Министерство науки и высшего образования Российской Федерации
Севастопольский Государственный Университет
Кафедра ИС

Расчётно-графическое задание

по дисциплине: «Корпоративные информационные системы»

на тему: «Разработка системы тестирования студентов по дисциплине «Объектно-ориентированное программирование»

Выполнил:
ст. гр. ИС/б-17-2-о
Клышко Н.А.

Проверил:
Дымченко И.В.

Севастополь

2021

1 ЦЕЛЬ РАБОТЫ

Исследовать алгоритмы автоматического тестирования заданий по программированию. Разработать систему автоматического тестирования студентов по дисциплине «Объектно-ориентированное программирование».

2 ПОСТАНОВКА ЗАДАЧИ

Разработанная тестирующая система должна обеспечивать автоматическую проверку решений задач в рамках дисциплины ООП.

Проверяемые задачи формулируются в общем виде следующим образом: «На языке программирования C++ составить N классов, содержащих M конструкторов и K методов, а также составить главную функцию программы, которая будет вызывать все/указанные конструкторы и методы».

Конкретная формулировка задачи может формулироваться следующим образом: «Создать класс «студент». Определить в нем три поля и два метода. Описать конструкторы (с параметрами и без параметров) и деструктор. Вынести описание одного из методов за рамки класса».

3 АНАЛИЗ ЗАДАЧИ

3.1 Для проверки задач с заданной формулировкой возможно использовать следующие методы:

1) Анализ информации о фактическом выполнении программы (gcov gcc) для получение информации как о наличии заданных языковых конструкций, так и факте их вызова.
2) Парсинг программы и проверка наличия языковых конструкций (если такую информацию невозможно получить из gcov), далее проверка вызовов уже с использованием gcov.
3) Реализация модуля clang для вставки вывода отладочной информации в валидируемые конструкторы/методы. Описание реализации подобного метода приведено в [1].

3.2 Процесс проверки соответствия структуры программы заданному описанию можно разделить на два этапа:

1) Извлечение фактов о наличии классов методов и других метаданных из исходного кода проверяемой программы.

2) С использованием фактической информации, полученной на предыдущем этапе, сопоставить структуру программы и описание.

Конкретные технические реализации каждого из этапов описаны в разделе 4.

3.3 Для проверки алгоритмической составляющей программы возможно использовать следующий подход:

1) Скорректировать постановку задачи так, чтобы реализуемые студентом методы выводили ответ в заданном формате.
2) С помощью кодогенерации сформировать вызовы заданных методов, инжектировать эти вызовы в код функции main() загруженной программы.
3) Проверить вывод программы стадартным методом проверки алгоритмических задач по программированию.

3.4 Интеграция разрабатываемого модуля в образовательную систему можно осуществить несколькими способами:

1) Интеграция на основе Moodle-плагина CodeRunner. Данный плагин реализует специальный тип вопроса в системе Moodle, который позволяет преподавателю задать тесты для программы, а студенту при прохождении урока загрузить свое решение в виде исходного кода программы. Оценивание написанной программы происходит автоматически, в соответствии с установленной бальной системой.
2) Разработка стороннего веб-интерфейса и веб-сервиса для тестирования, результаты из которого будут передаваться в Moodle. Реализация такой подсистемы должна включать следующие компоненты: WebIDE (браузерный редактор кода), серверный модуль, реализующий алгоритм тестирования пользовательской программы, панель администратора для конфигурации тестовых последовательностей и модуль интеграции с системой оценивания Moodle. Возможная архитектура такого сервиса изображена на рисунке 1.

