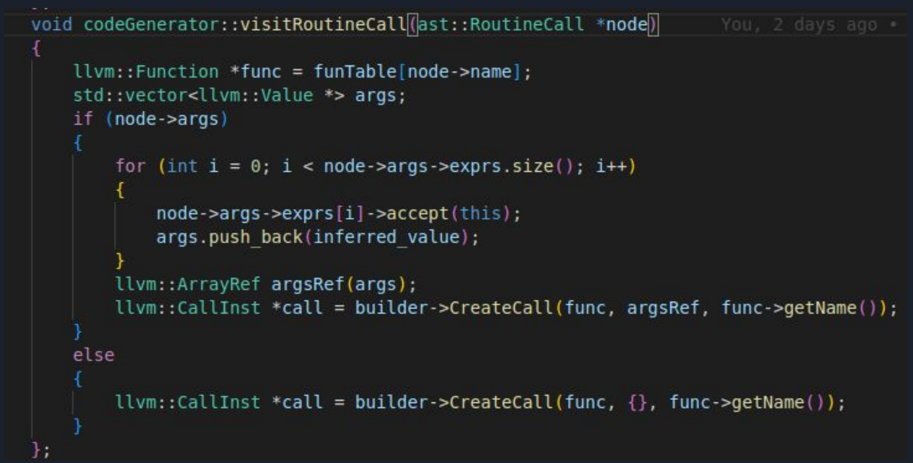
Carbonic-C Code-Generation

Asem, Jaffar, Menna, Mosab

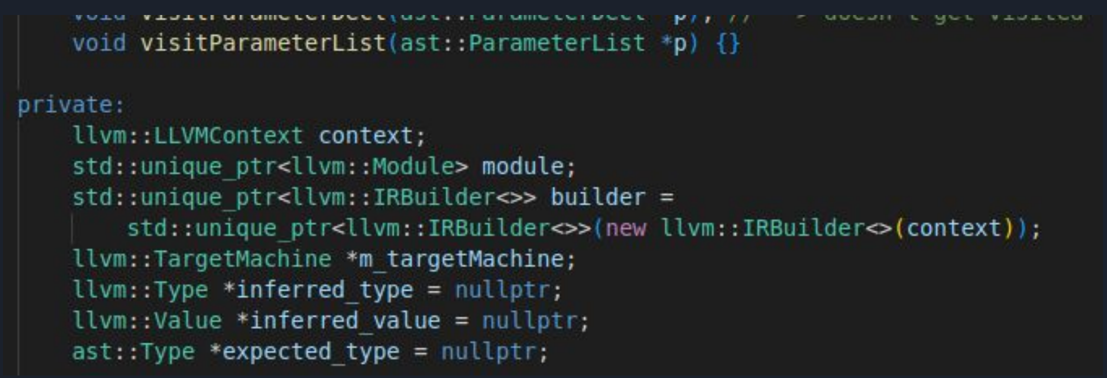
Language Features (So Far...)

- Routines
- Routine Call
- Return
- Variables (Integer, Real, Boolean, Array(partial), Record(partial))
- Global Variables
- Assignment
- Binary Expressions (*, ÷, +, -, %)
- Comparison Expressions (>, ≥, <, ≤, =, /=)
- Logical Expressions (AND, OR, XOR)
- If Conditions
- While Loops
- For Loops
- Print

