斐波那契数列程序探索过程

源程序（`source`）

1. `fib_baseline.cpp`

   基准程序，用于观察基本的、带有I/O副作用的循环是如何被编译的。

```
// fib_baseline.cpp
// 原始的斐波那契代码，带 cin 和循环内 cout。
// 作为所有对比的基准。

#include <iostream>

int main() {
    // 初始化
    int a = 0, b = 1, i = 1, t,n;

    // 从用户获取项数
    std::cin >> n;

    // 打印前两项
    std::cout << a << std::endl;
    std::cout << b << std::endl;

    // 在循环中计算并打印每一项
    while (i < n) {
        t = b;
        b = a + b;
        a = t;
        i = i + 1;
        std::cout << b << std::endl; // I/O操作在循环内部
    }

    return 0;
}
```

2. `fib_pure.cpp`

   用于探索**副作用 (Side Effects)** 对编译器优化的影响。

```
// fib_pure.cpp
// 将 cout 从循环内移到循环外，创建一个“纯净”的计算循环。
// 用于对比观察有无副作用时，循环优化的差异。
```

```cpp
#include <iostream>

int main() {
    int a = 0, b = 1, i = 1, t,n;

    // 从用户获取项数
    std::cin >> n;

    // 循环中只进行计算，没有任何I/O
    while (i < n) {
        t = b;
        b = a + b;
        a = t;
        i = i + 1;
    }

    // 所有计算完成后，只打印最终结果
    std::cout << b << std::endl;

    return 0;
}
```

3. `fib_const_pure.cpp`

用于探索常数折叠。

```cpp
// source/fib_const_pure.cpp
// 结合了常量输入和纯净循环的终极版本

#include <iostream>

int main() {
    const int n = 10; // 特性一：编译时常量

    int a = 0, b = 1, i = 1, t;

    // 特性二：纯计算循环，无副作用
    while (i < n) {
        t = b;
        b = a + b;
        a = t;
        i = i + 1;
    }

    // 只在最后打印最终结果
    std::cout << b << std::endl;
```

```
    return 0;
}
```

预处理器（`preprocessed`）

预处理命令

```
riscv64-unknown-elf-g++ source/fib_baseline.cpp -E -o
preprocessed/fib_baseline.i
riscv64-unknown-elf-g++ source/fib_pure.cpp -E -o preprocessed/fib_pure.i
riscv64-unknown-elf-g++ source/fib_const_pure.cpp -E -o
preprocessed/fib_const_pure.i
```

关于预处理器的思考

1. `g++ -E`是`cpp`的一层封装

   - **通过编译器驱动程序:** riscv64-unknown-elf-g++ source/fib_const_pure.cpp -E -o preprocessed/fib_const_pure.i
   - **直接调用预处理器:** riscv64-unknown-elf-cpp source/fib_const_pure.cpp > preprocessed/fib_const_pure_direct.i
   - 通过对比输出文件 fib_const_pure.i 和 fib_const_pure_direct.i，我们发现**两者内容完全一致**。这个结果有力地证明了，g++ (或 clang++) 这样的编译器驱动程序，在执行预处理任务时，其本质是调用了其工具链中一个名为 cpp (C Pre-Processor) 的独立程序来完成的。g++ 扮演的是一个“总指挥”的角色，而 cpp 则是负责预处理这道工序的“专家”。

2. 思考：为什么需要预处理器?

   预处理器并非C++语言本身的一部分，而是在编译器工作之前，对源代码进行结构化文本重组的强大自动化工具。它通过**头文件展开、宏替换和条件编译**三大核心功能，为C++提供了**代码复用、灵活性和跨平台**的关键支持，是整个编译流程中不可或缺的前置步骤。

编译器

词法分析

词法分析指令

```
clang++ -fsyntax-only -Xclang -dump-tokens source/fib_baseline.cpp >
compiler/1_lexical/fib_baseline.tokens 2>&1
clang++ -fsyntax-only -Xclang -dump-tokens source/fib_pure.cpp >
compiler/1_lexical/fib_pure.tokens 2>&1
clang++ -fsyntax-only -Xclang -dump-tokens source/fib_const_pure.cpp >
compiler/1_lexical/fib_const_pure.tokens 2>&1
```

词法分析结果

```
int 'int'    [StartOfLine]  Loc=<source/fib_baseline.cpp:7:1>
identifier 'main'    [LeadingSpace] Loc=<source/fib_baseline.cpp:7:5>
l_paren '('     Loc=<source/fib_baseline.cpp:7:9>
r_paren ')'     Loc=<source/fib_baseline.cpp:7:10>
l_brace '{'  [LeadingSpace] Loc=<source/fib_baseline.cpp:7:12>
int 'int'    [StartOfLine] [LeadingSpace]   Loc=<source/fib_baseline.cpp:9:5>
identifier 'a'   [LeadingSpace] Loc=<source/fib_baseline.cpp:9:9>
equal '='    [LeadingSpace] Loc=<source/fib_baseline.cpp:9:11>
numeric_constant '0'     [LeadingSpace] Loc=<source/fib_baseline.cpp:9:13>
comma ','       Loc=<source/fib_baseline.cpp:9:14>
identifier 'b'   [LeadingSpace] Loc=<source/fib_baseline.cpp:9:16>
equal '='    [LeadingSpace] Loc=<source/fib_baseline.cpp:9:18>
numeric_constant '1'     [LeadingSpace] Loc=<source/fib_baseline.cpp:9:20>
comma ','       Loc=<source/fib_baseline.cpp:9:21>
identifier 'i'   [LeadingSpace] Loc=<source/fib_baseline.cpp:9:23>
equal '='    [LeadingSpace] Loc=<source/fib_baseline.cpp:9:25>
numeric_constant '1'     [LeadingSpace] Loc=<source/fib_baseline.cpp:9:27>
comma ','       Loc=<source/fib_baseline.cpp:9:28>
identifier 't'   [LeadingSpace] Loc=<source/fib_baseline.cpp:9:30>
comma ','       Loc=<source/fib_baseline.cpp:9:31>
identifier 'n'      Loc=<source/fib_baseline.cpp:9:32>
semi ';'        Loc=<source/fib_baseline.cpp:9:33>
identifier 'std'     [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:12:5>
coloncolon '::'     Loc=<source/fib_baseline.cpp:12:8>
identifier 'cin'        Loc=<source/fib_baseline.cpp:12:10>
greatergreater '>>'  [LeadingSpace] Loc=<source/fib_baseline.cpp:12:14>
identifier 'n'   [LeadingSpace] Loc=<source/fib_baseline.cpp:12:17>
semi ';'        Loc=<source/fib_baseline.cpp:12:18>
identifier 'std'     [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:15:5>
coloncolon '::'     Loc=<source/fib_baseline.cpp:15:8>
identifier 'cout'       Loc=<source/fib_baseline.cpp:15:10>
lessless '<<'    [LeadingSpace] Loc=<source/fib_baseline.cpp:15:15>
```

```
identifier 'a'   [LeadingSpace] Loc=<source/fib_baseline.cpp:15:18>
lessless '<<'    [LeadingSpace] Loc=<source/fib_baseline.cpp:15:20>
identifier 'std'     [LeadingSpace] Loc=<source/fib_baseline.cpp:15:23>
coloncolon '::'     Loc=<source/fib_baseline.cpp:15:26>
identifier 'endl'       Loc=<source/fib_baseline.cpp:15:28>
semi ';'        Loc=<source/fib_baseline.cpp:15:32>
identifier 'std'     [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:16:5>
coloncolon '::'     Loc=<source/fib_baseline.cpp:16:8>
identifier 'cout'       Loc=<source/fib_baseline.cpp:16:10>
lessless '<<'    [LeadingSpace] Loc=<source/fib_baseline.cpp:16:15>
identifier 'b'   [LeadingSpace] Loc=<source/fib_baseline.cpp:16:18>
lessless '<<'    [LeadingSpace] Loc=<source/fib_baseline.cpp:16:20>
identifier 'std'     [LeadingSpace] Loc=<source/fib_baseline.cpp:16:23>
coloncolon '::'     Loc=<source/fib_baseline.cpp:16:26>
identifier 'endl'       Loc=<source/fib_baseline.cpp:16:28>
semi ';'        Loc=<source/fib_baseline.cpp:16:32>
while 'while'    [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:19:5>
l_paren '('  [LeadingSpace] Loc=<source/fib_baseline.cpp:19:11>
identifier 'i'      Loc=<source/fib_baseline.cpp:19:12>
less '<'     [LeadingSpace] Loc=<source/fib_baseline.cpp:19:14>
identifier 'n'   [LeadingSpace] Loc=<source/fib_baseline.cpp:19:16>
r_paren ')'     Loc=<source/fib_baseline.cpp:19:17>
l_brace '{'  [LeadingSpace] Loc=<source/fib_baseline.cpp:19:19>
identifier 't'   [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:20:9>
equal '='    [LeadingSpace] Loc=<source/fib_baseline.cpp:20:11>
identifier 'b'   [LeadingSpace] Loc=<source/fib_baseline.cpp:20:13>
semi ';'        Loc=<source/fib_baseline.cpp:20:14>
identifier 'b'   [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:21:9>
equal '='    [LeadingSpace] Loc=<source/fib_baseline.cpp:21:11>
identifier 'a'   [LeadingSpace] Loc=<source/fib_baseline.cpp:21:13>
plus '+'     [LeadingSpace] Loc=<source/fib_baseline.cpp:21:15>
identifier 'b'   [LeadingSpace] Loc=<source/fib_baseline.cpp:21:17>
semi ';'        Loc=<source/fib_baseline.cpp:21:18>
identifier 'a'   [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:22:9>
equal '='    [LeadingSpace] Loc=<source/fib_baseline.cpp:22:11>
identifier 't'   [LeadingSpace] Loc=<source/fib_baseline.cpp:22:13>
semi ';'        Loc=<source/fib_baseline.cpp:22:14>
identifier 'i'   [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:23:9>
equal '='    [LeadingSpace] Loc=<source/fib_baseline.cpp:23:11>
identifier 'i'   [LeadingSpace] Loc=<source/fib_baseline.cpp:23:13>
plus '+'     [LeadingSpace] Loc=<source/fib_baseline.cpp:23:15>
numeric_constant '1'     [LeadingSpace] Loc=<source/fib_baseline.cpp:23:17>
```

```
semi ';'        Loc=<source/fib_baseline.cpp:23:18>
identifier 'std'     [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:24:9>
coloncolon '::'     Loc=<source/fib_baseline.cpp:24:12>
identifier 'cout'       Loc=<source/fib_baseline.cpp:24:14>
lessless '<<'    [LeadingSpace] Loc=<source/fib_baseline.cpp:24:19>
identifier 'b'   [LeadingSpace] Loc=<source/fib_baseline.cpp:24:22>
lessless '<<'    [LeadingSpace] Loc=<source/fib_baseline.cpp:24:24>
identifier 'std'     [LeadingSpace] Loc=<source/fib_baseline.cpp:24:27>
coloncolon '::'     Loc=<source/fib_baseline.cpp:24:30>
identifier 'endl'       Loc=<source/fib_baseline.cpp:24:32>
semi ';'        Loc=<source/fib_baseline.cpp:24:36>
r_brace '}'  [StartOfLine] [LeadingSpace]   Loc=<source/fib_baseline.cpp:25:5>
return 'return'  [StartOfLine] [LeadingSpace]   Loc=
<source/fib_baseline.cpp:27:5>
numeric_constant '0'     [LeadingSpace] Loc=<source/fib_baseline.cpp:27:12>
semi ';'        Loc=<source/fib_baseline.cpp:27:13>
r_brace '}'  [StartOfLine]  Loc=<source/fib_baseline.cpp:28:1>
eof ''      Loc=<source/fib_baseline.cpp:28:2>
```