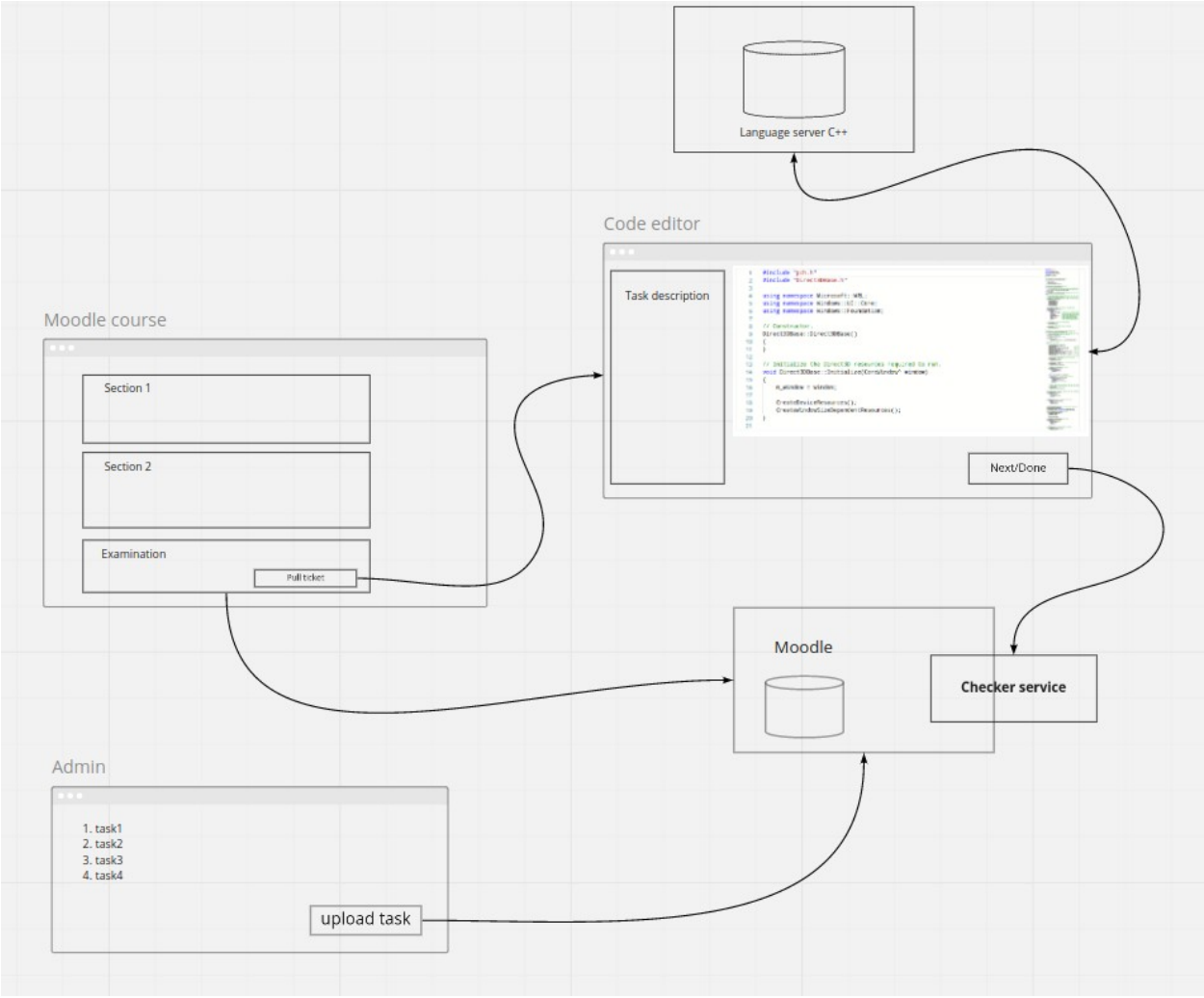

Рисунок 1 – Эскизная архитектура standalone-сервиса

4 ТЕХНИЧЕСКОЕ РЕШЕНИЕ

4.1 Реализация сбора фактов

Сбор фактов о структуре ООП-программы реализован на основе обработки абстрактного синтаксического дерева программы на языке C++ (AST). В листингах 1 и 2 приведены исходный текст C++ программы и соответствующее ей AST (дерево). Машинный анализ дерева в представленном виде не является эффективным и не содержит некоторой дополнительной метаинформации, поэтому тестирующая программа анализирует это дерево, заданное в формате JSON. Фрагмент AST в формате JSON приведён в приложении А.

Для получения AST средствами инструментария компилятора Clang используются следующие команды:

1) Для получения дерева в человеко-читаемом формате
   `clang -Xclang -ast-dump -fsyntax-only task1.cc`
2) Для получения в JSON формате, пригодном для машинной обработки
   `clang -Xclang -ast-dump=json -fsyntax-only -fno-color-diagnostics task1.cc`

Исходный код тестирующей Python-программы приведён в приложении Б.

