|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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)** |

**Дисциплина “Типы и структуры данных”**

**Лабораторный практикум №3**

**по теме: «обработка разреженных матриц»**

Выполнил студент: \_\_*Клименко Алексей Константинович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*фамилия, имя, отчество*

Группа: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*ИУ7-35Б*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил, к.п.н.: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*подпись, дата*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2020 г.*

**цель работы**

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

**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.

Указания к выполнению работы

Интерфейс программы должен быть понятен неподготовленному пользователю. При разработке интерфейса программы следует предусмотреть:

• указание формата и диапазона вводимых данных,

• блокирование ввода данных, неверных по типу,

• указание операции, производимой программой:

o добавление элемента в стек,

o удаление элемента из стека,

o вычисление (обработка данных);

• 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:

o проверить правильность ввода и вывода данных (в том числе, отследить попытки ввода данных, неверных по типу);

o обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);

o проверить правильность выполнения операций;

o обеспечить вывод соответствующих сообщений при попытке удаления элемента из пустого стека;

o отследить переполнение стека.

При реализации стека в виде списка необходимо:

• ограничить доступный объем оперативной памяти путем указания:

o максимального количества элементов в стеке; максимального адреса памяти, превышение которого будет свидетельствовать о переполнении стека;

• следить за освобождением памяти при удалении элемента из стека

**техническое задание**

**исходные данные**

Исходными данными являются числа, запрашиваемые у пользователя в программе, и помещаемые в стеки.

*Примеры вставки и удаления чисел из двух стеков:*

|  |  |  |
| --- | --- | --- |
| **Ввод пользователя** | **Стек a** | **Стек b** |
| push a 1 2 3  pop a  push 2 5  push B 3 | (1,2,3)  (1,2)  (1,2)  (1,2) | ()  ()  (5)  (5,3) |

**результат**

Результатом работы программы являются три стэка, два из которых содержат введёные пользователем числа, а третий является отсортированным объединением чисел из двух других стеков.

**описание задачи, реализуемой программой**

Программа решает задачу слияния с одновременной сортировкой элементов из двух стеков в третий. По мере работы программы можно запросить у неё вывод текущего состояния на экран командой **show**.

**способы обращения к программе**

Для запуска программы необходимо запустить файл **app.exe**.

**возможные аварийные ситуации и ошибки пользователя**

При неверном вводе команды программа попросит ввести команду снова, а при невозможности выполнения команды (например, извлечение из пустого стека) сообщает об этом, и предлагает ввести другую команду.

**Структуры данных**

Для реализации стека на массиве была выбрана следующая структура:

**struct stack\_arr**

**{**

**uint32\_t size;** // размер стека

**uint32\_t \_\_capacity;** // вместимость стека

**int32\_t \*\_\_data;** // массив данных стека

**int32\_t \*top;** // указатель на вершину стека

**};**

При хранении в стеке N элементов объем памяти будет:

**sizeof(stack\_arr) == (24 + 4N)Б**

Реализация стека на связном списке:

**struct \_\_st\_lst\_node**

**{**

**int32\_t data;** // данные узла списка

**struct \_\_st\_lst\_node \*prev;** // указатель на предыдущий узел

**};**

**struct stack\_lst**

**{**

**uint32\_t size;** // размер стека

**struct \_\_st\_lst\_node \*top;** // указатель на вершину стека

**};**

При хранении в стеке N элементов объем памяти будет:

**sizeof(stack\_lst) == (12 + 12N)Б**

Структура стэка со скрытой реализацией (универсальная):

**typedef int \_\_stack\_imp\_type;**

**#define STACK\_TYPE\_INVALID      0**

**#define STACK\_TYPE\_ARRAY        1**

**#define STACK\_TYPE\_LINKED\_LIST  2**

**struct stack**

**{**

**\_\_stack\_imp\_type \_\_type;** // тип реализации

**union**

**{**

**struct stack\_arr stack\_arr;** // реализация на массиве

**struct stack\_lst stack\_lst;** // реализация на связном списке

**} data;**

**};**

**Множетво допустимых значений**

Для

**Набор функций**

Над структурой, представляющей собой запись в таблице, допустимо совершать следующие действия:

* Инициализация структуры квартиры нулевыми значениями
* Освобождение памяти, выделенное для данной структуры
* Преобразование информации из структуры в строку и наоборот
* Печать данных структуры в файл или на экран
* Считывание данных структуры из файла или консоли

Над таблицей данных разрешено выполнять следующие действия:

* Инициализация таблицы нулевым количеством квартир
* Освобождение памяти, выделенной для данной таблицы
* Сортировка таблицы по одному из предложенных невариантных полей: адрес, площадь комнат, количество комнат
* Добавление новой записи в конец таблицы
* Считывание данных таблицы из файла
* Вывод данных таблицы на экран консоли
* Создание копии таблицы
* Инициализация массива ключей для сортировки

**Описание алгоритма обработки данных**

Для реализации сортировки мною будут рассмотрены такие методы как сортировка обменом и сортировка слиянием как представители более медленной и более быстрой сортировок.

Для пункта (а), в котором таблица хранит в себе только массив структур, обе сортировки будут оперировать этим массивом данных непосредственно, т. е. будут производить *обмен элементами на месте*.

