Задание к курсовой работе по дисциплине

**“Теория формальных языков и компиляторов”**

### Вариант N

**Цели работы:**

* практическое применение теоретических основ проектирования трансляторов для языков программирования;
* освоение средств автоматизации построения трансляторов;
* разработка основных элементов транслятора для заданного учебного языка.

### I. Содержание работы

1. Разработать учебный язык программирования согласно заданному варианту, описать его лексику, синтаксис и семантику. Написать несколько простых тестовых программ, содержащих все заданные элементы и управляющие конструкции языка для использования их при проверке составных частей разрабатываемого транслятора.
2. Разработать систему регулярных выражений, определяющую лексику заданного варианта языка. Используя пакет ВебТрансБилдер, построить автоматную реализацию лексического анализатора на выбранном инструментальном языке (рекомендуется выбирать язык javascript), добиться правильного преобразования последовательности символов в последовательность лексем.
3. Разработать формальную грамматику класса не выше, чем LL(1) или LALR(1), определяющую синтаксис заданного языка. Используя пакет ВебТрансБилдер, построить автоматную реализацию синтаксического акцептора, обеспечить правильность распознавания тестовых программ на заданном языке.
4. Разработать совокупность действий расширения синтаксического акцептора, выполняющую преобразование входной последовательности лексем в постфиксную форму записи (ПФЗ) или в абстрактное синтаксическое дерево (АСД). Добиться работоспособности полученного анализатора на тестовых программах.
5. Разработать семантический анализатор и преобразователь ПФЗ (или АСД) в псевдокод того формата, который определен заданием.
6. Разработать интерпретатор псевдокода. Добиться правильного исполнения интерпретатором тестовых программ на заданном языке.
7. Оформить (в электронном виде) расчетно-пояснительную записку, содержащую:
   * Задание (эта страница и выборка колонок из таблиц раздела II по варианту).
   * Оглавление.
   * Введение.
   * Описание учебного языка.
   * Тестовая программа (программы) на заданном языке.
   * По пунктам 2 и 3:
   * система регулярных выражений ( п. 2 ) и формальная грамматика ( п. 3 ) в виде текстов (не включать скриншоты или полные копии таблиц из окон, формируемых ВебТрансБилдер’ом, преобразовывать их содержимое в читаемые текстовые форматы);
   * фрагменты управляющих таблиц конечных автоматов, построенных ВебТрансБилдер’ом (не скриншоты), описания функционирования автоматов; примеры результатов работы автоматов с тестовыми примерами; объяснение и анализ этих результатов.
   * По каждому из пунктов 4, 5 и 6:
   * полное описание разработанного алгоритма и коды программ в виде приложений (в случае их большого объема включать в записку только фрагменты, содержащие наиболее важные части алгоритма; описание алгоритма можно не включать, если программы приводятся полностью и содержат детальные комментарии)
   * примеры результатов работы компонент транслятора с правильными и ошибочными входными программами; объяснение и анализ этих результатов.
   * Заключение.

Объем записки **не должен** превышать 25-30 страниц.

### II. Учебный язык, основные требования.

Программа на учебном языке представляет собой файл, который может содержать комментарии, объявления глобальных переменных и констант, определения функций, вызывающих друг друга (возможно рекурсивно) и возвращающих результат выполнения.

Расшифровка цифр варианта задания:

***Лексика***

1. *Идентификаторы*
2. $<пЦ><пБ> (например: $12aD, $7field, $Member, …)
3. <чпБЦ> (например: v3R5, Q9k, h, m1, …)
4. <б><пБЦ> (например: v5Zx, ig7h, popART, …)
5. <пБ><пЦ> (например: v5, i731, ArT19, …)
6. <бБ><пЦ><бБ> (например: x123y, N1n, a0B, xY…)
7. *Константы*
8. целые по основанию 10; вещественные и экспоненциальные; символьные
9. целые по основаниям 4, 8 и 10; вещественные; строковые и символьные
10. целые по основаниям 4, 10 и 16; вещественные; строковые и символьные
11. целые по основаниям 2, 8 и 10; вещественные; строковые и символьные
12. целые по основаниям 2, 4 и 10; вещественные; строковые и символьные

В символьных и строковых константах должна быть обеспечена возможность указывать любые символы, в том числе не имеющие графического изображения или соответствующей клавиши на клавиатуре.

