Templates:

*Templates* are used to have one block of code for different data types, or different class types.

* Templates need to be prototype defined before main -
* The syntax above template function or template class is:

**template <class T>**

// Template FUNCTION example for use with output

// prototype definition

template <class T>

void output (const T & n)

{

cout << n << endl;

}

// Implementation

int main()

{

int testInt = 3;

float testFloat = .2f;

std::string testString = “hello world”

Output(testInt);

Output(testFloat);

Output(testString);

}

// Template FUNCTION example for swapping

// prototype definition

template <class myT>

void swap\_these(myT & a, myT & b)

{

myT a\_old = a;

a = b;

b = a\_old;

}

// Implementation

int main(){

int array[5] = { 1, 2, 3, 5, 4 };

swap\_these(array[3], array[4]); } // main closing bracket

Template CLASS example

// Class prototype & definition

//template <class T>

//class A

//{

// private:

// T data;

//

//public:

// A(const T & init);

//

// void output();

//};

//template <class T>

//A<T>::A(const T & init)

//{

// data = init;

//}

//

//template <class T>

//void A<T>::output()

//{

// cout << data << endl;

//}

// Implementation

int main()

{

*A* <int> intInstance(10);

*A* <char> charInstance('c');

*A* <float> floatInstance(0.1f);

intInstance.output();

charInstance.output();

return 0;

}