Листинг 1. Текст C++ программы task1.cc

```
class Student {

    int property1;
    double property2;
    long property3;

    Student() {}
    Student(int param) {}

    void testMethod1();
    void testMethod2() {}

    ~Student() {}

};

void Student::testMethod1() {}
```

Листинг 2. AST программы

```
TranslationUnitDecl 0x556f5f8ac688 <<invalid sloc>> <invalid sloc>
|-CXXRecordDecl 0x556f5f8ec088 <examples/task1.cc:1:1, line:25:1> line:1:7 class Student definition
| |-DefinitionData standard_layout has_user_declared_ctor can_const_default_init
| | |-DefaultConstructor exists non_trivial user_provided
| | |-CopyConstructor simple trivial has_const_param needs_implicit implicit_has_const_param
| | |-MoveConstructor
| | |-CopyAssignment simple trivial has_const_param needs_implicit implicit_has_const_param
| | |-MoveAssignment
| | `-Destructor non_trivial user_declared
| |-CXXRecordDecl 0x556f5f8ec1b0 <col:1, col:7> col:7 implicit referenced class Student
| |-FieldDecl 0x556f5f8ec258 <line:3:5, col:9> col:9 property1 'int'
| |-FieldDecl 0x556f5f8ec2c0 <line:4:5, col:12> col:12 property2 'double'
| |-FieldDecl 0x556f5f8ec328 <line:5:5, col:10> col:10 property3 'long'
| |-CXXConstructorDecl 0x556f5f8ec420 <line:7:5, line:9:5> line:7:5 Student 'void ()'
| | `-CompoundStmt 0x556f5f8ec918 <col:15, line:9:5>
| |-CXXConstructorDecl 0x556f5f8ec5b0 <line:11:5, line:13:5> line:11:5 Student 'void (int)'
| | |-ParmVarDecl 0x556f5f8ec4e0 <col:13, col:17> col:17 param 'int'
| | `-CompoundStmt 0x556f5f8ec928 <col:24, line:13:5>
| |-CXXMethodDecl 0x556f5f8ec680 <line:15:5, col:22> col:10 testMethod1 'void ()'
| |-CXXMethodDecl 0x556f5f8ec740 <line:17:5, line:19:5> line:17:10 testMethod2 'void ()'
| | `-CompoundStmt 0x556f5f8ec938 <col:24, line:19:5>
| `-CXXDestructorDecl 0x556f5f8ec828 <line:21:5, line:23:5> line:21:5 ~Student 'void () noexcept'
|   `-CompoundStmt 0x556f5f8ec980 <col:16, line:23:5>
`-CXXMethodDecl 0x556f5f8ec9d0 parent 0x556f5f8ec088 prev 0x556f5f8ec680 <line:27:1, line:29:1>
line:27:15 testMethod1 'void ()'
  `-CompoundStmt 0x556f5f8ecac0 <col:29, line:29:1>
```

Полученная в результате анализа AST информация сохраняется в формате Prolog-фактов. Список полученных фактов из рассмотренной ранее программы показан в листинге 3.

Листинг 3. Фрагмент лога тестирующей программы

```
Collected facts: ["class('Student')"]
Collected facts: ["class('Student')"]
Collected facts: ["property('Student','property1')"]
Collected facts: ["property('Student','property2')"]
Collected facts: ["property('Student','property3')"]
Collected facts: ["constructor('Student',[])"]
Collected facts: ["constructor('Student',['int'])"]
Collected facts: ["method_declaration('Student','testMethod1','void',[])"]
Collected facts: ["method_declaration('Student','testMethod2','void',[])",
"method_implementation('Student','testMethod2',inside,'void',[])"]
Collected facts: ["destructor('Student')"]
Collected facts: ["method_implementation('Student','testMethod1',outside,'void',[])"]
```

4.2 Реализация сопоставления

Сохранение собранной информации в формате Prolog-фактов обусловлено тем, что сопоставление структуры программы и её абстрактного описания проще всего производить с помощью Prolog-запроса.

В случае проверки структуры одиночного класса использование Prolog-системы, возможно, избыточно. Однако, в случае необходимости сопоставления классовых иерархий при условии, что формулировка задачи позволяет именовать классы произвольным образом (невозможно тривиально сопоставить конкретный фрагмент описания класса в формулировке задачи и

конкретный класс в проверяемой программе), применение Prolog-системы упрощает реализацию сопоставления, так как по-умолчанию реализует перебор всех возможных решений (сопоставлений).

