

**templates can be used not just for classes, but also for functions. basically its just generic**

**TEMPLATES IN CLASSES:**

#include<iostream>

using namespace std;

**template <typename arru>**

**class Calculator {**

public:

arru n1;

arru n2;

arru add(arru n1, arru n2) {

return n1 + n2;

}

arru sub(arru n1, arru n2) {

return n1 - n2;

}

arru mul(arru n1, arru n2) {

return n1 \* n2;

}

arru div(arru n1, arru n2) {

return n1 / n2;

}

Calculator() {}

Calculator(arru x, arru y) {

n1 = x;

n2 = y;

}

};

int main() {

**Calculator<int> c1;** // name which type ka calculator

**Calculator<float> c2(2.2,1.1);**

cout << c1.sub(10, 9) << endl;

cout << c2.sub(2.2,1.1) << endl;

return 0;

}

**three days later:**

**24/5**

**tried two type names and dude I was right myself first try !!!**

**here’s a calculator that takes two variables of different types 🡪**

// revising templates

#include<iostream>

using namespace std;

template <typename a, typename b>

class Calculator {

public:

a first;

b second;

Calculator() {}

Calculator(a x, b y) {

first = x;

second = y;

}

void add(a x, b y) {

cout << " sum : " << x + y << endl;

}

void sub(a x, b y) {

cout << "difference : " << x - y << endl;

}

};

int main() {

Calculator <int, float> c1;

Calculator <float, int> c2;

cout << "int 4 and float 3.2: " << endl;

c1.add(4, 3.2);

c1.sub(4, 3.2);

cout << "float 3.4 and int 3: " << endl;

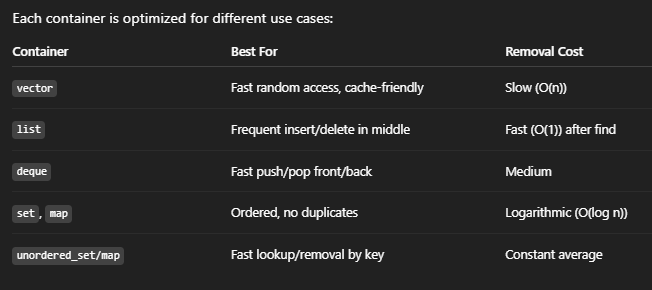
c2.add(3.4, 3);

c2.sub(3.4, 3);

return 0;

}

STL CONTAINERS 🡪

****