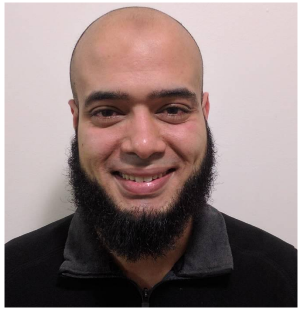
C++ Programming

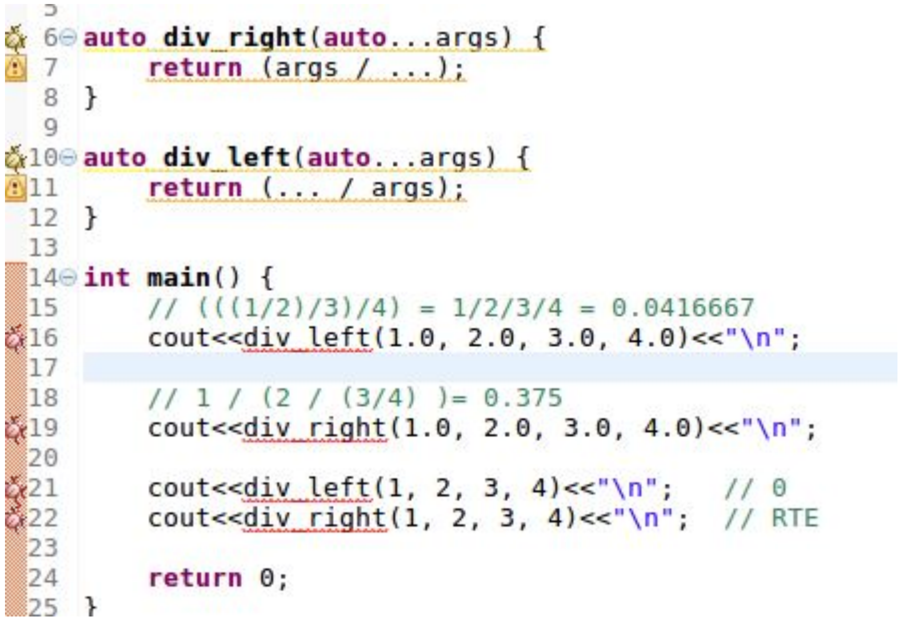
Fold Expression 2

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



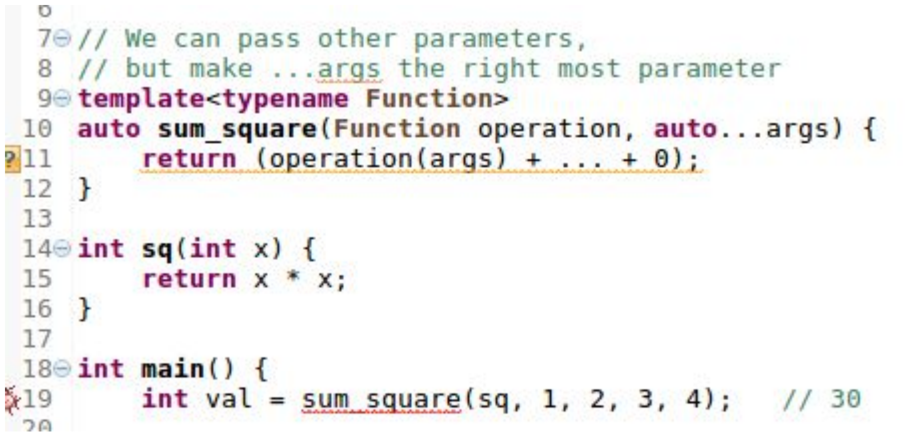
Division



```
auto div_right(auto...args) {
    return (args / ...);
}

auto div_left(auto...args) {
    return (... / args);
}

int main() {
    // (((1/2)/3)/4) = 1/2/3/4 = 0.0416667
    cout<<div_left(1.0, 2.0, 3.0, 4.0)<<"\n";

    // 1 / (2 / (3/4) )= 0.375
    cout<<div_right(1.0, 2.0, 3.0, 4.0)<<"\n";

    cout<<div_left(1, 2, 3, 4)<<"\n";   // 0
    cout<<div_right(1, 2, 3, 4)<<"\n";  // RTE

    return 0;
}
```

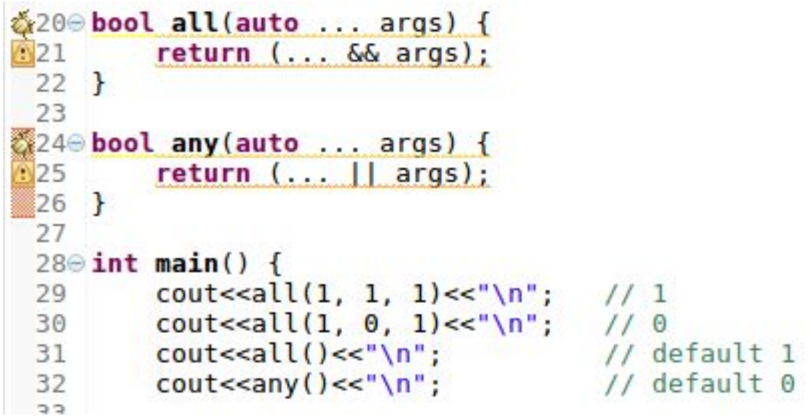
Applying function/functor

- Think in args as as **args(0)** after expansion.
  - We can apply operation directly arg +
  - some_function(args)
  - (some expression over args)



```
// We can pass other parameters,
// but make ...args the right most parameter
template<typename Function>
auto sum_square(Function operation, auto...args) {
    return (operation(args) + ... + 0);
}

int sq(int x) {
    return x * x;
}

int main() {
    int val = sum_square(sq, 1, 2, 3, 4);   // 30
```

