Note

From: ~~S.G.~~ S.G.

To:

Subject: ~~STACK (32, 2stack, par)~~

Date: ~~2018-10-20~~

Starta generic type
2018-11-02

(1)

1. ~~Procedu~~ Subroutines on arbitrary types:

push(array of T, T): array of T

⇒ a) for non-reference types, compile anew;
b) for reference types, use type erasure:

push(array of <T>, <T>): array of <T>

or even:

push (array of <ref T>, <ref T>): array of <ref T>

! solves also the 'new array of <T>' problem!

2. Containers of previously unknown type:

a) type array of T = ...

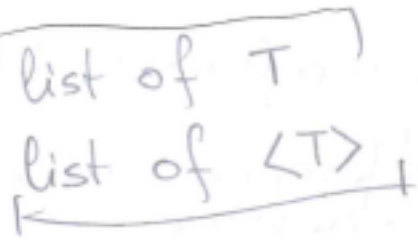
type list of T = struct {
next: list of T; //ok, ref
value: T; //ok, last field
}

⇒ resolved at the code generation time with real type attributes;

(2)

! Works only for bytecode inline subroutines! (mostly operators)

!! Problems with implementing native Starta subroutines



---

3. Current implementation assumes that all types fit one stackcell.

! A lot of problems:

- Can not compile to Java bytecode
- Can not implement 'new T[10]' for a generic T
- Implementations are inefficient and can not be easily optimised.

=> remove from the language?

3

Generic types of the „3-rd way" are ony needed for:

- push(T[]; T) → T[]

  also: pop() trim()

  ⇒ can be implemented once for all reference types T;

  ⇒ will have to include a parametrised module for non-reference Ts, but that's probably ok (i.e. not too much burden/overhead)

- length (array of T)

  ⇒ defined for reference type 'array of T' anyway...

- trim (array of T): array of T

  ⇒ defined only for the reference type 'array of T' anyway...

- same for 'clone()'

– sort(array of type T a; function cmp(T a; T b) -> bool): array of T;