Nguyên Lý Ngôn Ngữ Lập Trình

Bài Tập Lớn

ZCODE

nhóm thảo luận CSE
https://www.facebook.com/groups/211867931379013

Tp. Hồ Chí Minh, Tháng 1/2024

Mục lục

1 Lý Thuyết

1. **Class FuncZcode**

```
class FuncZcode(Zcode):
    def __init__(self, name, typ, param):
        self.typ = typ
        self.name = name
        self.param = param
        self.line = 0 #! hàng trong buff
```

- **self.typ** là kiểu của hàm này (return Type) có thể là **Void, Number, String, Bool, Array, None**
- **self.name** tên của hàm
- **self.param** danh sách param của hàm
- **self.line** vị trí hàng của **function** trong **self.buff của Class Emitter**

```
# ví dụ sau
func foo(number a)
begin
    var b <- 1
    return b
end
```

  - **func foo(number a)** lúc này chuyển sang java byte code có dạng **.method public static fooNone** (vì **self.typ=None**) hãy chú ý None làm sao tới bước **return b** ta có xử lí được bằng cách lưu **self.line** lại
  - **return b** tại bước này ta đã có trong **line của self.buff của Class Emitter** và cũng đã biết **self.typ=NumberType()** ta thay thế **None** thành **(F)F**

2. **Class VarZcode**

```
class VarZcode(Zcode):
    def __init__(self, name, typ, index, init = False):
        self.typ = typ
        self.name = name
        self.index = index #! vị trí biến trong bộ nhớ
        self.line = 0 #! hàng trong buff
        self.init = init
```

- **self.typ** là kiểu của hàm này (return Type) có thể là **Void, Number, String, Bool, Array, None**
- **self.name** tên của biến
- **self.index** vị trí khi khởi tạo bộ nhớ, ví dụ **.var 2 is i F from Label2 to Label3** biến i đang khởi tạo vị trí 2 trong bộ nhớ khi lấy ra **fload_2**
- **self.line** giống func
- **self.init** kiểm tra biến đã khởi tạo hay chưa

3. **class Emitter**
   - **self.buff** lưu danh sách các chuỗi sau này ghi vào file **.j**
   - **hàm printIndexNew** lấy line vừa mới tạo
   - **hàm setType** cập nhật Node thì type
4. **biến Global** chuyển thành biến static (trong java)
5. **các Function** thì chuyển thành hàm static
6. **các biến local và param** vẫn giống local trong java
7. **Hàm main** thì phải giống hàm main bên java
8. **class Access**

```
class Access():
    def __init__(self, frame, symbol, isLeft, checkTypeLHS_RHS = False):
        self.frame = frame
        self.symbol = symbol
        self.isLeft = isLeft
        self.checkTypeLHS_RHS = checkTypeLHS_RHS
```

   - **self.frame** này là frame của một hàm
   - **self.symbol** là danh sách các biến (giống prama ở BTL3)
   - **self.isLeft** dành cho phép gán kiểm tra có phải phía bên trái hay không dùng để dùng write/read
   - **self.checkTypeLHS_RHS** kiểm tra type nếu true nghĩa là code = None, type = ... (code không sài), ngược lại thì code = ..., type = None (type không sài) mà None hay không thì cũng được nói chung không sài đối với self.checkTypeLHS_RHS là true/false tùy TH
9. **các hàm Expr** thì bắt buộc phải trả về dưới dạng **tuple** với **return code, type**
10. **các hàm stmt** thì không cần return giá trị
11. **Class Frame**
    - **nhóm stack** dùng để xác định kích thước stack khi ta tính toán các expr
    - **nhóm index** xác định vị trí của biến
    - **Nhóm Label** dùng để đánh dấu nhản đến nhảy đến thông qua goto

