МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт Петербургский национальный-исследовательский университет Информационных технологий, механики и оптики»

Факультет программной инженерии и компьютерной техники

«Системное программное обеспечение»

Выполнил студент группы №P4116
Веденин Вадим Витальевич

Проверил
Кореньков Юрий Дмитриевич

САНКТ -ПЕТЕРБУРГ
2023

Цели

Использовать средство синтаксического анализа, реализовать модуль для разбора текста по заданному языку. Реализовать построение по исходному файлу с текстом синтаксического дерева с узлами, соответствующими элементам синтаксической модели языка. Полученное дерево вывести в файл в формате, поддерживающем просмотр графического представления.

Задачи

1. Изучить выбранное средство синтаксического анализа
    a. Средство должно поддерживать программный интерфейс, совместимый с языком Си
    b. Средство должно параметризоваться спецификацией, описывающей синтаксическую структуру разбираемого языка
    c. Средство может функционировать посредством кодогенерации и/или подключения необходимых для его работы дополнительных библиотек
    d. Средство может быть реализовано с нуля, в этом случае оно должно использовать обобщённый алгоритм, управляемый спецификацией
2. Изучить синтаксис разбираемого по варианту языка и записать спецификацию для средства синтаксического анализа, включающую следующие конструкции:
    a. Подпрограммы со списком аргументов и возвращаемым значением
    b. Операции контроля потока управления – простые ветвления if-else и циклы или аналоги
    c. В зависимости от варианта – определения переменных
    d. Целочисленные, строковые и односимвольные литералы
    e. Выражения численной, битовой и логической арифметики
    f. Выражения над одномерными массивами
    g. Выражения вызова функции
3. Реализовать модуль, использующий средство синтаксического анализа для разбора языка по варианту
    a. Программный интерфейс модуля должен принимать строку с текстом и возвращать структуру, описывающую соответствующее дерево разбора и коллекцию сообщений ошибке
    b. Результат работы модуля – дерево разбора – должно содержать иерархическое представление для всех синтаксических конструкций, включая выражения, логически представляющие собой иерархически организованные данные, даже если на уровне средства синтаксического анализа для их разбора было использовано линейное представление
4. Реализовать тестовую программу для демонстрации работоспособности созданного модуля
    a. 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
    b. Сообщения об ошибке должны выводиться тестовой программной (не модулем, отвечающим за анализ!) в стандартный поток вывода ошибок
5. Результаты тестирования представить в виде отчета, в который включить:
    a. В части 3 привести описание структур данных, представляющих результат разбора текста (3a)

b. В части 4 описать, какая дополнительная обработка потребовалась для результата разбора, предоставляемого средством синтаксического анализа, чтобы сформировать результат работы созданного модуля
c. В части 5 привести примеры исходных анализируемых текстов для всех синтаксических конструкций разбираемого языка и соответствующие результаты разбора

Описание решения

Описание структур данных

Узел формирующегося дерева представлен следующей структурой:

```
struct ASTNode {
    int id;
    char* type;
    char* value;
    ASTNode* left;
    ASTNode* right;
};
```

Описание дополнительной обработки:

Создание узла осуществляется при помощи функции:

```
ASTNode* createNode(char* type, char* value, ASTNode* left, ASTNode* right) {
    ASTNode* node = malloc(sizeof(ASTNode));

    node->type = type;

    char* buff = malloc(1024 * sizeof(char));
    strcpy(buff, value);
    node->value = buff;

    node->left = left;
    node->right = right;

    allNodes[allNodesCount] = node;
    allNodesCount++;

    return node;
}
```

Построенное дерево выводится в консоль в формате языка dot:

```
void printNodeValue(ASTNode* node) {
    printf("\"Type: %s, Id: %d", node->type, node->id);
    if (strlen(node->value) > 0) {
        printf(", Value: %s", node->value);
    }
    printf("\"");
}

void printNode(ASTNode* node) {
    if (node->left) {
        printNodeValue(node);
        printf(" -> ");
        printNodeValue(node->left);
        printf(";\n");
        printNode(node->left);
    }
    if (node->right) {
        printNodeValue(node);
        printf(" -> ");
        printNodeValue(node->right);
        printf(";\n");
        printNode(node->right);
    }
}

void printAST() {
    for (int i = 0; i < allNodesCount; i++) {
        allNodes[i]->id = i;
    }

    printf("digraph G {\n");
    printNode(allNodes[allNodesCount - 1]);

    printNodeValue(allNodes[allNodesCount - 1]);
    printf(" [shape=Mdiamond];\n");
    printf("}\n");
}
```

В дальнейшем можно представить граф в графическом виде при помощи утилиты. Для этого реализована цель Makefile:

```
run: result
	./result input.txt > output.txt && iconv -f ISO-8859-1 -t utf-8 output.txt >
utf8_output.txt && cat utf8_output.txt | graph-easy --html --output
result_tree.html
```

Пример

Входные данные:

```
//int commented_func (int arg1) {
//    while (i <= 10) {
//        if (i > 3) {
//            r = 5;
//        }
//}


int test_func() {
    while ( i >= 5) {
        if (i == 1) {
            x = 6;
            str1 = "strA";
        }
    }
//    b = (4 + 5) * 2;
    str2 = "strB";
}
```

Результат обработки: