Klasa kao realizacija strukture podataka

```
template <typename T, int MaxStackSize>
class Stack {
public:
  Stack ();
  int push (T in);
  int pop (T* out);

private:
  T stack[MaxStackSize];  // Stack
  int sp; // Stack pointer
};
```

```
template <typename T, int MaxStackSize>
Stack<T,MaxStackSize>::Stack () {
  this->sp = 0;
}

template <typename T, int MaxStackSize>
int Stack<T,MaxStackSize>::push (T in) {
  if (this->sp==MaxStackSize) return -1;
  this->stack[this->sp++] = in;
  return 0;
}

template <typename T, int MaxStackSize>
int Stack<T,MaxStackSize>::pop (T* out) {
  if (this->sp==0) return -1;
  *out = this->stack[--this->sp];
  return 0;
}
```

- Konkretne klase se generišu na zahtev, kada se prvi put upotrebi konkretan tip sa stvarnim parametrima šablona:

```
Stack<unsigned,256> parPositions;
Stack<Figure*,256> moves;

…

parPositions.push(pos);
```

- Klasu generiše prevodilac i ona ima sve karakteristike obične klase, s tim što joj naziv sadrži sve parametre:

```
Stack<unsigned,256>
```

Klasa kao realizacija strukture podataka

Standardna biblioteka jezika C++ (tzv. *Standard Template Library*, STL) sadrži mnogo definisanih šablonskih klasa, funkcija i drugih elemenata; na primer:

- niske elemenata i znakova:
```
template<typename Z> class basic_string;
typedef basic_string<char> string;
```
- uređen par:
```
template<typename A, typename B> struct pair;
```
- kompleksan broj čije su komponente datog tipa:
```
template<typename T> class complex;
```
- vektor (niz dinamički proširivog kapaciteta):
```
template<typename E> class vector;
```
- red sa pristupom elementima sa oba kraja:
```
template<typename E> class deque;
```
- lista:
```
template<typename E> class list;
```
- red:
```
template<typename E, typename Z=deque<E> > class queue;
```
- prioritetni red:
```
template<typename E, typename Z=vector<E>, typename U=less<E> > class priority_queue;
```
- stek:
```
template<typename E, typename Z=deque<E> > class stack;
```
- skupovi, multiskupovi, mape