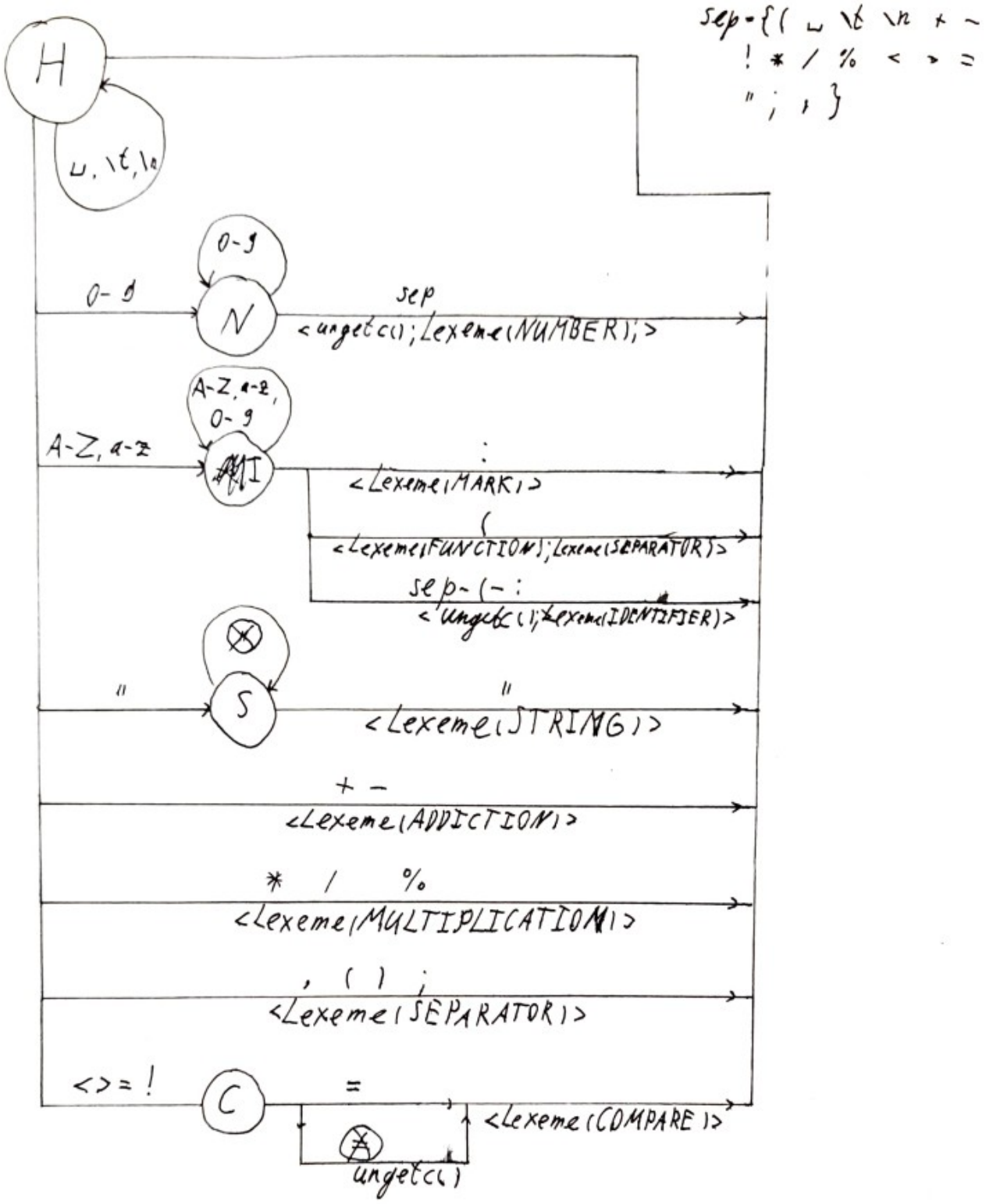
Сергеев Максим 212

Task 10. Отчёт.

Этап 1. Лексический анализ

На этапе лексического анализа текст данной программы обрабатывается по следующей схеме:



Лексический анализатор реализован в виде класса LexAnalyser.

