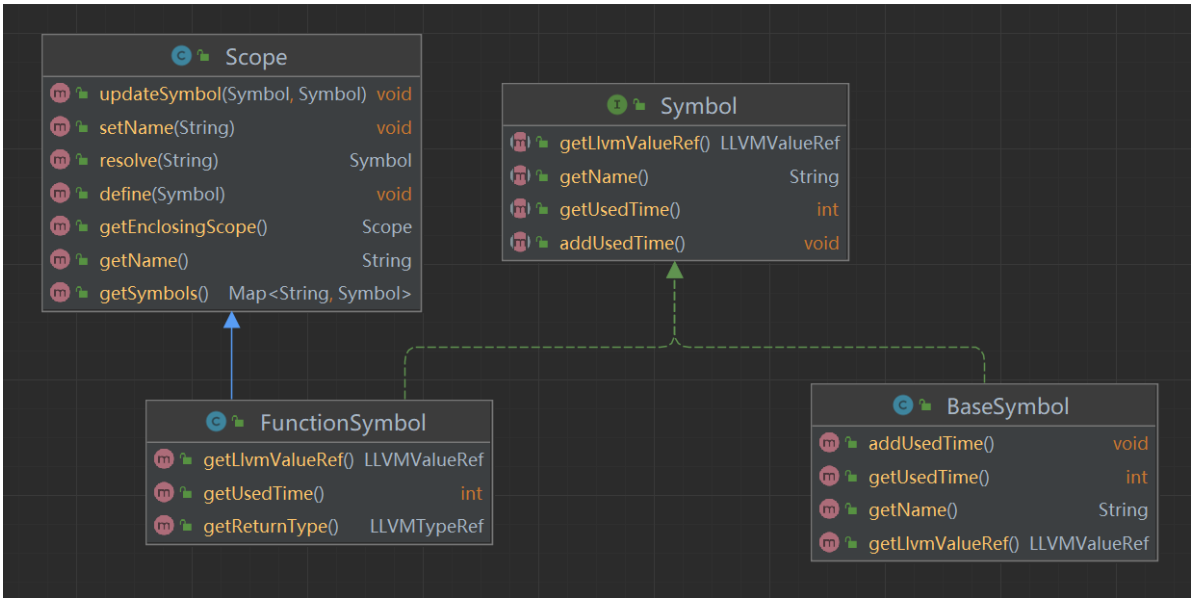
1. 程序实现功能

将函数定义与调用、局部变量的声明、定义、使用翻译为LLVM IR。

2. 如何实现

重新设计符号表，Symbol类中存储LLVMValueRef，相关类及关系如下图所示：



2.1 函数定义与调用

在lab4代码基础上，首先修改了argumentTypes。其次如果有形参的话，我将函数参数作为一个特殊的局部变量翻译，利用LLVMBuildAlloca申请内存，并利用LLVMGetParam获得参数值赋给它，然后将这个局部变量加入符号表。

调用时先从符号表中找到该函数，如果需要参数的话先构建实参列表arguments。然后调用LLVMBuildCall。这里要注意如果函数返回值是void的话，LLVMBuildCall的第四个参数必须为""，否则会报instructions returning void cannot have a name。

2.2 局部变量声明和定义

以变量为例，常量类似。

2.2.1 int型变量

如果用整形字面值常量初始化：

```
LLVMValueRef value = LLVMConstInt(i32Type, convert(s), /* signExtend */ 0);
LLVMBuildStore(builder, value, pointer);
BaseSymbol symbol = new BaseSymbol(name, pointer);
currentScope.define(symbol);
```

用其他变量或常量初始化，则通过visit子节点获得相应的LLVMValueRef：

```
LLVMValueRef value = visit(((SysYParser.InitVal1Context) ctx.initVal()).exp());
LLVMBuildStore(builder, value, pointer);
BaseSymbol symbol = new BaseSymbol(name, pointer);
currentScope.define(symbol);
```

2.2.2 数组初始化

```
int leftSize = convert(((SysYParser.NumContext)
ctx.constExp(0).exp()).number().INTEGR_CONST().getText());
LLVMTypeRef vectorType = LLVMVectorType(i32Type, leftSize);
LLVMValueRef vectorPointer = LLVMBuildAlloca(builder, vectorType, name);
SysYParser.InitVal2Context context = (SysYParser.InitVal2Context) ctx.initVal();
int rightSize = context.initVal().size();
for(int i=0; i<rightSize; i++){
    SysYParser.InitVal1Context init = (SysYParser.InitVal1Context)
context.initVal(i);
    LLVMValueRef value;
    if(init.exp() instanceof SysYParser.NumContext){
        value = LLVMConstInt(i32Type, convert(((SysYParser.NumContext)
init.exp()).number().INTEGR_CONST().getText()), 0);
    }else{
        value = visit(init.exp());
    }
    PointerPointer valuePointer = new PointerPointer(new LLVMValueRef[]{zero,
LLVMConstInt(i32Type, i, 0)});
    LLVMValueRef pointer = LLVMBuildGEP(builder, vectorPointer, valuePointer, 2,
"pointer");
    LLVMBuildStore(builder, value, pointer);
}
```

最后如果数组长度与初始值长度不相同，缺的初始值再补0。

3. 印象深刻的bug

3.1

void类型的函数如果没有return语句，需要手动加上。

3.2

最后一直剩hardtest1没有过，我一直以为是赋值语句的问题，因为把赋值语句注释了这个用例就是对的，找了很久也没有找出来。最后请同学帮忙看了一下，原来是上次实验的取模写错了，但是上次的用例都过了。上次取模的时候我是利用除数减去被除数乘商算出余数的，但是在有负数存在的情况下，取模运算与求余运算是不同的。最后直接调用LLVMBuildSRem就可以完成取模运算了。