Implementation

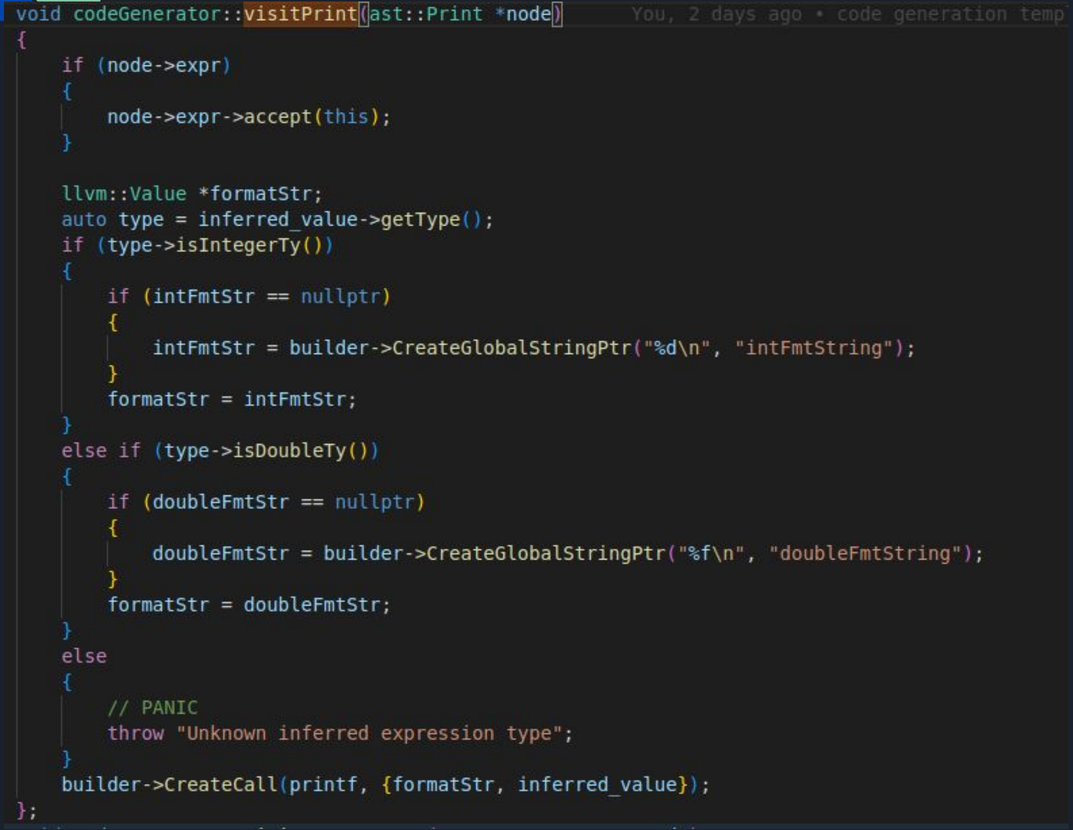


```
void codeGenerator::visitRoutineCall(ast::RoutineCall *node)        You, 2 days ago •
{
    llvm::Function *func = funTable[node->name];
    std::vector<llvm::Value *> args;
    if (node->args)
    {
        for (int i = 0; i < node->args->exprs.size(); i++)
        {
            node->args->exprs[i]->accept(this);
            args.push_back(inferred_value);
        }
        llvm::ArrayRef argsRef(args);
        llvm::CallInst *call = builder->CreateCall(func, argsRef, func->getName());
    }
    else
    {
        llvm::CallInst *call = builder->CreateCall(func, {}, func->getName());
    }
};
```

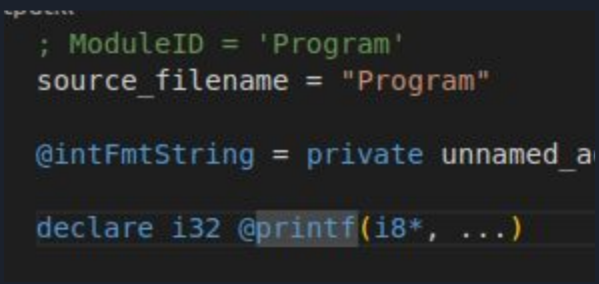


```
    void visitParameterList(ast::ParameterList *p) {}

private:
    llvm::LLVMContext context;
    std::unique_ptr<llvm::Module> module;
    std::unique_ptr<llvm::IRBuilder<>> builder =
        std::unique_ptr<llvm::IRBuilder<>>(new llvm::IRBuilder<>(context));
    llvm::TargetMachine *m_targetMachine;
    llvm::Type *inferred_type = nullptr;
    llvm::Value *inferred_value = nullptr;
    ast::Type *expected_type = nullptr;
```

