CS35X: Competitive Programming

Lecture 2: arrays, strings

Joshua Brody

Warmup Kattis Problems: dragafra, inputscandal

Problem debrief: pet

C++ has built-in arrays

Static array allocation:

- `int m[5];`
- ```
  for(int i=0; i<5; i++) {
      m[i] = i*i;
  }
  ```

Dynamic array allocation:

- `int n=5;`
- `int *m = new int[n];`
- …
  ```

C++ arrays

- C++ arrays are ***very simple***.
- indexing: `m[i] = i*i;`
- No slicing, no length.
- In practice, **arrays are fast and easy to use.**
- 2D-array allocation:
  - `int **m = new int*[5];`
  - ```
    for(int i=0; i<5; i++) {
        m[i] = new int[10];
    }
    ```
  - indexing in 2-D array: `m[3][8];`
- Note: often in CP problems it's easier to use 1D array of *strings*.
    ```

Passing C++ arrays into functions

- In practice you'll always need to pass the array and the size:

```
void foo(int *m, int size) {
    …
}
```

Kattis Problem: cprnummer

C++ strings

The C++ string library provides a data type for **strings** and many many **string methods**. Do **not** ***memorize*** all the methods.

You should know how to:

- Import the string library.
- Access the string documentation on cplusplus.com
- Access individual characters in a string
- Compare strings
- Convert to/from other data types
- Use common string methods: **length**, **find**, **+=**, **substr**

C++ string examples

- `#include <string>`
- `string s = “hello”, z = “banana”;`
- `cout << s.length();  // 5`
- `s += “ world”;  // string concatenation`
- `if(s<z) {…}  // string comparison`
- `cout << s[4];  // “o”`
- `cout << s.substr(6,3);  // “wor”`
- `cout << s.find(“llo”);  // 2`
- `z = to_string(554);  // “554”`
- `int m = stoi(“8890”);  // 8890.  Requires #include <stdio.h>`

Kattis Problem: natrij