词法分析做了什么

1. 从上面的输出结果中，我们可以清晰地看到几个关键的转化行为：

- **关键字识别：** 字符串 int 和 while 被准确地识别为关键字Token int 和 while。这表明词法分析器内置了一份C++的关键字列表。
- **标识符识别：** 我们定义的变量名如 a, b, i, main 以及命名空间 std 都被归类为 identifier。词法分析器遵循“由字母、数字、下划线组成，且不能以数字开头”的规则来识别它们。
- **常量识别：** 数字 0 和 1 被识别为 numeric_constant。
- **操作符与分隔符识别：** 诸如 ( ), { }, =, +, <, <<, >>, ; 甚至 :: 都被精确地切割并赋予了独立的Token类型，如 l_paren (左括号), equal (等于), plus (加号), lessless (左移), coloncolon (双冒号), semi (分号) 等。

2. **降维与简化——为语法分析铺路**

   词法分析的第一个伟大贡献是**降维**。它将语法分析器需要面对的问题从“如何处理每一个字符”简化为“如何处理每一个有意义的单词”。语法分析器不再需要关心 w-h-i-l-e 这五个字符，而只需要处理一个名为 while 的Token。它也不再需要关心一个=和一个==的区别，词法分析器已经明确地告诉它这是两个不同的Token (equal vs. equalequal)。这种简化是构建后续更复杂的语法结构（如抽象语法树）的前提。

3. **忽略“无关紧要”的空白**

> 我们注意到输出中有一个 [LeadingSpace] 标记，这表明词法分析器**识别并处理了**源代码中的空格和换行，但**并未**将它们作为有意义的Token传递给下一阶段（除了用于标记Token位置）。这体现了C++作为一种自由格式语言的特性——空白字符只用于分隔Token，不影响代码的逻辑结构。词法分析器是实现这一特性的“守门员”。

词法分析没做什么

- **只做“分词”，不解“文法”**

> 它完成了从字符到单词的飞跃，但如何将这些单词组织成有意义的短语、从句和句子，这个艰巨的任务被留给了下一阶段——**语法分析 (Syntax Analysis)。**

语法分析

语法分析命令

```
clang++ -fsyntax-only -Xclang -ast-dump source/fib_baseline.cpp >
compiler/2_syntax/fib_baseline.ast 2>&1
clang++ -fsyntax-only -Xclang -ast-dump source/fib_pure.cpp >
compiler/2_syntax/fib_pure.ast 2>&1
clang++ -fsyntax-only -Xclang -ast-dump source/fib_const_pure.cpp >
compiler/2_syntax/fib_const_pure.ast 2>&1
```

画图命令

```
clang++ -Xclang -ast-dump=json -fsyntax-only source/fib_baseline.cpp >
compiler/2_syntax/fib_baseline.json
python3 ast_parser.py compiler/2_syntax/fib_baseline.json
compiler/2_syntax/fib_baseline.dot --filter fib_baseline.cpp
dot -Tpng compiler/2_syntax/fib_baseline.dot -o
compiler/2_syntax/fib_baseline_ast.png
```

```
clang++ -Xclang -ast-dump=json -fsyntax-only source/fib_pure.cpp >
compiler/2_syntax/fib_pure.json
python3 ast_parser.py compiler/2_syntax/fib_pure.json
compiler/2_syntax/fib_pure.dot --filter fib_pure.cpp
dot -Tpng compiler/2_syntax/fib_pure.dot -o compiler/2_syntax/fib_pure_ast.png
```

```
clang++ -Xclang -ast-dump=json -fsyntax-only source/fib_const_pure.cpp >
compiler/2_syntax/fib_const_pure.json
```

```
python3 ast_parser.py compiler/2_syntax/fib_const_pure.json
compiler/2_syntax/fib_const_pure.dot --filter fib_const_pure.cpp
dot -Tpng compiler/2_syntax/fib_const_pure.dot -o
compiler/2_syntax/fib_const_pure_ast.png
```

baseline版本ast(clang 原始的文本转储)

```
`-FunctionDecl 0x130a0d40 <source/fib_baseline.cpp:7:1, line:28:1> line:7:5 main
'int ()'
  `-CompoundStmt 0x130b4228 <col:12, line:28:1>
    |-DeclStmt 0x130a10f0 <line:9:5, col:33>
    | |-VarDecl 0x130a0df8 <col:5, col:13> col:9 used a 'int' cinit
    | | `-IntegerLiteral 0x130a0e60 <col:13> 'int' 0
    | |-VarDecl 0x130a0e98 <col:5, col:20> col:16 used b 'int' cinit
    | | `-IntegerLiteral 0x130a0f00 <col:20> 'int' 1
    | |-VarDecl 0x130a0f38 <col:5, col:27> col:23 used i 'int' cinit
    | | `-IntegerLiteral 0x130a0fa0 <col:27> 'int' 1
    | |-VarDecl 0x130a0fd8 <col:5, col:30> col:30 used t 'int'
    | `-VarDecl 0x130a1058 <col:5, col:32> col:32 used n 'int'
    |-CXXOperatorCallExpr 0x130a4ca0 <line:12:5, col:17>
'std::basic_istream<char>::__istream_type':'std::basic_istream<char>' lvalue
'>>'
    | |-ImplicitCastExpr 0x130a4c88 <col:14>
'std::basic_istream<char>::__istream_type &(*)(int &)' <FunctionToPointerDecay>
    | | `-DeclRefExpr 0x130a4c10 <col:14>
'std::basic_istream<char>::__istream_type &(int &)' lvalue CXXMethod 0x13045678
'operator>>' 'std::basic_istream<char>::__istream_type &(int &)'
    | |-DeclRefExpr 0x130a1170 <col:5, col:10>
'std::istream':'std::basic_istream<char>' lvalue Var 0x130a0700 'cin'
'std::istream':'std::basic_istream<char>'
    | `-DeclRefExpr 0x130a11a0 <col:17> 'int' lvalue Var 0x130a1058 'n' 'int'
    |-CXXOperatorCallExpr 0x130b0ee0 <line:15:5, col:28>
'std::basic_ostream<char>::__ostream_type':'std::basic_ostream<char>' lvalue
'<<'
    | |-ImplicitCastExpr 0x130b0ec8 <col:20>
'std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &))' <FunctionToPointerDecay>
    | | `-DeclRefExpr 0x130b0e50 <col:20>
'std::basic_ostream<char>::__ostream_type &
(std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &))' lvalue CXXMethod 0x13024258
'operator<<' 'std::basic_ostream<char>::__ostream_type &
(std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &))'
    | |-CXXOperatorCallExpr 0x130b0390 <col:5, col:18>
```