Implementation - Print function

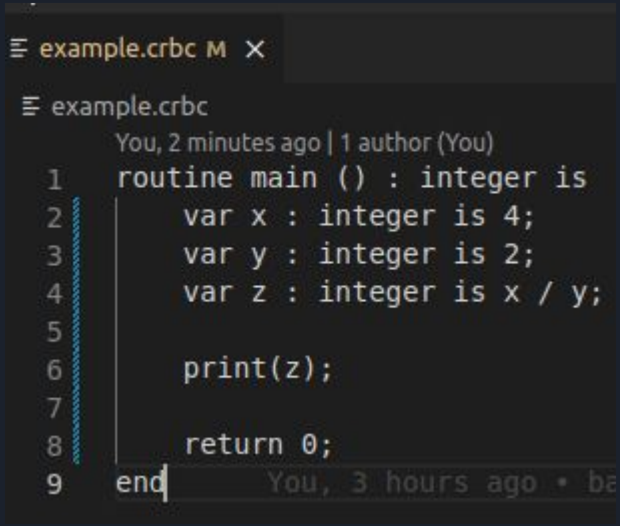


```
void codeGenerator::visitPrint(ast::Print *node)        You, 2 days ago • code generation temp
{
    if (node->expr)
    {
        node->expr->accept(this);
    }

    llvm::Value *formatStr;
    auto type = inferred_value->getType();
    if (type->isIntegerTy())
    {
        if (intFmtStr == nullptr)
        {
            intFmtStr = builder->CreateGlobalStringPtr("%d\n", "intFmtString");
        }
        formatStr = intFmtStr;
    }
    else if (type->isDoubleTy())
    {
        if (doubleFmtStr == nullptr)
        {
            doubleFmtStr = builder->CreateGlobalStringPtr("%f\n", "doubleFmtString");
        }
        formatStr = doubleFmtStr;
    }
    else
    {
        // PANIC
        throw "Unknown inferred expression type";
    }
    builder->CreateCall(printf, {formatStr, inferred_value});
};
```

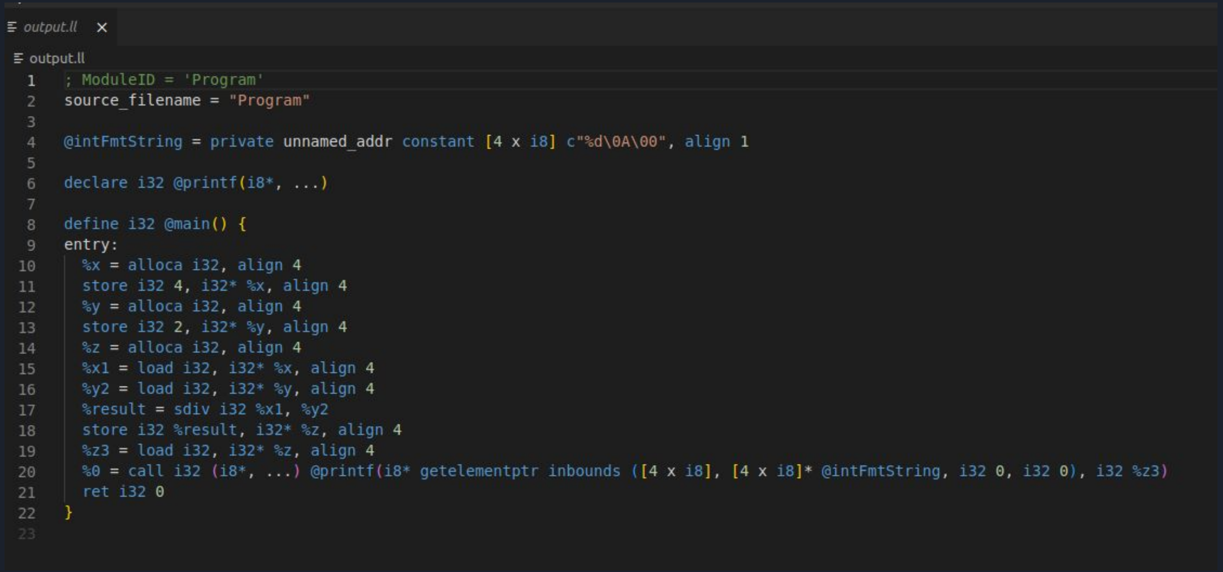


```
; ModuleID = 'Program'
source_filename = "Program"

@intFmtString = private unnamed_a

declare i32 @printf(i8*, ...)
```