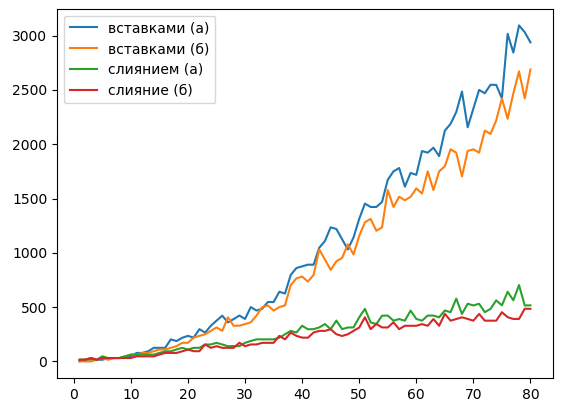
Для пункта (б), в котором помимо массива с данными будет и массив указателей на данные, сортироваться будет массив указателей, а это значит, что объем обрабатываемой при сортировке памяти будет меньше.

Приведём сложность упомянутых алгоритмов:

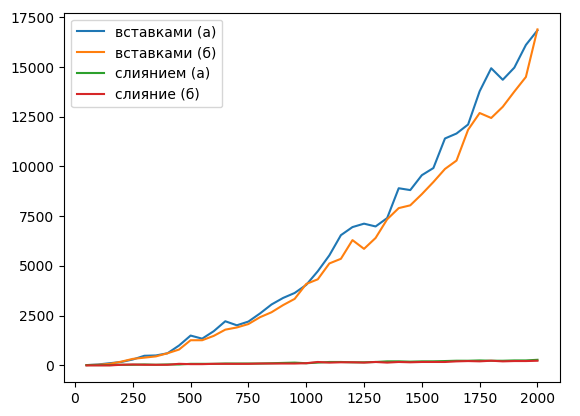
Сортировка вставками: **O(N2)**

Сортировка слиянием: **O(N\*log(N))**

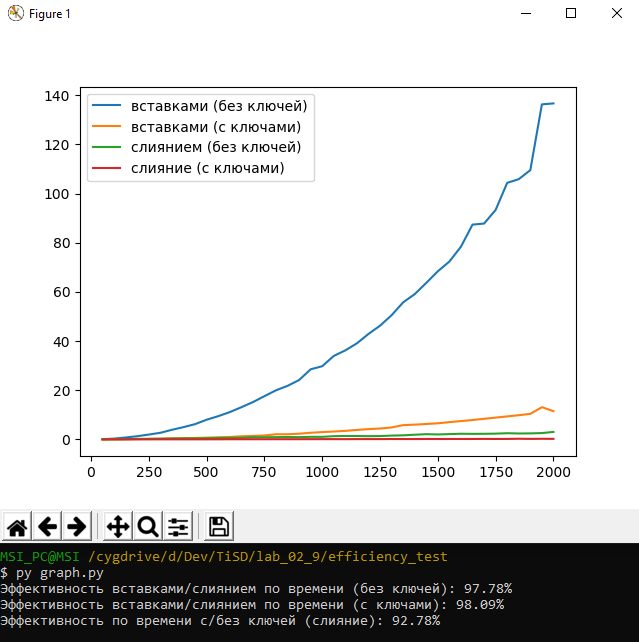
Результаты тестирования в виде графика (с учетом времени на инициализацию новой таблицы для сортировки):



Для большего числа элементов:



Без учёта времени инициализации копии таблицы графики преобретают иной вид:



Расчеты, проведённые в программе MathCAD для наиболее точного расчёта относительной эффективности двух алгоритмов при использовании дополнительной таблицы ключей:

Отностительная эффективность по времени (с учётом инициализаций):

**двух медленных сортировок: 3.1%**

**двух быстрых сортировок: 14.5%**

**набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Описание теста** | **Входные данные** | **Выходные данные** |
| 1 | Вывод таблицы | in.txt  3 | Форматированная таблица с данными |
| 2 | Ошибка при чтении файла | Null | Сообщение об ошибке. Завершение работы |
| 3 | Неверный выбор опции меню | in.txt  a | Отображение сообщения об ошибке и повторный запуск меню |
| 4 | Добавление новой записи | in.txt  1  *<flat data>*  3 | Добавление записи в конец таблицы |
| 5 | Ошибки при добавлении новой записи | in.txt  1  *<invalid data>* | Прерывание оперции, возврат в меню |
| 6 | Удаление записи | in.txt  5  *<flat id>*  3 | Удаление записи с указанным ID из таблицы |
| 7 | Ошибки при удалении записи | in.txt  5  *<not flat id>* | Отображение сообщения об ошибке и переход в меню |
| 8 | Сортировка таблицы по ключу | in.txt  2  *<1,2,3,4>*  *<1,2,3>*  *<0,1>* | Сортировка таблицы с последующим её отображением |
| 9 | Ошибки при сортировке таблицы | in.txt  2  *<invalid vals>* | Отображение сообщения об ошибке и переход в меню |
| 10 | Поиск записей по условию | in.txt  4  *<min price>*  *<max price>* | Вывод на экран всех подходящих под условия поиска записей или сообщения, что подходящих записей нет |
| 11 | Неверное условие при поиске записей | in.txt  4  *<wrong prices>* | Вывод на экран сообщения об ошибке и переход в меню |

**Выводы по проделанной работе**

По окончании работы мне удалось на практике сравнить эффективность двух различных алгоритмов сортировки, а также подтвердить практически их асимптотическую сложность, рассчитанную теоритически.

Судя по полученным результатам, сортировать таблицу с применением дополнительных массивов, оказывается немного более эффективным решением с точки зрения времени выполнения (приблизительно на **14.5%** для алгоритма сортировки слиянием). Однако уменьшение скорости оборачивается увеличением необходимого для работы объема памяти (приблизительно на **14.28%**).

Таким образом, при разработке программ необходимо выбирать структуры данных и алгоритмы по их обработке основываясь на имеющихся ресурсах для того, чтобы сделать разрабатываемый продукт наиболее эффективным.

**Контрольные вопросы**

1. *Как?*

Память