**Iterators**

An iterator identifies, access, and move between elements in a container (like pointers, but we won't need to work with pointers anymore)

"Random access" meaning that you can get access to elements in O(1)

* operator[] is not supported by all data structures
* Some functions requires iterator as parameters
* In STL, all containers have iterators defined
  + std::vector<int>::iterator
  + std::list<int>::iterator

**std::vector functions**

my\_vector.begin() 🡪 Returns an iterator to the first element

my\_vector.end() 🡪 Returns an iterator to the element following the last element

Graphical user interface, text, application

Description automatically generated

**Operating on iterators**

std::vector<int> numbers = {1, 2, 3, 4}

1. **Initializing**

std::vector<int>::iterator it = numbers.begin();

1. **Moving between elements**

it+; it--; it = numbers.begin() + 2;

1. **Using element value**

int x = \*it;

1. **Modify element values**

\*it = 27;

1. **Remove an element** erase(iterator)

🡪 Removes the element from the vector. However, O(n) complexity (no holes in vectors)

**auto keyword**

auto asks the compiler to define the type for us 🡪 we do not need to define the type when declaring an integer.

auto bruh\_bruh = 12.3;

for (auto iter = numbers.begin(); iter != numbers.end(); iter++)

**Ranged base for loop:**

We can also use auto for a syntactic sugar:

Graphical user interface, application

Description automatically generated