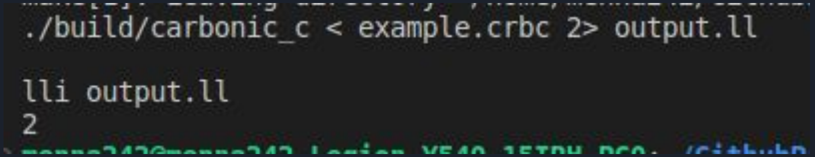
Code Generation (LLVM) #1



```
routine main () : integer is
    var x : integer is 4;
    var y : integer is 2;
    var z : integer is x / y;

    print(z);

    return 0;
end
```

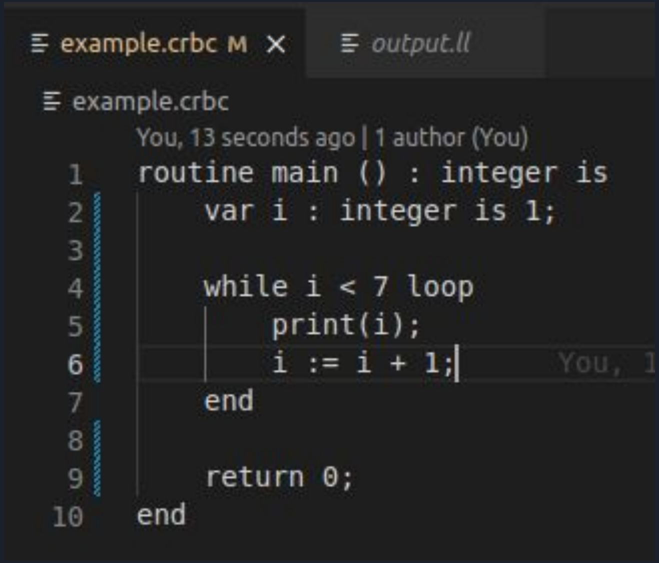


```
; ModuleID = 'Program'
source_filename = "Program"

@intFmtString = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1

declare i32 @printf(i8*, ...)

define i32 @main() {
entry:
  %x = alloca i32, align 4
  store i32 4, i32* %x, align 4
  %y = alloca i32, align 4
  store i32 2, i32* %y, align 4
  %z = alloca i32, align 4
  %x1 = load i32, i32* %x, align 4
  %y2 = load i32, i32* %y, align 4
  %result = sdiv i32 %x1, %y2
  store i32 %result, i32* %z, align 4
  %z3 = load i32, i32* %z, align 4
  %0 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @intFmtString, i32 0, i32 0), i32 %z3)
  ret i32 0
}
```

