IR自动生成
实验讲解

讲解人 课程助教 肖同欢

国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2025年10月09日

内容

- C++ 知识回顾（Dynamic_cast 和 Static_cast）
- AST
- 访问者模式
- Lab2 实验内容以及提交方式

C++ 知识回顾

编译原理课程组

中国科学技术大学

C++类知识回顾

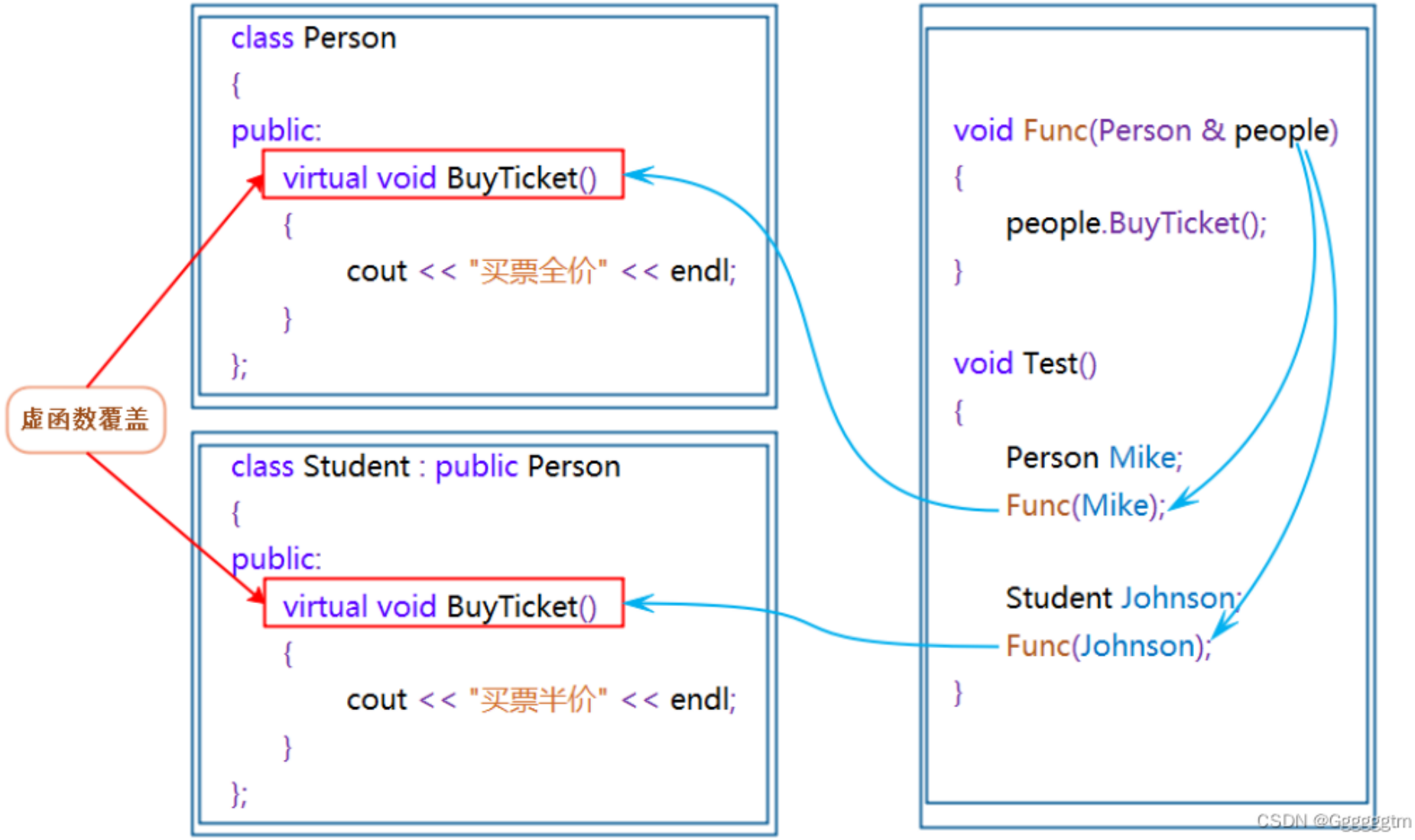
Class是 C++ 面向对象的基础，相当于对 C 中的结构体的扩展

除了保留了原来的结构体成员（即成员对象），增加了成员函数、访问控制、继承和多态等

使用继承，多态等特征，经常设计的操作是在父类指针和子类指针之间的转换，这里需要涉及到 dynamic_cast 和 static_cast

C++虚函数和多态

多态：同一个方法（接口）调用，由于对象不同可能会有不同的行为。



static_cast

- **static_cast是一个强制类型转换操作符，用于**
  - 不同变量类型之间的转换，例如short转int、int转double等
  - void指针和具体类型指针之间的转换，例如void *转int *、char *转void *等
  - 有转换构造函数或者类型转换函数的类与其它类型之间的转换

- **static_cast不能用于无关类型之间的转换**
  - int *转double *、Inst *转int *等
  - int和指针之间的转换

类型转换举例

```
int main(){
        int m = 0xffd7;
        long long n = static_cast<long long>(m);                    // 宽转换，没有信息丢失
        char ch = static_cast<char>(m);                             // 窄转换，可能会丢失信息
        int *p1 = static_cast<int*>(malloc(10 * sizeof(int)));      // 将void指针转换为具体类型指针
        void *p2 = static_cast<void*>(p1);                          // 将具体类型指针转换为void指针
        // 下面的用法是错误的
        // float *p3 = static_cast<float*>(p1);                     // 不能在两个具体类型的指针之间转换
        // float * p34 = static_cast<float*>(100);                  // 不能将整数转换为指针类型
        cout << m << endl;                                          // 65335
        cout << ch << endl;                                         // 7
        cout << p1 << endl;                                         // 0x5614372972b0
        cout << p2 << endl;                                         // 0x5614372972b0
        return 0;
}
```

dynamic_cast

- **和static_cast一样，dynamic_cast用于类型的转换**
- **功能**
  - 父类转子类（向下转型）
  - 子类转父类（向上转型）
- **语法形式**

  dynamic_cast < new-type > (expression)
- **注意事项**
  - new-type和expression必须同时是指针类型或者引用类型，即dynamic_cast只能转换指针类型和引用类型。
  - 对于指针类型，如果转换失败将返回NULL
  - 对于引用，如果转换失败将抛出std::bad_cast异常

dynamic_cast 向上转型

向上转型时，只要待转换的两个类型之间存在继承关系，并且基类包含了虚函数就一定能转换成功

- 示例

```
Derived *down = new Derived();
Base *up = dynamic_cast<Base*>(down);
```

派生类指针down被向上转型为基类指针并赋值给up

dynamic_cast 向下转型

- **向下转型是有风险的**
- **安全的向下转型**
  - 声明为基类的指针实际指向的是派生类对象，这时就可以将该指针向下转型为派生类指针，如果不是的话，就会返回一个空指针
- **示例**

```
Base *up= new Derived();
Derived *down = dynamic_cast<Derived*>(up);
```

基类指针up被向下转型为派生类指针并赋值给down
安全的原因是up实际指向的是派生类对象

AST

编译原理课程组

中国科学技术大学

语法分析树

- 右图是一颗语法分析树，详细的反应了程序的情况和语法规则是怎么样一步一步推导出具体的程序的，这部分是之前上课在讲的词法、语法分析的部分。
- 冗余部分太多！

```
int main(void) {
    int a;
    a = 1234;
    return a;
}
```

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
|  |  |  |  |  >--* {
|  |  |  |  |  >--+ local-declarations
|  |  |  |  |  |  >--+ local-declarations
|  |  |  |  |  |  |  >--* epsilon
|  |  |  |  |  |  >--+ var-declaration
|  |  |  |  |  |  |  >--+ type-specifier
|  |  |  |  |  |  |  |  >--* int
|  |  |  |  |  |  |  >--* a
|  |  |  |  |  |  |  >--* ;
......
```

AST

- 遍历这棵语法分析树，折叠非必要属性，得到一棵抽象语法树(AST)

```
int main(void) {
    int a;
    a = 1234;
    return a;
}
```

```
program
--fun-declaration: main
----compound-stmt
------var-declaration: a
```

AST

- AST（抽象语法树，Abstract Syntax Tree） 是编程语言中的一种树状数据结构，用于表示源代码的语法结构。
- 抽象语法树，每个结点的信息更丰富

```
struct _syntax_tree_node {
    struct _syntax_tree_node *parent;
    struct _syntax_tree_node *children[10];
    int children_num;

    char name[SYNTAX_TREE_NODE_NAME_MAX];
};
typedef struct _syntax_tree_node syntax_tree_node;

struct _syntax_tree {
    syntax_tree_node *root;
};
typedef struct _syntax_tree syntax_tree;
```

→

```
class AST {
  public:
    AST() = delete;
    AST(syntax_tree *);
    AST(AST &&tree) {
        root = tree.root;
        tree.root = nullptr;
    };
    ASTProgram *get_root() { return root.get(); }
    void run_visitor(ASTVisitor &visitor);

  private:
    ASTNode *transform_node_iter(syntax_tree_node *);
    std::shared_ptr<ASTProgram> root = nullptr;
};
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```
program
--fun-declaration: main
----compound-stmt
...
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```
struct ASTProgram : ASTNode {
    virtual Value* accept(ASTVisitor &) override final;
    virtual ~ASTProgram() = default;
    std::vector<std::shared_ptr<ASTDeclaration>> declarations;
};
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```
struct ASTProgram : ASTNode {
    virtual Value* accept(ASTVisitor &) override final;
    virtual ~ASTProgram() = default;
    std::vector<std::shared_ptr<ASTDeclaration>> declarations;
};
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```
struct ASTDeclaration : ASTNode {
    virtual ~ASTDeclaration() = default;
    CminusType type;
    std::string id;
};

struct ASTVarDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::shared_ptr<ASTNum> num;
};

struct ASTFunDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::vector<std::shared_ptr<ASTParam>> params;
    std::shared_ptr<ASTCompoundStmt> compound_stmt;
};
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```
struct ASTDeclaration : ASTNode {
    virtual ~ASTDeclaration() = default;
    CminusType type;
    std::string id;
};

struct ASTVarDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::shared_ptr<ASTNum> num;
};

struct ASTFunDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::vector<std::shared_ptr<ASTParam>> params;
    std::shared_ptr<ASTCompoundStmt> compound_stmt;
};
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```
struct ASTDeclaration : ASTNode {
    virtual ~ASTDeclaration() = default;
    CminusType type;
    std::string id;
};

struct ASTVarDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::shared_ptr<ASTNum> num;
};

struct ASTFunDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::vector<std::shared_ptr<ASTParam>> params;
    std::shared_ptr<ASTCompoundStmt> compound_stmt;
};
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```cpp
struct ASTDeclaration : ASTNode {
    virtual ~ASTDeclaration() = default;
    CminusType type;
    std::string id;
};

struct ASTVarDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::shared_ptr<ASTNum> num;
};

struct ASTFunDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::vector<std::shared_ptr<ASTParam>> params;
    std::shared_ptr<ASTCompoundStmt> compound_stmt;
};
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```
struct ASTDeclaration : ASTNode {
    virtual ~ASTDeclaration() = default;
    CminusType type;
    std::string id;
};

struct ASTVarDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::shared_ptr<ASTNum> num;
};

struct ASTFunDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::vector<std::shared_ptr<ASTParam>> params;
    std::shared_ptr<ASTCompoundStmt> compound_stmt;
};
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```cpp
struct ASTDeclaration : ASTNode {
    virtual ~ASTDeclaration() = default;
    CminusType type;
    std::string id;
};

struct ASTVarDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::shared_ptr<ASTNum> num;
};

struct ASTFunDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::vector<std::shared_ptr<ASTParam>> params;
    std::shared_ptr<ASTCompoundStmt> compound_stmt;
};
```

例子

```
>--+ program
|  >--+ declaration-list
|  |  >--+ declaration
|  |  |  >--+ fun-declaration
|  |  |  |  >--+ type-specifier
|  |  |  |  |  >--* int
|  |  |  |  >--* main
|  |  |  |  >--* (
|  |  |  |  >--+ params
|  |  |  |  |  >--* void
|  |  |  |  >--* )
|  |  |  |  >--+ compound-stmt
...
```

