2021 年度形式言語とコンパイラレポート課題

1　課題 1 Quartz 言語の実現

本講義で実装したプログラミング言語 Quartz をテストするにあったって作成したテストファイルの中身とそれらの実行結果を本章に示す.

a)　加減算，単項加減算演算子

```
1.  puts 1+1;
2.  puts 1+1+1;
3.  puts (1+3)/2;
4.  puts +2;
5.  puts -3;
6.  puts --10;
7.  puts 10--10;
```

図 1 任意個の加減算と単項加減算演算子のテスト

図 1 の実行結果を図 2 に示す.

```
[yui@HTT src]$ java Quartz ../tests/1-plusminus.txt
2
3
2
2
-3
10
20
```

図 2 図 1 の実行結果

b)　代入

```
8.  a = 2;
9.  puts a;
10. puts a + 3;
11. a = b = c = 10;
12. puts a;
13. puts b;
14. puts c;
```

図 3 代入のテスト

図 3 の実行結果を図 4 に示す.

```
[yui@HTT src]$ java Quartz ../tests/2-dainyu.txt
2
5
10
10
10
```

図４図３の実行結果

c)　論理演算子，比較演算子

```
15. puts 0||0;
16. puts 0||1;
17. puts 1||0;
18. puts 1||1;
19.
20. puts "--------";
21. puts 0&&0;
22. puts 0&&1;
23. puts 1&&0;
24. puts 1&&1;
25.
26. puts "--------";
27. puts 0==0;
28. puts 0==1;
29. puts 1==0;
30. puts 1==1;
31.
32. puts "--------";
33. puts 0<0;
34. puts 0<1;
35. puts 1<0;
36. puts 1<1;
37.
38. puts "--------";
39. puts 0>0;
40. puts 0>1;
41. puts 1>0;
42. puts 1>1;
43.
44. puts "--------";
45. puts 0<=0;
46. puts 0<=1;
47. puts 1<=0;
48. puts 1<=1;
49.
50. puts "--------";
51. puts 0>=0;
52. puts 0>=1;
53. puts 1>=0;
54. puts 1>=1;
```

図 5 論理演算子と比較演算子のテスト

図 5 の実行結果を図 6 に示す.

```
[yui@HTT src]$ java Quartz ../tests/3-hikaku.txt
0
1
1
1
--------
0
0
0
1
--------
1
0
0
1
--------
0
1
0
0
--------
0
0
1
0
--------
1
1
0
1
--------
1
0
1
1
```

図 6 図 5 の実行結果

d) 条件分岐構文

```
55. x = 1;
56. y = 0;
57. if x == 0 then
58.   y = 1;
59. else
60.   y = 2;
61. end
62. puts y;
```

図 7 条件分岐構文のテスト

図 7 の実行結果を図 8 に示す.

```
[yui@HTT src]$ java Quartz ../tests/4-if-test.txt
2
```

図 8 図 7 の実行結果

e)　繰り返し構文

```
i = 1;
sum = 0;
while i <= 10 do
sum = sum + i;
i = i + 1;
end
puts sum;
```

図 9 繰り返し構文のテスト

図 9 の実行結果を図 10 に示す.

```
[yui@HTT src]$ java Quartz ../tests/5-while-test.txt
55
```

図 10 図 9 の実行結果

f)　条件分岐構文と繰り返し構文

```
odd_num = even_num = 0;
  i = 0;
while i <= 10 do #comment
  if i % 2 == 0 then
    even_num = even_num + 1;
  else
    odd_num = odd_num + 1;
  end
  i = i + 1;
end
puts odd_num;
puts even_num;
```

図 11 条件分岐構文と繰り返し構文を合わせたプログラムのテスト

図 11 の実行結果を図 12 に示す.

```
[yui@HTT src]$ java Quartz ../tests/6-while-test2.txt
5
6
```

図 12 図 11 の実行結果

g)　文字列

```
82. a = "hello";
83. b = "kin iro";
84. c = "mozaic";
85.
86. puts a + " " + b + " " + c;
```

図 13 文字列の代入，結合，表示のテスト

図 13 の実行結果を図 14 に示す.

```
[yui@HTT src]$ java Quartz ../tests/7-string.txt
hello kin iro mozaic
```

図 14 図 13 の実行結果

h)　コメント

```
87. # puts "hello";
```

図 15 コメントのテスト

図 15 の実行結果を図 16 に示す.

```
[yui@HTT src]$ java Quartz ../tests/8-comment.txt
```

図 16 図 15 の実行結果

i)　関数

```
88. def fib(n)
89. if n < 2 then
90. n;
91. else
92. fib(n-1) + fib(n-2);
93. end
94. end
95. i = 0;
96. while i <= 10 do
97. puts "fib(" + i + ") = " + fib(i);
98. i = i + 1;
99. end
```

図 17 関数定義と関数呼び出しのテスト

図 17 の実行結果を図 18 に示す.

```
[yui@HTT src]$ java Quartz ../tests/9-fibonacchi.txt
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(3) = 2
fib(4) = 3
fib(5) = 5
fib(6) = 8
fib(7) = 13
fib(8) = 21
fib(9) = 34
```

図 18 図 17 の実行結果

2　クロージャの理解

本章で書き記すことは無い.

