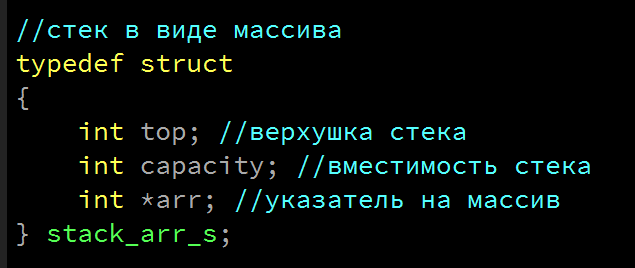
1. при вводе числового поля пользователь ничего не вводит – программа будет ждать ввода и не завершится, пока не встретит хоть один непробельный символ.
2. Компьютер не сможет выделить необходимое количество памяти – программа выведет сообщение об ошибке и завершится.

## Описание внутренних структур данных.

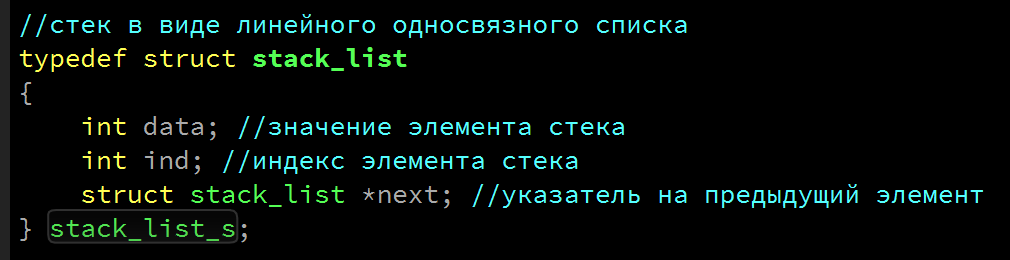
Массив свободных областей:



Стек в виде массива:



Стек в виде линейного односвязного списка



## Описание алгоритма

1. Выводится общая информация о программе и список возможных команд.
2. Пока пользователь не введет 0 (выход из программы), ему будет предложено выполнять действия с двумя реализациями стека – на основе массива или на основе линейного односвязного списка (стеки независимы друг от друга).
   1. Создается стек в выбранной реализации и заполняется начальными значениями
   2. Добавляется/удаляется элемент из стека в выбранной реализации.
   3. Выводится стек в выбранной реализации (сам стек в выбранной реализации не меняется)
   4. Выводится массив свободных областей
   5. Находятся убывающие серии, решение реализуется на выбранном представлении и выводится количественная характеристика выполнения задачи. (стек в выбранной реализации опустошается)
   6. Выводится информация об объеме занимаемой памяти в каждой реализации

На каждом этапе контролируется успешность ввода и выделения памяти, корректность последовательности действий, пустота или переполнение стека. В случае ошибки программа выводит соответствующее сообщение.

Код программы содержит комментарии по используемым в программе подпрограммам.