2 Chuyển Zcode thành Java

Đoạn code Zcode

```
number g <- 2
bool x[2]

func foo(number a, string b)
begin
    var c <- 3 % 4
    string d[2,3]
    d[0,0] <- "v"
    return c
end

func main()
begin
    var a
    number c
    if (true) c <- a
    else c <- a + 1

    var i <- 0;
    for i until i >= 2 by 1
        foo(i, "")
end
```

chuyển sang Đoạn code java

```
public class ZCodeClass {
    public static int g;
    public static bool x[2];

    public static float foo(float a, String b)
    {
        auto c <- 3 - 4 * int(3/4); // suy ra c là Float
        string d[][] = new string[2][3];
        d[0][0] <- "v";
        return c;
    }


    public static void main(String[] args) {
        auto a;
        float c;
        if (true)
        {
            a = 0;
            c <- a;
        }
        else
        {
            a = 0;
            c <- a + 1;
        }

        auto i <- 0;
        for <- i;
```

```
        for (; i < 2; i +=1) ZCodeClass.foo(i, "")
        i <- for
    }

    static void main <clinit>()
    {
        g = 2;
        x = new boolean[2];
    }
}
```

3 Hàm cơ bản

3.1 visitProgram

1. **Symbol** giống như BTL phần param mà đang hiện thực list<list<»
2. **"<init>"** hàm khởi tạo trong Zcode, luôn giống nhau (hay còn gọi contructor), hàm này không có tính static

```
.method public <init>()V
Label0:
.var 0 is this LZCodeClass; from Label0 to Label1
        aload_0
        invokespecial java/lang/Object/<init>()V
        return
Label1:
.limit stack 1
.limit locals 1
.end method
```

- **self.emit.emitMETHOD** dùng để khởi ra hàng 1
- **frame.enterScope(True)** tạo scope mới lúc này Scope lớn nhất trong function nên truyền vào True
- **self.emit.emitLABEL** đặt lable tại vị trí đó, **frame.getStartLabel()** là lable bắt đầu và **frame.getEndLabel()** là lable kết thúc
- **self.emit.emitVAR** khởi tạo biến ở đây vì không có tính static nên cần khởi tạo con trỏ **this**
- **self.emit.emitREADVAR** đọc địa chỉ con trỏ **this**
- **self.emit.emitINVOKESPECIAL** gọi hàm cha contrutor vì có tính thừa kế
- **self.emit.emitRETURN** trả về giá trị
- **self.emit.emitENDMETHOD** cuối hàm dùng xác định stack và local
- **frame.exitScope()** thoát khỏi hàm

3. **"<clinit>"** hàm này dùng để khởi tạo các biến static (biến Global) nếu có gán giá trị hoặc khởi tạo array

   `number a <- 0`
   `bool b[2]`

```
.method public static <clinit>()V
Label0:
        ldc 0.0000
        putstatic ZCodeClass/a F
```

```
        ldc 2.0000
        f2i
        newarray boolean
        putstatic ZCodeClass.b [Z
        return
Label1:
.limit stack 1
.limit locals 0
.end method
```

4. **các hàm static** cần duyệt qua và đánh dấu **self.function = self.Listfunction[i]** giống BTL3, bước này không gọi hàm main
5. **khởi tạo hàm main** hàm main có thêm **args dạng mảng string** và **for dạng number** dùng để xử lí for sau này (for thì hàm static nào cũng có)

```
func main ()
begin
end
```

```
.method public static main([Ljava/lang/String;)V
Label0:
.var 0 is args Ljava/lang/String; from Label0 to Label1
.var 1 is for F from Label0 to Label1
Label2:
Label3:
        return
Label1:
.limit stack 0
.limit locals 2
.end method
```