```cpp
struct ASTDeclaration : ASTNode {
    virtual ~ASTDeclaration() = default;
    CminusType type;
    std::string id;
};

struct ASTVarDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::shared_ptr<ASTNum> num;
};

struct ASTFunDeclaration : ASTDeclaration {
    virtual Value* accept(ASTVisitor &) override final;
    std::vector<std::shared_ptr<ASTParam>> params;
    std::shared_ptr<ASTCompoundStmt> compound_stmt;
};
```

访问者模式

编译原理课程组

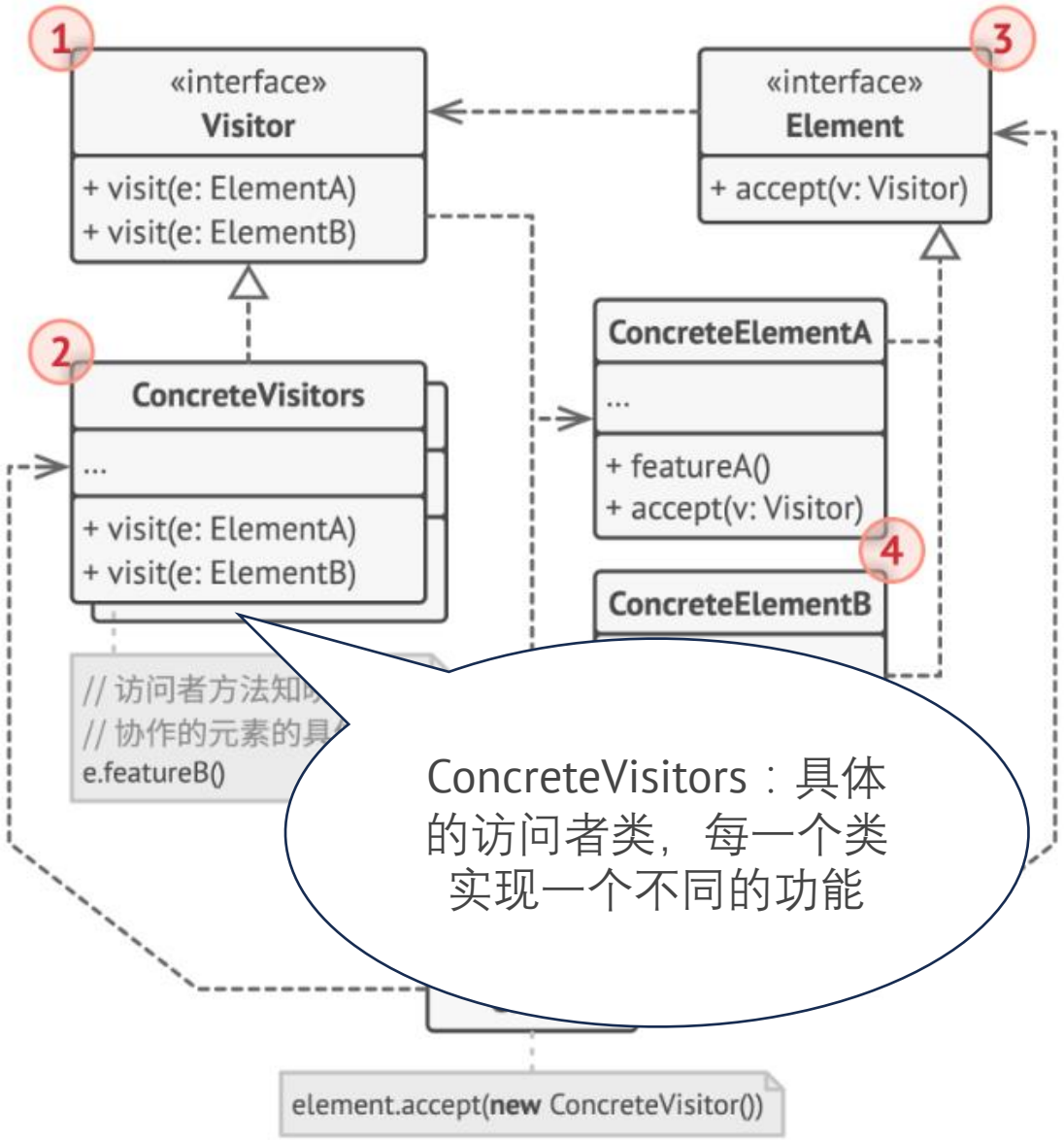
中国科学技术大学

访问者模式

Visitor：接口（抽象类）声明了一系列以对象结构为参数的访问者方法。

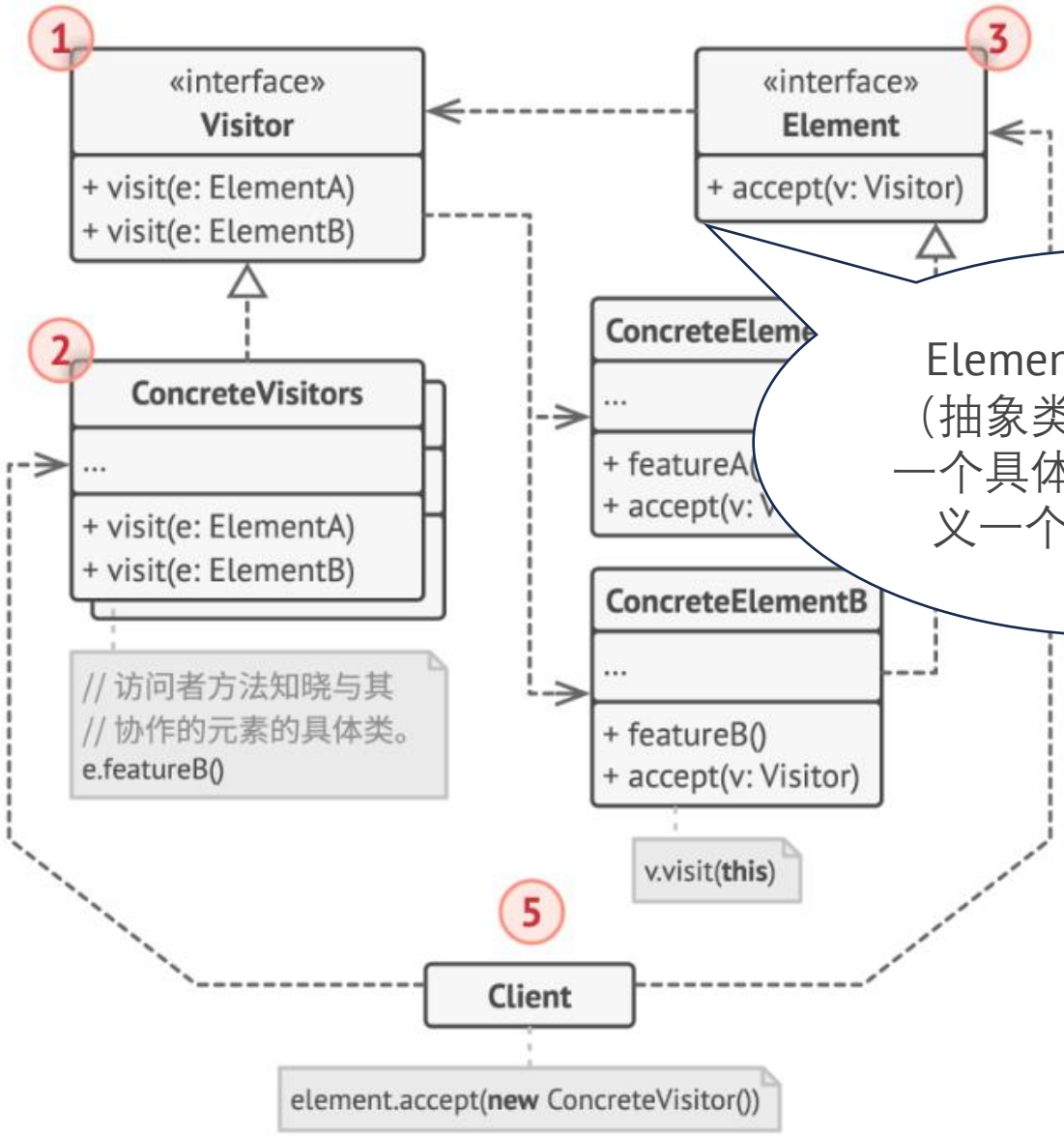
```
class ConcreteComponentA;
class ConcreteComponentB;

