🎯 Minic阶段0：加法编译器 - 15分钟从零到精通

🚀 学生快速上手（30秒体验）

立即开始

```
git clone https://github.com/zhnt/minic-student.git
cd minic-student
git checkout stage-0
cd stage-0
make
echo "3 + 5" | ./build/minic        # 8.00 ✨
```

📚 15分钟完整学习指南

🎯 学习目标

- **15分钟内**体验编译器神奇之处
- **零恐惧**理解编译器基础工作原理
- **建立信心**为后续复杂阶段打下基础
- **掌握方法**学会阅读和理解编译器代码

📈 四层渐进学习路径

第1步：30秒上手（立即见效）

```
make
echo "3 + 5" | ./build/minic        # 8.00
echo "100 + 200" | ./build/minic    # 300.00
```

第2步：2分钟观察（理解内部）

```
# 观察token化过程
./build/minic -vt   # 输入: 10 + 20
# 输出: NUMBER(10) PLUS NUMBER(20)

# 观察AST结构
./build/minic -va   # 输入: 10 + 20
# 输出: (+ 10 20)
```

第3步：5分钟深入（理解原理）

```
# 观察完整编译流程
./build/minic -v    # 输入: 42 + 58
# 完整显示:
# 源代码: 42 + 58
# Token: [NUMBER(42), PLUS, NUMBER(58)]
# AST: (+ 42 58)
# 字节码: PUSH 42, PUSH 58, ADD
# 结果: 100.00
```

第4步：8分钟实践（动手操作）

```
# 创建你的第一个程序
echo "total = 100 + 200 + 300; total" > myfirst.mc
./build/minic myfirst.mc  # 600.00

# 交互式编程
./build/minic
minic> 1 + 2 + 3 + 4 + 5
15.00
minic> x = 100 + 200; y = x + 50; y
350.00
minic> exit
```

🔍 编译器架构理解

1. 三层架构模型

```
源代码 → 词法分析 → 语法分析 → 代码生成 → 虚拟机执行
   ↑         ↑            ↑           ↑            ↑
"3+5" → [3,+,5] → (+ 3 5) → 字节码 → 8.00
```

2. 关键文件解读

文件	作用	关键函数	学习重点
`lexer.c`	词法分析器	字符→token	如何识别加号和数字
`parser.c`	语法分析器	token→AST	如何构建加法表达式
`compiler.c`	代码生成器	AST→字节码	如何生成加法指令
`vm.c`	虚拟机	字节码→结果	如何执行加法运算

3. 代码阅读重点

```
// 词法分析关键代码
switch (c) {
    case '+': return TOKEN_PLUS;    // 识别加号
    case '0'...'9': return scan_number();  // 识别数字
}

// 语法分析关键代码
Expr *parse_expression() {
    Expr *left = parse_primary();    // 左操作数
    if (match(TOKEN_PLUS)) {         // 遇到加号
        Expr *right = parse_expression();  // 右操作数
        return make_binary(left, right);   // 创建表达式
    }
    return left;
}

// 虚拟机执行关键代码
case OP_ADD: {
    double b = pop();           // 右操作数
    double a = pop();           // 左操作数
    push(a + b);                // 计算结果
    break;
}
```

📊 学习检查清单

基础理解（必须完成）

- [ ] 能够运行至少5个不同的加法运算
- [ ] 能够使用-vt观察token化过程
- [ ] 能够使用-va观察AST结构
- [ ] 能够使用-v观察完整编译流程

进阶理解（选做）

- [ ] 能够解释lexer.c如何识别加号和数字
- [ ] 能够理解parser.c如何构建加法表达式

- [ ] 能够描述从源代码到结果的完整流程

创新实践（挑战）

- [ ] 修改代码添加新的运算符（如减号）
- [ ] 扩展支持更多数字格式
- [ ] 创建复杂的加法表达式进行测试

🎓 学习建议

1. 分层学习法

- **第1层**：会用即可（5分钟）
- **第2层**：理解结构（10分钟）
- **第3层**：阅读关键代码（15分钟）
- **第4层**：尝试修改（额外时间）

2. 动手实践法

每学一个概念，立即用实际例子验证：

```
# 测试不同场景
echo "1+2+3+4+5" | ./build/minic      # 15.00
echo "x=10; y=20; x+y" | ./build/minic  # 30.00
```

3. 错误学习法

如果遇到编译错误，这是理解编译器如何处理错误的绝佳机会。

🔮 下一步预告

完成加法编译器后，你将进入阶段1：

- **词法分析观察**：深入理解编译器如何"看懂"代码
- **token概念**：学习编译器如何识别不同类型的语法元素
- **调试工具**：掌握-vt选项观察token化过程

📞 获取帮助

常见问题

1. **make失败**：确认已安装gcc和make
2. **运行无输出**：检查输入格式，确保使用echo管道
3. **理解困难**：先完成体验部分，理论可以慢慢理解

学习资源

- **代码注释**：每个关键函数都有详细注释
- **测试文件**：tests/目录下有示例用法
- **在线支持**：GitHub Issues提问

🎯 **记住：你已经迈出了编译器学习的第一步！加法虽小，五脏俱全。这15分钟的体验将为你打开编译器学习的大门！**

现在就开始你的15分钟编译器之旅吧！