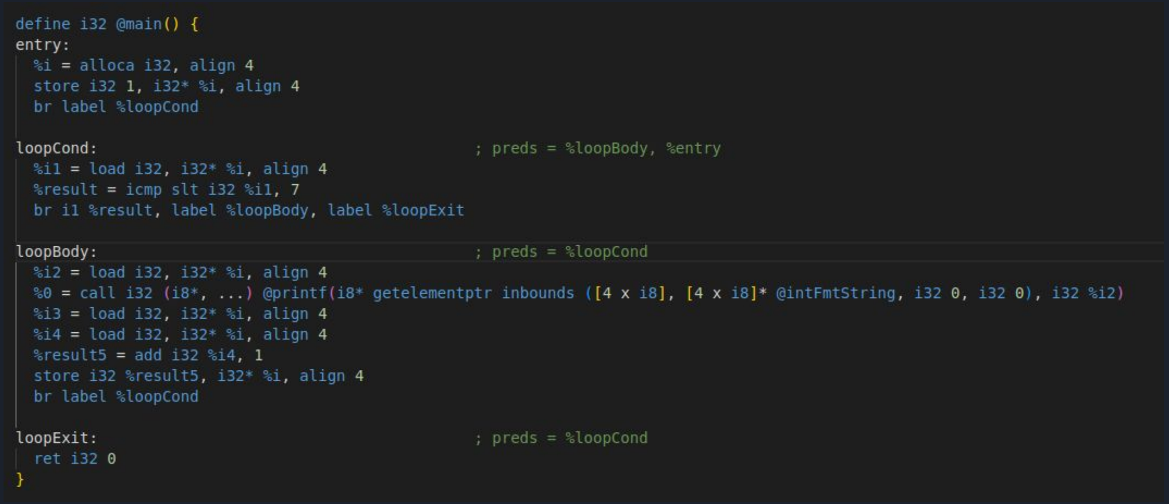


```
./build/carbonic_c < example.crbc 2> output.ll

lli output.ll
2
```

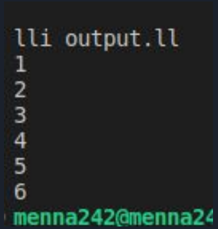
Code Generation (LLVM) #2



```
routine main () : integer is
    var i : integer is 1;

    while i < 7 loop
        print(i);
        i := i + 1;
    end

    return 0;
end
```

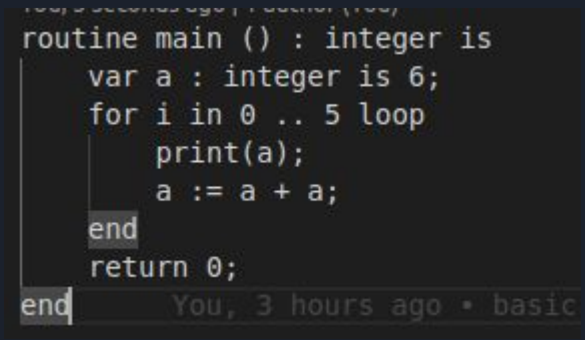


```
define i32 @main() {
entry:
  %i = alloca i32, align 4
  store i32 1, i32* %i, align 4
  br label %loopCond

loopCond:                                         ; preds = %loopBody, %entry
  %i1 = load i32, i32* %i, align 4
  %result = icmp slt i32 %i1, 7
  br i1 %result, label %loopBody, label %loopExit

loopBody:                                         ; preds = %loopCond
  %i2 = load i32, i32* %i, align 4
  %0 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @intFmtString, i32 0, i32 0), i32 %i2)
  %i3 = load i32, i32* %i, align 4
  %i4 = load i32, i32* %i, align 4
  %result5 = add i32 %i4, 1
  store i32 %result5, i32* %i, align 4
  br label %loopCond

loopExit:                                         ; preds = %loopCond
  ret i32 0
}
```

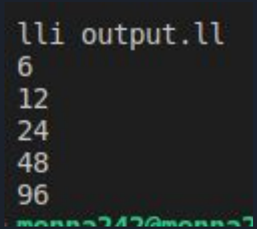


```
lli output.ll
1
2
3
4
5
6
menna242@menna24
```