Правильными словами языка являются также знаки арифметических операций, логических операций и операций сравнения, комментарии, а также все последовательности символов, выделенные жирным шрифтом в разделах «Синтаксис» и «Семантика» задания (за исключением символов «**[**» и «**]**», см. раздел «Обозначения»).

***Синтаксис***

1. *Оператор присваивания*
2. **let** <И> **on** <В> **;**
3. <И> **<–** <В> **;**
4. **set (** <В> , <И> **)** **;**
5. <И> **:=** <В> **;**
6. **put**<В> **to** <И> **;**
7. *Условный оператор*
8. **by (** <В> **)** <ОБ> **[ else** <ОБ> **]**
9. **when** <В> <ОБ> **[ other** <ОБ> **]**
10. **in (** <В> **)** **then** <ОБ> **[ not**<ОБ> **]**
11. **at** <В> **do** <ОБ> **[ or do** <ОБ> **]**
12. **? (** <В> **)** <ОБ> **[ ?: (** <В> **)** <ОБ> **] [ :** <ОБ> **]**
13. *Оператор цикла*

Для выхода из цикла его тело ОБ (оператор или блок) может содержать оператор, указанный в дополнительных квадратных скобках; таких операторов в теле одного цикла может быть несколько (**Внимание!** Эти операторы являются точными аналогами оператора «break;» в С-подобных языках)

1. **exec** <ОБ> **with** <И> **from** <К> **to** <К> **[ step** <K> **]** **[** **exit; ]**
2. **cycle ([** <Пр> **] ; [** <В> **] ; [** <Пр> **])** <ОБ> **[** **quit; ]**
3. **while (** <ЛВ> **) do** <ОБ> **[** **leаve; ]**
4. **foreach (** <И> **in** <K> **:** <K> **)** <ОБ> **[** **stop; ]**
5. **loop** <ОБ> **until [** <В> **)** **[** **retire; ]**
6. *Оператор переключателя*

Для выхода из переключателя последовательность операторов/блоков любой ветки может завершаться оператором, указанным в квадратных скобках. (**Внимание!** Эти операторы являются точными аналогами оператора «break;» в С-подобных языках)

1. **select** <В> **case** **(** <К> **)** <ПО> **[ break; ] [ case … ] … [ case ( false )** <ПО> **]** **end**
2. **case** <В> **{ when** <К> **then** <ПО> **[ exit; ]** **[ when…] … [** **otherwise** <ПО> **]** **}**
3. **switch** <В> **{** **by** <К> **do** <ПО> **[leave; ]** **[ by …] … [ any do** <ПО> **]** **}**
4. **??** <В> **{ ?=** <К> **:** <ПО> **[ quit; ] [ ?= … ] …** **[ ?~** **:** <ПО> **]** **}**
5. **choice** <В> **option** <К> **:** <ПО> **[ fin; ]** **[ option … ] … [ nooption** <ПО> **]** **end**
6. *Объявление функций*

Некоторые приведенные ниже примеры нужно адаптировать к заданному в варианте задания виду оператора присваивания.

1. **[** <Тип> **]** <И> **( [**<АргЛист>**] )** <Тело> (например int \_Mrg\_(float \_Af\_) {…})
2. <И> **( [** <АргЛист> **] ) [ ret** <Тип> **]** <Тело> (например $3Pi() ret long {…} )
3. **[** <Тип> **] ( [** <АргЛист> **] )** <Тело> (например $0var = int (double $5t) { … };)
4. <И> **=> [ as** <Тип> **] ( [** <АргЛист> **] )** <Тело> (например i1let => as char () {…} )
5. **function ( [** <АргЛист> **] )** <Тело> **->** <И> (например function(int a1Arg) { … } -> f0nc)

***Семантика***

1. *Объявления типов*
2. **Обязательны, нестрогая типизация (возможны неявные преобразования)**
3. **Могут присутствовать, но не обязательны**
4. **Отсутствуют (явные преобразования возможны)**
5. **Обязательны, строгая типизация (неявные преобразования запрещены)**
6. **Отсутствуют (явные преобразования запрещены)**
7. *Формат псевдокода виртуальной машины (интерпретатора)*
8. Триады: <Код><Оп><Р>
9. Пентады: <Метка><Код><Оп><Оп><Р>
10. Тетрады: <Метка><Код><Оп><Оп>
11. Триады: <Код><Оп><Оп>
12. Тетрады: <Код><Оп><Оп><Р>

### III. Обозначения:

**[...]** – квадратные скобки охватывают необязательную часть конструкции; в объявлениях функций обязательность присутствия типа возвращаемого значения определяется восьмой цифрой варианта задания;

**…** – предшествующая часть конструкции повторяется произвольное количество раз;

< … > – сокращения:

<б> – одна маленькая буква;

<Б> – одна большая буква;

<бБ> – одна любая буква;

<бЦ> – одна маленькая буква или одна цифра;

<Ц> – одна цифра;

<пБ> – непустая последовательность букв;

<пЦ> – последовательность цифр длины от 0 до 3;

<пБЦ> – возможно пустая последовательность букв и/или цифр

<чпБЦ> – непустая последовательность чередующихся букв и цифр, начинающаяся с букв

<И> – идентификатор (имя переменной или объекта);

<К> – константа;

<В> – выражение;

<ЛВ> – выражение, значение которого может быть приведено к логическому, т.е. к true или false;

<ОБ> – одиночный оператор или блок операторов;

<О> – одиночный оператор;

<Пр> – выражение присваивания;

<ПО> – возможно пустая последовательность из одного или нескольких операторов / блоков;

<Тип> – ключевое слово типа данных;

<Тело> – блок операторов (например { … } или begin … end);

<АргЛист> – список формальных аргументов функции;

<Код> – поле кода операции;

<Метка> – поле метки;

<Оп> – поле наименования операнда;

<Р> – поле наименования результата.

Любые конструкции, явно не определенные в этом задании (например – массивы, или указатели, или внутренние классы/интерфейсы/…), можно вводить в язык, однако делать этого не рекомендуется, сложность работы и без того высока.

Для того, чтобы по номеру варианта сформировать задание на учебный язык, нужно выбрать из пунктов *II-1 … II.9* подпункты, соответствующие цифрам варианта. Например, пусть номер варианта задания равен 445144411. Тогда набор основных конструкций учебного языка, транслятор для которого нужно реализовать в курсовой работе, будет определяться следующими требованиями:

*Идентификаторы:*

* + <пБ><пЦ> (например: i777, Art1, var528, tempo, …, но не v52987)

*Константы:*

* + целые по основаниям 2, 8 и 10; вещественные; строковые и символьные

*Оператор присваивания:*

* + **put**<В> **to** <И> **;**

*Условный оператор:*

* + **by (** <В> **)** <ОБ> **[ else** <ОБ> **]**

*Оператор цикла:*

* + **foreach (** <И> **in** <K> **:** <K> **)** <ОБ> **[** **stop; ]**

*Оператор переключателя:*

* + **??** <В> **{ ?=** <К> **:** <ПО> **[ quit; ] [ ?= … ] …** **[ ?~** **:** <ПО> **]** **}**

*Объявление функций:*

* + <И> **=> [ as** <Тип> **] ( [** <АргЛист> **] )** <Тело> (например i1t => as char () {…} )

*Объявления типов:*

* + **Обязательны, нестрогая типизация (возможны неявные преобразования)**

*Формат псевдокода виртуальной машины:*

* + Триады: <Код><Оп><Р>