ĐÁP ÁN

Đề thi cuối kỳ học kỳ 1 năm học 2017-2018

Môn Nguyên lý ngôn ngữ lập trình – Mã CO3005

1. Sử dụng đúng hàm sinh mã cho biểu thức (0.125), có gọi lệnh in mã biểu thức (0.125)

Sinh mã cho exp1, exp2, loop, exp3 đúng vị trí (0.25)

Sinh thêm các lệnh để tạo thành phát biểu for (0.5) trong đó

* thêm lệnh kiểm tra exp2 để jump ra ngoài (IFFALSE và label out) 0.125
* đặt đúng vị trí nhãn của continue (0.125)
* thêm lệnh goto về trước exp2 (GOTO và label ) 0.125
* sử dụng nhãn của break cho out (0.125)
* Sinh mã cho lệnh for
* **Chú ý:**
  + Tìm hiểu thêm lệnh do while, while

**visitFor**

|  |
| --- |
| *def* visitFor(*ast:For*, *o:Context*):  emit = o.stmt.method.emit  frame = o.stmt.frame  env = o.sym  exp1,exp1Type = self.visit(ast.expr1,ExprEnv(False,False,frame))  exp2, exp2Type= self.visit(ast.expr2,ExprEnv(False,False,frame))  exp3, exp3Type= self.visit(ast.expr3,ExprEnv(False,False,frame))    frame.getStartLabel()  beginLabel = frame.getNewLabel()  continueLabel = frame.getContLabel()  breakLabel = frame.getBreakLabel()  emit.printout(exp1)  emit.printout(emit.emitLABEL(beginLabel,frame))  emit.printout(exp2)  emit.printout(emit.emitIFFALSE(breakLabel,frame))  self.visit(stmt,StmtEnv(frame,env))  emit.printout(emit.emitLABEL(continueLabel,frame))  emit.printout(exp3)  emit.printout(emit.emitGOTO(beginLabel,frame))  emit.printout(emit.emitLABEL(breakLabel,frame))  frame.getEndLabel()  return |

**visitWhile**

|  |
| --- |
| *def* visitWhile(*ast:While*, *o:Context*):  emit = o.stmt.method.emit  frame = o.stmt.frame  env = o.sym  exp = self.visit(ast.expr,ExprEnv(False,False,frame))    frame.getStartLabel()  continueLabel = frame.getContLabel()  breakLabel = frame.getBreakLabel()  emit.printout(emit.emitLABEL(continueLabel,frame))  emit.printout(exp)  emit.printout(emit.emitIFFALSE(breakLabel,frame))  self.visit(stmt,StmtEnv(frame,env))  emit.printout(emit.emitGOTO(continueLabel,frame))  emit.printout(emit.emitLABEL(breakLabel,frame))  frame.getEndLabel()  return |

**visitDoWhile**

|  |
| --- |
| *def* visitDoWhile(*ast:DoWhile*, *o:Context*):  emit = o.stmt.method.emit  frame = o.stmt.frame  env = o.sym  exp = self.visit(ast.expr,ExprEnv(False,False,frame))    frame.getStartLabel()  continueLabel = frame.getContLabel()  breakLabel = frame.getBreakLabel()  emit.printout(emit.emitLABEL(continueLabel,frame))  self.visit(stmt,StmtEnv(frame,env))  emit.printout(exp)  emit.printout(emit.emitIFTRUE(continueLabel,frame))  emit.printout(emit.emitLABEL(breakLabel,frame))  frame.getEndLabel()  return |

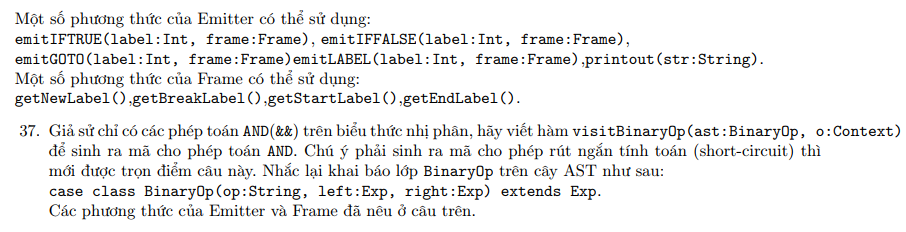
1. Sinh mã đúng cho biểu thức nhị phân với phép AND/OR thông thường 0.75, trong đó

* gọi các visit biểu thức đúng (0.25)
* gọi sinh mã cho phép toán AND (0.25)
* trả về mã của biểu thức và kiểu (0.25)

Xử lý short-circuit (0.25)

Nếu gọi emit.printout để in mã của biểu thức thì -0.25

* **Câu tương tự:**



**visitBinaryOp**

|  |
| --- |
| *def* visitBinaryOp(*ast:BinaryOp*, *o:Context*):  emit = o.stmt.method.emit  frame = o.stmt.frame  op = ast.op  left,leftType = self.visit(ast.left,ExprEnv(False,Fasle,o.stmt)  right,rightType = self.visit(ast.right,ExprEnv(False,Fasle,o.stmt)  if op is ‘&&’:  falseLabel = frame.getNewLabel()  endLabel = frame.getNewLabel()  lc, lt = self.visit(ast.left,o)  code = lc  code += emit.emitIFFALSE(falseLabel,frame)  rc, rt = self.visit(ast.right,o)  code += rc + emit.emitGOTO(endLabel,frame)  code += emit.emitLABEL(falseLabel,frame) + emit.emitPUSTICONST(0,frame) + emit.emitLABEL(endLabel,frame)  return code, BoolType()  if op is ‘||’:  trueLabel = frame.getNewLabel()  endLabel = frame.getNewLabel()  lc, lt = self.visit(ast.left,o)  code = lc  code += emit.emitIFTRUE(trueLabel,frame)  rc, rt = self.visit(ast.right,o)  code += rc + emit.emitGOTO(endLabel,frame)  code += emit.emitLABEL(trueLabel,frame) + emit.emitPUSTICONST(1,frame) + emit.emitLABEL(endLabel,frame)  return code, BoolType() |

