Back to Resources

# Templates

by Eddie Huang

Templates can help make your functions and classes more flexible over different data types

## A function template for the maximum of two items

```
#include <iostream>
using namespace std;

template <class T>
T GetMax (T a, T b) {
  T result;
  result = (a>b)? a : b;
  return (result);
}
```

```
int main () {
  int i=5, j=6, k;
  long l=10, m=5, n;
  k=GetMax<int>(i,j);
  n=GetMax<long>(l,m);
  cout << k << endl;
  cout << n << endl;
  return 0;
}
```

**Output**

```
6
10
```

## A class template of pairs of items

```
// class templates
#include <iostream>
using namespace std;

template <class T>
class mypair {
    T a, b;
  public:
    mypair (T first, T second)
      {a=first; b=second;}
    T getmax ();
};

template <class T>
T mypair<T>::getmax ()
{
  T retval;
  retval = a>b? a : b;
  return retval;
}
```

```
int main () {
  mypair <int> myobject (100, 75);
  cout << myobject.getmax();
  return 0;
}
```

**Output**

```
100
```