1. 程序实现功能

翻译while循环，以及break和continue语句。

2. 如何实现

1. 在MyVisitor类中增添两个Stack类型的成员变量，whileBlocks存储while循环的出口block，whileConditionBlocks存储while循环的循环条件block。

```
private Stack<LLVMBasicBlockRef> whileBlocks = new Stack<>();
private Stack<LLVMBasicBlockRef> whileConditionBlocks = new Stack<>();
```

2. 重写visit while循环的方法：

```
public LLVMValueRef visitStmt5(SysYParser.Stmt5Context ctx) {
    // 新建whileCondition的block
    LLVMBasicBlockRef whileConditionBlock = LLVMAppendBasicBlock(this.function,
"whileCondition");
    // 压栈
    whileConditionBlocks.push(whileConditionBlock);
    // 跳转到whileCondition block
    LLVMBuildBr(builder, whileConditionBlock);
    // 后续语句写在whileCondition block后
    LLVMPositionBuilderAtEnd(builder, whileConditionBlock);
    // visit循环条件
    LLVMValueRef tmp_ = visit(ctx.cond());
    // 判断循环条件结果
    tmp_ = LLVMBuildICmp(builder, LLVMIntNE, LLVMConstInt(i32Type, 0, 0), tmp_,
"tmp_");
    LLVMBasicBlockRef whileBodyBlock = LLVMAppendBasicBlock(this.function,
"whileBody");
    LLVMBasicBlockRef entryBlock = LLVMAppendBasicBlock(this.function, "entry");
    whileBlocks.push(entryBlock);
    // 条件为真，跳到循环体内；否则直接跳出循环
    LLVMBuildCondBr(builder, tmp_, whileBodyBlock, entryBlock);
    // 后续语句写在whileBody block后
    LLVMPositionBuilderAtEnd(builder, whileBodyBlock);
    // visit循环体
    visit(ctx.stmt());
    // visit循环体结束后，需要判断是否满足循环终止条件，所以跳转到whileCondition block
    LLVMBuildBr(builder, whileConditionBlock);
    // 后续语句不在循环内部，应该写到entryBlock之后
    LLVMPositionBuilderAtEnd(builder, entryBlock);
    // 一个循环结束，出栈
    whileBlocks.pop();
    whileConditionBlocks.pop();
    return null;
}
```

3. break语句的实现：利用whileBlocks.peek()得到当前while循环的出口块，然后直接跳转到这个block。
4. continue语句的实现：利用whileConditionBlocks.peek()得到当前while循环的条件判断块，然后直接跳转到这个block。

3. 印象深刻的bug

3.1 normaltest11、normaltest3和hardtest1

报错：JIT session error: Symbols not found: [k] lli-13: Failed to materialize symbols: { (main, { n, main }) }

解决：对于没有初始化的全局变量，将它初始化为0就可以了。

3.2 hardtest0和hardtest3

报错：lli-13: lli: out.ir:48:18: error: expected icmp predicate (e.g. 'eq') %tmp_12 = icmp unknown i32 %tmp_6, %tmp_11 ^

解决：报错的意思是比较语句缺少了比较符号。我一开始很奇怪，明明所有的LLVMBuildICmp参数都是齐全的。然后我想了一下上次实验写visitCond()中的cond AND cond和cond OR cond时没有好好思考，就跟大于小于一样直接tmp_ = LLVMBuildICmp(builder, LLVMAnd, left, right, "tmp_")以及tmp_ = LLVMBuildICmp(builder, LLVMOr, left, right, "tmp_")，试了一下果然是这两个有问题。那么应该如何表示条件之间与和或的真假？我想了一下，LLVMBuildICmp语句的结果要么是0要么是1。两个条件都为真，那么&&的结果才是真的，也就是说两个LLVMBuildICmp之和必须为2。只要有一个条件为真，那么||的结果就是真的，也就是说两个LLVMBuildICmp之和必定不为0。转化成代码如下：

```
// cond AND cond以及cond OR cond的情况
LLVMValueRef left = visit(ctx.cond(0));
LLVMValueRef right = visit(ctx.cond(1));
if(ctx.AND()!=null){
    LLVMValueRef valueRef = LLVMBuildAdd(builder, left, right, "add");
    tmp_ = LLVMBuildICmp(builder, LLVMIntEQ, valueRef, two, "tmp_");
}else if(ctx.OR()!=null){
    LLVMValueRef valueRef = LLVMBuildAdd(builder, left, right, "add");
    tmp_ = LLVMBuildICmp(builder, LLVMIntNE, valueRef, zero, "tmp_");
}
tmp_ = LLVMBuildZExt(builder, tmp_, i32Type, "tmp_");
return tmp_;
```