УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техникиНаправление подготовки 09.04.04 Программная инженерия Дисциплина «Системное программное обеспечение»

Лабораторная работа №2

Вариант 1

Студент
Алексеев М. Н.
P4116

Преподаватель

Кореньков Ю. Д.

Санкт-Петербург, 2023 г.

Задание лабораторной работы

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 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

Описание работы

Пример входных данных (содержимое файлов):

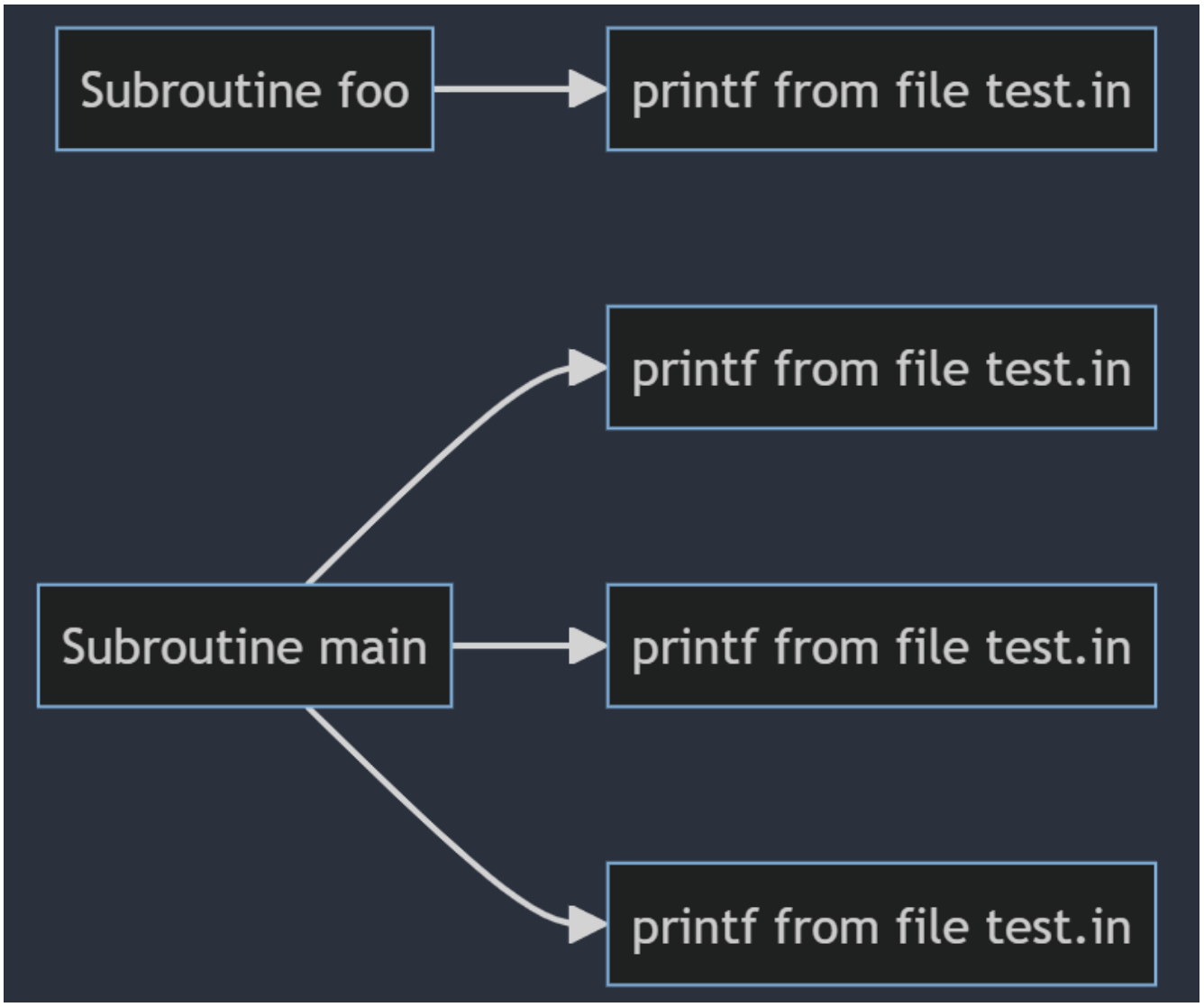
```
foo(oaoaoa);

int foo(oaoaoa);

main(argc, string[] argv) {
   printf("Hello, World!\n");

   if (argc == 0) {
      break;
   }

   // int x = argv[0];
   string y = argv[0];

   // x + y;

   int i = 0;
   while (i < 100) {
      printf("Iteration\n");
      ++i;
   }

   int j = 0;
   do {
      printf("Iteration\n");
      ++j;
   } while (j < 100);

   char x = 'a';
   0;
```

```
int foo(lol);

foo(kek oaoaoa) {
    printf("kek\n");
    // 1 == "kek";
}

int foo(kek oaoaoa);

printf(string text);
```

Пример результата работы программы (консольный вывод):

```
subroutine "foo" in file "test.in" at 2:14 (defined)
  with 1 arguments
  returns int at 2:1
- Local variables:
  - #1 kek oaoaoa at 4:5
- Flow graph:
  - #1 NOP at 4:17
    - next: #2
  - #2 EXPR at 5:5
    - expr -> <expr:call> at 5:5
      - type -> int at 9:21
      - subroutine -> subroutine "printf" in file "test.in" at 9:21 (not defined)
      - args[0] -> <expr:literal> at 5:12
        - type -> string at 5:12
        - literal -> <literal:str> "kek\n" at 5:12
    - next: RETURN
```

Пример результата работы программы (граф вызовов):



Вывод

Была создана программа, которая анализирует дерево разбора для набора входных данных и строит граф потока управления