Лекция. Словари и множества (C++)

Множества

- математические структуры, которые могут хранить в себе уникальные элементы (то есть, каждый элемент может входить в множество только один раз).

Работа с элементами множества

- Решим следующую задачу: даны N запросов трёх типов:

1. добавить элемент во множество;
2. проверить, входит ли элемент во множество;
3. удалить элемент из множества.

Сначала задается число N, а затем сами запросы. Каждый из них состоит из двух чисел. Первое обозначает тип запроса, а второе — элемент, с которым нужно произвести операцию.

Решение

```cpp
#include <iostream>
#include <set>

using namespace std;

int main() {
    set <int> s;
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        int type, x;
        cin >> type >> x;
        if (type == 1) { // добавление
            s.insert(x);
        } else if (type == 2) { // проверка
            if (s.find(x) == s.end()) {
                cout << "NO\n";
            } else {
                cout << "YES\n";
            }
        } else { // удаление
            s.erase(x);
        }
    }
    return 0;
}
```

Пояснение

- При объявлении получаем пустое множество
- Добавление элементов в него происходит с помощью метода insert.
- Чтобы проверить, входит ли элемент во множество, используется метод find.
- Если элемент в множестве не нашелся, то он выдает то же значение, что и метод end.
- Удаление отдельного элемента из множества выполняется с помощью метода erase.

Заполним N элементов множества целыми числами

```
set <int> s;
int n;
cin >> n;
for (int i = 0; i < n; i++) {
    int x;
    cin >> x;
    s.insert(x);
}
```

Вывод всех элементов множества

- 1 способ

```
for (auto now = s.begin(); now != s.end(); now++) {
    cout << *now << ' ';
}
```

- begin возвращает указатель на самый маленький элемент множества, end — это конец множества (он идёт после самого большого элемента)
- Чтобы посмотреть, что за элемент хранится по указателю, нужно перед его именем написать символ *.

Вывод всех элементов множества

- 2 способ

```
for (auto now : s) {
    cout << now << ' ';
}
```

Сортировка с помощью множества

- Поскольку проход по элементам множества осуществляется в возрастающем порядке, то можно использовать его для сортировки последовательностей.

- НО всё портит тот факт, что с одинаковыми элементами множество работает иначе.

Сортировка с помощью множества

- В C++ есть структура multiset, которая может хранить в себе одинаковые элементы.
- Multiset умеет все то же самое, что и обычный set, и лежит в той же библиотеке.

- Если в предыдущей программе вывода всех элементов заменить set на multiset, то мы как раз и получим элементы по возрастанию.

Количество разных элементов

- С помощью set очень легко подсчитать число различных элементов в последовательности.
- Для этого нужно просто добавить все элементы последовательности во множество, а затем посмотреть на его размер.

Количество разных элементов

- 1 способ
- при добавлении элемента во множества, если его нет увеличить счетчик
- 2 способ
- У set есть метод size(), который возвращает количество элементов во множестве. Добавить в него все элементы подряд, вывести size() от этого множества.

Подсчет количества вхождений элемента в последовательность

- посчитать, сколько раз встречается единица в последовательности

```
multiset <int> s;
int n;
cin >> n;
for (int i = 0; i < n; i++) {
    int x;
    cin >> x;
    s.insert(x);
}
int cnt = 0;
for (auto now = s.lower_bound(1); now != s.upper_bound(1);
now++) {
    cnt++;
}
cout << cnt;
```

Пояснение

- **lower_bound** возвращает указатель на первый элемент, значение которого больше либо равно переданному параметру.
- **upper_bound** — на первый элемент, который строго больше

- Так мы пробежим от первой единицы до первого элемента (или end'а нашего set'а), на каждом шаге увеличивая значение счётчика вхождений. Если ни одной единицы в последовательности нет, оба метода вернут указатели на больший элемент; будет выполнено 0 шагов.

Словари

- Структура, похожая на множество.
- Ставит в соответствие ключу значение, совсем как в обычном словаре, где каждому русскому слову ставится в соответствие иностранное.
- Словарь в C++ называется map (карта).
- Чтобы пользоваться словарями, нужно подключить библиотеку map.

Пример описания словаря(карты)

- map <int, string> s;
- Создания элемента словаря
- **s[112] = "sos";**
- Проверка существования элемента делается с помощью метода find, как и во множествах.

Пример

```
#include <iostream>
#include <map>
#include <string>

using namespace std;

int main() {
	map <int, string> s;
	s[112] = "sos";
	if (s.find(112) != s.end()) {
		cout << "YES\n";
	}
	return 0;
}
```

Проход по элементам словаря

- 1 способ

```
map <int, string> s;
s[112] = "sos";
s[102] = "emergency";
for (auto now : s) {
    cout << now.first << " " << now.second << "\n";
}
```

Особенности словаря

- В словаре на место now подставляются пары «ключ-значение»
- Обратиться к первому из них можно как к now.first (где now — название пары), а ко второму — now.second.
- Отдельные элементы пары также можно менять как обычные переменные.
- Как и во множестве, ключи в словаре упорядочены по возрастанию. Для поиска ключей можно также пользоваться методами find, lower_bound и upper_bound.

Сопоставление нескольких значений

- Часто требуется сопоставить одному ключу несколько значений.
- в телефонной книге — несколько номеров у одного и того же человека.

- Чтобы решить задачу такого сопоставления, мы будем использовать в качестве значения вектор.

Пример

```
#include <iostream>
#include <map>
#include <string>
#include <vector>

using namespace std;

int main() {
    map <string, vector <string>> s;
    s["Vasya"] = { "112133", "12341" };
    for (auto now : s["Vasya"]) {
        cout << now << " ";
    }
    return 0;
}
```

Пояснения

- В этой программе мы сразу инициализировали вектор конкретными значениями, используя фигурные скобки.
- В принципе, можно создать пустой вектор и добавлять в него элементы с помощью метода push_back.
- Это удобно в том случае, если мы не знаем заранее, сколько номеров в нашей телефонной книге может быть сопоставлено человеку.
- Если ключ ещё не встречался, нужно создать пустой вектор, а при каждом его обнаружении просто добавлять элемент в вектор.

Задача №1

- Дан список целых чисел, который может содержать до 100000 чисел. Определите, сколько в нем встречается различных чисел.

Задача №2

- Во входной строке записана последовательность чисел через пробел. Для каждого числа выведите слово YES (в отдельной строке), если это число ранее встречалось в последовательности или NO, если не встречалось.
- Ввод данных
- 6
- 1 2 3 2 3 4
- Вывод данных
- NO NO NO YES YES NO