Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по дисциплине
«Системное программное обеспечение»
Вариант: 2

Выполнил:
Шибаев Семен Сергеевич
Группа:
P4114

Преподаватель:
Кореньков Юрий Дмитриевич

Санкт-Петербург

2023

Задание

Реализовать построение графа потока управления посредством анализа дерева разбора для набора входных файлов. Выполнить анализ собранной информации и сформировать набор файлов с графическим представлением для результатов анализа.

Порядок выполнения:

1. Описать структуры данных, необходимые для представления информации о наборе файлов, наборе подпрограмм и графе потока управления, где:
    a. Для каждой подпрограммы: имя и информация о сигнатуре, граф потока управления, имя исходного файла с текстом подпрограммы.
    b. Для каждого узла в графе потока управления, представляющего собой базовый блок алгоритма подпрограммы: целевые узлы для безусловного и условного перехода (по мере необходимости), дерево операций, ассоциированных с данным местом в алгоритме, представленном в исходном тексте подпрограммы
2. Реализовать модуль, формирующий граф потока управления на основе синтаксической структуры текста подпрограмм для входных файлов
    a. Программный интерфейс модуля принимает на вход коллекцию, описывающую набор анализируемых файлов, для каждого файла – имя и соответствующее дерево разбора в виде структуры данных, являющейся результатом работы модуля, созданного по заданию 1 (п. 3.b).
    b. Результатом работы модуля является структура данных, разработанная в п. 1, содержащая информацию о проанализированных подпрограммах и коллекция с информацией об ошибках
    c. Посредством обхода дерева разбора подпрограммы, сформировать для неё граф потока управления, порождая его узлы и формируя между ними дуги в зависимости от синтаксической конструкции, представленной данным узлом дерева разбора: выражение, ветвление, цикл, прерывание цикла, выход из подпрограммы – для всех синтаксических конструкций по варианту (п. 2.b) С каждым узлом графа потока управления связать дерево операций, в котором каждая операция в составе

текста программы представлена как совокупность вида операции и соответствующих операндов (см задание 1, пп. 2.d-g)
    d. При возникновении логической ошибки в синтаксической структуре при обходе дерева разбора, сохранить в коллекции информацию об ошибке и её положении в исходном тексте
3. Реализовать тестовую программу для демонстрации работоспособности созданного модуля
    a. Через аргументы командной строки программа должна принимать набор имён входных файлов, имя выходной директории
    b. Использовать модуль, разработанный в задании 1 для синтаксического анализа каждого входного файла и формирования набора деревьев разбора
    c. Использовать модуль, разработанный в п. 2 для формирования графов потока управления каждой подпрограммы, выявленной в синтаксической структуре текстов, содержащихся во входных файлах
    d. Для каждой обнаруженной подпрограммы вывести представление графа потока управления в отдельный файл с именем “sourceName.functionName.ext” в выходной директории, по- умолчанию размещать выходной файлы в той же директории, что соответствующий входной
    e. Для деревьев операций в графах потока управления всей совокупности подпрограмм сформировать граф вызовов, описывающий отношения между ними в плане обращения их друг к другу по именам и вывести его представление в дополнительный файл, по-умолчанию размещаемый рядом с файлом, содержащим подпрограмму main.
    f. Сообщения об ошибке должны выводиться тестовой программной (не модулем, отвечающим за анализ!) в стандартный поток вывода ошибок
4. Результаты тестирования представить в виде отчета, в который включить:
    a. В части 3 привести описание разработанных структур данных
    b. В части 4 описать программный интерфейс и особенности реализации разработанного модуля
    c. В части 5 привести примеры исходных анализируемых текстов для всех синтаксических конструкций разбираемого языка и соответствующие результаты разбора

Детали реализации

Используются две структуры: вершина собственно графа потока управления и вершина дерева операций. Дерево операций соответствует блоку.

Нода графа потока управления бывает для условия (condition), а бывает для цикла (loop).

```
enum cfg_node_type {
    LOOP_CFG,

    CONDITION_CFG

};



struct cfg_loop {

    struct cfg_node *next_body;

    struct cfg_node *end; bool

    visited;

};



struct cfg_condition {

    struct cfg_node *then_block;

    struct cfg_node *else_block;

};
```

Общая структура (объединяет две через union).

```
struct cfg_node {

    char desciption[MAXIMUM_IDENTIFIER_LENGTH];

    enum cfg_node_type type;

    unsigned long long id;

    struct cfg_node *next;
```

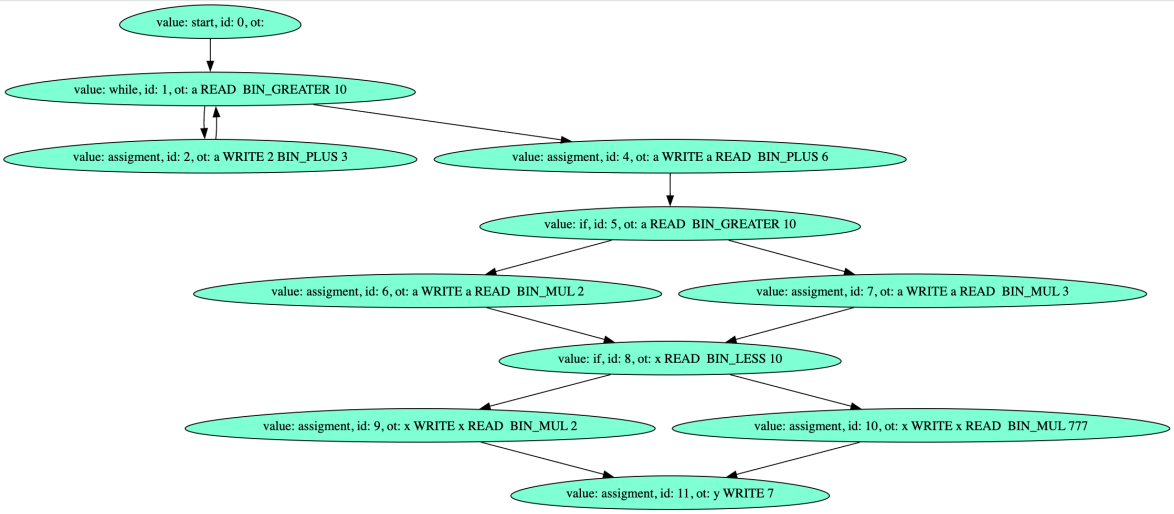
```
    struct operation_node *ot_root;
    bool visited;
    union {
    struct cfg_loop cfg_loop;
    struct cfg_condition cfg_condition;
    };
};
```

Структура вершины дерева операций:

```
struct operation_node {
    int id;
    char type[MAXIMUM_IDENTIFIER_LENGTH];
    bool is_left;
    bool is_right;
    union {
    char left_operand[MAXIMUM_IDENTIFIER_LENGTH];
    struct operation_node *left_next;
    };
    union{
    char     right_operand[MAXIMUM_IDENTIFIER_LENGTH];
    struct operation_node *right_next;
    };
};
```

Пример

```
method
callCthulhu(a:int):callingType
    begin
        while a>10 do
         begin
            a:=2+3;
         end;
        a:=a+6;
        if a>10 then a:=a*2;
        else a:=a*3;
        if x<10 then x:=x*2;
        else x:=x*777;
        y:=7;
    end;
```



Вывод

Я научился строить граф потока управления для программы.