In [1]:
```
/** La fonction factorielle
 *  @param n un nombre entier positif
 *  @return n!
 **/
int factorielle(int n) {
    int resultat = 1;
    resultat = 1;
    for ( int i = 1; i <= n; i++ ) {
        resultat = resultat * i;
    }
    return resultat;
}
```

Tests

In [2]:
```
factorielle(1)
```

Out[2]:
```
1
type: int
```

In [3]:
```
factorielle(2)
```

Out[3]:
```
2
type: int
```

In [4]:
```
factorielle(3)
```

Out[4]:
```
6
type: int
```

In [5]:
```
factorielle(4)
```

Out[5]:
```
24
type: int
```

In [6]:
```
factorielle(7)
```

Out[6]:
```
5040
type: int
```

Tests partiellement automatisés

In [7]:
```
factorielle(1) == 1
```

Out[7]:
```
true
type: bool
```

In [8]:
```
factorielle(2) == 2
```

Out[8]:
```
true
type: bool
```

In [9]:
```
factorielle(3) == 6
```

Out[9]:
```
true
type: bool
```

In [10]:
```
factorielle(4) == 24
```

Out[10]:
```
true
type: bool
```

In [11]:
```
factorielle(7) == 5040
```

Out[11]:
```
true
type: bool
```

Mini infrastructure de tests

In [12]:
```
ASSERT( 1 < 2 );
```

In [13]:
```
ASSERT( 1 > 2 );
```

```
Standard Exception: Test failed: 1 > 2
```

Tests automatisés

In [14]:
```
ASSERT( factorielle(1) == 1 );
ASSERT( factorielle(2) == 2 );
ASSERT( factorielle(3) == 6 );
ASSERT( factorielle(4) == 24 );
ASSERT( factorielle(7) == 5040 );
```