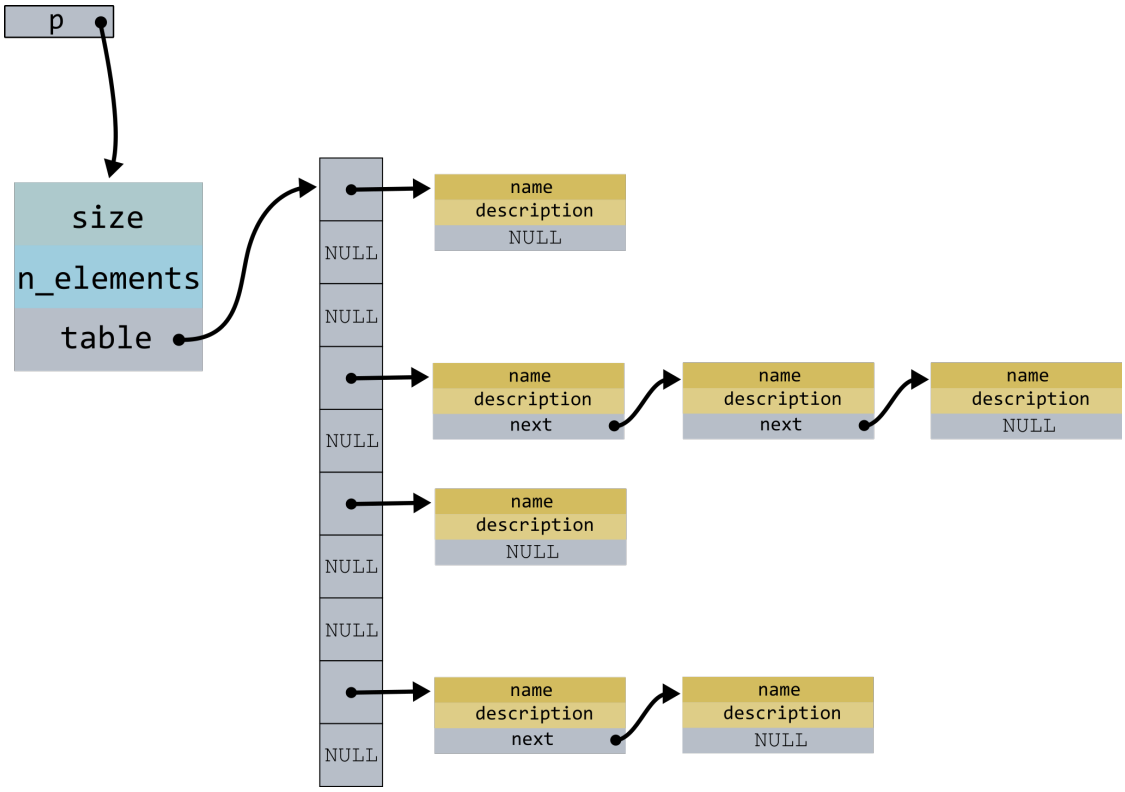
Хеш-таблицы: Домашнее задание



В файле `hash.c` лежит реализация хеш-таблицы (решение задач с классного занятия).

Задачи:

- **Перестройка таблицы:** Видоизмените функцию `hashtable_insert` так, чтобы она проверяла загруженность таблицы при каждом добавлении элемента. Если загруженность таблицы больше чем `MAX_LOAD_FACTOR`, то таблица должна будет вырасти в `GROWTH_FACTOR` раз. При этом положения каждого элемента таблицы могут измениться. Проще всего создать новую таблицу и добавить туда все элементы из старой.
- **Множество:** Абстрактный тип данных множество (Set) - это некоторая коллекция элементов с определёнными операциями:
  - `insert(S, x)` - добавляет элемент `x` в множество `S`; если элемент уже есть, то ничего не делает.
  - `erase(S, x)` - удаляет элемент `x` из множества `S`; если такого элемента нет, то ничего не делает.
  - `is_element_of(x, S)` - проверяет входит ли элемент `x` в множество `S`.

  Конечно, этот абстрактный тип данных можно реализовать с помощью простого массива или связного списка, но при этом операции над множеством будут занимать время $O(n)$, что очень долго. Поэтому множество реализуется с помощью хеш-таблицы $O(1)$, либо с помощью сбалансированного дерева $O(log(n))$. В этом задании вам нужно реализовать множество, хранящее целые числа, с помощью хеш-таблицы. Помимо операций `insert`, `erase`, `is_element_of`, нужно будет написать функции `create`, `print` и `destroy`.
  Пример работы с таким множеством (этот код должен напечатать `NO`):

```
Set* s = set_create();
set_insert(s, 15);
set_insert(s, 4);
set_insert(s, 15);
set_insert(s, 42);
set_erase(s, 15);
if (is_element_of(15, s))
        printf("YES\n");
else
        printf("NO\n");
set_destroy(s);
```

- **Уникальное число:** В файле `special_numbers.txt` лежит набор из 20001 чисел. Почти все числа из этого набора встречаются парами (содержатся в наборе чётное число раз). Но в наборе есть одно уникальное число, которое встречается лишь один раз. Найдите это число.
  Напоминание как считать числа из файла на C:

```c
int main()
{
        FILE* file = fopen("special_numbers.txt", "r");
        int N;
        int array[20001];
        fscanf(file, "%d", &N);
        for (int i = 0; i < N; i++)
                fscanf(file, "%d", &array[i]);

}
```

  Файл `special_numbers.txt` должен лежать в той же папке, что и *запускаемый исполняемый файл* (если вы используете IDE, то это не обязательно та папка в которой у вас лежит исходный код).

- **Словарь:** Ассоциативный массив или словарь (англ. dictionary или map или associative array) является важнейшим абстрактным типом данных в программировании. Он представляет собой коллекцию пар `<ключ, значение>` с определёнными следующими операциями:
  - `insert(M, key, value)` - вставляет в словарь `M` пару `<key, value>`. Если элемент с таким ключом уже есть, то ничего не делает.
  - `assign(M, key, value)` - вставляет в словарь `M` пару `<key, value>`. Если элемент с таким ключом уже есть, то устанавливает новое значение у этого ключа.
  - `erase(M, key)` - удаляет из словаря `M` пару с ключом `key`. Если элемента с таким ключом нет, то ничего не делает.
  - `find(M, key)` - ищет в словаре `M` пару с ключом `key`. (функция, соответствующая этой операции, должна возвращать указатель на элемент или `NULL`, если такого элемента нет)

  Обычно словарь реализуется с помощью хеш-таблицы, либо с помощью сбалансированного дерева. И ключ и значение может быть переменной любого типа. Словарь с точки зрения интерфейса можно рассматривать как массив у которого в качестве индекса может выступать значения любых типов, а не только целых чисел. По сути, в классной задаче мы создали славарь, у которого в качестве и ключа и значения были строки. В этой задаче вам нужно реализовать словарь, у которого в качестве ключа будет строка, а в качестве значения - целое число.
  Пример работы с таким словарём (пара город - численность населения):

```c
int main()
{
        Map* m = map_create();
        map_insert(m, "Dolgoprudny", 90956);
        map_insert(m, "London", 8908081);
        map_insert(m, "Montevideo", 1719453);
        map_assign(m, "Adelaide", 1345777);
        map_assign(m, "Dolgoprudny", 108861); // Как бы m["Dolgoprudny"] = 108861;

        Node* p = map_find(m, "London");
        if (p != NULL)
                printf("London is in the map. Population = %d\n", p->value);
        else
                printf("No London in the map\n");
        map_destroy(m);
}
```