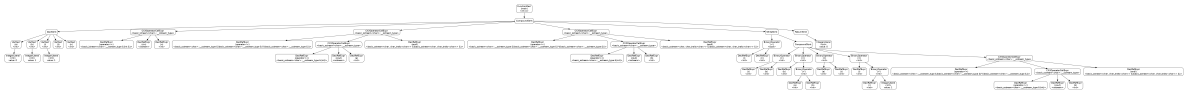
```
'std::basic_ostream<char>::__ostream_type':'std::basic_ostream<char>' lvalue
'<<'
    | | |-ImplicitCastExpr 0x130b0378 <col:15>
'std::basic_ostream<char>::__ostream_type &(*)(int)' <FunctionToPointerDecay>
    | | | `-DeclRefExpr 0x130b02f8 <col:15>
'std::basic_ostream<char>::__ostream_type &(int)' lvalue CXXMethod 0x13025228
'operator<<' 'std::basic_ostream<char>::__ostream_type &(int)'
    | | |-DeclRefExpr 0x130a4f18 <col:5, col:10>
'std::ostream':'std::basic_ostream<char>' lvalue Var 0x130a0778 'cout'
'std::ostream':'std::basic_ostream<char>'
    | | `-ImplicitCastExpr 0x130b02e0 <col:18> 'int' <LValueToRValue>
    | |   `-DeclRefExpr 0x130a4f48 <col:18> 'int' lvalue Var 0x130a0df8 'a'
'int'
    | `-ImplicitCastExpr 0x130b0e38 <col:23, col:28> 'basic_ostream<char,
std::char_traits<char>> &(*)(basic_ostream<char, std::char_traits<char>> &)'
<FunctionToPointerDecay>
    |   `-DeclRefExpr 0x130b0e00 <col:23, col:28> 'basic_ostream<char,
std::char_traits<char>> &(basic_ostream<char, std::char_traits<char>> &)' lvalue
Function 0x130289d8 'endl' 'basic_ostream<char, std::char_traits<char>> &
(basic_ostream<char, std::char_traits<char>> &)' (FunctionTemplate 0x13009748
'endl')
    |-CXXOperatorCallExpr 0x130b26b8 <line:16:5, col:28>
'std::basic_ostream<char>::__ostream_type':'std::basic_ostream<char>' lvalue
'<<'
    | |-ImplicitCastExpr 0x130b26a0 <col:20>
'std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &))' <FunctionToPointerDecay>
    | | `-DeclRefExpr 0x130b2680 <col:20>
'std::basic_ostream<char>::__ostream_type &
(std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &))' lvalue CXXMethod 0x13024258
'operator<<' 'std::basic_ostream<char>::__ostream_type &
(std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &))'
    | |-CXXOperatorCallExpr 0x130b1cd8 <col:5, col:18>
'std::basic_ostream<char>::__ostream_type':'std::basic_ostream<char>' lvalue
'<<'
    | | |-ImplicitCastExpr 0x130b1cc0 <col:15>
'std::basic_ostream<char>::__ostream_type &(*)(int)' <FunctionToPointerDecay>
    | | | `-DeclRefExpr 0x130b1ca0 <col:15>
'std::basic_ostream<char>::__ostream_type &(int)' lvalue CXXMethod 0x13025228
'operator<<' 'std::basic_ostream<char>::__ostream_type &(int)'
    | | |-DeclRefExpr 0x130b0f98 <col:5, col:10>
'std::ostream':'std::basic_ostream<char>' lvalue Var 0x130a0778 'cout'
'std::ostream':'std::basic_ostream<char>'
    | | `-ImplicitCastExpr 0x130b1c88 <col:18> 'int' <LValueToRValue>
    | |   `-DeclRefExpr 0x130b0fc8 <col:18> 'int' lvalue Var 0x130a0e98 'b'
```

```
'int'
    | `-ImplicitCastExpr 0x130b2668 <col:23, col:28> 'basic_ostream<char,
std::char_traits<char>> &(*)(basic_ostream<char, std::char_traits<char>> &)'
<FunctionToPointerDecay>
    |   `-DeclRefExpr 0x130b2630 <col:23, col:28> 'basic_ostream<char,
std::char_traits<char>> &(basic_ostream<char, std::char_traits<char>> &)' lvalue
Function 0x130289d8 'endl' 'basic_ostream<char, std::char_traits<char>> &
(basic_ostream<char, std::char_traits<char>> &)' (FunctionTemplate 0x13009748
'endl')
    |-WhileStmt 0x130b41d8 <line:19:5, line:25:5>
    | |-BinaryOperator 0x130b2760 <line:19:12, col:16> 'bool' '<'
    | | |-ImplicitCastExpr 0x130b2730 <col:12> 'int' <LValueToRValue>
    | | | `-DeclRefExpr 0x130b26f0 <col:12> 'int' lvalue Var 0x130a0f38 'i'
'int'
    | | `-ImplicitCastExpr 0x130b2748 <col:16> 'int' <LValueToRValue>
    | |   `-DeclRefExpr 0x130b2710 <col:16> 'int' lvalue Var 0x130a1058 'n'
'int'
    | `-CompoundStmt 0x130b41a0 <col:19, line:25:5>
    |   |-BinaryOperator 0x130b27d8 <line:20:9, col:13> 'int' lvalue '='
    |   | |-DeclRefExpr 0x130b2780 <col:9> 'int' lvalue Var 0x130a0fd8 't' 'int'
    |   | `-ImplicitCastExpr 0x130b27c0 <col:13> 'int' <LValueToRValue>
    |   |   `-DeclRefExpr 0x130b27a0 <col:13> 'int' lvalue Var 0x130a0e98 'b'
'int'
    |   |-BinaryOperator 0x130b28a8 <line:21:9, col:17> 'int' lvalue '='
    |   | |-DeclRefExpr 0x130b27f8 <col:9> 'int' lvalue Var 0x130a0e98 'b' 'int'
    |   | `-BinaryOperator 0x130b2888 <col:13, col:17> 'int' '+'
    |   |   |-ImplicitCastExpr 0x130b2858 <col:13> 'int' <LValueToRValue>
    |   |   | `-DeclRefExpr 0x130b2818 <col:13> 'int' lvalue Var 0x130a0df8 'a'
'int'
    |   |   `-ImplicitCastExpr 0x130b2870 <col:17> 'int' <LValueToRValue>
    |   |     `-DeclRefExpr 0x130b2838 <col:17> 'int' lvalue Var 0x130a0e98 'b'
'int'
    |   |-BinaryOperator 0x130b2920 <line:22:9, col:13> 'int' lvalue '='
    |   | |-DeclRefExpr 0x130b28c8 <col:9> 'int' lvalue Var 0x130a0df8 'a' 'int'
    |   | `-ImplicitCastExpr 0x130b2908 <col:13> 'int' <LValueToRValue>
    |   |   `-DeclRefExpr 0x130b28e8 <col:13> 'int' lvalue Var 0x130a0fd8 't'
'int'
    |   |-BinaryOperator 0x130b29d8 <line:23:9, col:17> 'int' lvalue '='
    |   | |-DeclRefExpr 0x130b2940 <col:9> 'int' lvalue Var 0x130a0f38 'i' 'int'
    |   | `-BinaryOperator 0x130b29b8 <col:13, col:17> 'int' '+'
    |   |   |-ImplicitCastExpr 0x130b29a0 <col:13> 'int' <LValueToRValue>
    |   |   | `-DeclRefExpr 0x130b2960 <col:13> 'int' lvalue Var 0x130a0f38 'i'
'int'
    |   |   `-IntegerLiteral 0x130b2980 <col:17> 'int' 1
    |   `-CXXOperatorCallExpr 0x130b4168 <line:24:9, col:32>
'std::basic_ostream<char>::__ostream_type':'std::basic_ostream<char>' lvalue
'<<'
    |     |-ImplicitCastExpr 0x130b4150 <col:24>
```

