

НИУ ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2
по дисциплине
«Системное программное обеспечение»
Вариант 1

Выполнил: студент группы Р4116

Крюков Андрей

Преподаватель: Кореньков Юрий Дмитриевич

Санкт-Петербург
2023 г.

Задание:	3
Описание структур данных	4
Примеры входных данных и результаты их обработки	4
Вывод	5

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)
 - d. 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)
 - e. 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 выходные 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 программой (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

Описание структур данных

В ходе обхода AST формируется граф потока управления(CFG), представленный следующими структурами

```
struct Block {
    int id;
    LinkList *predecessors;
    LinkList *exits;
};

struct BlockList {
    Block **blocks;
    int count;
};

struct CFG {
    char *procedureName;
    Block *entryblock;
    int nextId;
};

struct Link {
    Block *source;
    Block *target;
    char *comment;
};

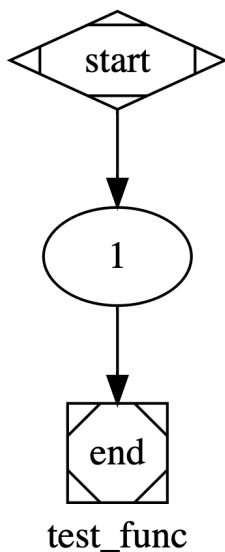
struct LinkList {
    Link **links;
    int count;
};
```

Обход AST выполняется посредством DFS

Результат при помощи рекурсивного алгоритма представляется в формате dot и для каждой процедуры выводится в отдельный файл, после чего его можно визуализировать при помощи, к примеру, graphviz.

Примеры входных данных и результаты их обработки

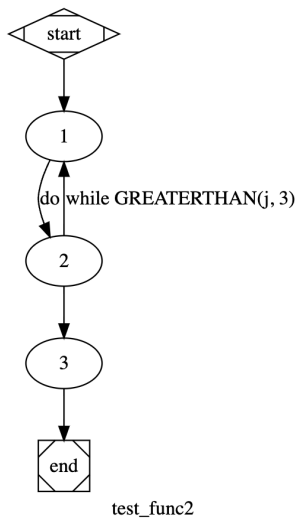
```
bool test_func() {
    int[,] g;
    g[r] = 4;
}
```



```

bool test_func2 (int arg_1 ) {
    do {
        r = 5+(6*4);
    } while (j > 3);
}

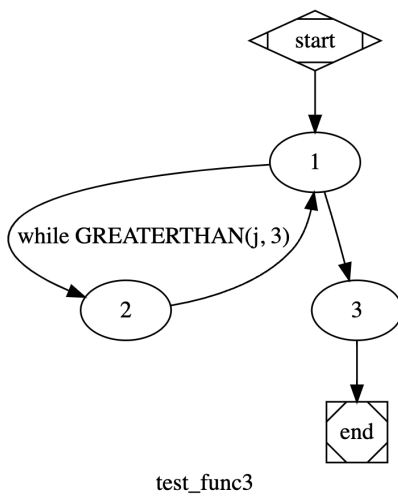
```



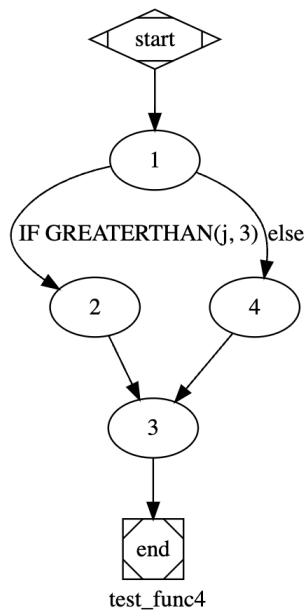
```

bool test_func3 (int arg_1 ) {
    while (j > 3) {
        r = 5+(6*4);
    }
}

```



```
bool test_func4 (int arg_1 ) {  
    if (j > 3) {  
        r = 5+(6*4);  
    }else {  
        g = 2;  
    }  
}
```



Вывод

В ходе данной лабораторной работы были имплементированы алгоритмы построения на основе AST и вывода в формате .dot графа потока управления.