3.2 visitVarDecl

1. cấp vùng nhớ mới **frame.getNewIndex**, **self.emit.emitVAR**
2. cập nhật trong **symbol**, cập nhật **line của var** đã khởi tạo bước trước đó
3. khởi tạo biến nếu có, khởi tạo array

```
func main ()
begin
    number a <- 0
    dynamic c
    bool b[2]
    string d[2,2]
end
```

```
.method public static main([Ljava/lang/String;)V
Label0:
.var 0 is args Ljava/lang/String; from Label0 to Label1
.var 1 is for F from Label0 to Label1
Label2:
.var 2 is a F from Label2 to Label3
        ldc 0.0000
        fstore_2
.var 3 is c None from Label2 to Label3
```

```
.var 4 is b [Z from Label2 to Label3
        ldc 2.0000
        f2i
        newarray boolean
        astore 4
.var 5 is d [[Ljava/lang/String; from Label2 to Label3
        ldc 2.0000
        f2i
        ldc 2.0000
        f2i
        multianewarray [[Ljava/lang/String; 2
        astore 5
.var 6 is a [F from Label2 to Label3
        ldc 1.0000
        f2i
        newarray float
        dup
        ldc 0.0000
        f2i
        ldc 1.0000
        fastore
        fstore_2
Label3:
        return
Label1:
.limit stack 6
.limit locals 7
.end method
```

3.3 visitFuncDecl

1. khởi tạo giống phần hàm main thôi nhưng phần param thì khác, gồm prama + for
2. **self.Return** xác định có return hay không

```
func foo1()
begin
end

func foo2()
    return 1

func foo3(number a)
    return

func foo4(number a[2,3])
    return

func main()
    return 1
```

```
.method public static foo1()V
Label0:
.var 0 is for F from Label0 to Label1
Label2:
Label3:
        return
```

```
Label1:
        return
.limit stack 0
.limit locals 1
.end method

.method public static foo2()F
Label0:
.var 0 is for F from Label0 to Label1
        ldc 1.0000
        freturn
        return
Label1:
.limit stack 1
.limit locals 1
.end method

.method public static foo3(F)V
Label0:
.var 0 is a F from Label0 to Label1
.var 1 is for F from Label0 to Label1
        return
        return
Label1:
.limit stack 0
.limit locals 2
.end method

.method public static foo4([[F)V
Label0:
.var 0 is a [[F from Label0 to Label1
.var 1 is for F from Label0 to Label1
        return
        return
Label1:
.limit stack 0
.limit locals 2
.end method

.method public static main([Ljava/lang/String;)V
Label0:
.var 0 is args Ljava/lang/String; from Label0 to Label1
.var 1 is for F from Label0 to Label1
        ldc 1.0000
        freturn
        return
Label1:
.limit stack 1
.limit locals 2
.end method
```

3.4 visitId

1. **self.emit.emitWRITEVAR** dành cho ghi giá trị biến cục bộ
2. **self.emit.emitREADVAR** dành cho đọc giá trị biến static
3. **self.emit.emitPUTSTATIC** dành cho ghi giá trị biến static

4. **self.emit.emitGETSTATIC** dành cho đọc giá trị biến static

```
number a <- 1
func main()
begin
    number b <- 1
    b <- a
    a <- b
    begin
        number c
        number b
    end
    number c
end
```

```
.method public static main([Ljava/lang/String;)V
Label0:
.var 0 is args Ljava/lang/String; from Label0 to Label1
.var 1 is for F from Label0 to Label1
Label2:
.var 2 is b F from Label2 to Label3
        ldc 1.0000
        fstore_2
        getstatic ZCodeClass/a F
        fstore_2
        fload_2
        putstatic ZCodeClass/a F
Label4:
.var 3 is c F from Label4 to Label5
.var 4 is b F from Label4 to Label5
Label5:
.var 3 is c F from Label2 to Label3
Label3:
        return
Label1:
.limit stack 1
.limit locals 5
.end method
```

3.5 visitCallExpr

- **phần io** gọi qua class io.java
- **self.emit.emitINVOKESTATIC** gọi hàm static và lấy stack các giá trị param

```
func main()
begin
    foo()
    foo1(2)
    foo2(1,2)
end
```

```
.method public static main([Ljava/lang/String;)V
Label0:
.var 0 is args Ljava/lang/String; from Label0 to Label1
.var 1 is for F from Label0 to Label1
Label2:
```

```
        invokestatic ZCodeClass/foo()F
        ldc 2.0000
        invokestatic ZCodeClass/foo1(F)F
        ldc 1.0000
        ldc 2.0000
        invokestatic ZCodeClass/foo2(FF)F
Label3:
        return
Label1:
.limit stack 4
.limit locals 2
.end method
```