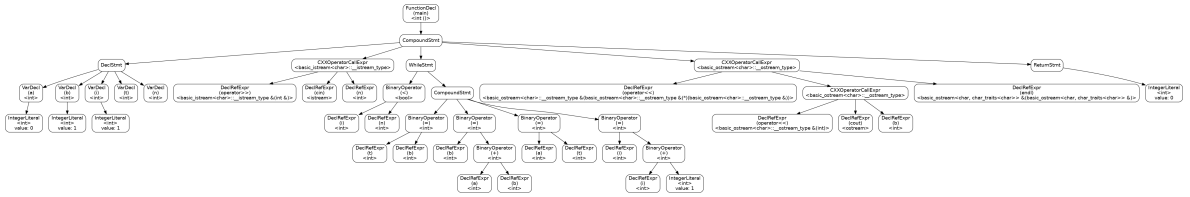
```
'std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &))' <FunctionToPointerDecay>
    |     | `-DeclRefExpr 0x130b4130 <col:24>
'std::basic_ostream<char>::__ostream_type &
(std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &))' lvalue CXXMethod 0x13024258
'operator<<' 'std::basic_ostream<char>::__ostream_type &
(std::basic_ostream<char>::__ostream_type &(*)
(std::basic_ostream<char>::__ostream_type &))'
    |     |-CXXOperatorCallExpr 0x130b3788 <col:9, col:22>
'std::basic_ostream<char>::__ostream_type':'std::basic_ostream<char>' lvalue
'<<'
    |     | |-ImplicitCastExpr 0x130b3770 <col:19>
'std::basic_ostream<char>::__ostream_type &(*)(int)' <FunctionToPointerDecay>
    |     | | `-DeclRefExpr 0x130b3750 <col:19>
'std::basic_ostream<char>::__ostream_type &(int)' lvalue CXXMethod 0x13025228
'operator<<' 'std::basic_ostream<char>::__ostream_type &(int)'
    |     | |-DeclRefExpr 0x130b2a48 <col:9, col:14>
'std::ostream':'std::basic_ostream<char>' lvalue Var 0x130a0778 'cout'
'std::ostream':'std::basic_ostream<char>'
    |     | `-ImplicitCastExpr 0x130b3738 <col:22> 'int' <LValueToRValue>
    |     |   `-DeclRefExpr 0x130b2a78 <col:22> 'int' lvalue Var 0x130a0e98 'b'
'int'
    |     `-ImplicitCastExpr 0x130b4118 <col:27, col:32> 'basic_ostream<char,
std::char_traits<char>> &(*)(basic_ostream<char, std::char_traits<char>> &)'
<FunctionToPointerDecay>
    |       `-DeclRefExpr 0x130b40e0 <col:27, col:32> 'basic_ostream<char,
std::char_traits<char>> &(basic_ostream<char, std::char_traits<char>> &)' lvalue
Function 0x130289d8 'endl' 'basic_ostream<char, std::char_traits<char>> &
(basic_ostream<char, std::char_traits<char>> &)' (FunctionTemplate 0x13009748
'endl')
    `-ReturnStmt 0x130b4218 <line:27:5, col:12>
      `-IntegerLiteral 0x130b41f8 <col:12> 'int' 0
```

自制的工具链生成的可视化图形

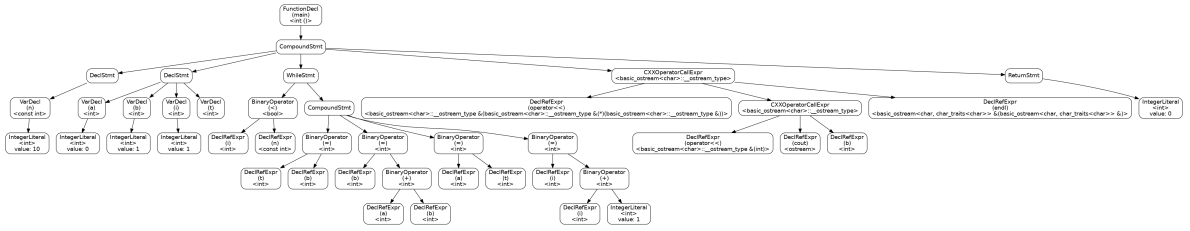
baseline



pure



const_pure



语法分析做了什么

1. AST是精确编码程序的语法结构，语法分析器依据C++语言的上下文无关文法（Context-Free Grammar），对输入的Token序列进行解析（Parsing），构建出一个能够反映源代码句法成分及其嵌套关系的树状结构。

    - **观察与分析：** 以fib_baseline.ast为例，源代码中的while循环语句在AST中被表示为一个WhileStmt节点。该节点拥有清晰的子节点结构，分别对应循环的**条件表达式**（Condition, 一个BinaryOperator节点，表示i < n）和**循环体**（Body, 一个CompoundStmt节点，表示{...}内的语句序列）。同样，函数调用、声明、赋值等语句也都被转换为具有明确父子和兄弟关系的相应AST节点类型。
    - **思考与意义：** 这种从线性Token序列到树状数据结构的转换，是编译器理解程序结构的关键。AST的层次性使得**作用域（Scoping）**、控制流（Control Flow）**和**表达式的运算优先级（Operator Precedence）**等重要的语法概念得到了形式化的、机器可处理的表示。这是后续所有依赖于程序结构的分析（如数据流分析、控制流分析）得以进行的基础。

2. AST在构建过程中附着了关键的静态语义信息，语法分析不仅构建了程序的结构骨架，还从源代码中提取了关键的静态信息，并将其作为属性标注在AST的节点上。

    - **观察与分析（横向对比）：**
        - **常量属性：** 在fib_const.ast中，变量n的VarDecl（Variable Declaration）节点被明确标注了const限定符，并关联了一个代表其初始值的IntegerLiteral子节点。这与fib_baseline.ast中普通的VarDecl节点形成了对比。
        - **类型信息：** 在fib_longlong.ast中，所有变量声明（VarDecl）和表达式运算（如BinaryOperator）节点的类型属性均被标注为'long long'，而非fib_baseline.ast中的'int'。

- **思考与意义：** AST并非纯粹的语法结构树，它是一个**属性图（Attributed Graph）的雏形。在语法分析过程中，类型、存储类别（如const）等静态语义信息被解析并绑定到相应的节点上。这棵“带标注”的树为下一阶段的语义分析**提供了必要的上下文信息，使其能够立即开始进行类型检查、常量表达式求值等更深层次的验证工作。

语义分析不做什么：不包含动态语义或优化决策

- **观察与分析（横向对比）：**
  fib_baseline.ast与fib_pure.ast的对比清晰地表明，CXXOperatorCallExpr（代表std::cout调用）在树中的**结构位置**因其在源代码中的位置不同而不同。AST忠实地记录了这一结构差异。
- **思考与意义：** AST本身不包含对**动态语义**的理解。例如，它不知道std::cout是一个具有**副作用（Side Effect）的函数调用，也不知道这个副作用在循环内部执行会对程序的最终行为产生何种影响。同样，AST也不包含任何优化信息**；它只是源代码的直接反映。对副作用的分析、常量折叠、循环不变量外提等优化决策，是在后续的语义分析和优化阶段，通过遍历和转换这棵AST来实现的。AST为这些高级转换提供了稳定、统一的输入数据结构。

语法分析阶段完成了从线性符号序列到**结构化、带类型标注的抽象语法树**的关键转换。AST作为编译器前端的核心数据结构，它不仅精确地编码了源代码的句法结构，还附着了后续阶段所必需的静态语义信息。它为编译器的后续工作——包括严格的语义验证、复杂的程序转换与优化——提供了一个规范化、机器可读的程序表示。

语义分析

中间代码生成

中间代码生成命令(不带优化)

```
clang++ -S -emit-llvm -O0 source/fib_baseline.cpp -o
compiler/4_ir_gen/fib_baseline_O0.ll
clang++ -S -emit-llvm -O0 source/fib_pure.cpp -o
compiler/4_ir_gen/fib_pure_O0.ll
clang++ -S -emit-llvm -O0 source/fib_const_pure.cpp -o
compiler/4_ir_gen/fib_const_pure_O0.ll
```