class Visitor {
 public:
  virtual void VisitConcreteComponentA(const ConcreteComponentA *element) const = 0;
  virtual void VisitConcreteComponentB(const ConcreteComponentB *element) const = 0;
};
```

访问者模式




```
class ConcreteVisitor1 : public Visitor {
 public:
  void VisitConcreteComponentA(const ConcreteComponentA *element) const override {
    std::cout << element->ExclusiveMethodOfConcreteComponentA() << " + ConcreteVisitor1\
  }

  void VisitConcreteComponentB(const ConcreteComponentB *element) const override {
    std::cout << element->SpecialMethodOfConcreteComponentB() << " + ConcreteVisitor1\n"
  }
};

class ConcreteVisitor2 : public Visitor {
 public:
  void VisitConcreteComponentA(const ConcreteComponentA *element) const override {
    std::cout << element->ExclusiveMethodOfConcreteComponentA() << " + ConcreteVisitor2\
  }
  void VisitConcreteComponentB(const ConcreteComponentB *element) const override {
    std::cout << element->SpecialMethodOfConcreteComponentB() << " + ConcreteVisitor2\n"
  }
};
```

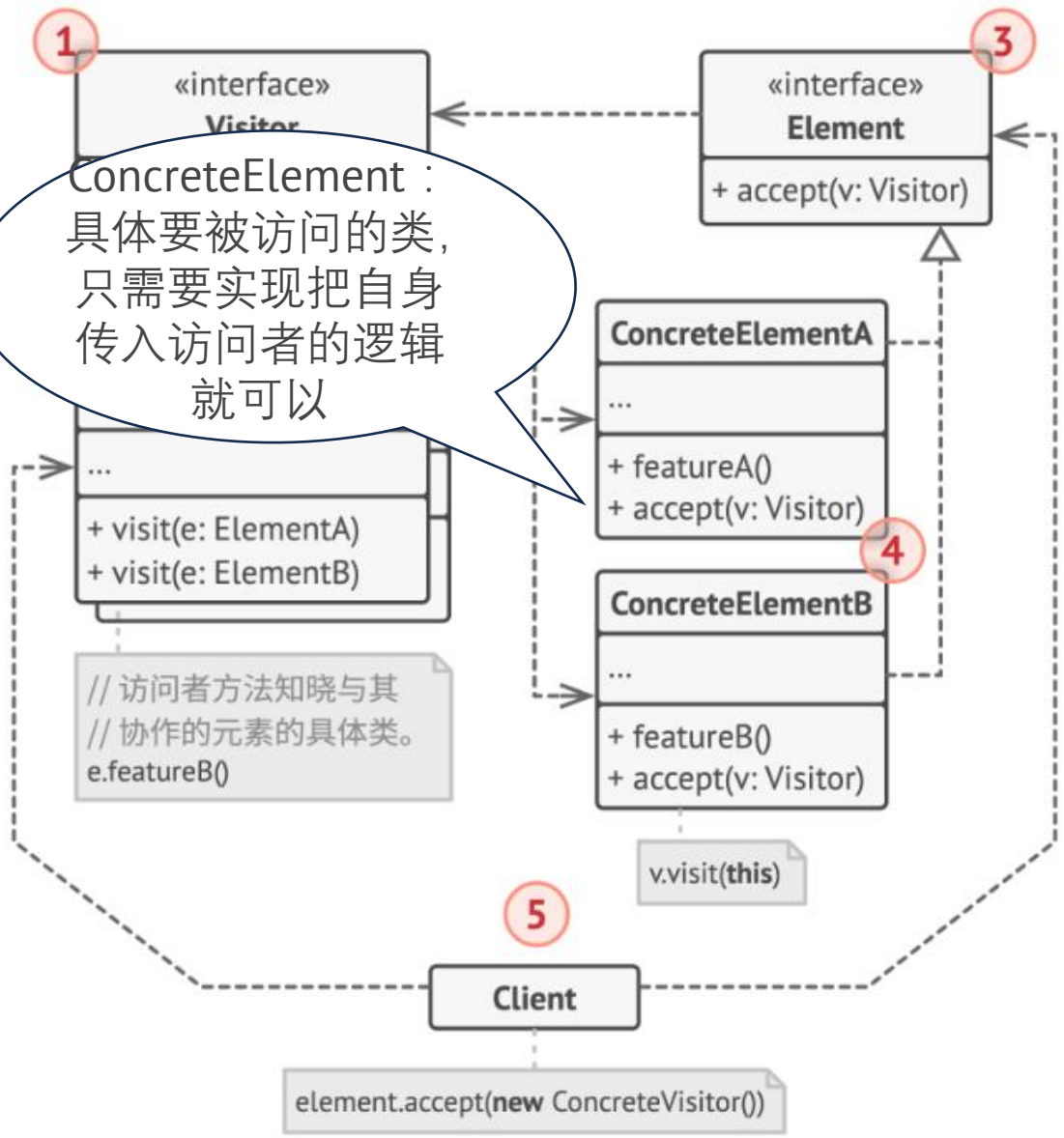
访问者模式



Element：一个接口（抽象类），用来给每一个具体的数据结构定义一个 accept 接口

```
/**
 * The Component interface declares an `accept` method that should take the base
 * visitor interface as an argument.
 */

class Component {
 public:
  virtual ~Component() {}
  virtual void Accept(Visitor *visitor) const = 0;
};
```

访问者模式




```
class ConcreteComponentA : public Component {
  /**
   * Note that we're calling `visitConcreteComponentA`, which matches the
   * current class name. This way we let the visitor know the class of the
   * component it works with.
   */
 public:
  void Accept(Visitor *visitor) const override {
    visitor->VisitConcreteComponentA(this);
  }
  /**
   * Concrete Components may have special methods that don't exist in their base
   * class or interface. The Visitor is still able to use these methods since
   * it's aware of the component's concrete class.
   */
  std::string ExclusiveMethodOfConcreteComponentA() const {
    return "A";
  }
};
```

IR 自动化生成框架总览

- **接下来介绍框架 CminusfBuilder 类通过访问者模式遍历AST，调用 Light IR C++ 库自动化的生成 IR**
- **Visitor Pattern (访问者模式) 概念**
  - AST 类有一个方法接受访问者，将自身引用传入访问者，而访问者类中集成了对不同 AST 节点的访问规则

访问者模式示例

```
//Visitor.h
class AddExp;
class IntExp;
class Visitor
{
public:
    int result = 0;
    virtual void visit(AddExp*);
    virtual void visit(IntExp*);
};
```

```
//Exp.h
#include  "Visitor.h"
class Exp{
public:
    virtual void accept(Visitor&v) = 0;};

class AddExp : public Exp{
public:
    Exp* rhs;
    Exp* lhs;
    AddExp(Exp* lhs, Exp* rhs) : lhs(lhs), rhs(rhs) {}
    virtual void accept(Visitor&v) override final;};

class IntExp : public Exp{
public:
    int value;
    IntExp(int value) : value(value) {}
    virtual void accept(Visitor&v) override final;};
```

访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

```
//main.cpp
#include<iostream>
#include<string>
#include "Exp.h"
using namespace std;

