Design Document

Yijian Liu (yil212)

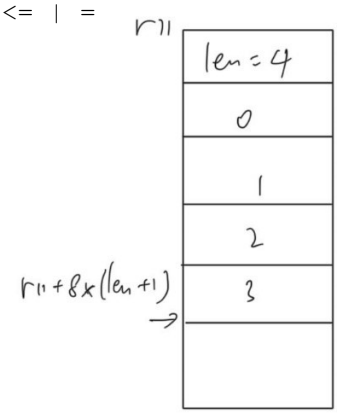
1. Grammar

Here is the grammar used by my compiler. The new part in egg eater is highlighted. The first new syntax is "`vec`" for constructing a structure for heap-allocated data. It must contain one or more `<expr>`. The other new syntax is "`vec-get`", which takes a vec and an index, and returns an element from the vec. "`vec-get`" will dynamically check the tag of the second argument and the third argument.

```
<prog> := <defn>* <expr>
<defn> := (fun (<name> <name>*) <expr>)
<expr> :=
  | <number>
  | true
  | false
  | input
  | <identifier>
  | (let (<binding>+) <expr>)
  | (<op1> <expr>)
  | (<op2> <expr> <expr>)
  | (set! <name> <expr>)
  | (if <expr> <expr> <expr>)
  | (block <expr>+)
  | (loop <expr>)
  | (break <expr>)
  | (<name> <expr>*)
  | (vec <expr>+)
  | (vec-get <expr> <expr>)
<op1> := add1 | sub1 | isnum | isbool | print | (new!)
<op2> := + | - | * | < | > | >= | <= | =
<binding> := (<identifier> <expr>)
```

2. Heap Allocation

The first word will always store the length of the array, which will be used for out-of-bound check. Each succeeding word will store the actual values. For example, a structure `(vec 0 1 2 3)`, it will be allocated on heap in the following way. The heap pointer then moves to the next available word, getting ready for the next allocation.



3. Error tests

Test 1: error-tag.snek (https://github.com/ucsd-cse231/ucsd-cse231-sp24-05-egg-eater-yjl450/blob/main/tests/error-tag.snek)
Result: `Get from non-vec object 123456`
Here the error is that the first argument, which is expected to be a vec, is a number. In `(vec-get e1 e2)`, after executing e1, we then compare the tag bits of e1 with 001. If the last three bits are of e1 are not 001, we throw an error with code 123456.

Test 2: error-bound.snek (github.com/ucsd-cse231/ucsd-cse231-sp24-05-egg-eater-yjl450/blob/main/tests/error-bounds.snek)
Input 1: `2` Result: `2`
Input 2: `-1` Result: `Index out of bounds 778899`
Input 3: `6` Result: `Index out of bounds 778899`
Here the index is determined at runtime by user input. In `(vec-get e1 e2)`, `e2` is first executed to get the index. This is then compared with 0 and an error is raised if the index is smaller than 0. After executing e1, the length is moved from `[rax]` to `rdx` and compared with the index. If the index is equal or larger than the length, an error is raised.

Test 3: error3.snek (https://github.com/ucsd-cse231/ucsd-cse231-sp24-05-egg-eater-yjl450/blob/main/tests/error3.snek)
Result: `invalid argument (type mismatch) 556677`
Here the index is a boolean value instead of a number. After executing e1, we will do a bit-wise and between the last bit of `rax` and 0x1 to get the tag for numbers. If the tag is not 0, an error is raised.

4. Success Tests

Test 1: simple_examples.snek (https://github.com/ucsd-cse231/ucsd-cse231-sp24-05-egg-eater-yjl450/blob/main/tests/simple_examples.snek)
Result:

```
(vec 0)
(vec 0 1)
(vec 0 1 2 3 true)
(vec 11 22 33)
(vec 4 5 6)
4
5
6
6
```

Test 2: points.snek (https://github.com/ucsd-cse231/ucsd-cse231-sp24-05-egg-eater-yjl450/blob/main/tests/points.snek)
Result:

```
(vec 1 2)
(vec -1 -2)
(vec 12 16)
(vec 0 0)
(vec 13 18)
(vec 13 18)
```

Test 3: bst.snek (https://github.com/ucsd-cse231/ucsd-cse231-sp24-05-egg-eater-yjl450/blob/main/tests/bst.snek)

```
(vec 8 (vec 4 (vec 2 (vec 1 false false) (vec 3 false false)) (vec 6 (vec 5 false false)
(vec 7 false false))) (vec 12 (vec 10 (vec 9 false false) (vec 11 false false)) (vec 14
(vec 13 false false) (vec 15 false false))))
true
false
false
```

This program builds a bst and tests if it contains the three specific values. Note that in our design, there is no way to retrieve the length of a vec. To distinguish between a leaf node and an internal node, we use `false` as a substitute for null pointer. So, every node will always have two children and a `false` child means there is no more node attached to that node.

5. Heap-Allocated Data in Other Languages

In C++ STL, the vector container also stores values in a contiguous chunk of memory on heap. Instead of storing the length explicitly, a vector stores a point to the beginning of the data, one to the end, and one to the end of the allocated memory. In this way, a vector can dynamically increase its capacity.
All objects in Python are heap allocated, including lists. Python lists also use contiguous memory blocks of memory. However, instead of directly storing each object, pointers to objects are stored in the list.
Thus, our design is more similar to C++ vector.

6. References

1. Piazza note @175
2. https://antonz.org/list-internals/
3. https://medium.com/@thetealpickle/c-vectors-and-memory-c925b8addfa2
4. https://stackoverflow.com/questions/12274060/does-python-use-linked-lists-for-lists-why-is-inserting-slow
5. https://stackoverflow.com/questions/8036474/when-vectors-are-allocated-do-they-use-memory-on-the-heap-or-the-stack