MT-02-lab

Владислав Кузнецов

November 2020

1 Задание 10. Заголовок функции в C

Заголовок функции в Си. Заголовок начинается именем возвращаемого типа или словом "`void` далее идет имя функции, скобка, затем разделенные запятой описания аргументов. Переменная может быть указателем, в этом случае перед ней идет звездочка (возможны и указатели на указатели, и т. д.). Аргументов может быть несколько.
Используйте один терминал для всех имен переменных и имен типов.
Пример: `int fib(int n);`

2 Разработка грамматики

Разработаем грамматику:

```
function -> name(*)*name(variables)
variables -> variables'
variables -> e
variables' -> variable,variables'
variables' -> variable
variable -> name(*)* name
name -> alphabet name'
name' -> all name'
alphabet -> [a-zA-Z]
all -> alphabet
all -> [0-9_]
```

3 Лексический анализатор

Лексический анализатор находится в пакете `ru.parser.terminal`. В этом пакете содержится три класса - `Terminal`, `Token` и `Tokenizer`. Первый содержит является множеством токенов. Для уменьшения нагрузки на анализатор, там реализованы функции `isFirst` (может ли данный символ являться началом токена), `isContinue` (может ли данный символ являться продолжением токена (актуально для имени)), `isContainData` (содержит ли токен какие либо данные (актуально для имени)). Таким образом, добавление одного токена почти любого токена затратит у вас не больше 9 строк кода (и вам не нужно будет искать это в case switch и так далее). Второй класс это просто токен с данными, если таковые потребуются. И, наконец, третий класс это токенайзер. Ему уже ничего не нужно, просто проверить все токены на первый символ, и продолжить идти по нему.

4 Синтаксический анализатор

Синтаксический анализатор находится в пакете `ru.parser.tree`. В этом пакете лежат все типы нетерминалов, и их класс-родитель. Конструктор всегда вызывается от токенайзера, заполняются все данные.

5 Визуализация дерева разбора

Лежит в `resources`. Вызвана относительно первых тестов

6 Тесты

Переход	Тест
function -> name(*)*name(variables)	any
variables -> variables’	any
variables -> e	test1_without_any
variables’ -> variable,variables’	test5_references
variables’ -> variable	test2_one_variable
variable -> name(*)* name	any
name -> alphabet name’	any
name’ -> all name’	test4_digit_in_name
alphabet -> [a-zA-Z]	test4_digit_in_name
all -> alphabet	test4_digit_in_name
all -> [0-9_]	test_2_variables