int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```

访问者模式示例

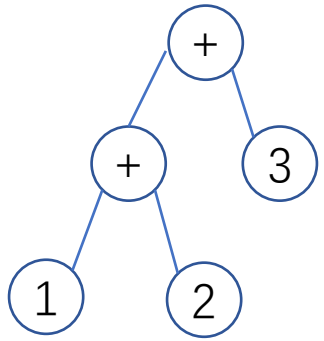
```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```

访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

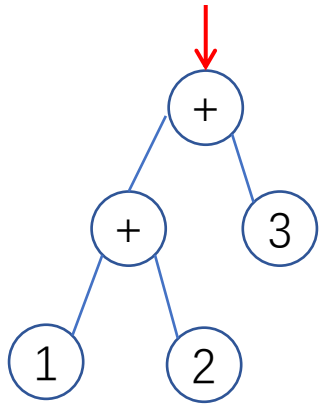
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

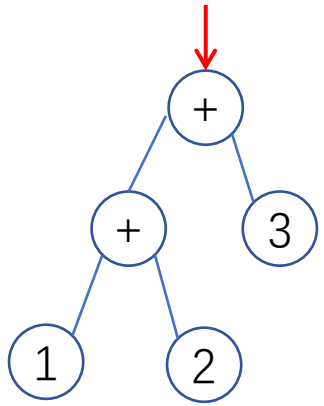
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

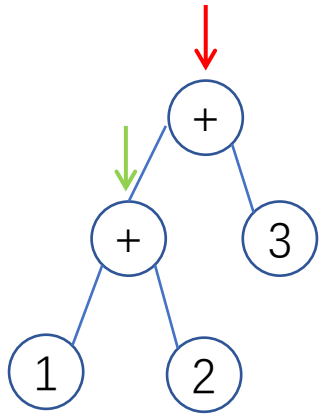
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

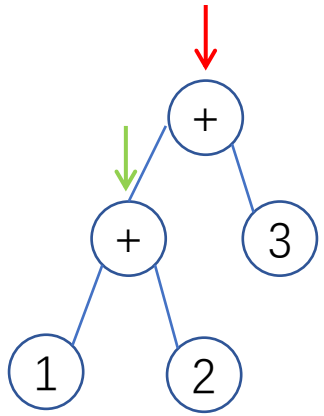
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

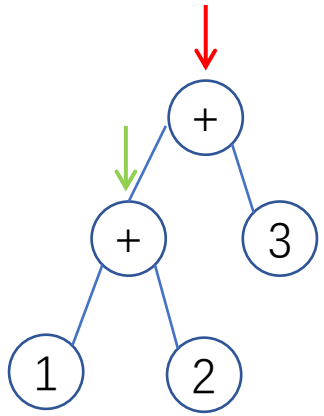
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

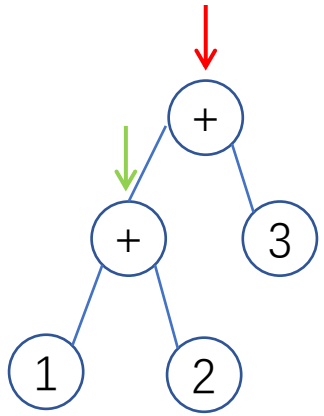
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

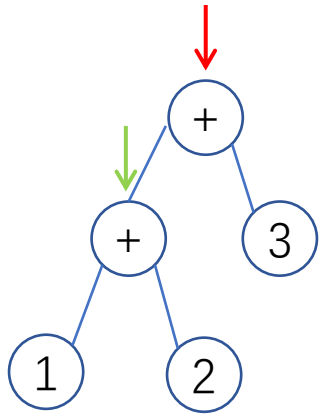
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

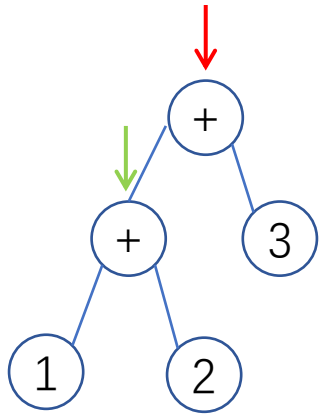
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

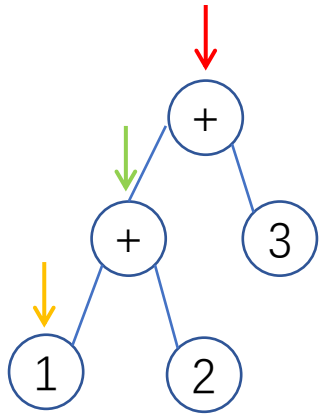
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

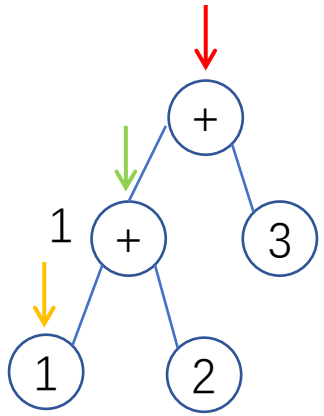
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

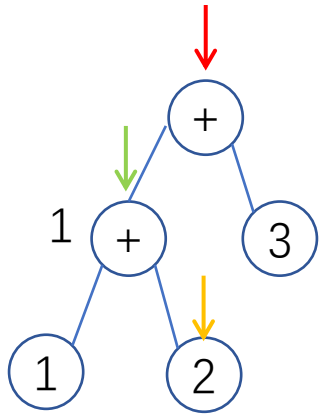
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

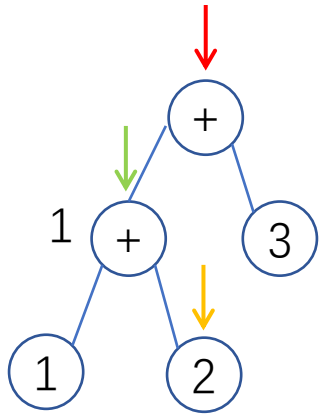
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

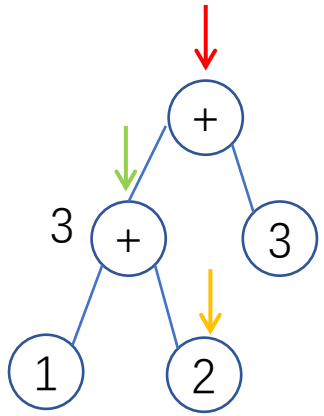
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

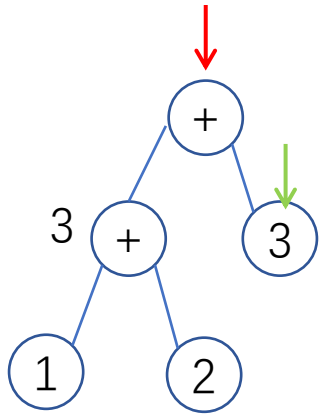
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

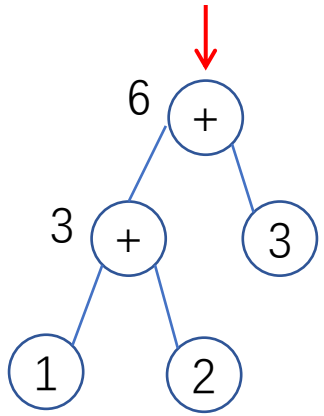
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

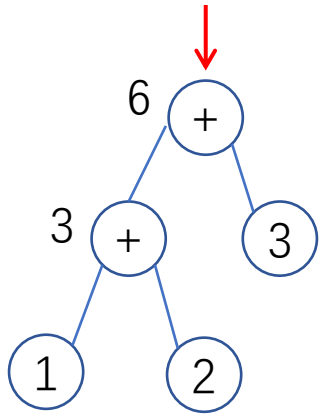
```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



访问者模式示例

```
//Visitor.cpp
#include "Exp.h"
void Visitor::visit(AddExp* add_exp){
    add_exp->lhs->accept(*this);
    add_exp->rhs->accept(*this);}

void Visitor::visit(IntExp* int_exp){
    result += int_exp->value;}

void AddExp::accept(Visitor & v){
    v.visit(this);}

void IntExp::accept(Visitor & v){
    v.visit(this);}
```

```
...
int main(){
    Exp* exp = new IntExp(1);
    for(int i = 2; i < 4; i++){
        exp = new AddExp(exp, new IntExp(i));    }

    Visitor CalSum;
    exp->accept(CalSum);
    cout << "Result is " << CalSum.result << endl;
    return 0;        }
```



Result is 6

IR autogen

- 从ASTProgram开始遍历

```
program
--fun-declaration: main
----compound-stmt
------var-declaration: a
```

```
Value *CminusfBuilder::visit(ASTProgram &node) {
    VOID_T = module->get_void_type();
    INT1_T = module->get_int1_type();
    INT32_T = module->get_int32_type();
    INT32PTR_T = module->get_int32_ptr_type();
    FLOAT_T = module->get_float_type();
    FLOATPTR_T = module->get_float_ptr_type();

    Value *ret_val = nullptr;
    for (auto &decl: shared_ptr<ASTDeclaration> & : node.declarations) {
        ret_val = decl->accept(&: *this);
    }
    return ret_val;
}
```

IR autogen

- 定义函数返回值类型

```
program
--fun-declaration: main
----compound-stmt
------var-declaration: a
```

```
Value *CminusfBuilder::visit(ASTFunDeclaration &node) {
    FunctionType *fun_type = nullptr;
    Type *ret_type = nullptr;
    std::vector<Type *> param_types;
    if (node.type == TYPE_INT) {
        ret_type = INT32_T;
    } else if (node.type == TYPE_FLOAT) {
        ret_type = FLOAT_T;
    } else {
        ret_type = VOID_T;
    }

    for (auto &param: shared_ptr<ASTParam> & : node.params) {…
    }
```