Пример задания, по которому было составлено решение (C++ программа), рассмотренное в разделе 4.1, приведён в листинге 4.

Листинг 4. Формулировка задачи (задания)

```
Создать класс «студент». Определить в нем три поля и два метода. Описать конструкторы (с параметрами и без параметров) и деструктор. Вынести описание одного из методов за рамки класса.
```

Листинг 5. Тестирующий Prolog-запрос

```
class(C),
    property(C,P1), property(C,P2), property(C,P3), all_diff([P1,P2,P3]),
    constructor(C,[]),
    constructor(C,PL1), length(PL1,PL1size), PL1size>0,
    destructor(C),
    method_declaration(C,M1,_,_), method_implementation(C,M1,inside,_,_),
    method_declaration(C,M2,_,_), method_implementation(C,M2,outside,_,_)
```

Таблица 1. Анализируемые типы фактов

Формат факта	Соответствующий узел AST	Описание смысла и параметров
`class(C)`	`CXXRecordDecl`	Наличие класса, C — имя класса
`parent(B,C,Type)`	`CXXRecordDecl`	Наследование классов, B — имя базового класса, C — имя дочернего класса, Type — модификатор наследования (public, private и т.д.)
`property(C,P)`	`FieldDecl`	Поле класса, C — имя класса, P — имя поля
`constructor(C,T)`	`CXXConstructorDecl`	Явный конструктор класса, C — имя класса, T — список типов параметров
`desctructor(C)`	`CXXDestructorDecl`	Явный деструктор класса, C — имя класса
`method_declaration(C,M,RET,PAR)`	`CXXMethodDecl`	Объявление метода класса, C — имя класса, M — имя метода, RET — тип возвращаемого значения, PAR — список типов параметров.
`method_implementation(C,M,Where,RET,PAR)`	`CXXMethodDecl`	Реализация метода класса, параметры аналогично method_declaration, Where — положение (внутри класса — inside, вне класса — outside)

ВЫВОДЫ

В ходе выполнения расчетно-графического задания исследованы методы анализа структуры ООП-программы на основе исходного кода и разработана техническая реализация такого анализа, а также реализация сопоставления полученной структуры и абстрактного описания программы в исходной формулировке задачи.

Составными частями разработанной тестирующей подсистемы являются: инструментарий компилятора C++ Clang, реализация Prolog-системы SWI-Prolog и библиотека PySwip для интеграции тестирующей python-программы и Prolog-системы.

Для проверки разработанной тестирующей программы составлены тестовые примеры, включающие: исходную формулировку задачи, соответствующую C++ программу, тестирующий Prolog-запрос. Разработанная подсистема выполняет необходимые функции и обеспечивает достаточную гибкость при составлении тестирующих запросов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Анализатор C++ на первом курсе: миф, иллюзия или выдумка? [Электронный ресурс]. – 2020. – Режим доступа: https://habr.com/ru/company/hsespb/blog/525794/, свободный. – Загл. с экрана.

2. Документация CODE RUNNER на GitHub [Электронный ресурс]. – Режим доступа: https://github.com/trampgeek/moodle-qtype_coderunner/blob/master/Readme.md.

3. Бондарев В.Н. Искусственный интеллект:Учеб. пособие для вузов / В.Н. Бондарев, Ф.Г. Аде. – Севастополь: Изд-во СевНТУ, 2002. – 615с. – ISBN 966-7473-45-7.

4. Clang 12 documentation. Introduction to the Clang AST [Электронный ресурс]. – Режим доступа: https://clang.llvm.org/docs/IntroductionToTheClangAST.html.