llvm ir

- baseline

```
; ModuleID = 'source/fib_baseline.cpp'
source_filename = "source/fib_baseline.cpp"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
```

```
%"class.std::ios_base::Init" = type { i8 }
%"class.std::basic_istream" = type { i32 (...)**, i64, %"class.std::basic_ios" }
%"class.std::basic_ios" = type { %"class.std::ios_base",
%"class.std::basic_ostream"*, i8, i8, %"class.std::basic_streambuf"*,
%"class.std::ctype"*, %"class.std::num_put"*, %"class.std::num_get"* }
%"class.std::ios_base" = type { i32 (...)**, i64, i64, i32, i32, i32,
%"struct.std::ios_base::_Callback_list"*, %"struct.std::ios_base::_Words", [8 x
%"struct.std::ios_base::_Words"], i32, %"struct.std::ios_base::_Words"*,
%"class.std::locale" }
%"struct.std::ios_base::_Callback_list" = type {
%"struct.std::ios_base::_Callback_list"*, void (i32, %"class.std::ios_base"*,
i32)*, i32, i32 }
%"struct.std::ios_base::_Words" = type { i8*, i64 }
%"class.std::locale" = type { %"class.std::locale::_Impl"* }
%"class.std::locale::_Impl" = type { i32, %"class.std::locale::facet"**, i64,
%"class.std::locale::facet"**, i8** }
%"class.std::locale::facet" = type <{ i32 (...)**, i32, [4 x i8] }>
%"class.std::basic_ostream" = type { i32 (...)**, %"class.std::basic_ios" }
%"class.std::basic_streambuf" = type { i32 (...)**, i8*, i8*, i8*, i8*, i8*,
i8*, %"class.std::locale" }
%"class.std::ctype" = type <{ %"class.std::locale::facet.base", [4 x i8],
%struct.__locale_struct*, i8, [7 x i8], i32*, i32*, i16*, i8, [256 x i8], [256 x
i8], i8, [6 x i8] }>
%"class.std::locale::facet.base" = type <{ i32 (...)**, i32 }>
%struct.__locale_struct = type { [13 x %struct.__locale_data*], i16*, i32*,
i32*, [13 x i8*] }
%struct.__locale_data = type opaque
%"class.std::num_put" = type { %"class.std::locale::facet.base", [4 x i8] }
%"class.std::num_get" = type { %"class.std::locale::facet.base", [4 x i8] }

@_ZStL8__ioinit = internal global %"class.std::ios_base::Init" zeroinitializer,
align 1
@__dso_handle = external hidden global i8
@_ZSt3cin = external global %"class.std::basic_istream", align 8
@_ZSt4cout = external global %"class.std::basic_ostream", align 8
@llvm.global_ctors = appending global [1 x { i32, void ()*, i8* }] [{ i32, void
()*, i8* } { i32 65535, void ()* @_GLOBAL__sub_I_fib_baseline.cpp, i8* null }]

; Function Attrs: noinline uwtable
define internal void @__cxx_global_var_init() #0 section ".text.startup" {
  call void @_ZNSt8ios_base4InitC1Ev(%"class.std::ios_base::Init"* noundef
nonnull align 1 dereferenceable(1) @_ZStL8__ioinit)
  %1 = call i32 @__cxa_atexit(void (i8*)* bitcast (void
(%"class.std::ios_base::Init"*)* @_ZNSt8ios_base4InitD1Ev to void (i8*)*), i8*
getelementptr inbounds (%"class.std::ios_base::Init",
%"class.std::ios_base::Init"* @_ZStL8__ioinit, i32 0, i32 0), i8* @__dso_handle)
#3
```

```
  ret void
}

declare void @_ZNSt8ios_base4InitC1Ev(%"class.std::ios_base::Init"* noundef
nonnull align 1 dereferenceable(1)) unnamed_addr #1

; Function Attrs: nounwind
declare void @_ZNSt8ios_base4InitD1Ev(%"class.std::ios_base::Init"* noundef
nonnull align 1 dereferenceable(1)) unnamed_addr #2

; Function Attrs: nounwind
declare i32 @__cxa_atexit(void (i8*)*, i8*, i8*) #3

; Function Attrs: mustprogress noinline norecurse optnone uwtable
define dso_local noundef i32 @main() #4 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  %4 = alloca i32, align 4
  %5 = alloca i32, align 4
  %6 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  store i32 0, i32* %2, align 4
  store i32 1, i32* %3, align 4
  store i32 1, i32* %4, align 4
  %7 = call noundef nonnull align 8 dereferenceable(16)
%"class.std::basic_istream"* @_ZNSirsERi(%"class.std::basic_istream"* noundef
nonnull align 8 dereferenceable(16) @_ZSt3cin, i32* noundef nonnull align 4
dereferenceable(4) %6)
  %8 = load i32, i32* %2, align 4
  %9 = call noundef nonnull align 8 dereferenceable(8)
%"class.std::basic_ostream"* @_ZNSolsEi(%"class.std::basic_ostream"* noundef
nonnull align 8 dereferenceable(8) @_ZSt4cout, i32 noundef %8)
  %10 = call noundef nonnull align 8 dereferenceable(8)
%"class.std::basic_ostream"* @_ZNSolsEPFRSoS_E(%"class.std::basic_ostream"*
noundef nonnull align 8 dereferenceable(8) %9, %"class.std::basic_ostream"*
(%"class.std::basic_ostream"*)* noundef
@_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_)
  %11 = load i32, i32* %3, align 4
  %12 = call noundef nonnull align 8 dereferenceable(8)
%"class.std::basic_ostream"* @_ZNSolsEi(%"class.std::basic_ostream"* noundef
nonnull align 8 dereferenceable(8) @_ZSt4cout, i32 noundef %11)
  %13 = call noundef nonnull align 8 dereferenceable(8)
%"class.std::basic_ostream"* @_ZNSolsEPFRSoS_E(%"class.std::basic_ostream"*
noundef nonnull align 8 dereferenceable(8) %12, %"class.std::basic_ostream"*
(%"class.std::basic_ostream"*)* noundef
@_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_)
  br label %14
```

```
14:                                               ; preds = %18, %0
  %15 = load i32, i32* %4, align 4
  %16 = load i32, i32* %6, align 4
  %17 = icmp slt i32 %15, %16
  br i1 %17, label %18, label %29

18:                                               ; preds = %14
  %19 = load i32, i32* %3, align 4
  store i32 %19, i32* %5, align 4
  %20 = load i32, i32* %2, align 4
  %21 = load i32, i32* %3, align 4
  %22 = add nsw i32 %20, %21
  store i32 %22, i32* %3, align 4
  %23 = load i32, i32* %5, align 4
  store i32 %23, i32* %2, align 4
  %24 = load i32, i32* %4, align 4
  %25 = add nsw i32 %24, 1
  store i32 %25, i32* %4, align 4
  %26 = load i32, i32* %3, align 4
  %27 = call noundef nonnull align 8 dereferenceable(8)
%"class.std::basic_ostream"* @_ZNSolsEi(%"class.std::basic_ostream"* noundef
nonnull align 8 dereferenceable(8) @_ZSt4cout, i32 noundef %26)
  %28 = call noundef nonnull align 8 dereferenceable(8)
%"class.std::basic_ostream"* @_ZNSolsEPFRSoS_E(%"class.std::basic_ostream"*
noundef nonnull align 8 dereferenceable(8) %27, %"class.std::basic_ostream"*
(%"class.std::basic_ostream"*)* noundef
@_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_)
  br label %14, !llvm.loop !6

29:                                               ; preds = %14
  ret i32 0
}

declare noundef nonnull align 8 dereferenceable(16) %"class.std::basic_istream"*
@_ZNSirsERi(%"class.std::basic_istream"* noundef nonnull align 8
dereferenceable(16), i32* noundef nonnull align 4 dereferenceable(4)) #1

declare noundef nonnull align 8 dereferenceable(8) %"class.std::basic_ostream"*
@_ZNSolsEi(%"class.std::basic_ostream"* noundef nonnull align 8
dereferenceable(8), i32 noundef) #1

declare noundef nonnull align 8 dereferenceable(8) %"class.std::basic_ostream"*
@_ZNSolsEPFRSoS_E(%"class.std::basic_ostream"* noundef nonnull align 8
dereferenceable(8), %"class.std::basic_ostream"* (%"class.std::basic_ostream"*)*
noundef) #1

declare noundef nonnull align 8 dereferenceable(8) %"class.std::basic_ostream"*
```

```
@_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_(%"class.std::basic_ostream"* noundef nonnull align 8 dereferenceable(8)) #1

; Function Attrs: noinline uwtable
define internal void @_GLOBAL__sub_I_fib_baseline.cpp() #0 section ".text.startup" {
  call void @__cxx_global_var_init()
  ret void
}

attributes #0 = { noinline uwtable "frame-pointer"="all" "min-legal-vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
attributes #2 = { nounwind "frame-pointer"="all" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
attributes #3 = { nounwind }
attributes #4 = { mustprogress noinline norecurse optnone uwtable "frame-pointer"="all" "min-legal-vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }

!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}

!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1.1"}
!6 = distinct !{!6, !7}
!7 = !{!"llvm.loop.mustprogress"}
```

llvm ir分析（基于baseline）

这份IR代码是源代码最直接的底层映射。关键特征如下：

- **变量的内存化：** 函数 @main 开头的一系列 alloca 指令为所有局部变量（a, b, i, t, n）在栈上分配了独立的内存空间。整个程序中对这些变量的访问都通过 load（读内存）和 store（写内存）指令完成，这体现了未优化代码的典型特征。

- **控制流的线性化：** C++ 中的 while 循环被分解为三个基本块（label %14, label %18, label %29）。label %14 块使用 icmp slt (有符号小于比较) 指令和 br (条件分支) 指令来决定执行流是进入循环体（label %18）还是退出循环（label %29），循环体执行完毕后通过无条件 br 跳回判断块，形成循环。
- **操作符到函数调用的转换：** C++中的 << 和 >> 操作符被翻译为对标准库中特定重载函数的 call 调用（例如 @_ZNSolsEi），其复杂的函数签名（Name Mangling结果）也得以保留。