3 配列の実現

a) 配列の代入と表示

```
100. a = [1,2,3];
101. puts a;
102.
103. puts a[0];
104. puts a[1];
105. puts a[2];
106.
107. w = [["one", 1], ["two", 2]];
108. puts w;
109. puts w[1];
110.
111. v = [[[["deep"]]]];
112. puts v[0];
113. puts v[0][0];
114. puts v[0][0][0];
115. puts v[0][0][0][0];
```

図 19 配列の代入と表示のテスト

図 19 の実行結果を図 20 に示す.

```
[yui@HTT src]$ java Quartz ../tests/12-arraytest.txt
[ 1, 2, 3, ]
1
2
3
[ [ one, 1, ], [ two, 2, ], ]
[ two, 2, ]
[ [ [ deep, ], ], ]
[ [ deep, ], ]
[ deep, ]
deep
```

図 20 図 19 の実行結果

4 配列操作の実現

a) 配列操作

```
116. a = [1,2,3,4,5];
117. b = [7,6,4,2];
118. puts a + b;
119. puts a – b;
120. puts a * b;
```

図 21 配列操作のテスト

図 21 の実行結果を図 22 に示す.

```
[yui@HTT src]$ java Quartz ../tests/13-arraytest2.txt
[ 1, 2, 3, 4, 5, 7, 6, 4, 2, ]
[ 1, 3, 5, ]
[ 2, 4, ]
```

図 22 図 21 の実行結果

5 for 文の実現

a) for 文

```
121. for i in [1, 2, 3] do
122. for j in ["a", "b", "c"] do
123. puts i + "," + j;
124. end
125. end
126. puts i;
```

図 23 for 文のテスト

図 23 の実行結果を図 24 に示す.

```
[yui@HTT src]$ java Quartz ../tests/14-for-test.txt
1,a
1,b
1,c
2,a
2,b
2,c
3,a
3,b
3,c
```

図 24 図 23 の実行結果

6　クラスとオブジェクトの実現

a)　クラスとオブジェクト

```
class Complex
    re = 0;
    im  = 0;

    def initialize(r, i)
        re = r;
        im = i;
    end

    def add(c)
        Complex.new(re + c.get_re(), im + c.get_im());
    end

    def add2()
        Complex.new(re, im);
    end

    def to_s()
        re + " + " + im + "i";
    end

    def show()
        env
    end

    def get_re()
        re;
    end

    def get_im()
        im;
    end
end

c1 = Complex.new(10, 20);
#c1.show();
c2 = Complex.new(30, 40);
c3 = c1.add(c2);
puts c3;
```

図 25 for 文のテスト

図 25 の実行結果を図 26 に示す.

```
[yui@HTT src]$ java Quartz ../tests/15-class-test.txt
40 + 60i
```

図 26 図 25 の実行結果

7　独自機能の実現

a)　変数テーブルの表示機能

現在定義されている変数・関数・クラスの一覧を表示する命令”env”を実装した．図 27 に使用例を示す．また，図 27 の実行結果を図 28 に示す．167 行目の”env”ではそれまでに定義されている a:10 のみ表示される．一方で，170 行目では局所変数 m,n に加えグローバルに定義されていた a と print 関数も表示される．

```
166.a = 10;
167.env
168.
169.def print(n, m)
170.    env
171.    puts n;
172.end
173.print(10, 20);
```

図 27 変数テーブルの表示機能を仕様するプログラム

```
[yui@HTT src]$ java Quartz ../tests/16-fun-print-test.txt
^^^^^^^^^^^^^^^^^^^^^^^^^^^^
a : 10
vvvvvvvvvvvvvvvvvvvvvvvvvvvv
^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^OUTER^^^^^^^^^^^^^
a : 10
print : Function@3abfe836
vvvvvvvvvvvvOUTERvvvvvvvvvvvvvvv
m : 20
n : 10
vvvvvvvvvvvvvvvvvvvvvvvvvvvv
10
```

図 28 図 27 の実行結果

b)　継承

オブジェクトの継承ができるような機能を追加した．クラスの継承を行うプログラムを図 29 に示す．実行結果は図 30 である．Human クラスにて，継承元 Animal の initialize 関数と to_s 関数を上書きするとともに，新しく定義した get_work 関数が動作することが分かる．

```
class Animal
    name = "";
    age = 0;
    def initialize(n, a)
        name = n;
        age = a;
    end
    def to_s()
        name + " (age = " + age + ")";
    end
    def show()
        env
    end
    def get_name()
        name;
    end
    def get_age()
        age;
    end
end

class Human extends Animal
    work = "";
    def initialize(n, a, w)
        name = n;
        age = a;
        work = w;
    end
    def get_work()
        work;
    end
    def to_s()
        name + " (age = " + age + "), (work = " + work
+ ")";
    end
end

a1 = Animal.new("lion", 10);
puts a1;

h1 = Human.new("sato", 30, "teacher");
puts h1.get_work();

h2 = Human.new("suzuki", 30, "neet");
puts h2;
```

図 29 クラスの継承を使ったプログラム

```
[yui@HTT src]$ java Quartz ../tests/17-class-test\ .txt
lion (age = 10)
teacher
suzuki (age = 30), (work = neet)
```

図 30 図 29 の実行結果