Set — конспект темы

Как работают множества

Контейнер `set` (множество) — упорядоченный набор уникальных элементов.

Чтобы проверить вхождение слова в вектор, пришлось бы идти по всему контейнеру циклом и сравнивать. Здесь вместо вектора множество `words`, а вместо цикла метод `count` — с ним код короче и эффективнее, чем последовательный цикл, особенно при большом количестве элементов:

```
set<string> words;
string word_to_find;
// ...
// Метод count возвращает количество вхождений элемента внутри множества
bool found = words.count(word_to_find) != 0;
```

Контейнер `set` автоматически сортирует элементы и удаляет дубликаты.

Чтобы создать `set`, нужно подключить библиотеку `<set>`, указать тип элементов множества и перечислить элементы в фигурных скобках:

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

using namespace std;

void PrintSet(set<string> s) {
    for (string x : s) {
        cout << x << endl;
    }
}

int main() {
    set<string> animals = {"cat"s, "dog"s, "aardvark"s, "dog"s, "sheep"s, "ape"s, "sheep"s};
    PrintSet(animals);
}
```

Названия животных будут упорядочены по алфавиту (по возрастанию) при выводе.

У множества есть свои методы:

- `size()` возвращает количество элементов,

- `empty()` проверяет множество на пустоту:

```
//...

int main() {
    set<string> animals = {"cat"s, "dog"s, "aardvark"s, "dog"s, "sheep"s, "ape"s, "sheep"s};
    if (!animals.empty()) {
        cout << "There are some animals in the set,"s << endl;
        cout << animals.size() << " different animals, to be exact."s;
    }
}
```

- `insert()` добавляет новые элементы во множество, упорядоченность элементов при этом сохраняется:

```
// ...

int main() {
    set<string> animals = {"cat"s, "dog"s, "aardvark"s, "dog"s, "sheep"s, "ape"s, "sheep"s};
    PrintSet(animals);
    cout << endl;
    animals.insert("another dog"s);
    animals.insert("cat"s);
    animals.insert("penguin"s);
    PrintSet(animals);
}
```

- `erase()` удаляет элементы из `set`:

```
// ...

int main() {
    set<string> animals = {"cat"s, "dog"s, "aardvark"s, "dog"s, "sheep"s, "ape"s, "sheep"s};
    PrintSet(animals);
    cout << endl << "Some animals are gone!"s << endl << endl;
    animals.erase("cat"s);
    animals.erase("dog"s);
    animals.erase("ape"s);

    // Удаление отсутствующего элемента не делает ничего
    animals.erase("mouse"s);

    PrintSet(animals);
}
```

Плюсы множеств

Множества легко сравнивать между собой:

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

using namespace std;

int main() {
    set<string> mammals = {"tiger"s, "elephant"s, "pig"s};
    set<string> other_mammals = {"pig"s, "elephant"s, "tiger"s, "pig"s};

    cout << boolalpha << (mammals == other_mammals) << endl;

    return 0;
}
```

Дубликат в `set` не считается. Поэтому множества совпадают, код выведет `true`.

Во множестве легко проверить наличие конкретного элемента. Для этого используют метод `count()`:

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

using namespace std;

void CheckBirdPresence(set<string> birds, string bird_name) {
    if (birds.count(bird_name) > 0) {
        // Птица всё ещё здесь
        cout << "The "s << bird_name << " is still here."s;
    } else {
        // Птица улетела
        cout << "The "s <<  bird_name << " has flown away!"s;
    }
}

int main() {
    // ласточка, орёл, сова, ласточка, скворец
    set<string> birds = {"swallow"s, "eagle"s, "owl"s, "swallow"s, "starling"s};

    cout << birds.count("swallow"s) << endl;
    CheckBirdPresence(birds, "swallow"s);
    cout << endl;
    birds.erase("swallow"s);
    cout << birds.count("swallow"s) << endl;
    CheckBirdPresence(birds, "swallow"s);
```

```
    return 0;
}
```

Результатом метода `count` будет 0 или 1.