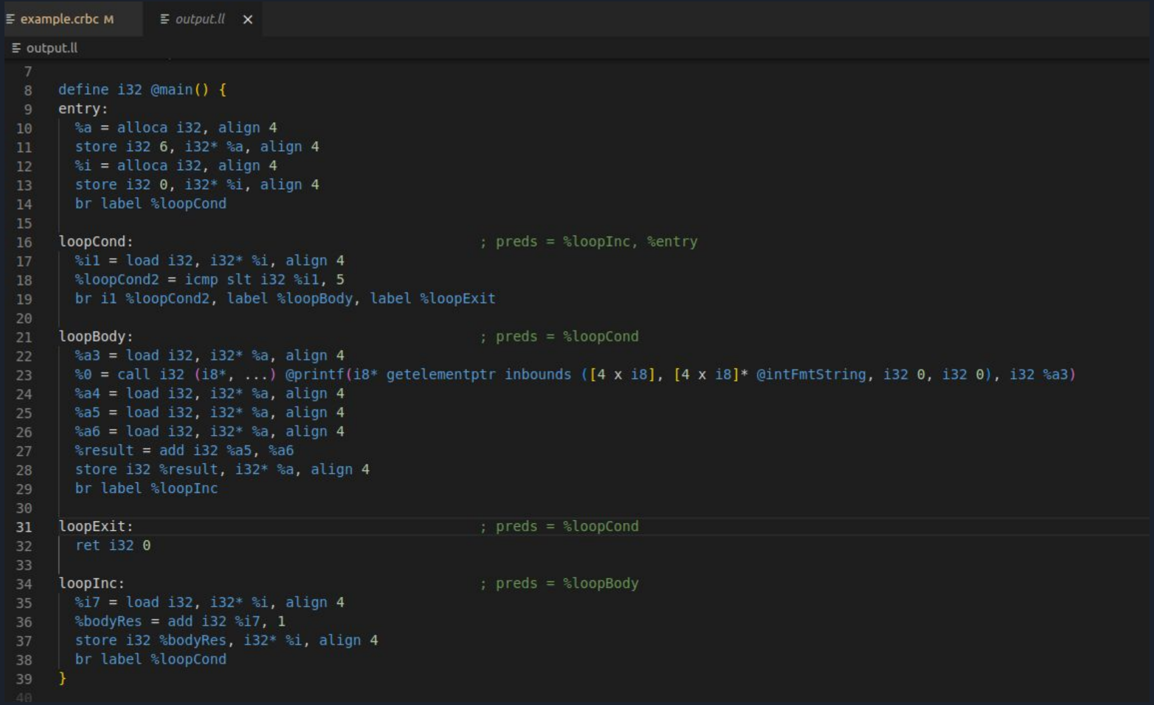
Code Generation (LLVM) #3



```
routine main () : integer is
    var a : integer is 6;
    for i in 0 .. 5 loop
        print(a);
        a := a + a;
    end
    return 0;
end       You, 3 hours ago • basic
```

