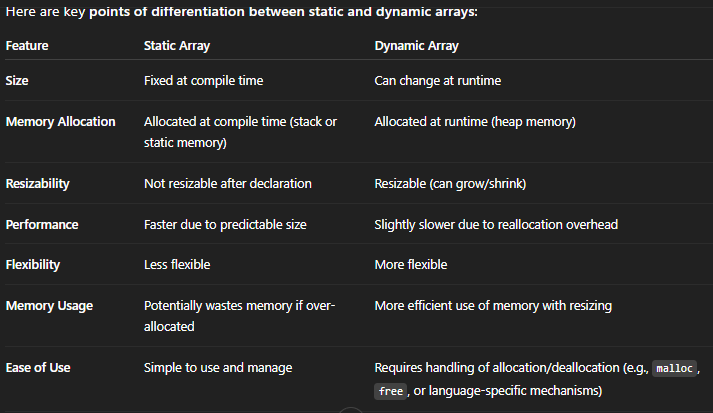
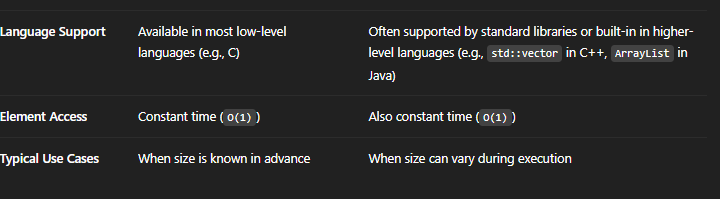
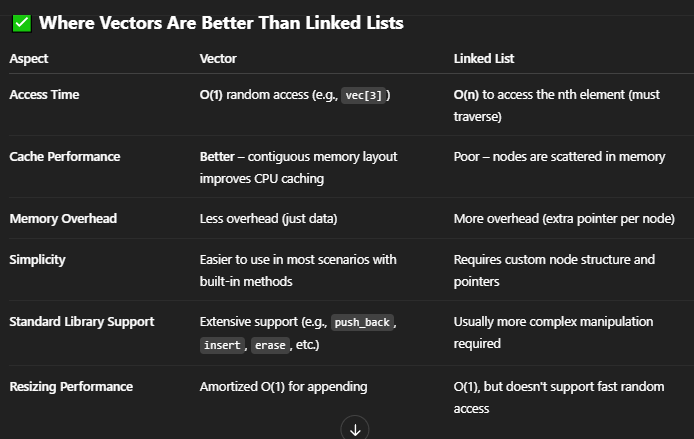
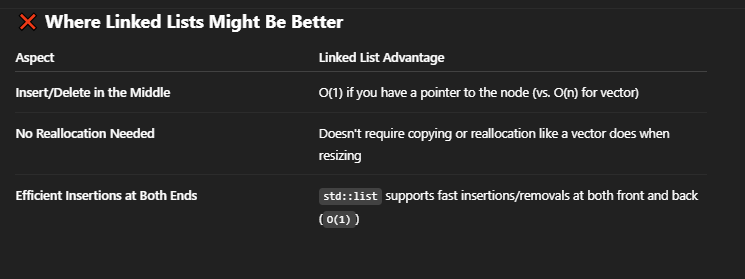
****

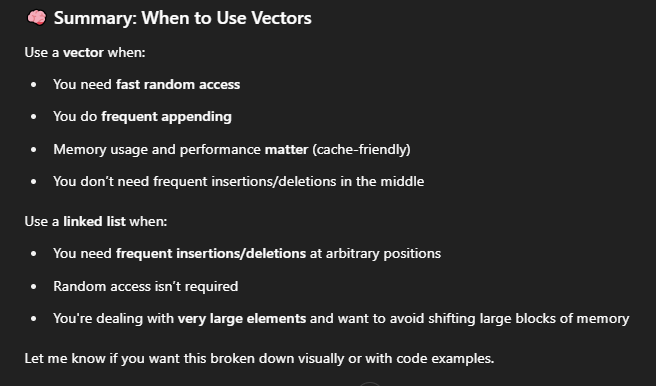
****

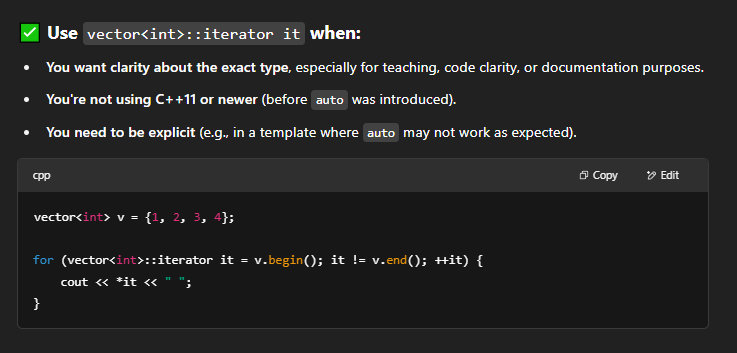
**DYNAMIC ARRAYS COPY INTO A NEW BIGGER/SMALLER ARRAY. THIS IS MEMORY EFFIECIENT BUT WASTES SOME TIME**

