**Типы и структуры данных.**

**Лабораторная работа №3**

**Записи с вариантами, обработка таблиц**.

**Выполнил**: Кузнецов Александр

**Группа**: ИУ7-33

**Цель работы:** приобрести навыки работы с типом данных «запись», содержащим вариантную часть, и с данными, хранящимися в таблицах. Сравнить несколько различных алгоритмов сортировок массива при использовании таблиц записей с большим числом полей и таблиц ключей. Оценить относительную эффективность программы (в процентах) по времени и по используемому объему памяти.

Создать таблицу, содержащую не менее 40-ка записей (тип – запись с вариантами). Упорядочить данные в ней по возрастанию ключей, где ключ – любое невариантное поле (по выбору программиста), используя: а) саму таблицу, б) массив ключей (возможность добавления и удаления записей в ручном режиме обязательна)

**Вариант 13 (задача 2)**

Ввести список литературы, содержащий фамилию автора, название книги, издательство, количество страниц, вид литературы (1: техническая – отрасль, отечественная, переводная, год издания; 2: художественная – роман, пьеса, стихи). Вывести список всей переводной лит-ры по указанной отрасли с годом издания не позднее указанного.

**Входные данные:**

Строки, целые числа, ключи, команды для обработки таблицы;

**Пример:**

//ввод команды для таблицы

>>1

//ввод целого числа в таблицу

int: 2

//ввод ключа для таблицы

key: 1

//ввод строки в таблицу

string: blabla

**Выходные данные**

Таблица со значениями, таблица ключей с флагом, время работы сортировки «пузырек» с таблицей ключей и без, объем памяти для таблицы ключей.

**Функция программы**

Вывод таблицы, вывод таблицы ключей, вывод исходной таблицы по тегам таблицы ключей, сортировка таблицы ключей, вставка строки в таблицу, удаление строки из таблицы, поиск по заданным критериям, замер времени выполнения сортировки.

**Допущения**

* В полях (string) вводятся корректные данные.
* В исходном файле содержатся корректные данные

**Аварийные ситуации**

**-** Переполнение таблицы (MAX\_LEN\_TABLE 350)(Ограничение 50)

(Error: Table is overflow!(50 symbols)**)**

**-** Удаление строки из пустой таблицы

**(**Error: Table is empty!**)**

**-** Вывод пустого стека

**(**Error: Table is empty!**)**

- Добавление элемента в таблицу не INT

(incorrect input, please return input...)

- Команда над таблицей не удовлетворяет интерфейсу

(incorrect input, please return input...)

**Описание использованных структур данных**

//структура, которая содержит заголовки таблицы

struct Head\_s{

char header[40];

} head[LEN\_HEAD];

//структура, содержащая данные для типа: Техническая

typedef struct {

char industry[30];

char national[4];

char translate[4];

char year[8];//int

}Technical;

//структура, содержащая данные для типа: Искусство

typedef struct {

char genre[10];

}Art;

//структура, объединяющая структуры: Искусство и Техническая

typedef union{

Technical tech;

Art art;

}Type;

//Основная структура для составления таблицы

typedef struct{

char surname[20];

char book[40];

char publishing[20];

char num\_les[6];

char typebook[10];

Type uni; //union

} Common;

//Структура для составления таблицы ключей

typedef struct{

int index\_main;

int main\_less;

} Keys;

//Массивы структур

Common table[MAX\_LEN\_TABLE];

Keys table\_keys[MAX\_LEN\_TABLE];

**Алгоритм**

**Обработка ввода:**

1. Вывести интерфейс программы
2. Проверить корректность ввода(Интерфейс)
3. Проверить корректность ввода элемента(int)

-//////- Вывод сообщение об ошибке, если она есть -//////-

Иначе

**Обработка операций над стеком:**

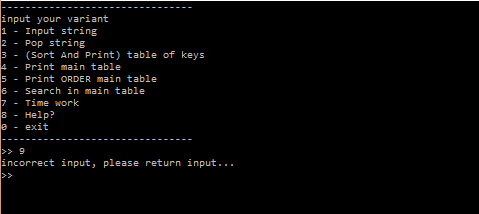
1. Добавление элемента в таблицу
2. Удаление элемента из таблицы
3. Вывод таблицы ключей и её сортировка
4. Вывод исходной таблицы
5. Вывод исходной таблицы с условием таблицы ключей
6. Поиск по определенным критериям

**Обработка времени:**

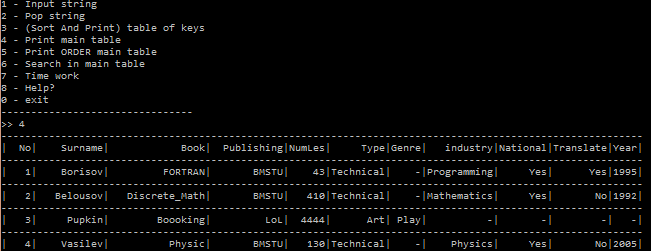
1. Расчет времени работы сортировки пузырьком
2. Вывод скорости работы сортировки с ключом и без него
3. Вывод памяти, которая потребовалась в результате работы ключа

**Тесты**

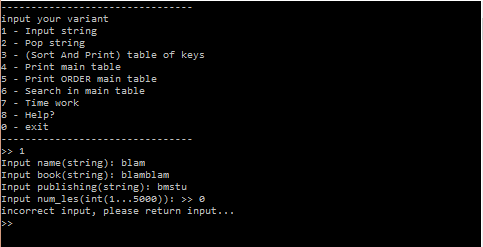
**Тест ввода:**Некорректный ввод номера операции над таблицей:



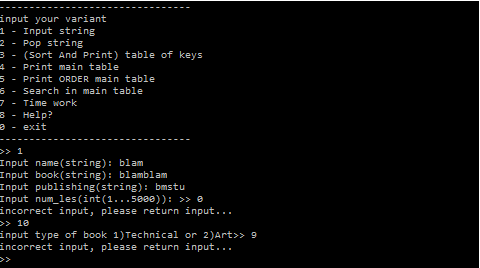
Корректный ввод номера операции над таблицей:



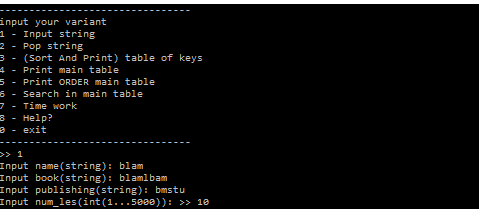
Некорректное добавление элемента в таблицу(не int):



Некорректное добавление элемента в таблицу(не ключ):

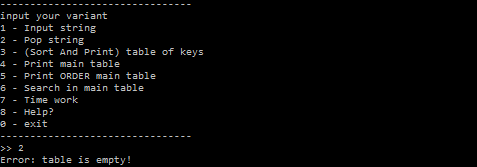


Корректное добавление элемента в таблицу(int):

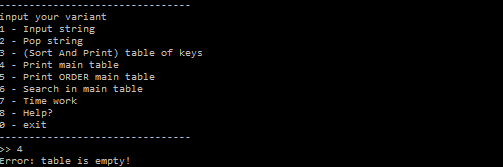


**Тест операций над таблицей:**

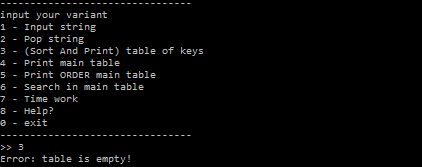
Удаление строки из пустой таблицы



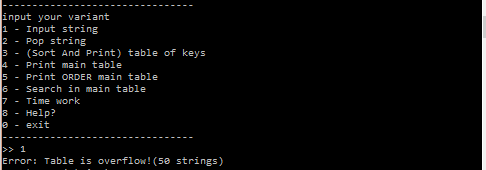
Вывод пустой таблицы



Вывод пустой таблицы ключей

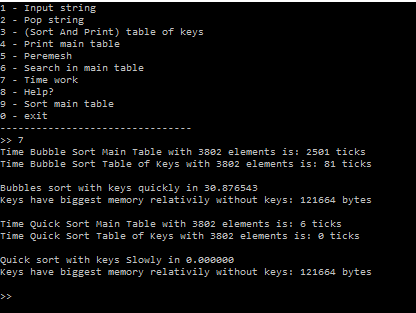


Переполнение Таблицы:



**Сравнение методов**

По результатам выполнения программы, видно, что реализация фильтра(сортировки) таблицы с ключом дает большой прирост в скорости выполнения фильтрации, в среднем в 30 раз, однако требуется дополнительная затрата памяти.



Отсюда видно, что

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N(элем) | Bubble(keys) | Bubble | Quick(keys) | Quick |
| 3802 | 81 | 2501 | 0 | 6 |

Отсюда видно, что при 3802 элементах Сортировка методом «Пузырек» по ключам заметно медленее, чем сортировка по ключам «Быстрая», более чем в 81 раз, так же «Быстрая» сортировка без ключей дает высочайший прирост скорости, по сравнению с «Пузырьком» в 416 раз.

**Вопрос – Ответ**

1. **Как выделяется память под вариантную часть записи?**

Объем памяти, необходимый для размещения записи, складывается из размера памяти под каждое из ее полей.

Запись может содержать поля, которые, в свою очередь, также являются записями, то есть, возможна структура вложенных записей.

1. **Что будет, если в вариантную часть ввести данные, несоответствующие описанным?**

Запись – это структурированный тип, состоящий из фиксированного числа компонентов различного типа. Компоненты записи называются полями записи. Каждое поле имеет свое собственное имя и тип. Тип поля может быть любым, кроме файлового.

1. **Кто должен следить за правильностью выполнения операций с вариантной частью записи?**

Тип данных в вариантной части при компиляции не проверяется, поэтому, контроль за правильностью ее использования возлагается на программиста.

1. **Что представляет собой таблица ключей, зачем она нужна?**

При больших размерах таблиц поиск данных, имеющих указанный ключ, может потребовать больших затрат времени. Если же помимо поиска требуется произвести сортировку данных, то временн**ые** затраты многократно возрастут, так как потребуется осуществлять их перестановку (перемещение). В этом случае можно уменьшить время обработки за счет создания дополнительного массива – таблицы ключей, содержащей индекс элемента в исходной таблице и выбранный ключ.

1. **В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?**

Таким образом, если мы сортируем таблицу ключей, то экономится время, поскольку перестановка записей в исходной таблице, которая иногда может содержать достаточно большое число полей, отсутствует. Этот выигрыш во времени особенно заметен при большой размерности таблиц и при правильно подобранных ключах. При этом, правда, надо помнить, что для размещения таблицы ключей требуется дополнительная память. Кроме того, следует учитывать, что если в качестве ключа используется символьное поле записи, то это влечет за собой необходимость посимвольной обработки данного поля в цикле, и, следовательно, приводит к увеличению времени выполнения любых операций. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, так же замедляет вывод этих данных.

1. **Какие способы сортировки предпочтительнее для обработки таблиц и почему?**

Принцип разделения хранения и структурирования информации часто применяется при хранении больших объемов информации в файлах, например, для хранения изображений. В этом случае создаются реестры (дополнительные массивы записей), содержащие ссылки на изображения, играющие роль ключевого значения, и имя файла.