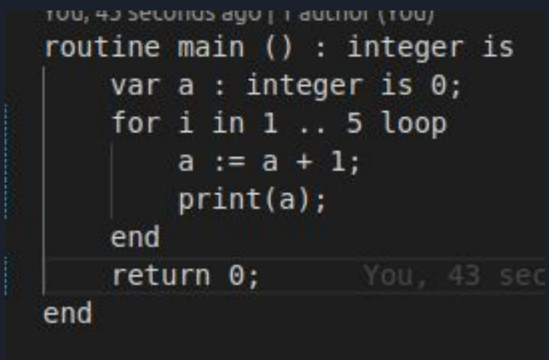


```
lli output.ll
6
12
24
48
96
```

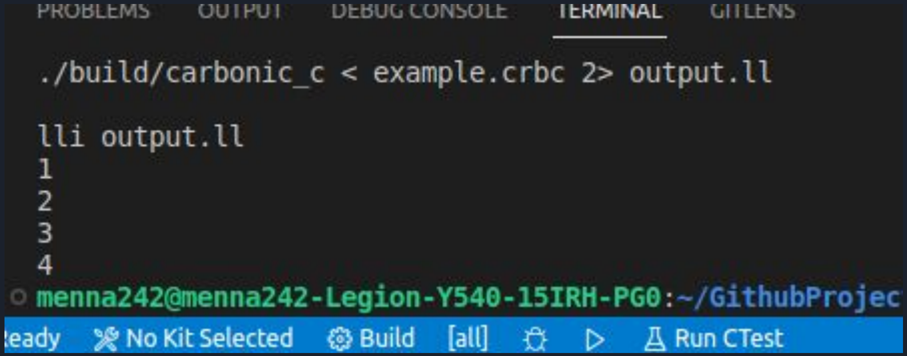


```llvm

define i32 @main() {
entry:
  %a = alloca i32, align 4
  store i32 6, i32* %a, align 4
  %i = alloca i32, align 4
  store i32 0, i32* %i, align 4
  br label %loopCond

loopCond:                                         ; preds = %loopInc, %entry
  %i1 = load i32, i32* %i, align 4
  %loopCond2 = icmp slt i32 %i1, 5
  br i1 %loopCond2, label %loopBody, label %loopExit

loopBody:                                         ; preds = %loopCond
  %a3 = load i32, i32* %a, align 4
  %0 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @intFmtString, i32 0, i32 0), i32 %a3)
  %a4 = load i32, i32* %a, align 4
  %a5 = load i32, i32* %a, align 4
  %a6 = load i32, i32* %a, align 4
  %result = add i32 %a5, %a6
  store i32 %result, i32* %a, align 4
  br label %loopInc

loopExit:                                         ; preds = %loopCond
  ret i32 0

loopInc:                                          ; preds = %loopBody
  %i7 = load i32, i32* %i, align 4
  %bodyRes = add i32 %i7, 1
  store i32 %bodyRes, i32* %i, align 4
  br label %loopCond
}
```

Example #1

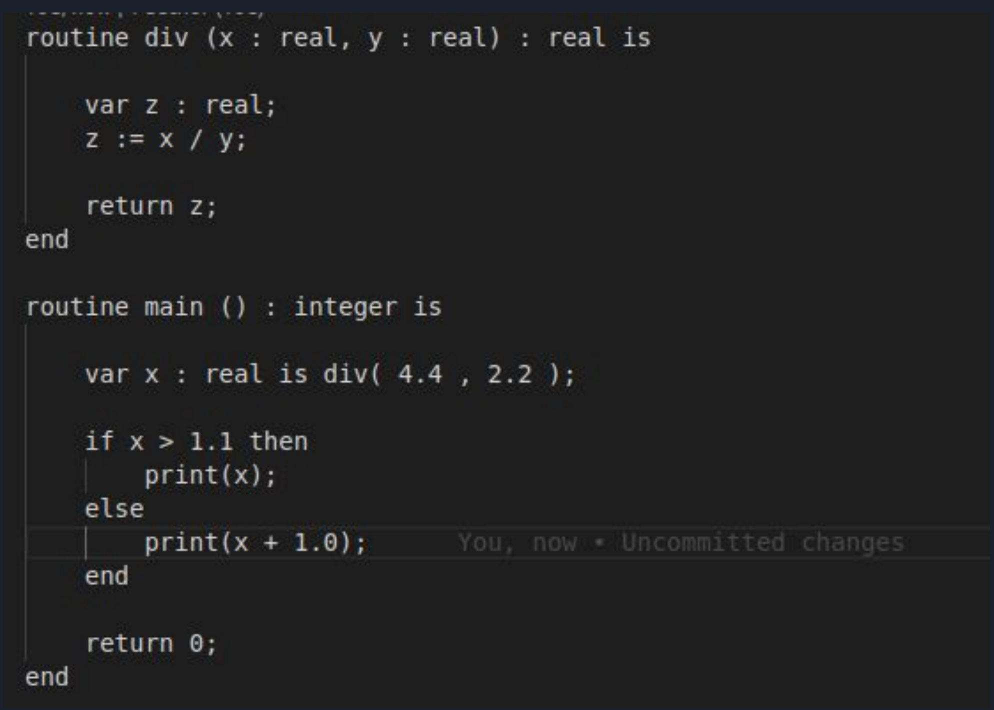


```
You, 43 seconds ago | 1 author (You)
routine main () : integer is
    var a : integer is 0;
    for i in 1 .. 5 loop
        a := a + 1;
        print(a);
    end
    return 0;        You, 43 sec
end
```