**VECTORS VS LINKED LIST 🡪**

****

****

****

****

**USE AUTO NORMALLY, UNLESS YOU WANT READABILITY OR WORKING WITH TEMPLATES CAUSE THEN YOU NEED TO SPECIFY THE TYPE USING ‘TYPENAME’ KEYWORD.**

****

auto it = numbers.begin();

cout << \*(it + 5);

**BASICALLY MOVES THE POINTER TO AAGE KI POSITION**

**five steps aage to be exact 🡪**

****

**CODES :: VECTORS**

**BEFORE FUNCTIONS 🡪**

#include<iostream>

#include<vector>

using namespace std;

int main() {

vector<int> vecnum;

// kinda like arrays so random access is O(1), but insertion/deletion is O(n)

// arrays but dynamic, ordered, so elements at indices can be accessed

vecnum.push\_back(1);

vecnum.push\_back(2);

vecnum.push\_back(3);

vecnum.push\_back(4);

vecnum.push\_back(5);

vecnum.push\_back(0);

vecnum.pop\_back();

for (auto it = vecnum.begin();it!= vecnum.end();it++ ) {

// \*it = 100;

// to avoid changing...use .cbegin() and .cend() --> constant :)

cout << \*it << endl;

}

**ALL VECTOR FUNCTIONS 🡪**

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

int main() {

vector<int> vecnum = {0,1,2,3,4,5 };

cout << "size of the vector: " << vecnum.size() << endl;

cout << "total possible size of the vector: " << vecnum.max\_size() << endl;

/\*cout << "resizing vector, shrinking to 2 elements: " << endl;

vecnum.resize(2);

cout << "new array: " << endl;

for (auto it = vecnum.begin(); it != vecnum.end(); it++) {

cout << \*it << endl;

}

\*/

cout << "total capacity pf vector before resizing is needed : " << vecnum.capacity() << endl;

cout << "no. of elements you can store in this vector,before new memory is dynamically allocated\n" << endl;

cout << "checking if vector is empty: " << endl;

if (vecnum.empty())

cout << "vector is empty" << endl;

else

cout << "vector is not empty" << endl;

cout << "\nprinting specific elements: " << endl;

cout << "\n if sure about range..like its gonna stay in range and faster performance, use []" << endl;

cout << vecnum[0] << endl;

cout << "if uncertain about range and wanna play safe, use .at()" << endl;

cout << vecnum.at(0) << endl;

cout << "\nprinting first and last element: " << endl;

cout << " printing first element: " << vecnum.front() << endl;

cout << " printing last element: " << vecnum.back() << endl;

cout << "\nREMEMBER --> if u wanna REMOVE last, use .pop\_back(), for PRINT use .back()\n" << endl;

// cout << "\n deleting all elements: " << endl;

// vecnum.clear();

cout << "inserting and deleting !!!" << endl;

cout << "INSERTION \n" << endl;

vecnum.insert(vecnum.begin() + 6, 100);

for (auto it = vecnum.begin(); it != vecnum.end(); it++) {

cout << \*it << endl;

}

cout << "/n deleting the second element\n" << endl;

vecnum.erase(vecnum.begin() + 1);

for (auto it = vecnum.begin(); it != vecnum.end(); it++) {

cout << \*it << endl;

}

vector<int> repeated = { 1,2,3,4,2 };

cout << "\n\n NEW VECTOR WITH REPEATED ELEMENTS\n";

for (auto it = repeated.begin(); it != repeated.end(); it++) {

cout << \*it << endl;

}

cout << "\n removing the first occurence of 2, include algorithm" << endl;

int target = 2;

auto it = find(repeated.begin(), repeated.end(), target);

if (it != repeated.end()) {

repeated.erase(it); // removes only the first 2

cout << "new vector: \n" << endl;

for (auto it = repeated.begin(); it != repeated.end(); it++) {

cout << \*it << endl;

}

}

return 0;

}