3.6 visitBinaryOp và visitUnaryOp

1. **self.emit.emitADDOP** dùng cho phép cộng và trừ
2. **self.emit.emitMULOP** dùng cho phép nhân và chia
3. **self.emit.emitREOP** các phép so sánh
4. **self.emit.emitANDOP** phép and
5. **self.emit.emitOROP** phép or
6. **self.emit.emitINVOKEVIRTUAL("java/lang/String/concat", ...)** phép nối chuỗi
7. **self.emit.emitINVOKEVIRTUAL("java/lang/String/equals", ...)** phép so sánh chuỗi
8. **self.emit.emitNEGOP** phép -
9. **self.emit.emitNOT** phép not
10. **self.emit.emitI2F** chuyển int thành float
11. **self.emit.emitF2I** chuyển float thành int
12. **xem test_4**

3.7 visitArrayCell

1. **len(typ.size) == len(ast.idx)** trả về một giá trị
2. **ngược lại** trả về một địa chỉ
3. **self.emit.emitALOAD** đọc giá trị từ địa chỉ
4. **self.emit.emitASTORE** ghi giá trị vào địa chỉ

```
func main()
begin
    number a[2,2]
    a[0,0] <- a[1,1]

    var c <- a[1]
    a[0] <- c
end
```

```
.method public static main([Ljava/lang/String;)V
Label0:
.var 0 is args Ljava/lang/String; from Label0 to Label1
.var 1 is for F from Label0 to Label1
Label2:
```

```
.var 2 is a [[F from Label2 to Label3
        ldc 2.0000
        f2i
        ldc 2.0000
        f2i
        multianewarray [[F 2
        astore_2
        aload_2
        ldc 0.0000
        f2i
        aaload
        ldc 0.0000
        f2i
        aload_2
        ldc 1.0000
        f2i
        aaload
        ldc 1.0000
        f2i
        faload
        fastore
.var 3 is c [F from Label2 to Label3
        aload_2
        ldc 1.0000
        f2i
        aaload
        astore_3
        aload_2
        ldc 0.0000
        f2i
        aload_3
        aastore
Label3:
        return
Label1:
.limit stack 8
.limit locals 4
.end method

```

3.8 visitArrayLiteral

1. **self.emit.emitNEWARRAY** dùng để khởi tạo mảng một chiều giá trị
2. **self.emit.emitANEWARRAY** dùng để khởi tạo mảng một chiều địa chỉ
3. **self.emit.emitMULTIANEWARRAY** dùng để khởi tạo mảng nhiều chiều
4. **self.emit.emitDUP** dùng nhân 2 giá trị stack để giữ lại địa chỉ

```
func main()
begin
    number a[2]
    string b[2,2]
    var c <- [1]
    var c <- [[1], [1]]
end
```

```
1
2   .method public static main([Ljava/lang/String;)V
3   Label0:
4   .var 0 is args Ljava/lang/String; from Label0 to Label1
5   .var 1 is for F from Label0 to Label1
6   Label2:
7   .var 2 is a [F from Label2 to Label3
8           ldc 2.0000
9           f2i
10          newarray float
11          astore_2
12  .var 3 is b [[Ljava/lang/String; from Label2 to Label3
13          ldc 2.0000
14          f2i
15          ldc 2.0000
16          f2i
17          multianewarray [[Ljava/lang/String; 2
18          astore_3
19  .var 4 is c [F from Label2 to Label3
20          ldc 1.0000
21          f2i
22          newarray float
23          dup
24          ldc 0.0000
25          f2i
26          ldc 1.0000
27          fastore
28          astore 4
29  .var 5 is c None from Label2 to Label3
30          ldc 2.0000
31          f2i
32          anewarray [F
33          dup
34          ldc 0.0000
35          f2i
36          ldc 1.0000
37          f2i
38          newarray float
39          dup
40          ldc 0.0000
41          f2i
42          ldc 1.0000
43          fastore
44          aastore
45          dup
46          ldc 1.0000
47          f2i
48          ldc 1.0000
49          f2i
50          newarray float
51          dup
52          ldc 0.0000
53          f2i
54          ldc 1.0000
55          fastore
56          aastore
57          astore 4
58  Label3:
```

```
59          return
60  Label1:
61  .limit stack 13
62  .limit locals 6
63  .end method
```