# Набор тестов с указанием, что проверяется

|  |  |  |
| --- | --- | --- |
| Что проверяет | Ввод | Результат |
| Несуществующая команда | 12 или -1 или 1.1 или a | Сообщение об ошибке, вывод списка возможных команд и предложение вновь ввести команду |
| Некорректное максимальное количество элементов | -1 или 2.2 или a | Сообщение об ошибке и предложение попробовать снова |
| Некорректное начальное количество элементов | -1 или 2.2 или a или (больше максимального количества элементов) | Сообщение об ошибке и предложение попробовать снова |
| Некорректный элемент стека | 1.1 или r | Сообщение об ошибке и предложение попробовать снова |
| Попытка создания стека на основе конкретной реализации при уже существующем стеке в данной реализации | (Повторная команда 1) или  (Повторная команда 6) | Сообщение о том, что стек уже существует |
| Попытка обработки стека до его создания | (Команды 2, 3, 4, 5 до команды 1) или  (Команды 7, 8, 9, 10,11 до команды 6) | Сообщение о том, что стек еще не создан |
| Переполнение стека | (команда 2) или (команда 7) при уже полном стеке | Сообщение об ошибке |
| Удаление элемента из пустого стека | (команда 3) или (команда 8) при пустом стеке | Сообщение об ошибке |
| Корректный ввод элементов стека  (команда 1) или  (команда 6) | 10  5  6 -1 3 4 3 | Сообщение об успешном заполнении стека. |
| Добавление элемента в стек при неполном стеке  (команда 2) или  (команда 7) | 101 | Сообщение об успешном добавлении в стек. |
| Удаление элемента при непустом стеке  (команда 3) или  (команда 8) |  | Сообщение об успешном извлечении из стека, значение извлеченного элемента и массив свободных адресов в случае стека-списка |
| Вывод массива освободившихся адресов  (команда 9) |  | массив свободных адресов |
| Поиск убывающих серий последовательности целых чисел  (команда 4 или 10) | (В стеке нет таких последовательностей) | Сообщение о том, что такие последовательности не найдены |
| Поиск убывающих серий последовательности целых чисел  (команда 4 или 10) | Содержание стека:  6 3 3 5 3 2 1 1 1 | 1 2 3 5  3 6  Время выполнения |

# Оценка эффективности

Оценим эффективность различных реализаций стека (в виде массива и в виде линейного односвязного списка) на различном количестве элементов в стеке (10, 100, 1000)

*Эффективность по памяти*

Каждый элемент стека-массива хранится как целое число-элемент массива (int = 4 байта), а общая информация о стеке-массиве находится в структуре

(int top- верхушка стека + int capacity-вместимость стека + int \*arr - указатель на массив = 16 байт)

Каждый элемент стека – списка хранится в виде отдельной структуры и содержит следующую информацию (int data - значение элемента стека + int ind - индекс элемента стека + struct stack\_list \*next - указатель на предыдущий элемент = 16 байт)

Память под стек-массив выделяется сразу под максимальное количество элементов, а под стек-список – только при добавлении очередного элемента.

Чтобы учесть эту особенность, будем указывать, при каком максимальном количестве занятых элементов стек–список не будет все – же проигрывать стеку-массиву.

**Объем занимаемой памяти (в байтах)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Максимальное количество элементов | Количество  занятых элементов | Стек-список | Стек-массив | Отношение список/массив |
| 10 | 10 | 16\*10 = 160 | 4\*10+16=56 | 2.9 |
| 3 | 16\*3 = 48 |  |
| 100 | 100 | 16\*100 = 1600 | 4\*100+16=416 | 3.8 |
| 26 | 16\*26 = 416 |  |
| 1000 | 1000 | 16\*1000 = 16000 | 4\*1000+16=4016 | 3.9 |
| 251 | 16\*251 = 4016 |  |

Таким образом, при полном заполнении стека стек–список выиграет по памяти у стека-массива только если в стеке не более одного элемента, но в таком случае в стеке нет необходимости.

Но если в стеке больше одного элемента, и заполнено более 30% максимального количества элементов, то стек-массив серьезно выигрывает у стека-списка. Причем с ростом процента заполненности стека, усиливается разрыв, вплоть до выигрыша по памяти стека-массива в четыре раза при полном заполнении стека.

*Эффективность по времени*

Время на выполнение операций подсчитывается в тактах процессора. Подсчитывается среднее время для 100 одинаковых операций. Операции производятся над одинаковыми данными

**Время удаления элемента (pop)**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер удаляемого элемента | Стек-массив | Стек-список | Отношение список/массив |
| 10 | 29 | 547 | 19 |
| 100 | 30 | 545 | 18 |
| 1000 | 28 | 524 | 19 |

**Время добавления элемента (push)**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер добавляемого элемента | Стек-массив | Стек-список | Отношение список/массив |
| 10 | 35 | 303 | 9 |
| 100 | 32 | 296 | 9 |
| 1000 | 34 | 330 | 10 |

Вне зависимости от количества элементов в стеке, операции добавления/удаления для каждой из реализации выполняются за схожее время, так как для стека-списка это работа с одним элементом одного и того же типа, а у стека-массива доступ к любому элементу выполняется за константное время.

