Scambiare due interi

```
void scambia (int& x, int& y){
    int temp = x;
    x = y;
    y = temp; }
```

Scambiare due caratteri

```
void scambia (char& x, char& y){
    char temp = x;
    x = y;
    y = temp; }
```

Scambiare due razionali

```
void scambia (Razionale& x, Razionale& y){
    Razionale temp = x;
    x = y;
    y = temp; }
```

Scambiare due pile

```
void scambia (Stack& x, Stack& y){
    Stack temp = x;
    x = y;
    y = temp; }
```

Template di funzione

```
template <typename T>
void scambia (T& x, T& y){
    T tmp = x;
    x = y;
    y = tmp; }
```

identificatore che denota il parametro tipo

la funzione scambia è parametrica rispetto al tipo T

- A seconda del datatype degli argomenti verranno istanziate funzioni diverse
- Vantaggi:
  - Genericità
  - No ridondanza e riuso del codice
  - Non implica perdita di rendimento

Esercizio

```
template <typename T>
T max_Return (T& x, T& y){
    if ( x > y){
        return x; }
    return y;
}
```

Template di classe

È possibile definire anche template di classe

Template di classe C parametrica in T

```
template <typename T>
class C {
        private:
                T z;
        public:
                int f(T x) {
                    ...
                }
                T f(z) {
                        T n;
                        ...
                        return n;
                }
};
```

la classe C è parametrica rispetto al tipo T

l'identificatore T può essere usato come un normale identificatore di datatype

Creazione di oggetti

```
Stack<int> s1;
```

il template di classe Stack<T> viene istanziato (specializzato) rispetto al tipo di int = Rimpiazza tutte le T con int

```
Stack<char> s2;
```

il template di classe Stack<T> viene istanziato (specializzato) rispetto al tipo di char = Rimpiazza tutte le T con char

```
Stack<Razionale> s3;
```

il template di classe Stack<T> viene istanziato (specializzato) rispetto al tipo di Razionale = Rimpiazza tutte le T con Razionale

Stack<int> ≠ Stack<char> ≠ Stack<Razionale>

anche se derivati dallo stesso template

Parametri di tipo multipli

```
template <class T1, class T2, class T3>

class C {
    ...
    }

int main () {

        C <T1, T2, T3> x;
          ...
        }
```