Out of Date Biases

Parameter Lists

Parameters are pushed onto a heterogenous[1] stack.

The idea of parameter lists was borne out of:

- concern for memory conservation
- mathematical notation (which came from pen-and-paper-only thinking)
- strictly textual notations
- "everything mus be synchronous" thinking.

Exceptions

The idea of exceptions was borne out of:

- strictly textual notations
- synchronous-only thinking
- the idea of parameter lists (which cause accidental complexity in the design of exception syntax)

Return Values

Return values are placed on the stack.

The idea of return values was borne out of:

[1] Parameters of all types share the same stack.

- concern for memory conservation
- mathematical notation
- strictly textual notations
- "everything must be synchronous" thinking
- the idea of parameter lists (which cause accidental complexity in the design of return syntax).

Extra Work

The compiler must do extra work:

- to determine type safety
- to determine alignment.

A Stack is a (Degenerate) Collection

A stack is an optimization of list.

A list is a collection.

The concept of *stack* conflates several issues:

- scoping
- optimization.

Two Stacks for Every Type

Separate Collections

Every *type* might be stored in a separate collection.

The *parameters* to a routine might be a collection of typed collections.

Type Checking

Assertion: if something is in a typed collection, the it has the correct type.

Type check is done at "push" time (where "push" means to add the item to its collection).

Type Name Clashes

What can we do if two parameters have the same type?

One solution - allow type synonyms.

Each parameter gets a unique name, but the name is synonymed to be of a given type.

E.G.

```
fn(a : int, b : int)
```

becomes

```
fn(a,b)
a = int
b = int
```

Type Stack Operations

TBD

I have implemented a version of the above ideas and will document the operations (about 5) that I use (push-and-check-type, pop, list-add, etc.)

PT Pascal and S/SL

The PT Pascal compiler, written in the language S/SL, used stacks for scoping and type checking. The PT Pascal source code and S/SL source code can be found at https://research.cs.queensu.ca/home/cordy/pub/downloads/ssl/.

A version of S/SL with input and output parameters was documented in a thesis - I can't find my copy of the thesis nor can I remember the author's name.