Templates

Templates are a good way of making classes more abstract by letting you define the behavior of the class without actually knowing what data type will be handled by the operations of the class. The advantage of having a single class - that can handle several different data types is that it makes the code easier to maintain, and it makes classes more reusable.

Syntax for one type of variable only:

```
template <typename T>
```

Syntax for two types of variables:

```
template <typename T, typename V>
```

For more than two types of variables, more type names can be added within the angular brackets.

For Example: You want to create a pair class and use it for different types of variables, like int, char, double etc... Follow the code below:

```
template <typename T>

class Pair {
        T x;            //variables x and y are declared of type T which
        T y;            //could take place of any known data type

        public :

        void setX(T x) {
                this -> x = x;
        }

        T getX() {
                return x;
        }
        void setY(T y) {
                this -> y = y;
        }

        T getY() {
                return y;
        }
};
```

Now suppose you want to create *x* of one type (like int) and *y* of another type (for eg. double), then you can do it in the following way:

```
template <typename T, typename V>

class Pair {
        T x;          //variables x and y are declared of type T and V respectively,
        V y;         //which could take place of any known data types(different/same)

        public :

        void setX(T x) {
                this -> x = x;
        }

        T getX() {
                return x;
        }

        void setY(V y) {
                this -> y = y;
        }

        V getY() {
                return y;
        }
};
```

To use these in the main() function, simply call the class like this:

```
Pair<int, double> p1;
```

and the pair of the required types will be created.