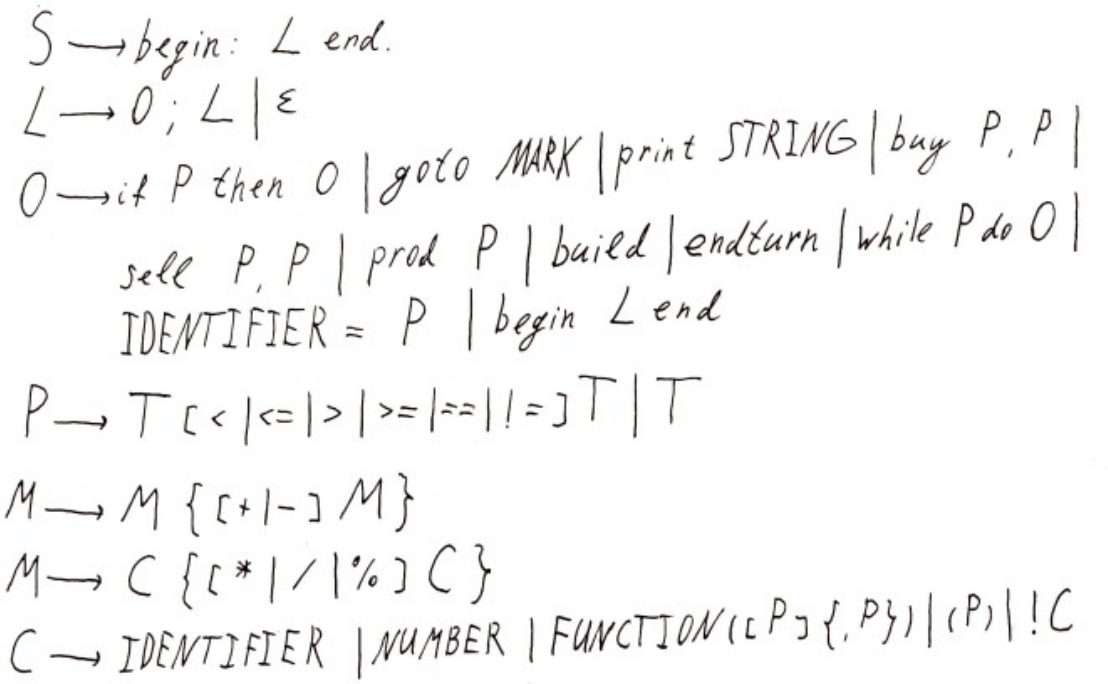
Результатом работы лексического анализатора является список объектов типа Lexeme. Хранится полученный список в виде открытого поля класса LexAnalyser.

Тип Lexeme содержит два поля: имя (строка — лексема из данного файла) и тип (переменная перечислимого типа). Типы лексем:

- NUMBER — для целых чисел, заданных в тексте программы
- MARK — для объявленных меток
- FUNCTION — для вызываемых функций
- KEY — для ключевых слов языка
- IDENTIFIER — для идентификаторов, используемых в тексте (переменные и метки в операторе перехода)
- STRING — для строк, заключённых в кавычки
- SEPARATOR — для символов-разделителей: , ; ( )
- COMPARE — для операторов сравнения и присваивания
- ADDICTION — для операторов того же приоритета, что и +
- MULTIPLICATION — для операторов того же приоритета, что и *
- NOT — для оператора ! (логическое отрицание)

Этап 2. Синтаксический анализатор

На этапе синтаксического анализа данная программа проверяется на соответствие языку рекурсивным спуском по следующей грамматике:



Реализован синтаксический анализатор в виде объекта класса SynAnalyser. Конструктор класса требует список лексем, сформированный на предыдущем этапе. При выявлении несоответсвия языку выбрасывается ошибка SynError с соответсвующим текстом.

Этап 3. Семантический анализ

Семантический анализ происходит одновременно с синтаксическим. Для этого используется стек типов, таблицы глобальных переменных, меток и функций. На этом этапе проверяется соответствие типов операндов операциям, правильное использование идентификаторов.

Этап 4. Генерация внутреннего представления

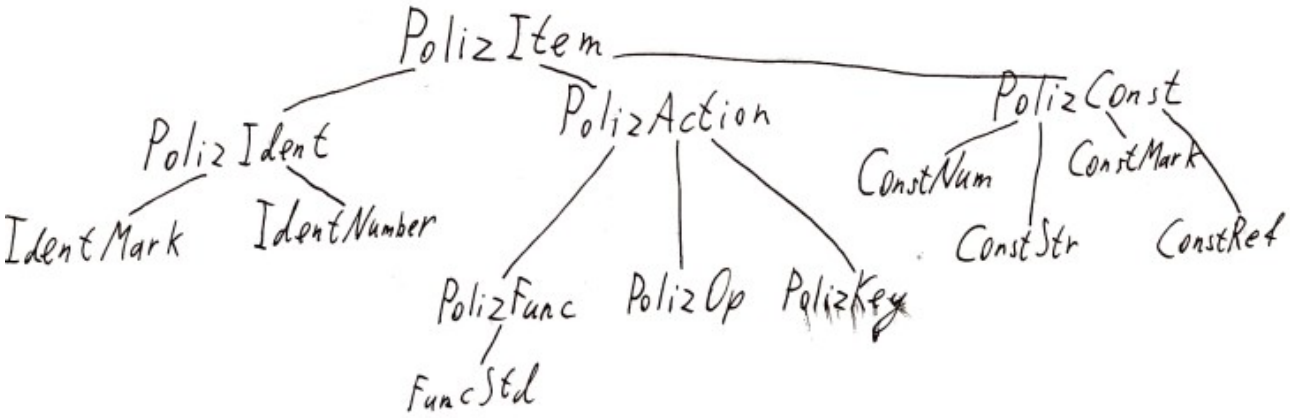
Одновременно с синтаксическим анализом формируется внутреннее представление программы в виде ПОЛИЗа. ПОЛИЗ зранится в списке ссылок на IdentItem (см. приложение). На этом этапе ключевые слова if и while заменяются на соответствующие конструкции, содержащие условные и безусловные переходы.

Этап 5. Выполнение

На этом этапе программа, записанная в виде ПОЛИЗа выполняется при помощи стека ссылок на ПОЛИЗ-константы. Для каждого элемента ПОЛИЗа выполняется виртуальная функция run(). Методы элементов-констант и идентификаторов вносят в стек свои значения. Методы-операции выбирают из стека нужное количество значений, выполняют действия над ними и вносят результат обратно в стек. Особый оператор ; полностью очищает стек от значений, оставшихся от предыдущих операций.

Приложение

Типы ПОЛИЗа



- PolizItem. Абстрактный класс с одной число виртуальной функцией run. Используется для хранинеия объектов классов-наследников в виде мессива ссылок. В функцию run по ссылке передаётся итератор ПОЛИЗа, который будет перемещён на следующую команду при выполнении (не зависимо от объекта), и стек ссылок на тип PolizConst, из которого операции будут извлекать операнды и в который будут помещать результаты.
- PolizIdent. Абстрактный класс для идентификаторов (run не замещяется). Содержит закрытые поля: строка name и прерменная перечислимого типа type (принимает значения INT, BOOL, STR, MRK, REF, но не все эти типы встречаются в идентификаторах).
- IdentMark. Класс для хранения меток. Содержит закрытое поле value — итератор ПОЛИЗа. Для класса определено 2 конструктора: по строки и итератору, вносящий значение, и по одной строке, определюящей только имя и оставляющей значение пустым. Оба конструктора устанавливают тип MRK. Открытый метод get_i возвращает значения поля value. Функция run помещает в стек объект типа ConstMark, инициализировынный собственным значением.
- IdentNumber. Класс для хранения числовых переменных. Закрытое поле типа int — value. Один коструктор с трея параметрами: строка-имя, тип (допускается INT и BOOL), значения. Для последних двух параметров установлены значения по умолчанию. Определена открытая функция set, получающая объект ConstNum в качестве параметра и устанавливающая по нему значение переменной. Функция run помещает в стек объект типа ConstNum, инициализированный собственным значением.
- PolizAction. Абстрактный класс для объектов, выполняющих с данными какие-то действия. Функция run не замещена.
- PolizFunc. Абстрактный класс для функций. Содержит 4 закрытых поля: строку-имя, список объектов типа IdentNumber — параметров функции, объект типа IdentNumber — результирующее значение, переменная типа int — количество параметров.

Конструктор требуте только строку-имя. Метод check необходим для этапа семантического анализа. Он извлекает типы из стека семантического анализа и проверяет их соответствие типа параметров. Функция par_num возвращает количество параметров. Определены операторы стравниния с объектами класса PolizFunc и строками. Сравниение происходит только п имени.

- FuncStd. Класс для предопределённых функций. Конструктор преобразования создаёт объект по строке-имени. Определены методы для добавления параметров и установки типа результата. Функция run извлекает параметры из стека и выполняет действия с ними. Результат вносится обратно в стек.
- PolizOp. Класс для операций (арифметических, логических, переходи и изменения значения). Содержит закрытое поле type. Конструктор требует страку-запись поерации в программе. По этой строке устанавливается type. Функция run, в зависимости от type, извлекает нужные параметры и выполняет соответствующие действия. Результат (если есть) заносится в стек.
- PolizKey. Класс для действий, заданных ключевыми словами. Содержит закрытое поле type. Конструктор по строке-записи устанавливает значение type. Функция run, в зависимости от type, извлекает нужные параметры и выполняет соответствующие действия. Действия ключевых слов не имеют результата, поэтому в стек ничего не заносится.
- PolizConst. Абстрактный тип для хранинния значений-констант. Содержит поле type того же типа, что и в идентификаторах. Конструктор требует значение type. Функция run не замещена.
- ConstNum. Тип для числовых переменных. Содержит поле value типа int. Конструктор с параметрами строкой-записью и типом находит по записи значение и устанавливает тип. Конструктор с параметром-идентификатором инициализирует объект параметром. Конструкторы с параметрами типов int и bool устанавливают соответствующие тип и значение. Определены операторы приведения к int и bool. Второй допускает только явное применение. Функция run вносит в стек копию самого объекта.
- ConstStr. Тип для строковых констант. В закрытом поле value хранится значение. Инициализируется объкт строкой, из которой удаляются кавычки. Функция run вносит в стек копию самого объекта. Так же определена функция-друг — operator<< для вывода значения в ostream.
- ConstMark. Класс для констант-меток. Содержит поле value — итератор ПОЛИЗА. Инициализируется итератором или маркой-идентификатором. Конструктор умолчания с пустым телом. Функция set_v устанавливает value. Определён оператор преобразования в итератор ПОЛИЗа. Функция run вносит в стек копию самого объекта.
- ConstRef. Класс для хранинния указателей на идентификаторы. Содержит закрытое поле value — указатель на IdentNumber. Инициализируется ссылкой на IdentNumber. Функция get_ref возвращает ссылку на переменную, на которую указывает объект. Функция run вносит в стек копию самого объекта.