Templates in C++

- **Template** function in C++ makes it easier to reuse classes and functions.
- A template can be viewed as a variable that can be instantiated to any data type, irrespective of whether this data type is a fundamental C++ type or a user-defined type.

Selection Sort Template

```
1.  template <class  T>
2.  void SelectionSort(T  *a,  const int n)
3.  {// sort a[0] to a[n-1] into nondecreasing order.
4.       for (int i = 0;  i < n;  i++)
5.       {
6.            int j = i;
7.            // find smallest integer in a[i] to a[n-1]
8.            for (int k = i+1;  k < n;  k++)
9.                 if (a[k] < a[j])  {j = k;}
10.           swap(a[i],  a[j]);
11.      }
12. }
```

```
float farray[100];
int intarray[200];
……..
sort(farray, 100);
sort(intarray, 200);
```

Program 3.1: Selection sort using templates

Program 3.4

```
Class Bag
{
  public:
    Bag();
    ~Bag();
    …
  private:
     int *array;
     int  capacity;
    …
}
```

```
Bag::~Bag(){delete array;}
```

Program 3.6

```
Template <class T>
Class Bag
{
  public:
    Bag();
    ~Bag();
    …
  private:
     T *array;
     int  capacity;
    …
}
```

```
template <class T>
Bag<T>::~Bag(){delete array;}
```

Declaration:
Bag <int> a;
Bag <Rectangle> a;