Для решения предлагаю использовать два вида структур данных:

* std::vector (массив) для хранения информации о товарах;
* std::unordered\_map (хеш-таблица) для осуществления поиска.

Типы данных:

* int для штрих-кодов;
* string для имен и строки со всей информацией о товаре.

Алгоритм:

Подготовка данных:

1. Построчно читаем из файла, каждую строку пишем в вектор.
2. Проходимся по вектору:
   1. Из строки выделяем штрих-код, пишем пару <штрих-код, указатель на соответствующий товар в векторе> в хеш-таблицу с штрих-кодами;
   2. Из строки выделяем имя, пишем пару <имя, указатель на соответствующий товар в векторе> в хеш-таблицу с именами.

P.s. Из-за перераспределения памяти при достижении максимума элементов в векторе, нужно запоминать ссылки только после окончания записи в него.

Поиск:

1. Ищем в хеш-таблице со штрих-кодами/именами пару по ключу, далее по ссылке из вектора выводим данные о товаре.

|  |  |
| --- | --- |
| vector<string> product\_info | |
| product\_info[i] | … |

|  |  |
| --- | --- |
| unordered\_map<int, string\*> batch\_indexed\_map; | |
| Key: batch\_code | pointer to data |

|  |  |
| --- | --- |
| unordered\_map<string, string\*> name\_indexed\_map; | |
| Key: name | pointer to data |

Информация о товаре не дублируется в хеш-таблицах, т.к. она хранится в векторе и к ней обращаются по необходимости.

Почему выбраны эти структуры:

* Вектор выбран для хранения из-за быстрого обращения к элементу по индексу (чтение из вектора O(1));
* Unordered\_map оптимальна для поиска O(1), будет быстрее, чем map, а порядок нам не важен.