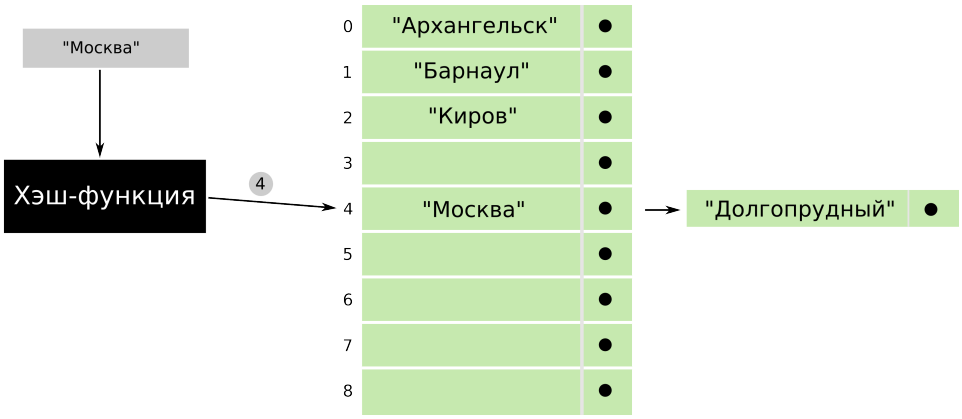
Хеш-таблицы




```
#define INITIAL_SIZE (1024)
#define GROWTH_FACTOR (2)
#define MAX_LOAD_FACTOR (1)

struct node
{
    struct node* next;
    char* key;
    char* value;
};
typedef struct node Node;

struct hashtable
{
    /* Размер таблицы */
    int size;
    /* Количество элементов в таблице */
    int n;
    struct node** table;
};
typedef struct hashtable Hashtable;
```

```
// Функция, которая добавляет элемент в
    хештаблицу-
Hashtable* hashtable_create(int size)
{
    int i;
    Hashtable* ht =
         malloc(sizeof(Hashtable));

    ht->size = size;
    ht->n = 0;
    ht->table = malloc(sizeof(struct
    node*)*ht->size);

    for(i = 0; i < ht->size; i++)
        ht->table[i] = 0;

    return ht;
}
```


Задачи

1. Написать функцию `void hashtable_insert(Hashtable* ht, char* key, char* value)`, которая добавляет элемент в хештаблицу. Для этого вам понадобится простейшая хеш-функция:

```
unsigned long hash_function(const char *s)
{
    unsigned long h = 0;
    for(unsigned char* p = s; *p; p++)
        h = h * 97 + *p;
    return h;
}
```

2. Написать функцию `void print_hashtable(Hashtable* ht)`, которая будет печатать все элементы хештаблицы.

3. Используйте функции из предыдущих двух задач, чтобы добавить в хештаблицу 10 элементов. Напечатайте эту хещ-таблицу.

4. Написать функцию `char* search_hashtable(Hashtable* ht, char* key)`, которая будет искать элемент в хеш-таблице по ключу.

5. Написать функцию `void grow_hashtable(Hashtable* ht)`, которая будет увеличивать хеш-таблицу в GROWTH_FACTOR раз.

6. Считать данные из файла `citiydescriptions.txt` в хеш-таблицу. Добавьте в хеш-таблицу большое количество элементов и проследите как меняется её размер.