3.9 visitReturn và visitAssign

1. **self.emit.emitRETURN** hàm trả về return chia làm 2 lội là void và expr khác void
2. **Phép gán** đối với phép gán sẽ có TH đặt biệt là gán cho array

```
func foo()
    return 1
func foo1()
    return
func main()
begin
    number a
    number b[2]
    a <- 1
    b[1] <- 1
end
```

```
1   .method public static foo()F
2   Label0:
3   .var 0 is for F from Label0 to Label1
4           ldc 1.0000
5           freturn
6           return
7   Label1:
8   .limit stack 1
9   .limit locals 1
10  .end method
11
12  .method public static foo1()V
13  Label0:
14  .var 0 is for F from Label0 to Label1
15          return
16          return
17  Label1:
18  .limit stack 0
19  .limit locals 1
20  .end method
21
22  .method public static main([Ljava/lang/String;)V
23  Label0:
24  .var 0 is args Ljava/lang/String; from Label0 to Label1
25  .var 1 is for F from Label0 to Label1
26  Label2:
27  .var 2 is a F from Label2 to Label3
28  .var 3 is b [F from Label2 to Label3
29          ldc 2.0000
30          f2i
31          newarray float
32          astore_3
33          ldc 1.0000
```

```
        fstore_2
        aload_3
        ldc 1.0000
        f2i
        ldc 1.0000
        fastore
Label3:
        return
Label1:
.limit stack 4
.limit locals 4
.end method
```

4 Suy diễn kiểu

1. **khởi tạo number** gán 0.0, các trường hợp không được khởi tạo khi dùng
2. **khởi tạo string** gán "", các trường hợp không được khởi tạo khi dùng
3. **khởi tạo bool** gán false, các trường hợp không được khởi tạo khi dùng
4. **khởi tạo array** cấp phát , các trường hợp không được khởi tạo khi dùng

code ban đầu

```
func main()
begin
    dynamic a
    dynamic b
    dynamic c
    writeNumber(a) ## a <- 0
    writeBool(b) ## b <- false
    writeString(c) ## c <- "
end
```

tự mặt định

```
func main()
begin
    dynamic a
    dynamic b
    dynamic c
    a <- 0
    writeNumber(a) ## a <- 0
    b <- false
    writeBool(b) ## b <- false
    c <- ""
    writeString(c) ## c <- ""
end
```

```
.method public static main([Ljava/lang/String;)V
Label0:
.var 0 is args Ljava/lang/String; from Label0 to Label1
.var 1 is for F from Label0 to Label1
Label2:
.var 2 is a F from Label2 to Label3
.var 3 is b Z from Label2 to Label3
```

```
.var 4 is c Ljava/lang/String; from Label2 to Label3
        ldc 0.0000
        fstore_2
        fload_2
        invokestatic io/writeNumber(F)V
        iconst_0
        istore_3
        iload_3
        invokestatic io/writeBool(Z)V
        ldc ""
        astore 4
        aload 4
        invokestatic io/writeString(Ljava/lang/String;)V
Label3:
        return
Label1:
.limit stack 1
.limit locals 5
.end method
```

5 Các Khóa Học HK232

nhóm thảo luận CSE

https://www.facebook.com/groups/211867931379013

- Lớp BTL1 + GK + LAB + Lý thuyết + Harmony của môn DSA HK232
- Lớp BTL2 + CK + LAB + Lý thuyết + Harmony của môn DSA HK232
- Lớp BTL1 + LAB + Lý thuyết + Harmony của môn KTLT HK232
- Lớp BTL2 + LAB + Lý thuyết + Harmony của môn KTLT HK232
- Lớp BTL1 + BTL2 + GK + Harmony của môn PPL HK232
- Lớp BTL3 + BTL4 + CK + Harmony của môn PPL HK232

CHÚC CÁC EM HỌC TỐT