Однако стек-список серьезно проигрывает по времени (в 9 раз и более) при выполнении этих операций, так как для их воспроизводства ему нужно работать с памятью – выделять или освобождать память из-под элемента -, а также изменять указатели на предыдущий элемент.

**Время поиска убывающих серий (без учета печати)**

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов в стеке | Стек-массив | Стек-список | Отношение список/массив |
| 10 | 9.417 | 12.890 | 1.4 |
| 100 | 27.396 | 58.513 | 2.1 |
| 1000 | 105.230 | 25.910.860 | 246 |
| 10000 | 2.958.739 | 13.691.212.662 | 4627 |

А при сравнении времени на выполнение данной операции видно, что с ростом количества элементов в стеке увеличивается и относительный разрыв во времени выполнения при различных реализациях стека. Это происходит из-за того, что теперь при работе со стеком-списком время теряется не только на выделение/освобождение памяти, но и на доступ к предыдущему элементу через указатель (а в стеке-массиве – просто по предыдущему индексу, а доступ по индексу занимает константное время).

Тестирование показало, что на моем компьютере фрагментации памяти не происходит.

# Выводы по проделанной работе

Стек-список выгоднее по памяти только когда занято не более 30% от максимального количества элементов стека. В остальных же случаях стек-массив эффективнее по памяти, причем чем больше количество элементов в стеке, тем он эффективнее.

Добавление и удаление элемента, а также выполнение операции, реализованной в данной программе (поиск убывающих серий) – также эффективнее выполнять со стеком-массивом, так как нет необходимости постоянно выделять/высвобождать память, и осуществлять доступ к предыдущему элементу по указателю. Причем выполнение реализованной в данной программе операции тем эффективнее выполнять при реализации стека массивом, чем больше элементов в стеке

Таким образом, если нужно реализовать такую структуру данных как стек, то эффективнее (как по памяти, так и по времени) использовать массив, а не линейный односвязный список.

# Ответы на 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. 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?

При хранении стека с помощью линейного односвязного списка: память выделяется в куче, под каждый элемент выделяется sizeof(int) байт на хранение значения элемента, sizeof(int) байт на хранение номера элемента, sizeof(int\*) байт на хранение указателя на предыдущий элемент.

При хранении стека с помощью массива, память выделяется либо в куче, либо на стеке (в зависимости от того, динамический массив или статический) непрерывным участком, под каждый элемент выделяется sizeof(int) байт на хранение значения элемента, а также необходимо отдельно хранить вместимость стека, номер текущего элемента и указатель на начало стека.

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

При хранении стека с помощью линейного односвязного списка: верхний элемент удаляется освобождением памяти для него и смещением указателя, указывающего на начало стека.

При хранении стека с помощью статического массива: память освобождается только после завершения программы – сразу из-под всего массива.

4) Что происходит с элементами стека при его просмотре?

При работе со стеком доступен только его верхний элемент, который адресуется специальным указателем стека PS (Pointer Stack). Причем классическая реализация стека предполагает, что просмотреть содержимое стека без извлечения (удаления) его элементов невозможно. Поэтому при просмотре очередного элемента, тот удаляется.

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

Реализовывать стек эффективнее с помощью динамического массива. Он эффективнее списка (в классическом случае) как по времени, так и по памяти. Реализация стека в виде списка может иметь преимущества только в двух случаях:

1) если сравнивать ее со стеком, реализованным статическим массивом. В этом случае память для списка ограничена размером оперативной памяти (так как память выделяется в куче), а память для статического массива ограничена размером стека.

2) если необходимо иметь возможность сохранить много элементов в стеке, но в основном в нем будет занято менее 30% от максимального количества элементов. Тогда список можно назвать эффективным по памяти.