PFL User Guide

This guide explains how to write valid programs using the custom pfl language.

---

1. Program Structure

A program consists of zero or more lines. Each line must end with a semicolon `;`.

```
<line>;
<line>;
...
```

---

2. Line Types

Each line can be one of the following:

- Integer declaration
- List declaration
- Function definition
- Write call

Empty lines are also allowed.

---

3. Declarations

Integer Declaration

```
int x = <integer expression>;
```

Example:

```
int a = 10;
```

List Declaration

From a function call:

```
list l = F(...);
```

From a list of elements:

```
list l = [<int expr>, <int expr>, ...];
```

Example:

```
list l = [1, 2, 3];
```

---

4. Identifiers

- Function names must begin with an uppercase letter (e.g., `F`, `Compute`).
- Variable names must begin with a lowercase letter (e.g., `x`, `myList`).

---

5. Function Definitions

```
fun FunctionName(type param1, type param2, ...) {
    <statements>;
    ? (<boolean expression>) return <value>;
    : return <value>;
};
```

Example:

```
fun Add(int a, int b) {
    return (a + b);
};
```

---

6. Expressions

Integer Expressions

- Numbers: `10`
- Variables: `x`
- Binary operations: `(x + 1)`, `(a * b)`
- Function calls returning integers

Boolean Expressions

- Comparisons: `(x < y)`, `(a = b)`
- List checks: `(list.empty)`

---

7. Function Calls

```
FunctionName(arg1, arg2, ...)
```

Arguments can be expressions or values returned from other functions.

---

8. List Operations and Accessors

You may chain attributes on a list variable:

- `.head` — returns the first element
- `.tail` — returns the last element
- `.length` — returns the number of elements
- `.empty` — returns true if the list is empty

Mutating methods:

- `.pushHead(value)` — returns a new list with value added to the head
- `.pushTail(value)` — returns a new list with value added to the tail
- `.popHead` — returns the list without its head
- `.popTail` — returns the list without its tail

Example:

```
list l = [1, 2, 3];
int x = l.head;
list m = l.pushHead(0);
```

---

9. Conditionals (Ternary Structure)

```
? (condition) return <value>;
: return <value>;
```

This acts like a simple `if-else` structure.

Example:

```
? (x < 5) return a;
: return b.pushHead(x);
```

---

10. Return Statements

Return statements may return:

- Variables
- List operations (like `.popHead`, `.pushTail(1)`)
- Expressions (`(a + b)`)

Example:

```
return a;
return l.pushHead(1);
return (x + 1);
```

---

11. Write Statement

Print a value to the output.

```
write(<expression or variable>);
```

Examples:

```
write(a);
write(l.head);
```

---

12. Operators

Arithmetic

- `+` — Addition
- `-` — Subtraction
- `*` — Multiplication
- `/` — Division

Comparison

- `=` — Equality
- `<`, `>`, `<=`, `>=`

---

13. Notes

- Lists are immutable: operations like `.pushHead` return a new list.
- All lists are doubly linked and support head/tail access and push/pop operations.