Санкт-петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 2

Вариант 2

По дисциплине «Системное программное обеспечение»

Выполнила: Носкова Е.Е.
Группа № P4114
Проверил: Кореньков Ю. Д.

Санкт-Петербург
2023

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

Узел графа потока управления (целевые узлы для безусловного и условного перехода (по мере необходимости), дерево операций, ассоциированных с данным местом в алгоритме, представленном в исходном тексте подпрограммы).

```
typedef struct GraphNode {
    int traverseTag;
    struct GraphNode* usl;
    struct GraphNode* busl;
    char* data;
} GraphNode;
```

Порождение узлов графа потока управления и формирование между ними дуг в зависимости от синтаксической конструкции.

```
GraphNode* drawBranch(GraphNode* curr, ASTNode* ast)
{
    printf("drawBranch for %s\n", ast->type);
    if (strcmp("if_statement", ast->type) == 0) {
        GraphNode* a = createCfgNode(ast->left->type);
        GraphNode* b = createCfgNode(NULL);
        GraphNode* c = createCfgNode(NULL);
        curr->busl = a;
        a->busl = b;
        a->usl = c;

        GraphNode* d =  drawBranch(b,ast->right->left);
        GraphNode* e = drawBranch(c, ast->right->right);
        GraphNode* f = createCfgNode(NULL);
        d->busl = f;
        e->busl = f;
        return f;

    }
    else if (strcmp("while", ast->type) == 0){
        GraphNode* a = createCfgNode(ast->left->type);
        GraphNode* b = createCfgNode(NULL);
        GraphNode* c = createCfgNode(NULL);
        curr->busl = a;
        a->usl = b;
        a->busl = c;

        GraphNode* d = drawBranch(b, ast->right);
        d->busl = a;
        return c;
    }
    else if (strcmp("expression_statement", ast->type) == 0) {
        GraphNode* a= createCfgNode(ast->left->type);
        curr->busl = a;

        return a;

    }
```

```
    else if (strcmp("compound_statement", ast->type) == 0) {
        GraphNode* a = drawBranch(curr, ast->left);
        return a;
    }
    else if (strcmp("block_item_list", ast->type) == 0) {
        GraphNode* a = drawBranch(curr, ast->left);
        GraphNode* b = drawBranch(a, ast->right);
        return b;
    }
    else {
        printf("unexpected control-flow statement %s\n", ast->type);
        return curr;
    }


}
```

Файл control_flow_graph.dgml для вывода графа потока управления.

```
void traverseForNode(FILE* f, int tag, GraphNode* node)
{
    if (node != NULL && node->traverseTag != tag) {
        node->traverseTag = tag;
        fprintf(f, "<Node Id=\"#%lx\" Label=\"#%lx&#10;%s\" />\n", (long)node, (long)node, node->data);
        traverseForNode(f, tag, node->busl);
        traverseForNode(f, tag, node->usl);
    }
}
void traverseForLinks(FILE* f, int tag, GraphNode* node)
{
    if (node != NULL && node->traverseTag != tag) {
        node->traverseTag = tag;
        if (node->busl != NULL) {
            fprintf(f, "<Link Source=\"#%lx\" Target=\"#%lx\"/>\n", (long)node, (long)node->busl);
            traverseForLinks(f, tag, node->busl);
        }
        if (node->usl != NULL) {
            fprintf(f, "<Link Source=\"#%lx\" Target=\"#%lx\"/>\n", (long)node, (long)node->usl);
            traverseForLinks(f, tag, node->usl);
        }
    }
}

void traverseForCfgDgml(GraphNode* node, char* fileName)
{
    FILE* file = fopen(fileName, "w+");
    fprintf(file, "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n");
    fprintf(file, "<DirectedGraph xmlns=\"http://schemas.microsoft.com/vs/2009/dgml\">\n");
    fprintf(file, "<Nodes>\n");
    traverseForNode(file, 3, node);
    fprintf(file, "</Nodes>\n");
    fprintf(file, "<Links>\n");
    traverseForLinks(file, 4, node);
    fprintf(file, "</Links>\n");
    fprintf(file, "</DirectedGraph>");
    fclose(file);

}
```

void traverseAST(ASTNode* node):

```
void traverseAST(ASTNode* node) {

    if (node) {

        printf("Type: %s\n", node->type);
        printf("Value: %s\n", node->value);
        printf("Id: %d\n", node->id);


        if (strcmp("body", node->type) == 0) {

            GraphNode* cfgStart = createCfgNode(node->right->type);
            printf("1: %s\n", node->right->type);
            drawBranch(cfgStart, node->right);
            traverseForCfgDgml(cfgStart, "control_flow_graph.dgml");
        }
        else {
            traverseAST(node->left);
            traverseAST(node->right);
        }
    }
}
```

Примеры входных и выходных данных:

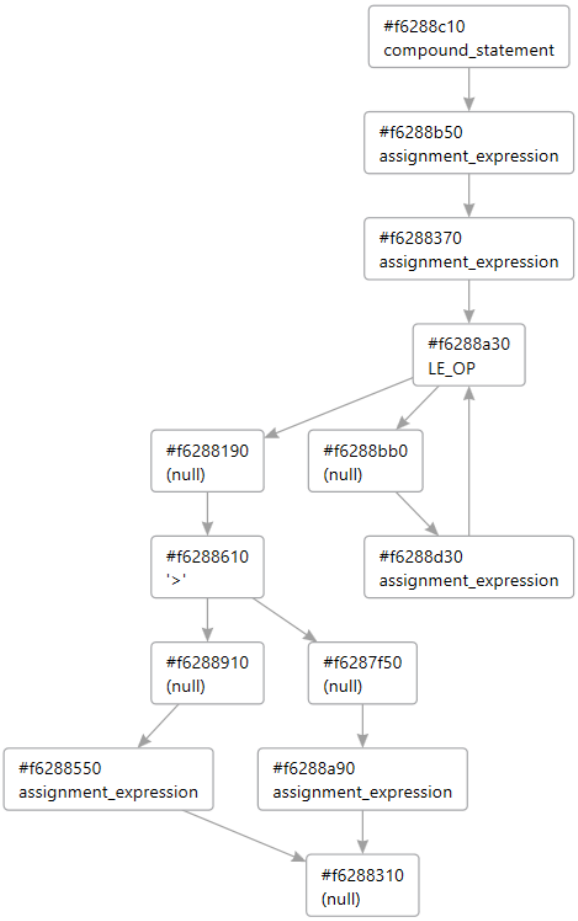
Входные данные 1.

```
method f(x:int,y:int) :int
var
a,b:long;
f,g:int;
begin
    a=g+b;
    c=d;
    while w<=0 do
    q=x+e;

   if w>33 then
        a = x+y+z;
        else
        a = z+y+x;

end;
```

Результат 1.

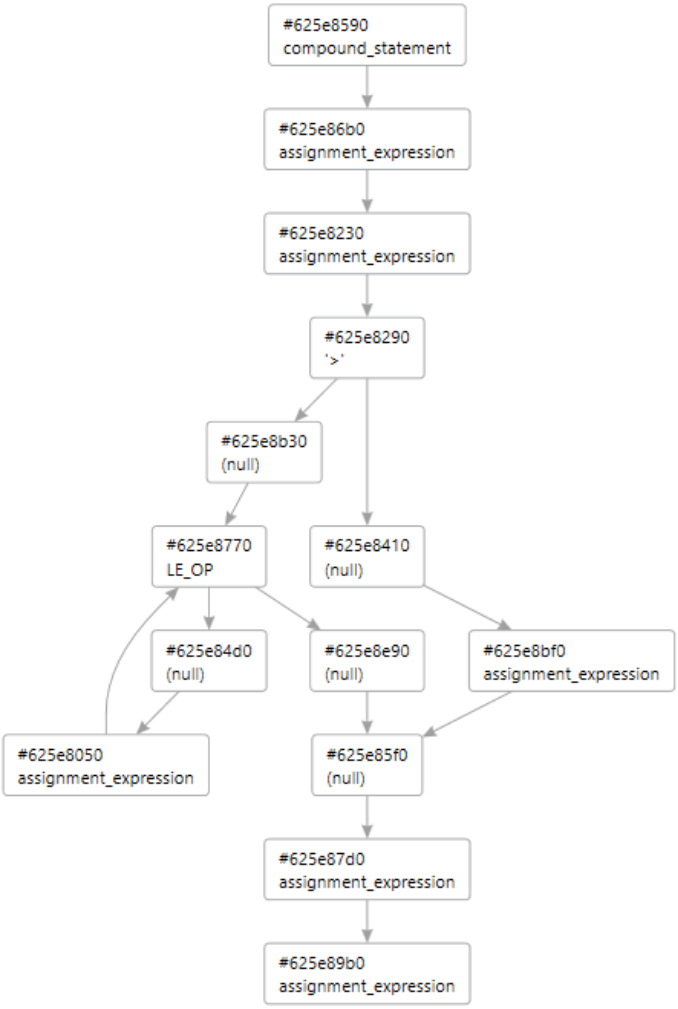


Входные данные 2.

```
method f(x:int,y:int) :int
var
a,b:long;
f,g:int;
begin
    a=g+b;
    c=d;
        if w>33 then
            while w<=0 do
            a = x+y+z;
        else
        a = z+y+x;
    a=q+b;
    c=d;

end;
```

Результат 2.



Вывод:

В ходе выполнения лабораторной работы реализовано построение графа потока управления посредством анализа дерева разбора для набора входных файлов. Выполнен анализ собранной информации и сформировать набор файлов с графическим представлением для результатов анализа.