PHÂN TÍCH NGỮ NGHĨA TĨNH CHO SIMPLECODE

Qui định

- Hạn nộp: xem trên trang web môn học
- Dạng bài tập: nhóm.
- Ngôn ngữ: Java
- Cấu trúc bài nộp:

<MSSV>

|__ libs // Thư viện hỗ trợ

|__ Bao_cao //chứa tập tin báo cáo (.pdf /.doc / .docx)

|__ Source_code // code của chương trình

Nén theo định dạng .rar hoặc .zip và nộp vào link tương ứng.

Yêu cầu

Viết chương trình để bắt các lỗi ngữ nghĩa tĩnh của một chương trình được viết bằng ngôn ngữ Simple Code. Danh sách các ngữ nghĩa tĩnh được mô tả trong tập tin đặc tả ngôn ngữ Simple Code.

Chương trình chạy với tham số dòng lệnh như sau:

java -jar dist/<tên_chuong_trinh>.jar -target inter <tập tin >

Trong đó,

- ***<tập tin>***: đường dẫn tập tin chứa đoạn code được viết bằng ngôn ngữ SimpleCode.
- Chương trình xuất kết quả phân tích ra tập tin. Tập tin này có tên trùng với tên của tập tin đầu vào, nhưng phần mở rộng là .out. Tập tin chứa tất cả các lỗi của đoạn code, bao gồm lỗi cú pháp (của bài scanner& parser) và lỗi ngữ nghĩa tĩnh.

Định dạng của tập tin đầu vào

Tập tin văn bản bình thường có thể chứa đoạn mã viết bằng ngôn ngữ SimpleCode hoặc không. Sinh viên xem thêm trong tập dữ liệu cung cấp.

Định dạng của tập tin đầu ra

- Trong trường hợp, đoạn chương trình trong tập tin đầu vào là hợp lệ thì tập tin đầu ra là rỗng (không ghi gì)
- Trong trường hợp, đoạn chương trình trong tập tin đầu vào không hợp lệ thì tập tin đầu ra liệt kê các lỗi đã xảy ra. Mỗi lỗi nằm trên một dòng. Sinh viên có thể xuất thông điệp lỗi theo định dạng do cá nhân tự định nghĩa.

Các bước thực hiện

Xây dựng cây biểu diễn trung gian (IR)

- Định nghĩa các lớp đối tượng biểu diễn các node trên cây IR.
- Sử dụng lớp Visitor được phát sinh bởi ANTLR để tạo cây IR.
    - Điều chỉnh Run >> External Tools >> External Tools Configurations của parser thành “-visitor” thay vì “-no-visitor”
- Cấu trúc cây IR được mô tả ở phần sau.

Xây dựng Symbol Table

Là bảng băm, cho phép ánh xạ các định danh vào các đối tượng ngữ nghĩa (chẳng hạn khai báo biến.

Kiểm tra lỗi ngữ nghĩa tĩnh

Duyệt trên cây IR và sử dụng Symbol Table để thực hiện kiểm tra ngữ nghĩa.

Cấu trúc cây IR

Khai báo các lớp cho mỗi node trên cây IR. Trong nhiều trường hợp, phân cấp các lớp này giống với văn phạm ngôn ngữ. Dưới đây là một đề nghị sự phân cấp các lớp biểu diễn các node trong IR cho SimpleCode:

```
abstract class Ir
abstract class     IrExpression
abstract class         IrLiteral
           class             IrIntLiteral
           class             IrBooleanLiteral
           class         IrCallExpr
           class             IrMethodCallExpr
           class             IrCalloutExpr
           class         IrBinopExpr
abstract class     IrStatement
           class         IrAssignStmt
           class         IrPlusAssignStmt
           class         IrBreakStmt
           class         IrContinueStmt
           class         IrIfStmt
           class         IrForStmt
           class         IrReturnStmt
           class         IrInvokeStmt
           class         IrBlock
           class     IrClassDecl
abstract class     IrMemberDecl
           class         IrMethodDecl
           class         IrFieldDecl
           class     IrVarDecl
           class     IrType
           class     IrId
```

Lưu ý: Sự phân cấp lớp ở trên chỉ là một đề nghị. Sinh viên có thể thêm/bớt lớp để phục vụ cho bài làm của mình.

Dưới đây là một ví dụ định nghĩa các lớp nêu trên:

```
public class IrType extends IrNode {

	private int _size = 0; //size in byte(s)
	private String _name = ""; // name of type

	public String GetName(){
		return _name;
	}

	public int GetSize(){
		return _size;
	}
	public IrType(String name, int size){
		_size = size;
		_name = name;
	}

	public static final IrType INT = new IrType("int", 4);
	public static final IrType BOOLEAN = new IrType("boolean", 1);
}
```

Cấu trúc Symbol Table

Dùng để lưu trữ các định danh đã được khai báo, cho phép tham chiếu về sau

```
public class SymbolTable{
	private Hashtable<String, SymbolId> table;
	protected SymbolTable outer; //  symbol table cha (block lệnh lồng ngoài).

	public SymbolTable(SymbolTable parent){
		table = new Hashtable();
		outer = parent;
	}
	public void Push(String token, IrType type, int offset)
	{
		// Thêm một token (định danh) và
		//các thông tin liên quan vào symbol table
		SymbolId id = new SymbolId(token, type, offset);
		table.put(token, id);
	}

	public IrSymbolId Lookup(String token)
	{
		// Tìm thông tin liên quan đến token
```

```
                // trong symbol table (bao gồm cả trong outer)
                for (SymbolTable t = this; t != null; t = t.outer)
                {
                        SymbolId id = t.table.get(token);
                        if (id != null) return id;
                }
                return null;
        }
}
```

Tài liệu tham khảo:

1. Part II, section 7 - Implement Applications with Visitors, sách The Definitive ANTLR 4 Reference.
2. http://elemarjr.com/en/2016/04/21/learning-antlr4-part-1-quick-overview/