C++ Confidence Course 2018

Hands-On #1

Task: Fond Memories

Write a function that takes in two integers n and r and computes $\binom{n}{r}$. The function signature should be `int binom(int n, int k)`. Turn this into a complete program that reads input and prints output.

Note that `int N; cin >> N;` is used to read an integer input into an integer variable `N`.

Hands-On #2

Task: Ice Milo without Ice, Mergesort without Sort

Write a function that takes two sorted integer arrays of undetermined length and merges them into a single sorted array. Your function should have the signature `int* merge(int* A, int* B)`.

Each input array contains only nonzero elements and is terminated by a zero. The terminal zero is **not** treated as one of the array elements.

Hands-On #3

Task 1 - Reinventing the `std::vector`

We have seen a `FixedArray` class in the lecture slides. Now create a `DynamicArray` class that behaves as a dynamically resizable array.

Your class should contain the following methods:

- `void add(int x)` : appends an element to the back of the array
- `int remove()` : removes an element from the back of the array
- [OPTIONAL] `int remove (int pos)` : removes an element from any arbitrary position in the array (no need to bounds-check)

Your `DynamicArray` should grow and shrink as needed. Do **not** use `std::vector` . That defeats the whole purpose of this exercise. xD

[OPTIONAL] Try to do this efficiently… `add` and `remove` should run in $O(1)$ amortised.

Task 2 - Game Development

Your task is to create a `Vector3` class that represents a vector in $\mathbb{R}^3$. Objects of this class are expected to behave 'naturally', or in particular are expected to support:

- addition using `Vector3 operator+(Vector3 const& o)`
- scalar product using `Vector3 operator*(double c)`
- dot product using `Vector3 operator*(Vector3 const& o)`

[OPTIONAL] Extend `Vector3` to `VectorN` , representing a vector in $\mathbb{R}^n$. If you're really bored, extend it further to `VectorNC` that represents a vector in $\mathbb{C}^n$, making use of (or extending) the `Complex` class.

Note that the dot product in $\mathbb{C}^n$ is defined as $v \bullet w = \sum_{i=1}^{n} v_i \overline{w_i}$, where $\overline{a+bi} = a - bi$ (complex conjugation). It reduces to $v \bullet w = \sum_{i=1}^{n} v_i w_i$ in $\mathbb{R}^n$.