得到CFG命令（不带优化）

baseline

```
# 1. 编译并生成所有.dot文件在当前目录
g++ -fdump-tree-all-graph -O0 -S source/fib_baseline.cpp -o /dev/null

# 2. 从中挑选出CFG的.dot文件，直接生成PNG图片
dot -Tpng fib_baseline.cpp.*.cfg.dot -o
compiler/4_ir_gen/fib_baseline_O0_cfg.png

# 3. 创建一个归档目录，并把这次生成的所有.dot文件都移进去
mkdir -p compiler/4_ir_gen/fib_baseline_dumps
mv fib_baseline.cpp.*.dot compiler/4_ir_gen/fib_baseline_dumps/
```

pure

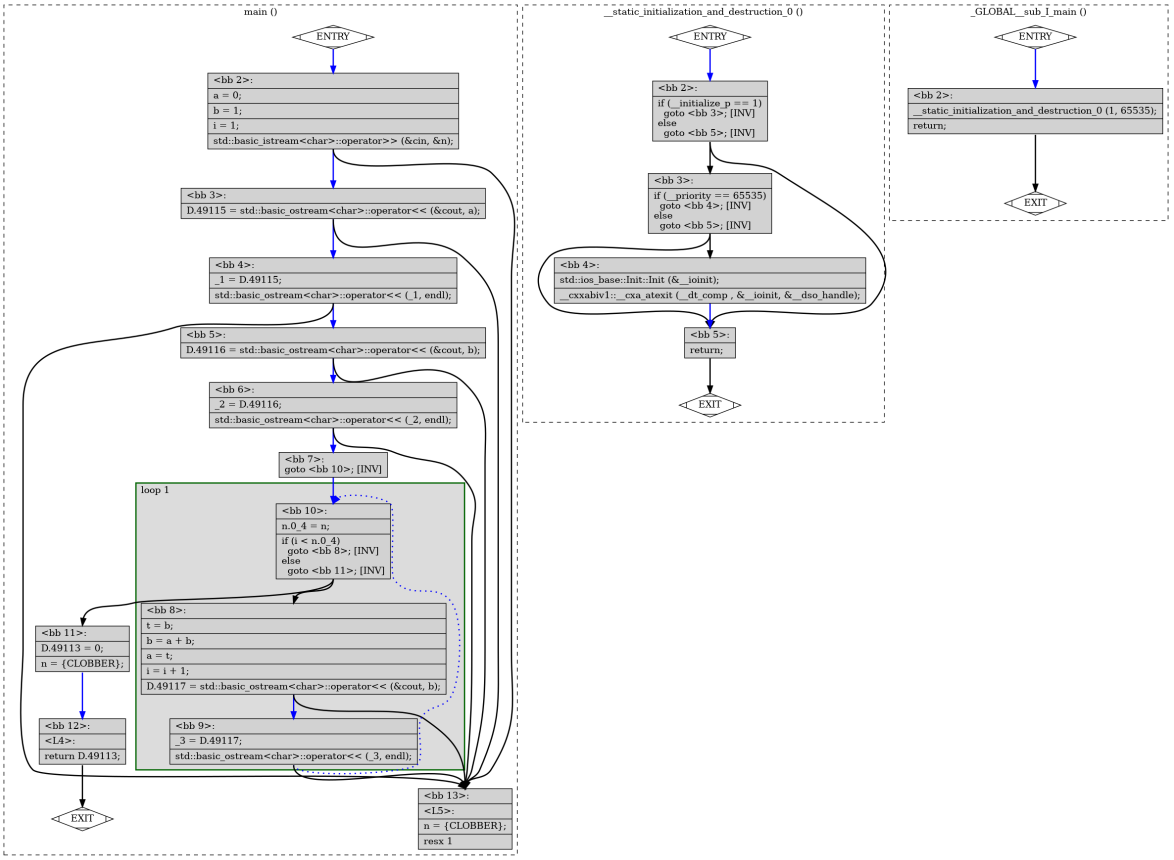
```
g++ -fdump-tree-all-graph -O0 -S source/fib_pure.cpp -o /dev/null
dot -Tpng fib_pure.cpp.*.cfg.dot -o compiler/4_ir_gen/fib_pure_O0_cfg.png
mkdir -p compiler/4_ir_gen/fib_pure_dumps
mv fib_pure.cpp.*.dot compiler/4_ir_gen/fib_pure_dumps/
```

const_pure

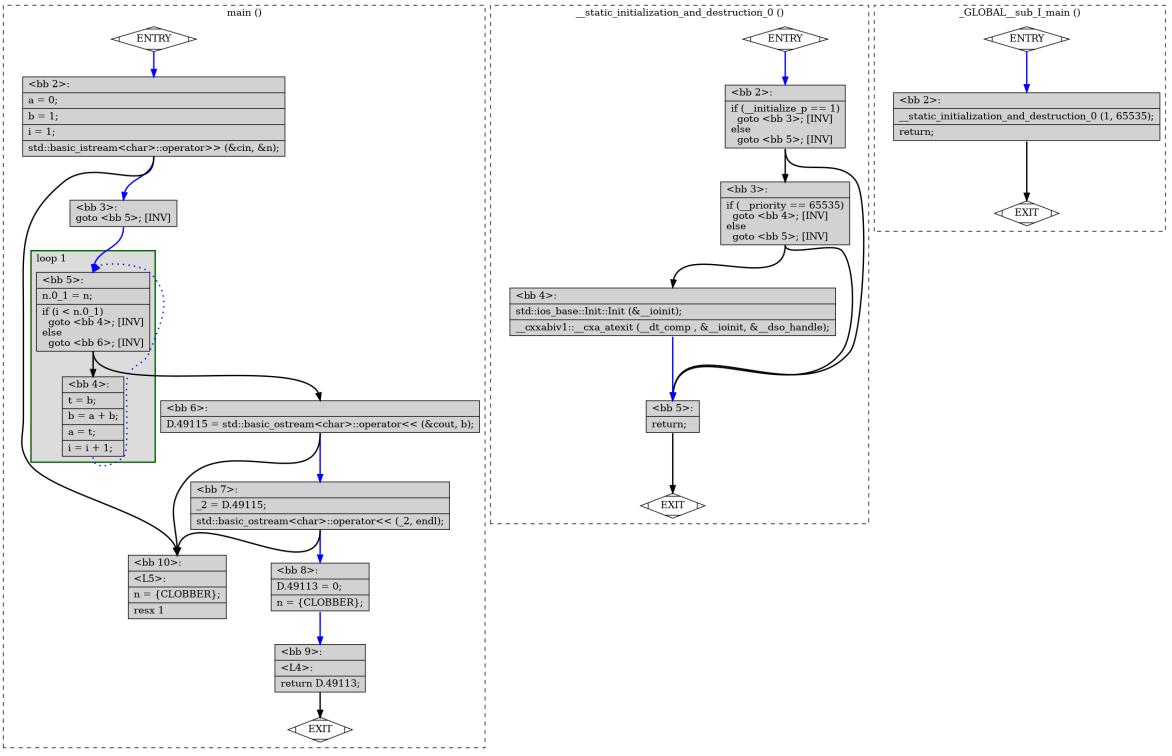
```
g++ -fdump-tree-all-graph -O0 -S source/fib_const_pure.cpp -o /dev/null
dot -Tpng fib_const_pure.cpp.*.cfg.dot -o
compiler/4_ir_gen/fib_const_pure_O0_cfg.png
mkdir -p compiler/4_ir_gen/fib_const_pure_dumps
mv fib_const_pure.cpp.*.dot compiler/4_ir_gen/fib_const_pure_dumps/
```

CFG

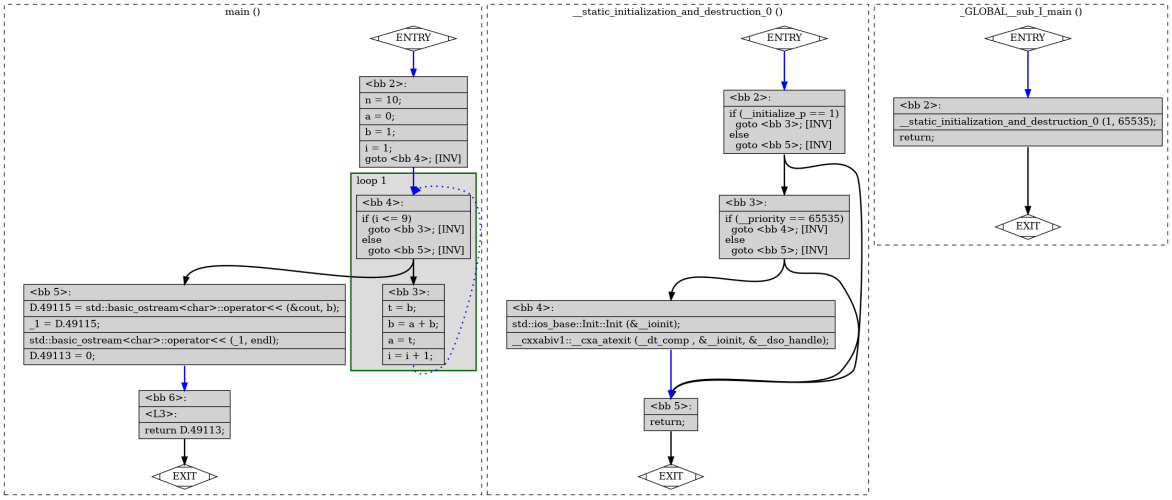
baseline



pure



const_pure



中间代码生成做了什么

中间代码生成阶段是一个**高保真**的翻译过程，它将前端AST的结构、类型和常量信息，几乎无损地转换成了线性的、平台无关的LLVM IR和其图形化表示CFG。这份未优化的中间表示，是源代码最底层的、最忠实的逻辑画像，它通过解耦前后端，为后续的大量、跨平台的优化工作提供了一个统一、规范的分析和转换平台。

