# c1w3

28 сентября 2020 г.

## 0.1 task1

В стандартном потоке дана одна строка, состоящая из N + 1 целых чисел. Первым числом идёт само число N. Далее следуют ещё N чисел, обозначим их за массив A. Между собой числа разделены пробелом.

Отсортируйте массив A по модулю и выведите его в стандартный поток.

### 0.1.1 Ограничения

$0 \leq N \leq 1000$

$-1000000 \leq A_i \leq 1000000$

### 0.1.2 Примеры:

| stdin | stdout |
|---|---|
| 2 -4 3 | 3 -4 |
| 3 1 -3 2 | 1 2 -3 |

```
>>> #include <iostream>
... #include <vector>
... #include <algorithm>
...
... using namespace std;
...
... int main() {
...     size_t count;
...     cin >> count;
...     vector<int> nums(count);
...     for (auto& num : nums)
...         cin >> num;
...
...     sort(begin(nums), end(nums), [](int x, int y) {
...         return abs(x) < abs(y);
...     });
...
...     for (const auto& num : nums)
...         cout << num << " ";
...
...     return 0;
... }
```

## 0.2 task2

В стандартном потоке дана одна строка, состоящая из числа N и следующих за ним N строк S. Между собой число и строки разделены пробелом.

Отсортируйте строки S в лексикографическом порядке по возрастанию, игнорируя регистр букв, и выведите их в стандартный поток вывода.

### 0.2.1 Ограничения

$0 \leq N \leq 1000$

$1 \leq |S| \leq 15$

Каждая строка $S_i$ может состоять из следующих символов: $[0-9, a-z, A-Z]$

### 0.2.2 Примеры

| stdin | stdout |
|---|---|
| 2 q A | A q |
| 3 a C b | a b C |

```
>>> #include <iostream>
... #include <vector>
... #include <algorithm>
...
... using namespace std;
...
... int main() {
...     size_t count;
...     cin >> count;
...     vector<string> strings(count);
...     for (auto& str : strings)
...         cin >> str;
...
...     sort(begin(strings), end(strings), [](string x, string y) {
...         std::transform(x.begin(), x.end(), x.begin(),
...                        [](unsigned char c){ return std::tolower(c); });
...         std::transform(y.begin(), y.end(), y.begin(),
...                        [](unsigned char c){ return std::tolower(c); });
...         return x < y;
...     });
...
...     for (const auto& str : strings)
...         cout << str << " ";
...
...     return 0;
... }
```

## 0.3 task3

Реализуйте класс для человека, поддерживающий историю изменений человеком своих фамилии и имени.

Считайте, что в каждый год может произойти не более одного изменения фамилии и не более одного изменения имени. При этом с течением времени могут открываться всё новые факты из прошлого

человека, поэтому года в последовательных вызовах методов *ChangeLastName* и *ChangeFirstName* не обязаны возрастать.

Гарантируется, что все имена и фамилии непусты.

Строка, возвращаемая методом *GetFullName*, должна содержать разделённые одним пробелом имя и фамилию человека по состоянию на конец данного года.

- Если к данному году не случилось ни одного изменения фамилии и имени, верните строку “Incognito”.
- Если к данному году случилось изменение фамилии, но не было ни одного изменения имени, верните “last_name with unknown first name”.
- Если к данному году случилось изменение имени, но не было ни одного изменения фамилии, верните “first_name with unknown last name”.

Пример:

```
>>> int main() {
...   Person person;
...
...   person.ChangeFirstName(1965, "Polina");
...   person.ChangeLastName(1967, "Sergeeva");
...   for (int year : {1900, 1965, 1990}) {
...     cout << person.GetFullName(year) << endl;
...   }
...
...   person.ChangeFirstName(1970, "Appolinaria");
...   for (int year : {1969, 1970}) {
...     cout << person.GetFullName(year) << endl;
...   }
...
...   person.ChangeLastName(1968, "Volkova");
...   for (int year : {1969, 1970}) {
...     cout << person.GetFullName(year) << endl;
...   }
...
...   return 0;
... }
```

### 0.3.1 Вывод:

```
Incognito
    Polina with unknown last name
    Polina Sergeeva
    Polina Sergeeva
    Appolinaria Sergeeva
    Polina Volkova
    Appolinaria Volkova
```

```
>>> #include <iostream>
...
... #include <iostream>
... #include <vector>
... #include <map>
... #include <algorithm>
...
```

```
... using namespace std;
...
... class Person {
... public:
...     void ChangeFirstName(int year, const string& first_name) {
...         firstNameHistory[year] = first_name;
...     }
...     void ChangeLastName(int year, const string& last_name) {
...         lastNameHistory[year] = last_name;
...     }
...     string GetFullName(int year) {
...         // in c++ maps are sorted by keys :)
...         string first_name;
...         string last_name;
...
...         for (const auto& [current_year, current_first_name] : firstNameHistory) {
...             if (year >= current_year)
...                 first_name = current_first_name;
...         }
...
...         for (const auto& [current_year, current_last_name] : lastNameHistory) {
...             if (year >= current_year)
...                 last_name = current_last_name;
...         }
...
...         if (first_name.empty() && last_name.empty())
...             return "Incognito";
...
...         if (!first_name.empty() && last_name.empty())
...             return first_name + " with unknown last name";
...
...         if (first_name.empty() && !last_name.empty())
...             return last_name + " with unknown first name";
...
...         return first_name + " " + last_name;
...     }
... private:
...     map<int, string> firstNameHistory;
...     map<int, string> lastNameHistory;
... };
```

## 0.4 task4

Реализуйте класс ReversibleString, хранящий строку и поддерживающий методы Reverse для переворота строки и ToString для получения строки.

### 0.4.1 Пример:

```
>>> int main() {
...   ReversibleString s("live");
...   s.Reverse();
...   cout << s.ToString() << endl;
...
```

```
...     s.Reverse();
...     const ReversibleString& s_ref = s;
...     string tmp = s_ref.ToString();
...     cout << tmp << endl;
...
...     ReversibleString empty;
...     cout << '"' << empty.ToString() << '"' << endl;
...
...     return 0;
... }
```

### 0.4.2 Вывод:

```
evil
    live
```

```
>>> class ReversibleString {
... private:
...     string _value;
... public:
...     explicit ReversibleString(string value="") : _value(std::move(value))
...     {}
...
...     void Reverse()
...     {
...         reverse(begin(_value), end(_value));
...     }
...
...     [[nodiscard]] string ToString() const
...     {
...         return _value;
...     }
... };
```