More C++ Features

Troy Serapio

Functions

- It's just like the ones from math!!
- Takes in input, does stuff to it, and gives an output!

Functions

- It has the following key features:
  - Name (that you call it by)
  - Parameters (the input, could be none)
  - Return Type (type of data it will output, could be none)

Function (Example)

```
string reverseString(string inputString) {
    string reversed = "";
    for (int i = inputString.length() - 1; i >= 0; i++) {
        reversed += inputString[i];
    }
    return reversed;
}
```

Vectors

- Exactly like `list` in Python
- Acts as a **list of objects** in C++.
- You can even keep a list of `vector`s (also known as 2D vector)!

Vectors

- $O(1)$ operations:
  - `v.size()` → length of `v`
  - `v.begin()` → beginning of `v`
  - `v.end()` → end of `v`
  - `v.push_back(val)` → append `val` to `v`
  - `v.pop_back()` → remove back of `v`
  - `v[i]` → `i` th element of `v`, 0-indexed
- $O(n)$ operations:
  - `v.resize(N)` → make `v` have length of `N`
  - `v.assign(N, val)` → make first `N` elements have value `val`

Vectors (Example)

```
int main() {

    vector<string> goatsOfNOI2024;
    goatsOfNOI2024.resize(4);

    goatsOfNOI2024[0] = "Gabee De Vera";
    goatsOfNOI2024[1] = "Filbert Wu";
    goatsOfNOI2024[2] = "Jerome Te";
    goatsOfNOI2024[3] = "Walsh Letran";

    goatsOfNOI2024.push_back("Farmer John's Goat #1");
    goatsOfNOI2024.pop_back();

    goatsOfNOI2024.push_back("Marco Arcallana");

}
```

Pairs

- Has exactly **two** elements
- Can vary in data type
- Useful for storing data that, comes in pairs!

Pairs (Example)

```
int main() {

    pair<int, int> fibonacci = {1, 1};
    int N = 1000;

    while (N--) {
        int tmp = fibonacci.first;
        fibonacci.first = fibonacci.second;
        fibonacci.second = fibonacci.first + tmp;
    }

    cout << fibonacci.second << "\n";

}
```

Structs

- For our purposes, its "`pair` but better"
- Groups different variables (members) under one name.
- Ideal for bundling related data (e.g., coordinates, intervals).
- Make custom comparators to do operations with other `struct`s of the same type (covered in Greedy Algorithms with Custom Comparators)

Structs (Example)

```
struct Interval {
    char label;
    int start;
    int end;
    int importance;
};

Interval a = {'A', 0, 10, 100};
Interval b = {'B', 0, 5, 1};
Interval c = {'C', 5, 10, 10};
```

Got more questions and clarifications?

Ask the Reboot Discord server!!