Page 1

Scale for project VM Abstract

Git repository

Introduction

We ask you, for the good progress of this notation:

- To remain courteous, polite, respectful, constructive, during this exchange. The link of trust between the community 42 and you depend on it.
- To clearly indicate to the person (or group) possible malfunctions.
- to accept that there may sometimes be differences of interpretation on the requests the subject or the scope of the features. Stay open minded to the vision of the other (is he right or wrong?), and write as honestly as possible.

Good defense to all!

Guidelines

REMEMBER THAT YOU NEED TO CORRECT THAT WHICH IS ON THE STUDENT'S DECLARATION TO RENDER.

This is to make a \ "git clone \" of the rendering repository, and correct what is there.

ratings

Define the type of error, which ended the correction.

Ok Empty workIncomplete work

No author file

Invalid compilation

StandardCheat Crash

vogsphere@vogsphere.42.fr: intra / 2015 / activities / abstract_vm / kpoirier

---

Page 2

Attachments

Subject

sections

preliminaries

Preliminary tests

First check the following:

- There is a rendering (in the deposit git)
- No cheating, students must be able to explain their code.

If an element does not conform to the subject, the notation stops. You are encouraged to continue to discuss the project, but the scale is not applied.

Functionality tests

Test 1

Test the following program:

```
push int32 (42)
push int32 (33)
add; pony
push float (44.55)
mul
```

double push (42.42)
comment of phew
push int32 (42)
dump
pop
assert double (42.42)
exit
Is the program running correctly?
Yes
No.

Page 3

Test 2
Test the following program:
push int32 (32)
push int32 (0)
div
exit
Does the program stop in error because of the division by 0?
Test 3
Test the following program:
push int16 (99999999999999999999999999)
exit
The program stops properly in error because of the overflow?
Test 4
Test the following program:
push int16 (32;)
pu int (32))
exit
The program stops on a syntax error?
Test 5
Test the following program:
pop
exit
The program stops properly in error because of the pop on empty stack?
Yes
No.
Yes
No.
Yes
No.
Yes
No.
Yes
No.

Page 4

Test 6
Test the following program:
push int32 (42)
assert int32 (0)
exit
The program stops properly in error on the assert?
Test 7
Test the following program:
push int32 (42)
add
exit
The program stops in error on the missing operand?
Test 8
Test the following program:
push int8 (33);
push int8 (112);
push int8 (111);
push int8 (108);
push int8 (112);
print
pop
print
pop
print
pop
print
pop
print
pop
exit
The program runs and displays "plop!" ?
Yes
No.
Yes
No.
Yes
No.

Page 5

Custom test
Test a program of your invention. For example, do cross operations
types with very large and very small numbers (not overflow / underflow).
It works as you expected?

Custom hard test
Try to test a difficult program of your invention (a vicious kind of thing).
It works as you expected?
Implementation
inputs
The VM must be able to read its input from a file, or the standard input (with ;;
delimit the end of the entry).
Battery
The VM contains a "stack". It may not be a std :: stack unless it is justified
rigorously (std :: stack is not iterable, it can serve as a base class at best).
Polymorphic operands
Operands are manipulated polymorphically through IOperand *.
Otherwise, rendering is off topic. Click on the
flag \ "crash \", the notation is no longer applied, but you are
encouraged to continue to discuss.
Operands factory
There must be an Operand factory that implements the following function:
Yes
No.
Yes
No.
Yes
No.
Yes
No.
Yes
No.
Yes
No.

Page 6

IOperand * SomeClass :: createOperand (eOperandType type, const std :: string & value);
Precision management
The VM manages the accuracy in a non trivial way - no batteries of yews or other dirt. A
enum is quite acceptable for example.
parser
The VM has a clean and complete parsing?
exceptions
The VM must use exceptions to handle certain errors.
Select the corresponding graduation:
- No exceptions: 0
- Scalar Exceptions (string, char *, int, ...): 1
- Presence of pre-made exceptions (only std :: exception or other): 2
- Presence of custom exceptions inheriting from std :: exception: 3

- Presence of custom exceptions inheriting from a more specialized class than std :: exception: 4

Rate it from 0 (failed) through 5 (excellent)

bonus

Full verification

The VM is able to give all the errors of a file, and does not stop at the first error encountered (excluding interpretation).

Advanced parsing

The parsing is well structured, namely couple lexer / parser with well-defined roles such as that they are supposed to be in reality.

Yes

No.

Yes

No.

Yes

No.

Yes

No.

Yes

No.

Page 7

Other bonuses

Count in this part if there are other bonuses. You can rate up to 5 bonuses disctincts.

Each bonus must be:

- A useful minimum (at your discretion)
- Well done and 100% functional

Rate it from 0 (failed) through 5 (excellent)

Conclusion

Leave a comment on this correction

1. verbose mode
2. interactive
3. color
4. instruction (max, min, avg, pow)
5. bitwise instructions: AND / OR / XOR
6. log (verbose strings)