ПРИЛОЖЕНИЕ А

Фрагмент представления C++ Clang AST в формате JSON


```
… {
      "id": "0x563f36bcb398",
      "kind": "CXXRecordDecl",
      "loc": {
        "offset": 6,
        "file": "examples/task1.cc",
        "line": 1,
        "col": 7,
        "tokLen": 7
      },
      "range": {
        "begin": {
          "offset": 0,
          "col": 1,
          "tokLen": 5
        },
        "end": {
          "offset": 225,
          "line": 25,
          "col": 1,
          "tokLen": 1
        }
      },
      "name": "Student",
      "tagUsed": "class",
      "completeDefinition": true,
      "definitionData": {
        "canConstDefaultInit": true,
        "copyAssign": { "hasConstParam": true, "implicitHasConstParam": true, "needsImplicit": true,
"simple": true, "trivial": true
        },
        "copyCtor": {
          "hasConstParam": true, "implicitHasConstParam": true, "needsImplicit": true, "simple": true,
"trivial": true
        },
        "defaultCtor": {},
        "dtor": {
          "nonTrivial": true,
          "userDeclared": true
        },
        "hasUserDeclaredConstructor": true,
        "isStandardLayout": true,
        "moveAssign": {},
        "moveCtor": {}
      },
      "inner": [
        {
          "id": "0x563f36bcb4c0",
          "kind": "CXXRecordDecl",
          "loc": { "offset": 6, "line": 1, "col": 7, "tokLen": 7
          },
          "range": {
            "begin": { "offset": 0, "col": 1, "tokLen": 5 },
            "end": { "offset": 6, "col": 7, "tokLen": 7 }
          },
          "isImplicit": true,
          "isReferenced": true,
          "name": "Student",
          "tagUsed": "class"
        },
        {
          "id": "0x563f36bcb568",
          "kind": "FieldDecl",
          "loc": { "offset": 25, "line": 3, "col": 9, "tokLen": 9 },
          "range": {
            "begin": { "offset": 21, "col": 5, "tokLen": 3 },
            "end": { "offset": 25, "col": 9, "tokLen": 9 }
          },
          "name": "property1",
          "type": {
```

```
    "qualType": "int"
  }
},
{
  "id": "0x563f36bcb5d0",
  "kind": "FieldDecl",
  "loc": { "offset": 47, "line": 4, "col": 12, "tokLen": 9 },
  "range": {
    "begin": { "offset": 40, "col": 5, "tokLen": 6 },
    "end": { "offset": 47, "col": 12, "tokLen": 9 }
  },
  "name": "property2",
  "type": {
    "qualType": "double"
  }
},
{
  "id": "0x563f36bcb638",
  "kind": "FieldDecl",
  "loc": { "offset": 67, "line": 5, "col": 10, "tokLen": 9 },
  "range": {
    "begin": { "offset": 62, "col": 5, "tokLen": 4 },
    "end": { "offset": 67, "col": 10, "tokLen": 9 }
  },
  "name": "property3",
  "type": {
    "qualType": "long"
  }
},
{
  "id": "0x563f36bcb7c0",
  "kind": "CXXConstructorDecl",
  "loc": { "offset": 113, "line": 11, "col": 5, "tokLen": 7 },
  "range": {
    "begin": { "offset": 113, "col": 5, "tokLen": 7 },
    "end": { "offset": 139, "line": 13, "col": 5, "tokLen": 1 }
  },
  "name": "Student",
  "mangledName": "_ZN7StudentC1Ei",
  "type": {
    "qualType": "void (int)"
  },
  "inner": [
    {
      "id": "0x563f36bcb6a0",
      "kind": "ParmVarDecl",
      "loc": { "offset": 125, "line": 11, "col": 17, "tokLen": 5 },
      "range": {
        "begin": { "offset": 121, "col": 13, "tokLen": 3 },
        "end": { "offset": 125, "col": 17, "tokLen": 5 }
      },
      "name": "param",
      "mangledName": "_ZZN7StudentC1EiE5param",
      "type": {
        "qualType": "int"
      }
    },
    {
      "id": "0x563f36bcbb58",
      "kind": "CompoundStmt",
      "range": {
        "begin": { "offset": 132, "col": 24, "tokLen": 1 },
        "end": { "offset": 139, "line": 13, "col": 5, "tokLen": 1 }
      }
    }
  ]
}, ...
```

