Template functions can be used to define generic algorithms.

template <class T>

T max(T x, T y)

{

return x < y ? y: x;

}

template <class T>

const T& max(const T& x, const T& y)

{

return x < y ? y: x;

}

The latter definition is used in STL

int u = 3, v= 4;

double d = 4.7, e = 5.9;

cout << max(u, v) << endl;

cout << max(d, e) << endl;

cout << max(u, d) << endl; //error : types do not match

The compiler requires that values of the same type be passed for x and y since the same template type parameter, T, is used for both in the declaration.

Writing the following

pair<double, long> pair1(3.145, 999);

pair<double, long> pair2(3.146, 998);

max(pair1, pair2);

will not compile because there is no definition of operator< on pair<double, long> objects. But it would compile if we first defined some meaning for operator< on objects of this type, such as

bool operator<(const pair<double, long>& x,

const pair<double, long>& y)

//Compare x and y on their first members

{

return x.first < y.first;

}