代码优化

代码优化命令及对应gcc与llvm流程图生成命令

```
# 1. 对 fib_baseline_O0.ll 运行 mem2reg 和 dot-cfg-only
#    注意：这里使用了现代LLVM的 -passes=... 语法
opt -passes=mem2reg -dot-cfg-only
compiler/5_ir_opt/fib_baseline/fib_baseline_O0.ll > /dev/null

# 2. 将生成的 .main.dot 文件重命名并移动到正确位置
mv .main.dot compiler/5_ir_opt/fib_baseline/fib_baseline_O0_llvm.dot

# 3. 使用 dot 工具将 .dot 文件渲染成 PNG 图像
dot -Tpng compiler/5_ir_opt/fib_baseline/fib_baseline_O0_llvm.dot -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O0_llvm_cfg.png

# (可选) 打印成功信息
echo "Successfully generated CFG for fib_baseline_O0.ll"
```

```
# --- 创建 fib_pure 的专属文件夹 (如果还没创建的话) ---
mkdir -p compiler/5_ir_opt/fib_pure

# --- 生成 O0 的 .ll 文件 (如果还没生成的话) ---
```

```
clang++ -S -emit-llvm -O0 source/fib_pure.cpp -o
compiler/5_ir_opt/fib_pure/fib_pure_O0.ll

# 1. 对 fib_pure_O0.ll 运行 mem2reg 和 dot-cfg-only
opt -passes=mem2reg -dot-cfg-only compiler/5_ir_opt/fib_pure/fib_pure_O0.ll >
/dev/null

# 2. 重命名并移动 .main.dot
mv .main.dot compiler/5_ir_opt/fib_pure/fib_pure_O0_llvm.dot

# 3. 渲染成 PNG
dot -Tpng compiler/5_ir_opt/fib_pure/fib_pure_O0_llvm.dot -o
compiler/5_ir_opt/fib_pure/fib_pure_O0_llvm_cfg.png

# (可选) 打印成功信息
echo "Successfully generated CFG for fib_pure_O0.ll"
```

```
# --- 创建 fib_const_pure 的专属文件夹 (如果还没创建的话) ---
mkdir -p compiler/5_ir_opt/fib_const_pure

# --- 生成 O0 的 .ll 文件 (如果还没生成的话) ---
clang++ -S -emit-llvm -O0 source/fib_const_pure.cpp -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0.ll

# 1. 对 fib_const_pure_O0.ll 运行 mem2reg 和 dot-cfg-only
opt -passes=mem2reg -dot-cfg-only
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0.ll > /dev/null

# 2. 重命名并移动 .main.dot
mv .main.dot compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0_llvm.dot

# 3. 渲染成 PNG
dot -Tpng compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0_llvm.dot -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0_llvm_cfg.png

# (可选) 打印成功信息
echo "Successfully generated CFG for fib_const_pure_O0.ll"
```

```
# --- 创建 fib_baseline 的专属文件夹 ---
mkdir -p compiler/5_ir_opt/fib_baseline

# --- O0 ---
clang++ -S -emit-llvm -O0 source/fib_baseline.cpp -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O0.ll
```

```
opt -dot-cfg-only compiler/5_ir_opt/fib_baseline/fib_baseline_O0.ll > /dev/null
&& mv .main.dot compiler/5_ir_opt/fib_baseline/fib_baseline_O0_llvm.dot && dot -
Tpng compiler/5_ir_opt/fib_baseline/fib_baseline_O0_llvm.dot -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O0_llvm_cfg.png
g++ -fdump-tree-all-graph -O0 -S source/fib_baseline.cpp -o /dev/null && mv
fib_baseline.cpp.*.cfg.dot
compiler/5_ir_opt/fib_baseline/fib_baseline_O0_gcc.dot && dot -Tpng
compiler/5_ir_opt/fib_baseline/fib_baseline_O0_gcc.dot -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O0_gcc_cfg.png && rm
fib_baseline.cpp.*.dot

# --- O1 ---
clang++ -S -emit-llvm -O1 source/fib_baseline.cpp -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O1.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_baseline/fib_baseline_O1.ll > /dev/null
&& mv .main.dot compiler/5_ir_opt/fib_baseline/fib_baseline_O1_llvm.dot && dot -
Tpng compiler/5_ir_opt/fib_baseline/fib_baseline_O1_llvm.dot -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O1_llvm_cfg.png
g++ -fdump-tree-all-graph -O1 -S source/fib_baseline.cpp -o /dev/null && mv
fib_baseline.cpp.*.cfg.dot
compiler/5_ir_opt/fib_baseline/fib_baseline_O1_gcc.dot && dot -Tpng
compiler/5_ir_opt/fib_baseline/fib_baseline_O1_gcc.dot -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O1_gcc_cfg.png && rm
fib_baseline.cpp.*.dot

# --- O2 ---
clang++ -S -emit-llvm -O2 source/fib_baseline.cpp -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O2.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_baseline/fib_baseline_O2.ll > /dev/null
&& mv .main.dot compiler/5_ir_opt/fib_baseline/fib_baseline_O2_llvm.dot && dot -
Tpng compiler/5_ir_opt/fib_baseline/fib_baseline_O2_llvm.dot -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O2_llvm_cfg.png
g++ -fdump-tree-all-graph -O2 -S source/fib_baseline.cpp -o /dev/null && mv
fib_baseline.cpp.*.cfg.dot
compiler/5_ir_opt/fib_baseline/fib_baseline_O2_gcc.dot && dot -Tpng
compiler/5_ir_opt/fib_baseline/fib_baseline_O2_gcc.dot -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O2_gcc_cfg.png && rm
fib_baseline.cpp.*.dot

# --- O3 ---
clang++ -S -emit-llvm -O3 source/fib_baseline.cpp -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O3.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_baseline/fib_baseline_O3.ll > /dev/null
&& mv .main.dot compiler/5_ir_opt/fib_baseline/fib_baseline_O3_llvm.dot && dot -
Tpng compiler/5_ir_opt/fib_baseline/fib_baseline_O3_llvm.dot -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O3_llvm_cfg.png
g++ -fdump-tree-all-graph -O3 -S source/fib_baseline.cpp -o /dev/null && mv
fib_baseline.cpp.*.cfg.dot
```

```
compiler/5_ir_opt/fib_baseline/fib_baseline_O3_gcc.dot && dot -Tpng
compiler/5_ir_opt/fib_baseline/fib_baseline_O3_gcc.dot -o
compiler/5_ir_opt/fib_baseline/fib_baseline_O3_gcc_cfg.png && rm
fib_baseline.cpp.*.dot
```

```
# --- 创建 fib_pure 的专属文件夹 ---
mkdir -p compiler/5_ir_opt/fib_pure

# --- O0 ---
clang++ -S -emit-llvm -O0 source/fib_pure.cpp -o
compiler/5_ir_opt/fib_pure/fib_pure_O0.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_pure/fib_pure_O0.ll > /dev/null && mv
.main.dot compiler/5_ir_opt/fib_pure/fib_pure_O0_llvm.dot && dot -Tpng
compiler/5_ir_opt/fib_pure/fib_pure_O0_llvm.dot -o
compiler/5_ir_opt/fib_pure/fib_pure_O0_llvm_cfg.png
g++ -fdump-tree-all-graph -O0 -S source/fib_pure.cpp -o /dev/null && mv
fib_pure.cpp.*.cfg.dot compiler/5_ir_opt/fib_pure/fib_pure_O0_gcc.dot && dot -
Tpng compiler/5_ir_opt/fib_pure/fib_pure_O0_gcc.dot -o
compiler/5_ir_opt/fib_pure/fib_pure_O0_gcc_cfg.png && rm fib_pure.cpp.*.dot

# --- O1 ---
clang++ -S -emit-llvm -O1 source/fib_pure.cpp -o
compiler/5_ir_opt/fib_pure/fib_pure_O1.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_pure/fib_pure_O1.ll > /dev/null && mv
.main.dot compiler/5_ir_opt/fib_pure/fib_pure_O1_llvm.dot && dot -Tpng
compiler/5_ir_opt/fib_pure/fib_pure_O1_llvm.dot -o
compiler/5_ir_opt/fib_pure/fib_pure_O1_llvm_cfg.png
g++ -fdump-tree-all-graph -O1 -S source/fib_pure.cpp -o /dev/null && mv
fib_pure.cpp.*.cfg.dot compiler/5_ir_opt/fib_pure/fib_pure_O1_gcc.dot && dot -
Tpng compiler/5_ir_opt/fib_pure/fib_pure_O1_gcc.dot -o
compiler/5_ir_opt/fib_pure/fib_pure_O1_gcc_cfg.png && rm fib_pure.cpp.*.dot

# --- O2 ---
clang++ -S -emit-llvm -O2 source/fib_pure.cpp -o
compiler/5_ir_opt/fib_pure/fib_pure_O2.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_pure/fib_pure_O2.ll > /dev/null && mv
.main.dot compiler/5_ir_opt/fib_pure/fib_pure_O2_llvm.dot && dot -Tpng
compiler/5_ir_opt/fib_pure/fib_pure_O2_llvm.dot -o
compiler/5_ir_opt/fib_pure/fib_pure_O2_llvm_cfg.png
g++ -fdump-tree-all-graph -O2 -S source/fib_pure.cpp -o /dev/null && mv
fib_pure.cpp.*.cfg.dot compiler/5_ir_opt/fib_pure/fib_pure_O2_gcc.dot && dot -
Tpng compiler/5_ir_opt/fib_pure/fib_pure_O2_gcc.dot -o
compiler/5_ir_opt/fib_pure/fib_pure_O2_gcc_cfg.png && rm fib_pure.cpp.*.dot

# --- O3 ---
clang++ -S -emit-llvm -O3 source/fib_pure.cpp -o
```

```
compiler/5_ir_opt/fib_pure/fib_pure_O3.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_pure/fib_pure_O3.ll > /dev/null && mv
.main.dot compiler/5_ir_opt/fib_pure/fib_pure_O3_llvm.dot && dot -Tpng
compiler/5_ir_opt/fib_pure/fib_pure_O3_llvm.dot -o
compiler/5_ir_opt/fib_pure/fib_pure_O3_llvm_cfg.png
g++ -fdump-tree-all-graph -O3 -S source/fib_pure.cpp -o /dev/null && mv
fib_pure.cpp.*.cfg.dot compiler/5_ir_opt/fib_pure/fib_pure_O3_gcc.dot && dot -
Tpng compiler/5_ir_opt/fib_pure/fib_pure_O3_gcc.dot -o
compiler/5_ir_opt/fib_pure/fib_pure_O3_gcc_cfg.png && rm fib_pure.cpp.*.dot
```

```
# --- 创建 fib_const_pure 的专属文件夹 ---
mkdir -p compiler/5_ir_opt/fib_const_pure

# --- O0 ---
clang++ -S -emit-llvm -O0 source/fib_const_pure.cpp -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0.ll >
/dev/null && mv .main.dot
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0_llvm.dot && dot -Tpng
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0_llvm.dot -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0_llvm_cfg.png
g++ -fdump-tree-all-graph -O0 -S source/fib_const_pure.cpp -o /dev/null && mv
fib_const_pure.cpp.*.cfg.dot
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0_gcc.dot && dot -Tpng
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0_gcc.dot -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O0_gcc_cfg.png && rm
fib_const_pure.cpp.*.dot

# --- O1 ---
clang++ -S -emit-llvm -O1 source/fib_const_pure.cpp -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O1.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_const_pure/fib_const_pure_O1.ll >
/dev/null && mv .main.dot
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O1_llvm.dot && dot -Tpng
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O1_llvm.dot -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O1_llvm_cfg.png
g++ -fdump-tree-all-graph -O1 -S source/fib_const_pure.cpp -o /dev/null && mv
fib_const_pure.cpp.*.cfg.dot
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O1_gcc.dot && dot -Tpng
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O1_gcc.dot -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O1_gcc_cfg.png && rm
fib_const_pure.cpp.*.dot

# --- O2 ---
clang++ -S -emit-llvm -O2 source/fib_const_pure.cpp -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O2.ll
```

```
opt -dot-cfg-only compiler/5_ir_opt/fib_const_pure/fib_const_pure_O2.ll >
/dev/null && mv .main.dot
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O2_llvm.dot && dot -Tpng
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O2_llvm.dot -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O2_llvm_cfg.png
g++ -fdump-tree-all-graph -O2 -S source/fib_const_pure.cpp -o /dev/null && mv
fib_const_pure.cpp.*.cfg.dot
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O2_gcc.dot && dot -Tpng
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O2_gcc.dot -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O2_gcc_cfg.png && rm
fib_const_pure.cpp.*.dot

# --- O3 ---
clang++ -S -emit-llvm -O3 source/fib_const_pure.cpp -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O3.ll
opt -dot-cfg-only compiler/5_ir_opt/fib_const_pure/fib_const_pure_O3.ll >
/dev/null && mv .main.dot
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O3_llvm.dot && dot -Tpng
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O3_llvm.dot -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O3_llvm_cfg.png
g++ -fdump-tree-all-graph -O3 -S source/fib_const_pure.cpp -o /dev/null && mv
fib_const_pure.cpp.*.cfg.dot
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O3_gcc.dot && dot -Tpng
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O3_gcc.dot -o
compiler/5_ir_opt/fib_const_pure/fib_const_pure_O3_gcc_cfg.png && rm
fib_const_pure.cpp.*.dot
```

