Bài 8.2: Bảng băm người dùng tự định nghĩa

- ✓ Sơ lược về các thao tác trên bit
- ✓ Tạo lớp Entry
- ✓ Tạo lớp HashTable
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Thao tác &|^~

- Các thao tác trên bit ta tìm hiểu gồm: &, |, ^, ~, <<, >>.
- Phép AND: thực hiện nhân 2 bit, cho kết quả là 1 khi cả hai bit tham gia cùng bằng 1.
- Phép OR: cho kết quả là 0 khi và chỉ khi cả hai bit tham gia đều bằng 0.
- Phép XOR: cho kết quả là 1 nếu hai bit tham gia khác nhau. Kết quả là 0 nếu hai bit giống nhau.

Operator	Bit 1	Bit 2	Bit Value (Result)
AND &	1	1	1
	1	0	0
	0	1	0
	0	0	0
OR |	1	1	1
	1	0	1
	0	1	1
	0	0	0
XOR ^	1	1	0
	1	0	1
	0	1	1
	0	0	0

Thao tác &|^~

- ~: NOT-lấy phần bù, bù của 0 là 1 và của 1 là 0. Đây là phép bù 2. NOT x = -x – 1.
- Khi ta thực hiện các phép toán trên với hai biến nguyên, chương trình tự chuyển đổi giá trị hệ 10 sang hệ 2 sau đó tiến hành thao tác được yêu cầu trên dãy nhị phân sau chuyển đổi.
- Ví dụ a & b với a = 5, b = 6 thì kết quả là 4.
  - 5 = 0101
  - 6 = 0110
  - a&b = 0100 => 4 ở hệ 10

Thao tác dịch bit

- << dịch trái dãy bit sang trái n bit, điền 0 vào các bit phải cùng. Dịch trái n bit tương đương nhân với 2^n.
- Ví dụ: xét số a 1 byte có biểu diễn 00010000 = 16. a << 2 cho biểu diễn là 01000000 = 64.
- >> dịch phải dãy bit sang phải n bit, điền bit dấu vào các bit trái cùng. Dịch phải n bit tương đương chia cho 2^n.
- Số dương +a sẽ có bit dấu là 0, số âm có bit dấu là 1. Bit dấu luôn là bít trái cùng.
- Ví dụ: a = +16, a >> 2 cho kết quả là +4: 00000100.
- Ví dụ: a = 10010111 = -105. khi a >> 1 cho kết quả 11001011 = -53.

Tạo lớp Entry

- Tạo lớp template Entry chứa thông tin về key, value của từng phần tử trong bảng băm.
- Các thuộc tính:
  - Mã băm: hash.
  - Khóa: key.
  - Giá trị gắn với khóa: value.

```
// lớp template lưu trữ thông tin các phần tử trong bảng băm
template<class V> class Entry {
    friend ostream& operator << <>(ostream&, const Entry<V>&);
public:
    string key;
    V value;
    size_t hash;

    Entry(string key, V value) { ... }

    // hàm băm đơn giản
    // quy tắc băm:
    // hashValue = s[0] * 31 ^ (n - 1) + s[1] * 31 ^ (n - 2) + ... + s[n - 1]
    // trong đó: s là chuỗi kí tự trong key, n là độ dài chuỗi đó
    // và ^ là phép lũy thừa
    size_t hashCode() { ... }
};
```

Tạo lớp HashTable

```
// lớp template mô tả thông tin và các hành động của bảng băm
template<class V> class HashTable {
private:
    size_t capacity;
    list<Entry<V>>* table;
public:
    HashTable(int capacity = 10) {
        if (capacity <= 0) {
            capacity = 10;
        }
        this->capacity = capacity;
        this->table = new list<Entry<V>>[capacity];
    }

    size_t hashCode(size_t code) { ... }

    void insert(string key, V value) { ... }

    void displayHashTable() { ... }

    ~HashTable() {
        delete[] table; // thu hồi bộ nhớ đã cấp phát động
    }
};
```

Nội dung tiếp theo

Xóa cặp key-value