ПРИЛОЖЕНИЕ Б

Исходный код тестирующей программы

```
from pyswip import Prolog
import argparse
import json
import re

FUNCTION_TYPE_PATTERN = r"([^\(\)\s]+)\s*\(([^\(\)]*)\).*"

TYPE_CLASS = 'class'
TYPE_METHOD = 'method'

LIBRARIES = [
    "use_module(library(clpfd))"
]

RULES = [
    "all_diff(L) :- \+ (append(_,[X|R],L), memberchk(X,R))"
]

declared_methods = {}

parser = argparse.ArgumentParser(description='Tests OOP program structure.')
parser.add_argument('ast', type=str, help='Json file of CLang AST')
parser.add_argument('src_name', type=str, help='Name of source code file')
parser.add_argument('test', type=str, help='Prolog test query')

args = parser.parse_args()

def create_object(parent, obj_type, name):
    return {
        'parent': parent,
        'type': obj_type,
        'name': name
    }


def collect_facts(prolog, parent, ast):
    node_kind = ast['kind']
    facts = []
    if node_kind == 'CXXRecordDecl':
        class_name = ast['name']
        facts.append("class('{}')".format(class_name))
        parent = create_object(parent, TYPE_CLASS, class_name)
        if 'bases' in ast:
            for base in ast['bases']:
                base_class_name = base['type']['qualType']
                facts.append("parent('{}','{}',{})".format(
                        base_class_name, class_name, base['access']
                    ))
    if node_kind == 'FieldDecl' \
            and parent != None and parent['type'] == TYPE_CLASS:
        class_name = parent['name']
        property_name = ast['name']
        facts.append("property('{}','{}')".format(class_name, property_name))
    if node_kind == 'CXXConstructorDecl' \
            and parent != None and parent['type'] == TYPE_CLASS \
```

```
            and ('isImplicit' not in ast or not ast['isImplicit']):
            # and 'type' in ast and 'qualType' in ast['type'] \
        match = re.match(FUNCTION_TYPE_PATTERN, ast['type']['qualType'])
        assert(match.lastindex == 2)
        parameter_types = match.group(2)
        if parameter_types.strip() == '':
            parameter_types = []
        else:
            parameter_types = parameter_types.split(', ')
        class_name = parent['name']
        facts.append("constructor('{}',{})".format(class_name, str(parameter_types)))
    if node_kind == 'CXXDestructorDecl' \
            and parent != None and parent['type'] == TYPE_CLASS \
            and ('isImplicit' not in ast or not ast['isImplicit']):
        class_name = parent['name']
        facts.append("destructor('{}')".format(class_name))
    if node_kind == 'CXXMethodDecl':
        method_name = ast['name']
        match = re.match(FUNCTION_TYPE_PATTERN, ast['type']['qualType'])
        assert(match.lastindex == 2)
        return_type = match.group(1)
        parameter_types = match.group(2)
        if parameter_types.strip() == '':
            parameter_types = []
        else:
            parameter_types = parameter_types.split(', ')
        # parent = create_object(parent, TYPE_METHOD, method_name)
        if parent is not None and parent['type'] == TYPE_CLASS:
            class_name = parent['name']
            facts.append("method_declaration('{}','{}','{}',{})".format(
                    class_name, method_name, return_type, str(parameter_types)
                ))
            if 'inner' in ast:
                facts.append("method_implementation('{}','{}',inside,'{}',
{})".format(
                    class_name, method_name, return_type, str(parameter_types)
                ))
            declared_methods[ast['id']] = parent
        elif 'previousDecl' in ast and ast['previousDecl'] in declared_methods \
                and 'inner' in ast:
            class_obj = declared_methods[ast['previousDecl']]
            facts.append("method_implementation('{}','{}',outside,'{}',{})".format(
                    class_obj['name'], method_name, return_type, str(parameter_types)
                ))
        else:
            print('Warning: Invalid CXXMethodDecl node: {}'.format(ast['id']))



    if len(facts) > 0:
        print('Collected facts: {}'.format(str(facts)))
        for fact in facts:
            prolog.assertz(fact)

    if 'inner' in ast:
        for children in ast['inner']:
            collect_facts(prolog, parent, children)


def main():
    prolog = Prolog()
```

```
    with open(args.ast, 'r') as f:
        try:
            ast = json.loads(f.read())
        except:
            print("Can't read ast file!")
            exit()

    for library in LIBRARIES:
        prolog.assertz(library)
    for rule in RULES:
        prolog.assertz(rule)

    if ast['kind'] == 'TranslationUnitDecl':
        items = ast['inner']
        current_file = ''
        for item in items:
            if 'loc' in item and 'file' in item['loc'] and item['loc']['file'] != '':
                current_file = item['loc']['file']
                # print('Handling file: {}'.format(current_file))

            if current_file == args.src_name:
                collect_facts(prolog, None, item)
    else:
        print('Invalid root declaration')
        exit()

    solutions = prolog.query(args.test)
    try:
        specific_solution = next(solutions)
        print('PASSED')
    except:
        print('FAILED')

    solutions.close()


main()
```