LLVM 中如何将指令规范化为统一的字符串，

即：free(i8* %10)这样的函数调用统一为 free(i8*)的形式，return i32 %4 统一为 return i32, %5 = call i8* @malloc(i32 %4)统一为 i8* = malloc(i32)？

在我们的项目中，只关注 call 和 ret 两种指令，如下代码给出了给定这两种指令中的一种，规范化为字符串（string）的实现，结果储存在 normalizedStr 中。

变量 inst 是“Instruction *”类型。

raw_string_ostream 是 LLVM 中的一个类，需要引入相应的头文件；C++标准库中好像也有类似的字符串输出类，将“<<”的输出导入到字符串中。

isa 用于判断某个指针是否是给定类型，cast 用于将参数转换为给定类型指针的类型，均需要引入相应的头文件。

```
string normalizedStr;
raw_string_ostream rso(normalizedStr);
if (isa<ReturnInst>(inst))
{
    rso << "return ";
    Type *rType = inst->getType();
    rType->print(rso);
}
else
{
    CallInst *cinst = cast<CallInst>(inst);
    Function *cfunc = cinst->getCalledFunction();
    FunctionType *ftype = cinst->getFunctionType();
    Type *rtype = ftype->getReturnType();
    if (!rtype->isVoidTy())
    {
        ftype->getReturnType()->print(rso);
        rso << " = ";
    }
    if (cfunc->hasName())
    {
        rso << cfunc->getName();
    }
    rso << "(";
    for (auto iter = ftype->param_begin(); iter != ftype->param_end(); iter++)
    {
        if (iter != ftype->param_begin())
        {
            rso << ", ";
        }
        Type *ptype = *iter;
        ptype->print(rso);
    }
    if (ftype->isVarArg())
    {
        if (ftype->getNumParams())
            rso << ", ";
        rso << "...";
    }
    rso << ")";
}
rso.flush();
```