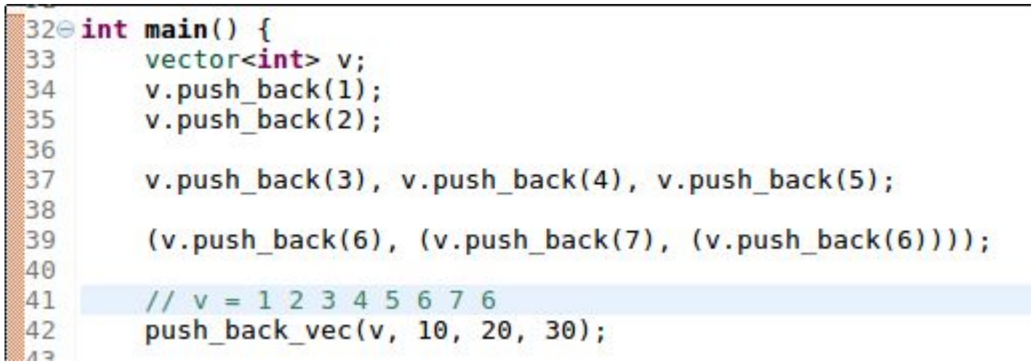
No need for initial values for && || ,



```
bool all(auto ... args) {
    return (... && args);
}

bool any(auto ... args) {
    return (... || args);
}

int main() {
    cout<<all(1, 1, 1)<<"\n";   // 1
    cout<<all(1, 0, 1)<<"\n";   // 0
    cout<<all()<<"\n";          // default 1
    cout<<any()<<"\n";          // default 0
```

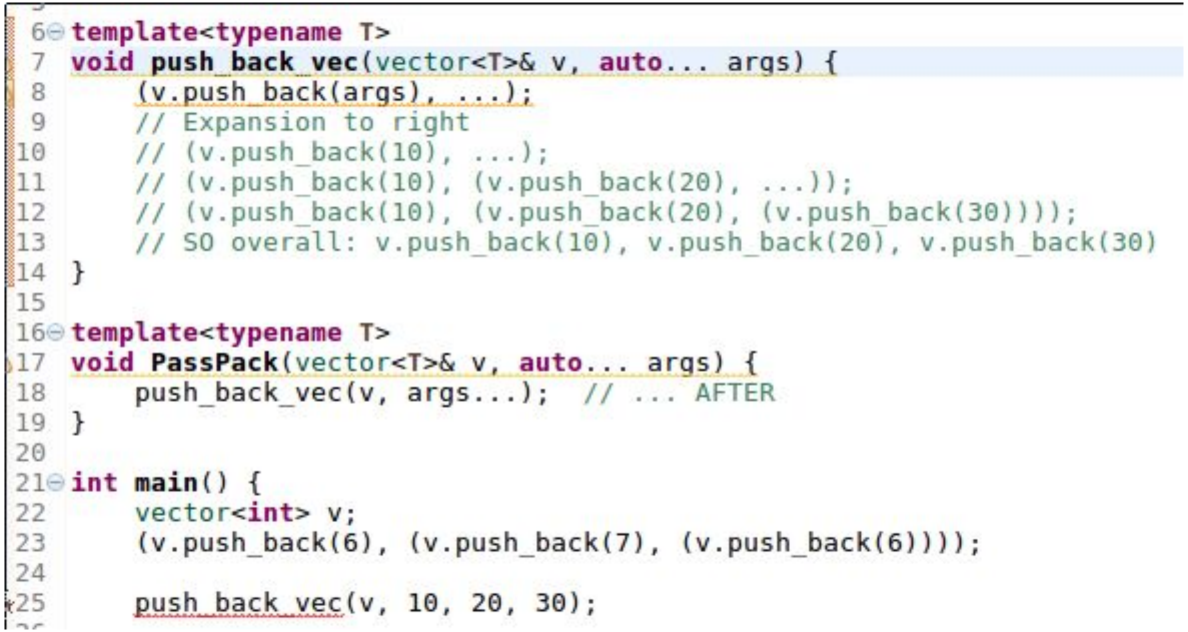
Recall comma operator

- Evaluate left to write (and return value of last expression)
- Use comma operator to do sequential steps
  - E.g. Push items to the vector: Give a trial



```
int main() {
    vector<int> v;
    v.push_back(1);
    v.push_back(2);

    v.push_back(3), v.push_back(4), v.push_back(5);

    (v.push_back(6), (v.push_back(7), (v.push_back(6))));

    // v = 1 2 3 4 5 6 7 6
    push_back_vec(v, 10, 20, 30);
```

Comma operator (no initial)



```
template<typename T>
void push_back_vec(vector<T>& v, auto... args) {
    (v.push_back(args), ...);
    // Expansion to right
    // (v.push_back(10), ...);
    // (v.push_back(10), (v.push_back(20), ...));
    // (v.push_back(10), (v.push_back(20), (v.push_back(30))));
    // SO overall: v.push_back(10), v.push_back(20), v.push_back(30)
}

template<typename T>
void PassPack(vector<T>& v, auto... args) {
    push_back_vec(v, args...);  // ... AFTER
}

int main() {
    vector<int> v;
    (v.push_back(6), (v.push_back(7), (v.push_back(6))));

    push_back_vec(v, 10, 20, 30);
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."