工作内容

动态错误检查

RuntimeError中添加对应的String，在TacEmitter.visitClassCast和visitBinary添加判断。

抽象类

TacGen中添加判断

局部类型推导

无需推导

扩展Call

按照指导内容，修改TacEmitter.visitVarSel逻辑。FuncVisitor中，仿照visitMemberCall实现visitCall.

将方法名直接当做函数使用

在ProgramWriter中新建静态方法的虚表，实现visitVTablesSucceed方法处理建立虚表（visitVTables）后的处理（在TacGen中调用）。

针对静态方法再写一个FuncVisitor.visitFuncEntry，并在TacEmitter.visitVarSel中调用静态方法即可。

修改FuncLabel中对于main相关的判定，使得其与其他静态方法的调用相同。

Lambda表达式

记录捕获变量

在LambdaScope中增加Map以及一些方法来添加、查找和存储lambda作用域中所捕获的变量（方法和变量名以cap开头）

建立lambda的虚表

在GlobalScope中添加List以存储所有的lambda，并在Namer中记录、然后再ProgramWriter中新增加变量来存储lambda的虚表。针对lambda方法写一个FuncVisitor.visitFuncEntry，并在新增加的函数TacEmitter.visitLambda调用它们，从而建立虚表。

lambda表达式的函数实现

实现FuncVisitor.visitLambdaFunc以生存lambda表达式的函数体。

TacEmitter.visitLambda函数中，通过无条件跳转指令获得一段不会被访问的代码空间，在其中调用FuncVisitor.visitLambdaFunc以建立函数体。

判断当前lambda表达式的层级

需要判断当前在哪个lambda表达式内部，于是仿照TacEmitter.loopStack实现TacEmitter.lambdaStack。在TacEmitter.visitVarSel和visitThis中添加相关逻辑，使得可以在lambda表达式内部时查找待捕获的变量。

lambda语法实现的流程

见"工作内容.lambda表达式"一节。

工程中遇到的困难

无法识别main函数

在修改完毕静态方法后，发现main函数无法识别，这是因为FuncLabel中对于main方法进行了特殊处理，其标签为"main"而非"_L_Main_main"。Simulator查找虚表中静态方法时，在_label_to_addr中进行查找，其中存储的标签为"main"，但搜索时使用"_L_Main_main"进行搜索，因此查找不到。删去特殊处理后解决了该问题。

该问题的出现位置让人意想不到，因此排查此问题花费了一定的时间。

lambda表达式的函数调用的问题

出现了一系列问题，例如，捕获变量个数多于外层参数个数时，在FuncVisitor.getArgTemp中，对argsTemps[index]的访问会造成数组越界。这些问题是因为，最初实现lambda表达式函数体时，仅生成了FuncLabel，没有生成函数体，因此需要实现FuncVisitor.visitLambdaFunc。

该问题的出现会产生异常，沿着异常的来源可以一步步解决问题。