

# Syntax

1. For simplicity, all variables are integers
2. The program consists of statements

- **if-else** statement

```
if (expression) {  
    ....  
} else {  
    ....  
}
```

- **while** loop

```
while (expression) {  
    ....  
}
```

- Assignment of an expression or read from stdin

```
x = 69;  
x = ?;
```

- Postfix increment and decrement

```
x++;  
x--;
```

- Print to stdout

```
print x;
```

3. Comments

```
// single-line comment  
  
/*  
    multiline  
    comment  
*/
```

4. Operator priority

1. ++, --
2. +, - // unary
3. \*, /
4. +, - // binary
5. >, <, >=, <=, ==, !=

# Functions and scopes in ParaCL

## 1. Functions

- Functions definition

```
name_of_function = func(argument declarations) {  
    body_of_function  
}
```

```
summ = func(x, y) {  
    x + y;  
}
```

- Only variables declared as function arguments or local variables can be accessed inside a function
- Result of function is result of last expression or explicit **return** statement

```
strange_summ = func(x, y) {  
    if (x < 0) {  
        return x - y;  
    }  
  
    -x + y;  
}  
  
y = 69;  
z = strange_sum(y, y); // z == 0  
a = strange_sum(-y, y); // a == -138
```

- It is possible to declare a global function name (for recursion, for example, because the local name is only visible in the scope of the declaration)

```
fibonacci = func(x): fib {  
    result = 1;  
    if (x > 1) {  
        res = fib(x - 1) + fib(x - 2);  
    }  
    res;  
}
```

## 2. Scopes

- The lifetimes is limited by scope

```
{  
    x = 69;  
}
```

```
print x;
```

```
Error: use of undeclared identifier 'x'  
5:7 | print x;  
    |      ^
```

- Scope returns result of last expression

```
x = {  
  y = 69;  
  z = { print y; }  
  y + z--;  
}  
  
print x; // x == 137
```