1. **Vector (std::vector)**: It’s a dynamic array that can grow and shrink in size. Commonly used functions include push\_back(), pop\_back(), size(), begin(), end(), etc.

Example:

std::vector<int> v;

v.push\_back(1); // v: [1]

v.push\_back(2); // v: [1, 2]

v.pop\_back(); // v: [1]

1. **Stack (std::stack)**: It’s a container adapter that provides Last-In-First-Out (LIFO) type of data structure. Commonly used functions include push(), pop(), top(), empty(), etc.

Example:

std::stack<int> s;

s.push(1); // s: [1]

s.push(2); // s: [1, 2]

s.pop(); // s: [1]

1. **Queue (std::queue)**: It’s a container adapter that provides First-In-First-Out (FIFO) type of data structure. Commonly used functions include push(), pop(), front(), back(), empty(), etc.

Example:

std::queue<int> q;

q.push(1); // q: [1]

q.push(2); // q: [1, 2]

q.pop(); // q: [2]

1. **Priority Queue (std::priority\_queue)**: It’s a container adapter that provides a queue with the highest priority element always on top. Commonly used functions include push(), pop(), top(), empty(), etc.

Example:

std::priority\_queue<int> pq;

pq.push(1); // pq: [1]

pq.push(2); // pq: [2, 1]

pq.pop(); // pq: [1]

1. **Unordered Map (std::unordered\_map)**: It’s an associative container that contains key-value pairs with unique keys. Commonly used functions include insert(), erase(), find(), size(), etc.

Example:

std::unordered\_map<std::string, int> um;

um.insert({"apple", 1}); // um: [("apple", 1)]

um["banana"] = 2; // um: [("apple", 1), ("banana", 2)]

um.erase("apple"); // um: [("banana", 2)]

Sure, here are a few more commonly used C++ Standard Library (STL) components:

1. **Set (std::set)**: It’s an associative container that contains a sorted set of unique objects. Commonly used functions include insert(), erase(), find(), size(), etc.

Example:

std::set<int> s;

s.insert(1); // s: [1]

s.insert(2); // s: [1, 2]

s.erase(1); // s: [2]

1. **Unordered Set (std::unordered\_set)**: It’s an associative container that contains a set of unique objects of the same type. Commonly used functions include insert(), erase(), find(), size(), etc.

Example:

std::unordered\_set<int> us;

us.insert(1); // us: [1]

us.insert(2); // us: [1, 2]

us.erase(1); // us: [2]

1. **Map (std::map)**: It’s an associative container that contains sorted key-value pairs. Commonly used functions include insert(), erase(), find(), size(), etc.

Example:

std::map<std::string, int> m;

m.insert({"apple", 1}); // m: [("apple", 1)]

m["banana"] = 2; // m: [("apple", 1), ("banana", 2)]

m.erase("apple"); // m: [("banana", 2)]

1. **String (std::string)**: It’s a sequence of characters. Commonly used functions include length(), substr(), find(), replace(), etc.

Example:

std::string str = "Hello, World!";

std::cout << str.length(); // Output: 13

1. **Algorithm (std::algorithm)**: It’s a collection of functions especially designed to be used on ranges of elements. Commonly used functions include sort(), reverse(), max\_element(), min\_element(), etc.

Example:

std::vector<int> v = {3, 1, 4, 1, 5, 9};

std::sort(v.begin(), v.end()); // v: [1, 1, 3, 4, 5, 9]

Absolutely, here are some more commonly used C++ Standard Library (STL) components:

1. **List (std::list)**: It’s a container that supports constant time insert and erase operations anywhere within the sequence. Commonly used functions include push\_back(), push\_front(), pop\_back(), pop\_front(), size(), etc.

Example:

std::list<int> l;

l.push\_back(1); // l: [1]

l.push\_front(2); // l: [2, 1]

l.pop\_back(); // l: [2]

1. **Deque (std::deque)**: It’s a container with dynamic sizes that can be expanded or contracted on both ends. Commonly used functions include push\_back(), push\_front(), pop\_back(), pop\_front(), size(), etc.

Example:

std::deque<int> d;

d.push\_back(1); // d: [1]

d.push\_front(2); // d: [2, 1]

d.pop\_back(); // d: [2]

1. **Pair (std::pair)**: It’s a simple container defined in <utility> header consisting of two data elements or objects. The first element is referenced as first and the second element as second.

Example:

std::pair<std::string, int> p;

p.first = "apple";

p.second = 1; // p: ("apple", 1)

1. **Tuple (std::tuple)**: It’s a single data structure that can hold elements of different types. It’s defined in <tuple>header.

Example:

std::tuple<std::string, int, double> t;

t = std::make\_tuple("apple", 1, 0.5); // t: ("apple", 1, 0.5)

1. **Bitset (std::bitset)**: It’s a container that can store bits (elements with only two states: 0 or 1). It’s defined in <bitset> header.

Example:

std::bitset<8> b(std::string("10110011")); // b: 10110011

These are just a few examples and there are many other STL components and functions that are commonly used in competitive programming. For a more comprehensive list, you may want to refer to the C++ STL documentation or other online resources. If you need more specific information or examples, feel free to ask! 😊