IR autogen

- 定义函数类型
- 定义函数
- 定义入口BB
- 定义函数的scope（用于管理变量的生命周期）

```
program
--fun-declaration: main
----compound-stmt
------var-declaration: a
```

```
fun_type = FunctionType::get(result: ret_type, params: param_types);
auto *func: Function * = Function::create(ty: fun_type, name: node.id, parent: module.get());
scope.push(name: node.id, val: func);
context.func = func;
auto *funBB: BasicBlock * = BasicBlock::create(m: module.get(), name: "entry", parent: func);
builder->set_insert_point(bb: funBB);
scope.enter();
context.pre_enter_scope = true;
std::vector<Value *> args;
for (auto &arg: Argument & : func->get_args()) {
    args.push_back(x: &arg);
}
for (unsigned i = 0; i < node.params.size(); ++i) {…
}
```

IR autogen

- 递归地遍历函数体部分
- 补充返回指令

```
program
--fun-declaration: main
----compound-stmt
------var-declaration: a
```

```
node.compound_stmt->accept(&: *this);
// can't deal with return in both blocks
if (not builder->get_insert_block()->is_terminated()) {
    if (context.func->get_return_type()->is_void_type()) {
        builder->create_void_ret();
    } else if (context.func->get_return_type()->is_float_type()) {
        builder->create_ret(val: CONST_FP(0.));
    } else {
        builder->create_ret(val: CONST_INT(val: 0));
    }
}
scope.exit();
return nullptr;
```

IR autogen

- 进入一层scope
- 递归遍历所有变量定义与语句(stmt)
- 退出这层scope

```
program
--fun-declaration: main
----compound-stmt
------var-declaration: a
```

```
Value *CminusfBuilder::visit(ASTCompoundStmt &node) {
    bool need_exit_scope = !context.pre_enter_scope;
    if (context.pre_enter_scope) {
        context.pre_enter_scope = false;
    } else {
        scope.enter();
    }
    for (auto &decl: shared_ptr<ASTVarDeclaration> & : node.local_declarations) {
        decl->accept(&: *this);
    }
    for (auto &stmt: shared_ptr<ASTStatement> & : node.statement_list) {
        stmt->accept(&: *this);
        if (builder->get_insert_block()->is_terminated()) {
            break;
        }
    }
    if (need_exit_scope) {
        scope.exit();
    }
    return nullptr;
}
```

IR autogen

- 定义变量类型
- 变量存入scope中

```
program
--fun-declaration: main
----compound-stmt
------var-declaration: a
```

```
Value *CminusfBuilder::visit(ASTVarDeclaration &node) {
    Type *var_type = nullptr;
    if (node.type == TYPE_INT) {
        var_type = module->get_int32_type();
    } else {
        var_type = module->get_float_type();
    }

    if (node.num == nullptr) {
        if (scope.in_global()) {…
        } else {
            auto *var: AllocaInst * = builder->create_alloca(ty: var_type);
            scope.push(name: node.id, val: var);
        }
    } else {…
    }
    return nullptr;
}
```

Lab2 自动化 IR 生成

编译原理课程组

中国科学技术大学

AST->lightIR 转换示例

Cminusf源文件

```
int main(void)
{
    return 0;
}
```

emit-ast

AST实例

```
program
--fun-declaration: main
----compound-stmt
------return-stmt
--------simple-expression
----------additive-expression
------------term
--------------num (int): 0
```

emit-llvm

lightir 源文件

```
define i32 @main() {
label_entry:
  ret i32 0
}
```

Lab 2 自动化 IR 生成

- **基于已有 AST 通过访问者模式进行 IR 自动化生成**

- **AST**

- **自动化生成工具：CminusBuilder 类**
  - 参考 include/cminusf/cminusf_builder.hpp
  - TODO: 需添加一些属性信息
- **访问者模式：CminusBuilder 中不同 ASTnode 的 visit 函数**
  - src/cminusfc/cminusf_builder.cpp
  - TODO：visit 函数待实现

Lab2 实验内容

- **正如前面所讲，本次实验需要补充 autogen 环节中的便利 AST 生成 Lighter IR 代码，给大家准备了 4 个可以补充的地方**
  - Fork 代码仓 https://github.com/XiaoTonghuan/USTC-Compiler-Engineering-2025
  - 补充 /src/cminus/cminusf_builder.cpp 中的 4 个 TODO （需要补充的地方都给大家标记出来了）

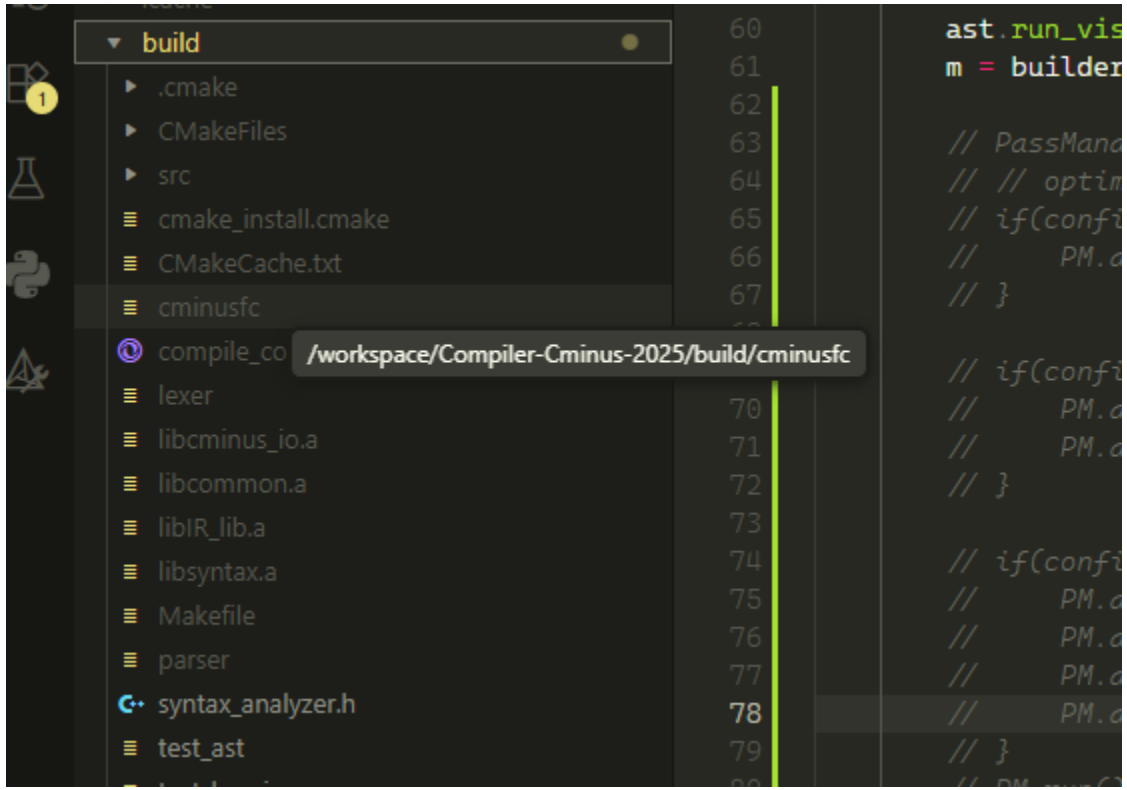
```
Value* CminusfBuilder::visit(ASTIterationStmt &node) {
    // TODO: This function is empty now.
    // Add some code here.
    return nullptr;
}
```

