Национальный исследовательский университет ИТМО
Мегафакультет компьютерных технологий и управления
Факультет программной инженерии и компьютерной техники

Отчет по лабораторной работе №2

по курсу «Языки программирования»

Выполнил студент группы P4119	_____	М.С. Маршалов
	(подпись)	

Руководитель	_____	Ю. Д. Кореньков
	(подпись)	

Санкт-Петербург

2025

Цель

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.

Задачи

1 Описать структуры данных, необходимые для представления информации о наборе файлов, наборе подпрограмм и графе потока управления, где:

1.1 Для каждой подпрограммы: имя и информация о сигнатуре, граф потока управления, имя исходного файла с текстом подпрограммы.

1.2 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

2.1 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).

2.2 . Результатом работы модуля является структура данных, разработанная в п. 1, содержащая информацию о проанализированных подпрограммах и коллекция с информацией об ошибках

2.3 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)

2.4 С каждым узлом графа потока управления связать дерево операций, в котором каждаяоперация в составе текста программы представлена как совокупность вида операции и соответствующих операндов (см задание 1, пп. 2.d-g)

2.5 . 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

3.1 Программный интерфейс модуля должен принимать строку с текстом и возвращать структуру,

3.2 описывающую соответствующее дерево разбора и коллекцию сообщений ошибке

3.3 Результат работы модуля – дерево разбора – должно содержать иерархическое

3.4 представление для всех синтаксических конструкций, включая выражения, логически

3.5 представляющие собой иерархически организованные данные, даже если на уровне средства

3.6 синтаксического анализа для их разбора было использовано линейное представление

4 Реализовать тестовую программу для демонстрации работоспособности созданного модуля

4.2 Через аргументы командной строки программа должна принимать имя входного файла для чтения и анализа, имя выходного файла записи для дерева, описывающего синтаксическую структуру разобранного текста

4.3 Сообщения об ошибке должны выводиться тестовой программной (не модулем, отвечающим за анализ!) в стандартный поток вывода ошибок

Ход работы

Для выполнения задания были созданы классы AST-парсера и непосредственно CFG, под собственную грамматику, представленную в Листинге 1. Результат ее разбора представлен на Рисунке 1.

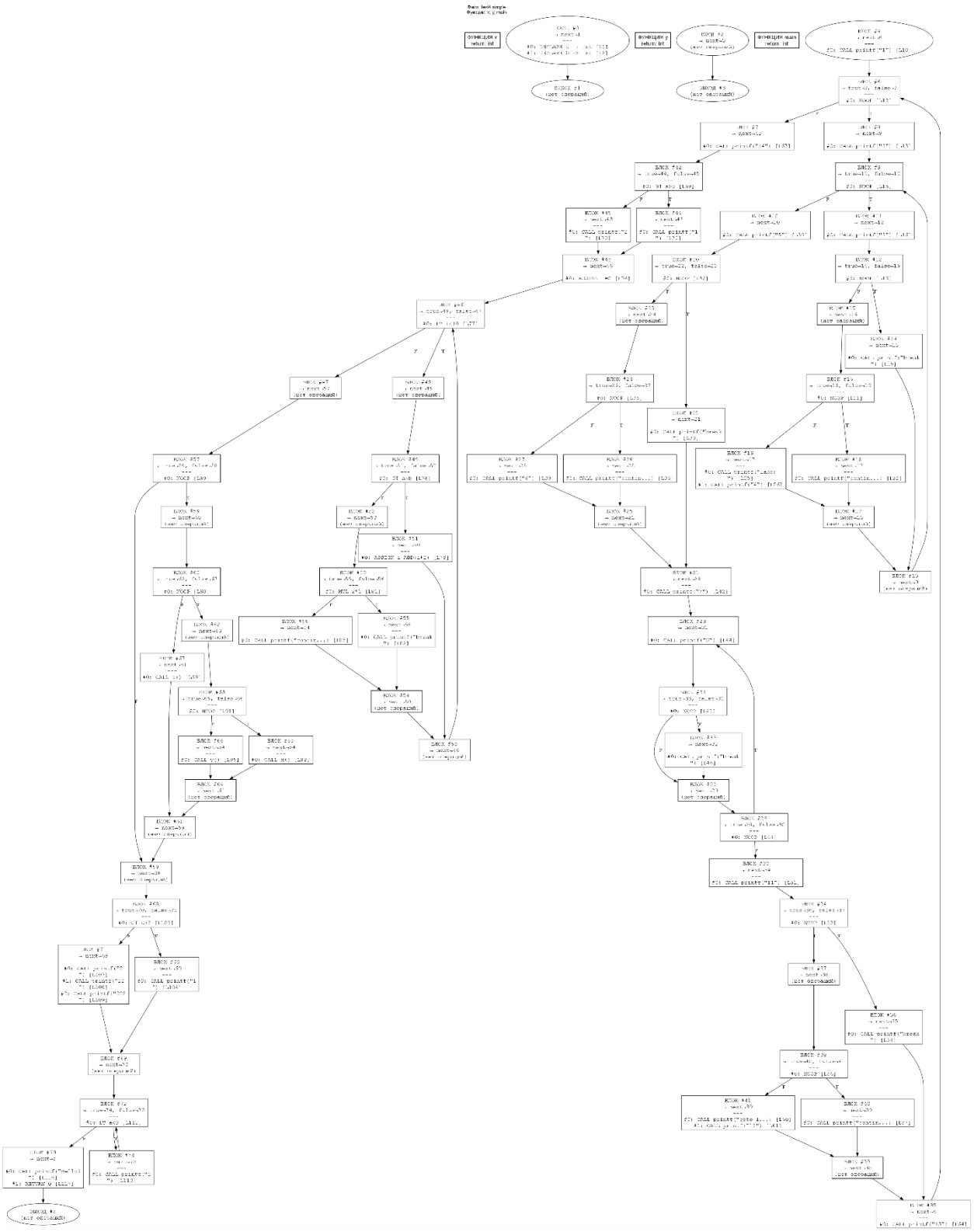

Рисунок 1 – Результат построения CFG

Листинг 1 – Тестовый код

```
function x() -> int{
    a -> int;
    b -> int;
}

function y() -> int{

}

function main() -> int {
    printf("1");

    while (1) {
        printf("2");

        while (2) {
            printf("3");

            if (3) {
                printf("break\n");
            }
            else if (4) {
                printf("continue\n");
```

```
    }
    else {
        printf("labb:\n");
        printf("4");
    }
}

printf("5");

if (5) {
    printf("break\n");
}
else if (6) {
    printf("continue\n");
}
else {
    printf("6");
}

printf("7");

do {
    printf("8");
    if (9) {
```

```
            printf("break\n");
        }
    } while (10);

    printf("11");

    if (11) {
        printf("break\n");
    }
    else if (12) {
        printf("continue\n");
    }
    else {
        printf("goto labb\n)
        printf("12");
    }

    printf("13");
}

printf("14");

if (a > 3) {
    printf("1\n");
```

```
}
else {
    printf("2\n");
}

i = 0;
while (i < 10) {
    if (a > b) {
        i = i + 2;
    }
    else if (2 * 3) {
        printf("break\n");
    }
    else {
        printf("continue\n");
    }
}

if (1) {
    if (2) {
        if (3) {
            x()
        }
        else {
```

```
            y();
        }
    }
    else {
        z();
    }
}

if (a > 3) {
    printf("1\n");
}
else {
    printf("2\n");
    printf("22\n");
    printf("222\n");
}

while (a < 3) {
    printf("3\n");
}

printf("Hello!\n");
return 0;
}
```