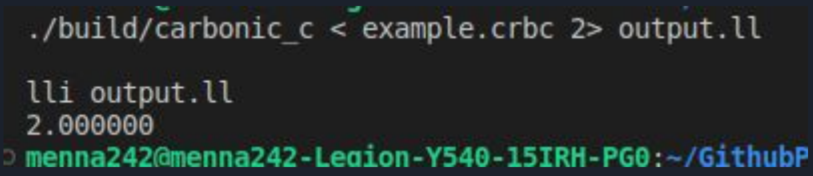


```
./build/carbonic_c < example.crbc 2> output.ll

lli output.ll
1
2
3
4
menna242@menna242-Legion-Y540-15IRH-PG0:~/GithubProjec
```

Example #2

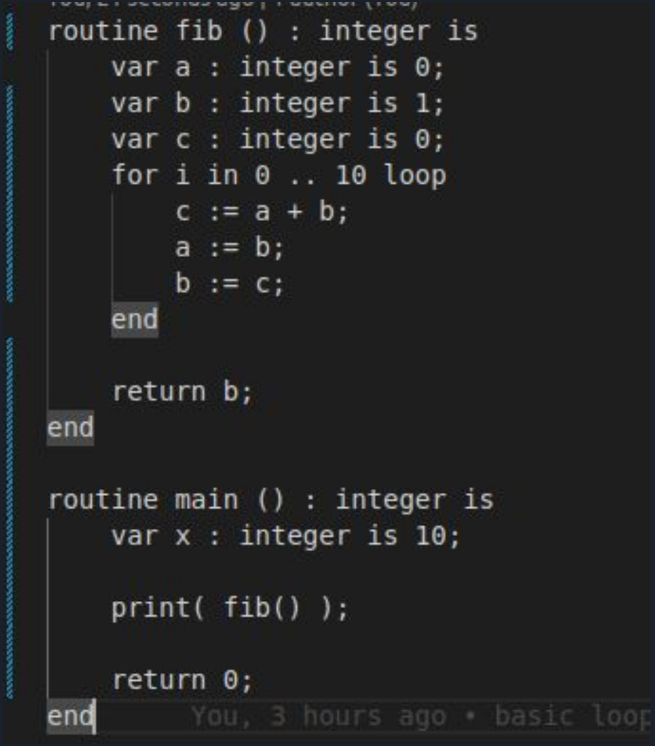


```
routine div (x : real, y : real) : real is

    var z : real;
    z := x / y;

    return z;
end

routine main () : integer is

    var x : real is div( 4.4 , 2.2 );

    if x > 1.1 then
        print(x);
    else
        print(x + 1.0);      You, now • Uncommitted changes
    end

    return 0;
end
```

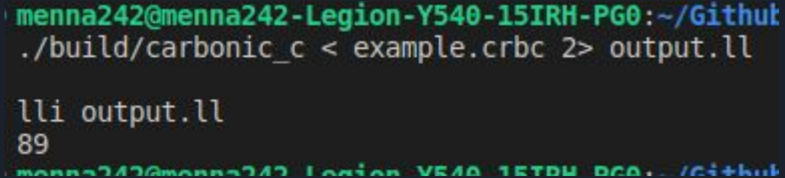


```
./build/carbonic_c < example.crbc 2> output.ll

lli output.ll
2.000000
menna242@menna242-Legion-Y540-15IRH-PG0:~/GithubP
```

Example #3



```
routine fib () : integer is
    var a : integer is 0;
    var b : integer is 1;
    var c : integer is 0;
    for i in 0 .. 10 loop
        c := a + b;
        a := b;
        b := c;
    end

    return b;
end

routine main () : integer is
    var x : integer is 10;

    print( fib() );

    return 0;
end       You, 3 hours ago • basic loop
```



```
menna242@menna242-Legion-Y540-15IRH-PG0:~/Githut
./build/carbonic_c < example.crbc 2> output.ll

lli output.ll
89
```

Responsibilities

Asem: Code Generation

Jaffar: CMake, Automation, Configurations

Menna: Code Generation

Mosab: Automation, Configurations

Demo Time!

Thank you for Listening!