

Лабораторная работа 14-2 (2 часа)
Языки программирования

Разработка лексического распознавателя (IV часть)

1. Разработайте регулярные выражения, постройте графы переходов и соответствующие конечные автоматы для лексем из следующей таблицы (см. лабораторные 11-12).

Фраза SVV-2015	лексема	примечание
integer string	t	ТИ: integer или string, значение по умолчанию: для integer – нуль, для string – пустая строка
идентификатор	i	ТИ: строка идентификатора, усеченная до 5 символов. Префикс: имя конструкции
литералы	l	integer или string, значение.
function	f	
declare	d	
return	r	
print	p	
main	m	
;	;	
,	,	
{	{	
}	}	
((
))	
+ - * /	v	

2. Доработайте приложение, удовлетворяющее следующим требованиям.

Приложение:

- осуществляет лексический анализ всех фраз исходного текста программы на языке SVV-2015 (см. лабораторную 14-1) и строит таблицу лексем (ТЛ) и таблицу идентификаторов (ТИ);

- формирует протокол работы: представляет содержимое таблицы лексем с разбивкой по строкам с нумерацией, соответствующей номерам строк исходного текста программы (во фрагменте ниже представлен примерный вид), а также диагностические сообщения (с применением таблицы ошибок) с указанием номера ошибочной строки, позиции в исходном тексте и описанием ошибки;

```
01 tfi(ti,ti)
02 {
03 dti;
04 i=i*(i+i);
05 ri;
06 };
...
```

- выводит содержимое таблицы идентификаторов со ссылками на индексы первого вхождения соответствующей лексемы в ТЛ;
- при обработке ошибок использовать таблицу ошибок из лабораторной 10, добавив в нее необходимые коды ошибок.

3. Необходимо подготовить примеры, в которых демонстрируется обработка всех типов ошибок.

4. При построении таблицы лексем используйте следующую спецификацию (h-файл LT.h)

```
#pragma once
#define LEXEMA_FIXSIZE 1 // фиксированный размер лексемы
#define LT_MAXSIZE 4096 // максимальное количество строк в таблице лексем
#define LT_TI_NULLIDX 0xffffffff // нет элемента таблицы идентификаторов
#define LEX_INTEGER 't' // лексема для integer
#define LEX_STRING 't' // лексема для string
#define LEX_ID 'i' // лексема для идентификатора
#define LEX_LITERAL 'l' // лексема для литерала
#define LEX_FUNCTION 'f' // лексема для function
#define LEX_DECLARE 'd' // лексема для declare
#define LEX_RETURN 'r' // лексема для return
#define LEX_PRINT 'p' // лексема для print
#define LEX_SEMICOLON ';' // лексема для ;
#define LEX_COMMA ',' // лексема для ,
#define LEX_LEFTBRACE '{' // лексема для {
#define LEX_BRACELET '}' // лексема для }
#define LEX_LEFTHESIS '(' // лексема для (
#define LEX_RIGHTHESIS ')' // лексема для )
#define LEX_PLUS '+' // лексема для +
#define LEX_MINUS '-' // лексема для -
#define LEX_STAR '*' // лексема для *
#define LEX_DIRSLASH '/' // лексема для /
```

```

namespace LT          // таблица лексем
{
    struct Entry      // строка таблицы лексем
    {
        char lexema[LEXEMA_FIXSIZE]; // лексема
        int sn;          // номер строки в исходном тексте
        int idxTI;       // индекс в таблице идентификаторов или LT_TI_NULLIDX
    };

    struct LexTable    // экземпляр таблицы лексем
    {
        int maxsize;    // емкость таблицы лексем < LT_MAXSIZE
        int size;       // текущий размер таблицы лексем < maxsize
        Entry* table;   // массив строк таблицы лексем
    };

    LexTable Create(    // создать таблицу лексем
        int size       // емкость таблицы лексем < LT_MAXSIZE
    );

    void Add(           // добавить строку в таблицу лексем
        LexTable& lextable, // экземпляр таблицы лексем
        Entry entry       // строка таблицы лексем
    );

    Entry GetEntry(     // получить строку таблицы лексем
        LexTable& lextable, // экземпляр таблицы лексем
        int n           // номер получаемой строки
    );

    void Delete(LexTable& lextable); // удалить таблицу лексем (освободить память)
};

```

5. При построении таблицы идентификаторов используйте следующую спецификацию (h-файл IT.h)

```

#pragma once
#define ID_MAXSIZE    5          // максимальное количество символов в идентификаторе
#define TI_MAXSIZE    4096      // максимальное количество строк в таблице идентификаторов
#define TI_INT_DEFAULT 0x00000000 // значение по умолчанию для типа integer
#define TI_STR_DEFAULT 0x00      // значение по умолчанию для типа string
#define TI_NULLIDX    0xffffffff // нет элемента таблицы идентификаторов
#define TI_STR_MAXSIZE 255
namespace IT           // таблица идентификаторов
{
    enum IDDATATYPE {INT=1, STR=2}; // типы данных идентификаторов: integer, string
    enum IDTYPE     {V=1, F=2, P=3, L=3}; // типы идентификаторов: переменная, функция, параметр, литерал

    struct Entry    // строка таблицы идентификаторов
    {
        int idxfirstLE; // индекс первой строки в таблице лексем
        char id[ID_MAXSIZE]; // идентификатор (автоматически усекается до ID_MAXSIZE)
        IDDATATYPE iddatatype; // тип данных
        IDTYPE idtype; // тип идентификатора
        union
        {
            {
                int vint; // значение integer
                struct
                {
                    char len; // количество символов в string
                    char str[TI_STR_MAXSIZE-1]; // символы string
                } vstr[TI_STR_MAXSIZE]; // значение string
            } value; // значение идентификатора
        };
    };
};

```

```

struct IdTable          // экземпляр таблицы идентификаторов
{
    int  maxsize;        // емкость таблицы идентификаторов < TI_MAXSIZE
    int  size;           // текущий размер таблицы идентификаторов < maxsize
    Entry* table;        // массив строк таблицы идентификаторов
};

IdTable Create(          // создать таблицу идентификаторов
                int size // емкость таблицы идентификаторов < TI_MAXSIZE
                );

void Add(               // добавить строку в таблицу идентификаторов
    IdTable& idtable,    // экземпляр таблицы идентификаторов
    Entry entry          // строка таблицы идентификаторов
);

Entry GetEntry(         // получить строку таблицы идентификаторов
    IdTable& idtable,    // экземпляр таблицы идентификаторов
    int n               // номер получаемой строки
);

int IsId(               // возврат: номер строки (если есть), TI_NULLIDX(если нет)
    IdTable& idtable,    // экземпляр таблицы идентификаторов
    char id[TI_MAXSIZE] // идентификатор
);

void Delete( IdTable& idtable); // удалить таблицу лексем (освободить память)
};

```