```
let compare x p q =
  if x = p then
    if x = q then "both"
      else "first"
    else
      if x = q then "second"
      else "neither";;
```

```
function min(x, y : integer)
    return integer is
begin
    if x < y then return x;
    else return y;
    end if;
end min;

function min(x, y : long_float)
    return long_float is
begin
    if x < y then return x;
    else return y;
    end if;
end min;
```

```
generic
    type T is private;
    with function "<"(x, y : T) return Boolean;
function min(x, y : T) return T;

function min(x, y : T) return T is
begin
    if x < y then return x;
    else return y;
    end if;
end min;

function int_min is new min(integer, "<");
function real_min is new min(long_float, "<");
function string_min is new min(string, "<");
function date_min is new min(date, date_precedes);
```

```
template<class item, int max_items = 100>
class queue {
    item items[max_items];
    int next_free, next_full, num_items;
public:
    queue() : next_free(0), next_full(0), num_items(0) { }
    bool enqueue(const item& it) {
        if (num_items == max_items) return false;
        ++num_items;  items[next_free] = it;
        next_free = (next_free + 1) % max_items;
        return true;
    }
    bool dequeue(item* it) {
        if (num_items == 0) return false;
        --num_items;  *it = items[next_full];
        next_full = (next_full + 1) % max_items;
        return true;
    }
};
...
queue<process> ready_list;
queue<int, 50> int_queue;
```

	Explicit (generics)				Implicit	
	Ada	**C++**	**Java**	**C#**	**Lisp**	**ML**
Applicable to	subroutines, modules	subroutines, classes	subroutines, classes	subroutines, classes	functions	functions
Abstract over	types; subroutines; values of arbitrary types	types; enum, int, and pointer constants	types only	types only	types only	types only
Constraints	explicit (varied)	implicit	explicit (inheritance)	explicit (inheritance)	implicit	implicit
Checked at	compile time (definition)	compile time (instantiation)	compile time (definition)	compile time (definition)	run time	compile time (inferred)
Natural implementation	multiple copies	multiple copies	single copy (erasure)	multiple copies (reification)	single copy	single copy
Subroutine instantiation	explicit	implicit	implicit	implicit	—	—