目标代码生成

不同优化下目标代码生成命令

```


# --- 处理 fib_baseline.cpp ---
# 为 fib_baseline 创建一个主目录
mkdir -p compiler/6_target_code/fib_baseline

# 循环处理 O0, O1, O2, O3 四个优化等级
for level in 0 1 2 3
do
  # 1. 为当前优化等级创建专属的归档目录
  mkdir -p compiler/6_target_code/fib_baseline/O${level}_dumps

  # 2. 生成汇编文件
  riscv64-unknown-elf-g++ -S -O${level} source/fib_baseline.cpp -o
compiler/6_target_code/fib_baseline/fib_baseline_O${level}.s

  # 3. 运行GCC，让它在当前目录 dump 出所有 .dot 文件
```

```
  riscv64-unknown-elf-g++ -fdump-tree-all-graph -O${level} -S
source/fib_baseline.cpp -o /dev/null

  # 4. 从中挑选出我们最关心的 CFG 图，并渲染成 PNG
  # 我们挑选文件名中带有 .cfg.dot 的文件
  dot -Tpng fib_baseline.cpp.*.cfg.dot -o
compiler/6_target_code/fib_baseline/fib_baseline_O${level}_cfg.png 2>/dev/null

  # 5. 把所有生成的 .dot 文件归档到专属目录中
  mv fib_baseline.cpp.*.dot compiler/6_target_code/fib_baseline/O${level}_dumps/

  echo "fib_baseline @ -O${level} processed."
done
```

```
# --- 处理 fib_pure.cpp ---
mkdir -p compiler/6_target_code/fib_pure
for level in 0 1 2 3
do
  mkdir -p compiler/6_target_code/fib_pure/O${level}_dumps
  riscv64-unknown-elf-g++ -S -O${level} source/fib_pure.cpp -o
compiler/6_target_code/fib_pure/fib_pure_O${level}.s
  riscv64-unknown-elf-g++ -fdump-tree-all-graph -O${level} -S
source/fib_pure.cpp -o /dev/null
  dot -Tpng fib_pure.cpp.*.cfg.dot -o
compiler/6_target_code/fib_pure/fib_pure_O${level}_cfg.png 2>/dev/null
  mv fib_pure.cpp.*.dot compiler/6_target_code/fib_pure/O${level}_dumps/
  echo "fib_pure @ -O${level} processed."
done
```

```
# --- 处理 fib_const_pure.cpp ---
mkdir -p compiler/6_target_code/fib_const_pure
for level in 0 1 2 3
do
  mkdir -p compiler/6_target_code/fib_const_pure/O${level}_dumps
  riscv64-unknown-elf-g++ -S -O${level} source/fib_const_pure.cpp -o
compiler/6_target_code/fib_const_pure/fib_const_pure_O${level}.s
  riscv64-unknown-elf-g++ -fdump-tree-all-graph -O${level} -S
source/fib_const_pure.cpp -o /dev/null
  dot -Tpng fib_const_pure.cpp.*.cfg.dot -o
compiler/6_target_code/fib_const_pure/fib_const_pure_O${level}_cfg.png
2>/dev/null
  mv fib_const_pure.cpp.*.dot
compiler/6_target_code/fib_const_pure/O${level}_dumps/
```

```
  echo "fib_const_pure @ -O${level} processed."
done
```

汇编器

```
汇编
riscv64-unknown-elf-g++ -c compiler/6_target_code/fib_baseline/fib_baseline_O0.s
-o assembler/fib_baseline_O0.o
反汇编
riscv64-unknown-elf-objdump -d assembler/fib_baseline_O0.o
符号表
riscv64-unknown-elf-nm assembler/fib_baseline_O0.o
```

- **汇编器的工作：** 将汇编助记符翻译成二进制机器码，并创建符号表和重定位表。
- **.o 文件的本质：** 一个**可重定位目标文件**，它包含了已翻译的机器码，但也明确记录了对外部符号的依赖（通过符号表），以及哪些指令的地址需要被修正（通过重定位表）。
- **不能运行的原因：** 其**外部依赖**尚未被满足，其内部的**函数调用地址**尚未被修正。

链接器

```
链接
riscv64-unknown-elf-g++ assembler/fib_baseline_O0.o -o
linker/fib_baseline_O0_exec
# 对比 .o 文件和可执行文件的大小
ls -lh assembler/fib_baseline_O0.o linker/fib_baseline_O0_exec
# 查看可执行文件的符号表
riscv64-unknown-elf-nm linker/fib_baseline_O0_exec
# 反汇编可执行文件
riscv64-unknown-elf-objdump -d linker/fib_baseline_O0_exec
# 执行静态链接
riscv64-unknown-elf-g++ assembler/fib_baseline_O0.o -o
linker/fib_baseline_O0_static_exec -static
# 对比动态链接和静态链接的可执行文件大小
ls -lh linker/fib_baseline_O0_exec linker/fib_baseline_O0_static_exec
# 运行动态链接的版本
qemu-riscv64 linker/fib_baseline_O0_exec
# 运行静态链接的版本
qemu-riscv64 linker/fib_baseline_O0_static_exec
```