Lab2 结果测试

- **Include ---------- light ir 代码库文件**
- **src ----------IR 自动生成框架源文件**
- **tests**
  - 1-parser ---------- 生成抽象语法树阶段的测试（不用管）
  - 2-ir-gen ---------- lab2 测试相关文件
    - autogen ---------- lab2 相关测试
    - warmup ---------- 大家可以用这里的内容来练手（单个测试）
      - answers ---------- 每个测试用例对应的输出
      - testcases ---------- 每个测试用例
      - eval_lab2.py
      - eval_lab2.sh ---------- 测试用脚本

Lab2 结果测试（对单个代码调试）

- 代码 build 成功后，可以使用可执行文件先测试一些简单的例子，比如我的可执行文件在 build/cminusfc



Lab2 结果测试（对单个代码调试）

- 先观察判断 ir 生成是否正确，再进行后面步骤
- 比如我执行了下面的命令

```
./build/cminusfc -o  ./test.ll -emit-llvm ./test.cminus
```

- 其中 ./test.cminus 是我们要处理的文件，./test.ll 是我们要生成的文件，内容如下

test.cminus

```
1  void main(void) {
2      float a;
3      return;
4  }
5
```

test.ll

```
1  ; ModuleID = 'cminus'
2  source_filename = "/workspace/Compiler-Cminus-2025/test.cminus"
3
4  declare i32 @input()
5
6  declare void @output(i32)
7
8  declare void @outputFloat(float)
9
10 declare void @neg_idx_except()
11
12 define void @main() {
13 label_entry:
14   %op0 = alloca float
15   ret void
16 }
17
```

Lab2 结果测试（对单个代码调试）

- ir 生成后，可以通过 clang 生成可执行文件

```
# clang -o0 -w -no-pie ./test.ll -o ./test -L ./build/ -lcminus_io
#
```

优化层级，拦截所有警告，生成位置有关代码，生成的.ll 文件 可执行文件的位置，　　连接的库位置，需要连接实验提供的io库

- 执行可执行文件后，可以通过 echo $? 来查看 return code

```
(base) root@cf85b2980346:/workspace/Compiler-Cminus-2025# clang -o0 -w ./test.ll -o ./test -L ./build/ -lcminus_io
(base) root@cf85b2980346:/workspace/Compiler-Cminus-2025# echo $?
0
(base) root@cf85b2980346:/workspace/Compiler-Cminus-2025#
```

Lab2实验内容（大规模测试）

- 可以执行 /test/2-ir-gen/autogen 目录下的 eval_lab2.sh 脚本，查看对所有测试样例，结果在 tests/2-ir-gen/autogen/eval_result
- 可以查看每一个测试用例的成功与失败，80个用例按照难度分为5个等级，总共是100分，成功的测试用例会显示 Success，失败就显示 Fail

Lab2实验内容（大规模测试）

- **如果没问题如右图所示**
  - 所有样例都显示 Success
  - 总分(total points)是满分
- **如果有问题，比如**

```
==========lv3 START========
complex1:    Success
complex2:    Fail
complex3:    Success
complex4:    Success
points of lv3 is: 8
==========lv3 END========

total points: 97
```

- **发现是 complex2 有问题**

```
negidx_voidfuncall: Success
selection1: Success
selection2: Success
selection3: Success
iteration1: Success
iteration2: Success
scope:   Success
transfer_float_to_int:   Success
transfer_int_to_float:   Success
points of lv1 is: 31
==========lv1 END========

==========lv2 START========
funcall_chain:   Success
assign_chain:    Success
funcall_var:     Success
funcall_int_array:   Success
funcall_float_array:     Success
funcall_array_array:     Success
return_in_middle1:   Success
return_in_middle2:   Success
funcall_type_mismatch1: Success
funcall_type_mismatch2: Success
return_type_mismatch1:   Success
return_type_mismatch2:   Success
points of lv2 is: 23
==========lv2 END========

==========lv3 START========
complex1:    Success
complex2:    Success
complex3:    Success
complex4:    Success
points of lv3 is: 11
==========lv3 END========

total points: 100
```

Lab2实验内容（大规模测试）

- **用例位置在**
  - tests/2-ir-gen/autogen/testcases/lv3/complex2.cminus
- **参考答案和输入位置（可执行文件的输入输出，不是参考的 IR！）**
  - tests/2-ir-gen/autogen/answers/lv3/complex2.in
  - tests/2-ir-gen/autogen/answers/lv3/complex2.out
- **按照最开始的单例测试方法，进行针对性测试，看看是哪里的问题**

Lab2实验提交方法

- **将 GitHub 仓地址发送到邮箱 tonghuanxiao@gmail.com**
- **将验证正确性成功的文件(eval_result)一并发送到上述邮箱**
- **注意**
  - 请确保 GitHub 有正常的提交记录以证明本实验是自己完成的
  - 请不要大规模使用 AI 完成任务
  - 请注意实验截止时间（两周，11月2日晚12点）