Необходимо написать программу на C++, которая реализует работу с простой базой данных (сокращённо «БД»). Программа будет общаться с пользователем через стандартный ввод и вывод (потоки stdin и stdout).
Будем хранить в нашей БД пары вида: дата, событие. Определим дату как строку вида Год-Месяц-День, где Год — это число от 0 до 9999 включительно, Месяц — это номер месяца от 1 до 12 включительно, День — это номер дня от 1 до 31 включительно. Год, Месяц и День всегда разделяются ровно одним символом дефиса (-) и никаким другим. Событие определим как строку из произвольных печатных символов без разделителей внутри (пробелов, табуляций и пр.). Событие не может быть пустой строкой. В одну и ту же дату может произойти много разных событий — БД должна суметь их все сохранить. Одинаковые события, произошедшие в один и тот же день сохранять не нужно — достаточно сохранить только одно из них.
Наша БД должна понимать определённые команды, чтобы с ней можно было взаимодействовать:

```
- добавление события:                          Add Дата Событие
- удаление события:                           Del Дата Событие
- удаление всех событий за конкретную дату:   Del Дата
- поиск событий за конкретную дату:           Find Дата
- печать всех событий за все даты:            Print
```

Все команды, даты и события при вводе разделены пробелами. Команды считываются из стандартного ввода. В одной строке может быть ровно одна команда, но можно ввести несколько команд в несколько строк. На вход также могут поступать пустые строки - их следует игнорировать и продолжать обработку новых команд в последующих строках.

Добавление события

При добавлении события БД должна его запомнить и затем показывать его при поиске (командой Find) или печати (командой Print). В случае корректного ввода программа ничего не должна выводить на экран.

Удаление события

Удалить можно только ранее добавленные события. Если событие найдено и удалено, то программа должна вывести строку «Deleted successfully» (без кавычек). Если событие в указанную дату не найдено, то программа должна вывести строку «Event not found» (без кавычек).

Удаление нескольких событий

Команда удаляет все ранее добавленные события за указанную дату. Программа всегда должна выводить строку вида «Deleted N events», где N — это количество всех найденных и удалённых событий. N может быть равно нулю, если в указанную дату не было ни одного события.

Поиск событий

Ищем и печатаем ранее добавленные события в указанную дату. Программа должна вывести на печать только сами события, по одному на строке. События должны быть отсортированы по возрастанию в порядке сравнения строк между собой (тип string).

Печать всех событий

С помощью этой команды можно показать полное содержимое нашей БД. Программа должна вывести на печать все пары: Дата Событие по одной на строке. Все пары должны быть отсортированы по дате, а события в рамках одной даты должны быть отсортированы по возрастанию в порядке сравнения строк между собой (тип string). Даты должны быть отформатированы специальным образом: ГГГГ-ММ-ДД, где Г, М, Д — это цифры чисел года, месяца и дня соответственно. Если какое-то число имеет меньше разрядов, то оно должно дополняться нулями, например, 0001-01-01 — первое января первого года.

Обработка ошибок ввода

Если пользователь ввёл неизвестную команду, то программа должна об этом сообщить, выведя строку «Unknown command: COMMAND», где COMMAND — это та команда, которую ввёл пользователь. Командой считается первое слово в строке (до пробела).

Если пользователь ввёл дату в неверном формате там, где она ожидалась, то программа должна напечатать «Wrong date format: DATE», где DATE — это то, что пользователь ввёл вместо даты (до пробела).

Если формат даты верный, но сама дата неверна, то программа должна напечатать более конкретную проблему:

«Month value is invalid: MONTH», где MONTH — это неверный номер месяца, например, 13 или 0.

«Day value is invalid: DAY», где DAY — это неверный номер дня в месяце, например, 39 или 0.

Значение года проверять отдельно не нужно.

Не нужно проверять календарную корректность даты: количество дней в каждом месяце считается равным 31, високосные года учитывать не нужно.

Если неверны как месяц, так и день, то нужно вывести одно сообщение об ошибке в месяце.

После любой ошибки ввода и печати сообщения программа должна завершать своё выполнение.

Примеры

Корректный ввод:

```
Add 0-1-2 event1
Add 1-2-3 event2
Find 0-1-2
Del 0-1-2
Print
```

Вывод:

```
event1
Deleted 1 events
0001-02-03 event2
```

Неверный формат даты:

```

Add 0-13-32 event1
```

Вывод:

C++Выделить код

```

Month value is invalid: 13
```

Примечания

Преобразование числа к строке

Чтобы, имея число MONTH, составить строку «Month value is invalid: MONTH», можно использовать функцию to_string, преобразующую число к строке. Таким образом, составить необходимую строку можно следующим образом:

C++Выделить код

```


string error = "Month value is invalid: " +
to_string(month);
```

Поиск в константном словаре

При реализации данного шаблона вам может понадобится использовать поиск с помощью квадратных скобок для словаря, переданного в функцию по константной ссылке. Как было показано ранее, это сделать не удастся, так как обращение к несуществующему ключу с помощью квадратных скобок добавит его в словарь, что недопустимо для константного словаря.

В этом случае вместо квадратных скобок используйте метод at: в случае отсутствия ключа он выбросит исключение и потому может быть использован для константного объекта.

Например, вместо кода

C++Выделить код

```
1
2
3
4  void DoSomething(const map<int, int>& m) {
5    // ...
6    if (m.count(key) > 0) {
7      value = m[key];  // не компилируется
8    }
9    // ...
10 }
```

используйте код

C++Выделить код

```
1
2
3
4  void DoSomething(const map<int, int>& m) {
5    // ...
6    if (m.count(key) > 0) {
7      value = m.at(key);  // теперь всё хорошо
8    }
9    // ...
10 }
```

Не пойманные исключения

Так как для решения задачи требуется программа, работающая верно на большом количестве разных входных данных, то неизбежно в ней могут обнаружиться ошибки, про некоторые из которых мы не рассказали в нашем курсе лекций. Одной из таких ошибок может быть не пойманное исключение: ошибка заключается в том, что исключение, будучи выброшенным, не попадает ни под одно из выражений блоков catch вплоть до функции main. В этом случае программа тут же завершится аварийно, и в качестве ошибки в тесте вы увидите «Unknown signal 6».