1. Xử lý (viết được hàm và thực hiện gọi hàm đúng) đếm được phần tử trực tiếp của danh sách (0.5); đếm đệ qui được các phần tử gián tiếp (0.5)

* Viết chương trình

**Chịu ôn lại curry function**

1. (tổng 2 điểm)Viết đúng môi trường tham khảo của mỗi ctc (0.25 \* 4)

Mỗi bản ghi hoạt động vẽ đúng các phần tử cục bộ (0.125 \* 4)

Nếu các bản ghi hoạt động đều có vẽ IP, EP và SCP (0.25) nếu chỉ có IP, EP (0.2)

Viết đúng kết quả của b (0.25)

* Tầm vực động, tầm vực tĩnh

**static-scoping: Tim từ ngoài vào**

**dynamic-scoping:**

**shallow binding: tìm từ thằng gọi hàm đó**

**deep binding: tìm từ thằng khai báo và đặc tả hàm đó**

1. Nêu đúng 5 đặc điểm (0.5), đúng các cơ chế tương ứng (0.5)

**5 Đặc điểm:**

**No recursion** Không đệ quy

**Explicit Call Site** Lệnh gọi phải tường minh

**Single Entry Point** Mỗi chương trình chỉ có môt điểm vảo

**Immediate Control Passing**  Truyền tức thời khi một chương trình được gọi lập tức nó sẽ được thực thi, trình điều khiển được chuyển qua cho chương trình được gọi.

**Single Execution** Đơn thực thi tại một thời điểm chỉ có một chương trình chạy

* Các cơ chế gọi chương trình con:
* **Recursive Call**
* **Exception**
* **Coroutines**
* **Tasks**

**Truyền tham số có các kiểu:**

**Input-Output:**

**By value-result**

**By reference**

**By name**

**Input Only**

**By value**

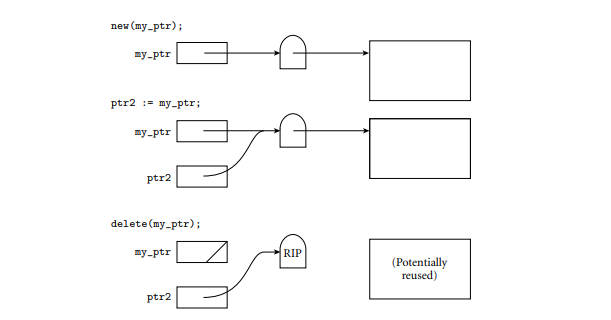
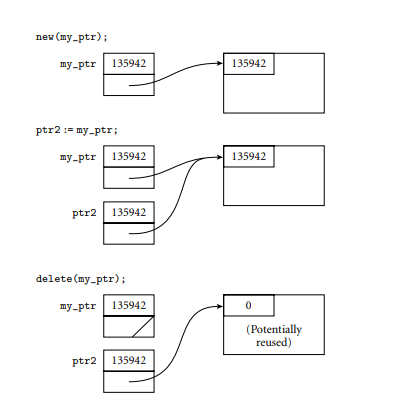
**By constant reference**

**Output Only**

**By result**

**As a result of a function**

1. Mỗi phương pháp trình bày được 0.5 trong đó

* hình vẽ 0.125
* giải thích hình vẽ 0.125
* giải thích khi huỷ bỏ đối tượng 0.125
* cách phát hiện tham chiếu treo 0.125
* Giải pháp giải quyết tham chiếu treo.
* TombStone
* 
* Lock-and-Key
* 

1. Viết đúng các operand 0.2, đúng các phép toán 0.5, đúng các () 0.3

* Prefix, Infix,

**Tiền tố bình thường:** Viết theo dạng *toán tử \*(toán hạng, toán hạng)*

|  |  |
| --- | --- |
| (a + b) \* (c - d) | \*(+(a,b),-(c,d)) |

**Tiền tố Polish:** Đẩy hết dấu ra, toán hạng dính liền lại, bao nhiêu phép toán thì biễu diễn bấy nhiêu toán tử liền nhau

|  |  |
| --- | --- |
| a - b - c \* e \* f - g | ---ab\*\*cefg |

**Tiền tố Cambridge Polish:** Toán tử giống nhau được lượt bỏ, các phép toán ghi trong bộ () để biết cần có bao nhiêu toán hạng được thực hiên tương ứng với 1 toán tử.

|  |  |
| --- | --- |
| a \* b \* ( c - e - f) \* g | (\*ab(-cef)g) |

1. Giải thích đúng về đặc điểm union (0.5)

Cho được 1 ví dụ về union (0.25)

Giải thích vì sao phải là union (0.25)

Kiểu union là một kiểu mà những biến trong kiểu này là một các cấp những giá trị kiểu khác nhau ở những thời điểm khác nhau trong suốt quá trình thực thi. Do đó vùng nhớ cấp phát cho một kiểu union được dùng chung cho các thành phần của nó.

union {

Image img;

Sound smd;

} data;

Là union để tiết kiệm dữ liệu khi khái báo kiểu vì mỗi dữ liệu hoặc là ảnh hoặc là âm do nó ta ko cần phải cấp phát thêm tránh dư thừa.

Kiểu record là kiểu cho phép kết hợp nhiều kiểu thành phần khác nhau.

Để truy suất các thành phần của nó thì dựa vào tên của các thành phần.

1. Viết đúng và đủ các ràng buộc kiểu 0.5, giải và tính đúng kết quả 0.5