Templates:-

Generic Programming

```
int sum(int x,int y)
{
    int res;
    res = x + y;
    return res;
}
float sum(float x,float y)
{
    float res;
    res = x + y;
    return res;
}
char sum(char x,char y)
{
    char res;
    res = x + y;
    return res;
}
Complex sum(Complex c1,Complex c2)
{
    Complex res = c1 + c2;
    result res;
}
```

Not Need

Function Templates

```
template<typename T>
T sum(T x,T y)
{
    T res;
    res = x + y;
    return res;
}
int main()
{
    int a=10,b=20,c;
    float x=2.3f,y=5.6f,z;
    char c1='A', c2=' ', c3;

    c=sum(a,b);
    z=sum(x,y);
    c3=sum(c1,c2);
    return 0;
}
```

Type deduction of template parameter
- Implicit, based on arguments
- Explicit, as per type mentioned

```
template<typename T>
void myswap(T& ref1, T& ref2)
{
    T temp = ref1;
    ref1 = ref2;
    ref2 = temp;
}

myswap(a,b);
myswap(x,y);
myswap(c1,c2);
```

```
sum(a,y);           //error
sum<int>(a,y);

sum(x,b);           //error
sum<float>(x,b);

sum(a,c2);          //error
sum<char>(a,c2);
```

```
void iswap(int& x,int& y)
{
    int res = x;
    x = y;
    y = res;
}

iswap(a,b);
```

```
template<typename T>
T sumarr(T arr[], int len)
{
   T total = 0;
   for(int i=0;i<len;i++)
        total += arr[i];
   return total;
}

template<typename T>
T gsearch(T arr[], int len, T key)
{
   //TODO
}
template<typename T>
void gsort(T arr[], int len)
{

}
```

Few probs:-
* Generic multiplication
* Generic min/max
* Generic swap

* Generic sort
* Generic search

Template class
- Stack
- Queue
- Array, Iterator
- LinkedList

Limited Scope (limited to primitive types
- Point
- Complex

```
MyArray<int> a1(10);
```

STL Containers
- std::vector
- std::list
- std::map
- std::set

std::vector
- create, ctors
- size()
- capacity()
- front()
- back()
- front()
- back()
- empty()

- at(index)
- [](index)

```
std::vector<int> v0(10);
std::vector<int> v2(15,8);
std::vector<int> v3;
std::vector<int> v1{11,12,13,14,15};

v0.capacity()   //10

v1.capacity()   //5
v1.size()       //5
v1.push_back(16);
v1.size()       //6
v1.capacity()   //10
v1.push_back(17);
v1.size()       //7
v1.capacity()   //10

v1.front()      //11
v1.back()       //17

v1.at(3)        //14
v1[3]           //14

v1.pop_back()   //17
v1.size()       //6
```

```
//M1 : subscript based
for(int i=0;i<v1.size();i++)
   sum += v1.at(i);
   // sum += v1[i]


//M2 : range based loop
for(int val: v1)
    sum += val
```

```
//M3 : iterator based
MyVector<int>::iterator iter;

for(iter=v1.begin();iter!=v1.end();++iter)
    sum += *iter;
```