*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана***  ***(национальный исследовательский университет)»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_\_\_\_\_ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА \_\_\_\_\_\_\_ Компьютерные Системы и сети (ИУ6)\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по домашней работе № 2**

**Вариант 22**

**Дисциплина: машинно-зависимые языки и основы компиляции**

**Название лабораторной работы:** **Лексические и Синтетические анализаторы**

Студент гр. ИУ6-44  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шумаков А.А.**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Данилюк С.С.**

(Подпись, дата) (И.О. Фамилия)

Москва, 2018

**Задание**

***Целью*** домашнего задания № 2 по дисциплине «Машинно-зависимые языки и основы компиляции» является закрепление знаний теоретических основ и основных методов приемов разработки лексических и синтаксических анализаторов регулярных и контекстно-свободных формальных языков.

C++. Разработать грамматику и распознаватель языка описаний структурных типов данных, включающего описание массивов, строк и структур. Предусмотреть следующие типы элементов: float, int, char, unsigned char. Например:

**int aer[8];**

**char record[80];**

**struct student { char name[22]; char family[22]; int old; } st1,ft;**

**Выполнение**

Описание грамматики в форме Бэкуса-Наура

Выделим конструкции языка для массива:

<Массив> ::= <Тип данных> <Идентификатор><[><Индекс><];>

<Тип данных> ::= int | float | char | unsigned char

<Идентификатор> ::= <Буква> <;>| <Буква><цифра><;>

<Индекс> ::= <цифра>

<Буква> ::= [A..z]

<Цифра> ::= [0..9]

Выделим конструкции языка для структуры:

<Структура> ::=<Служебное слово><Наименование структурного типа данных><{> <элемент структуры><}><Идентификатор структуры;>

<Служебное слово> ::= struct

<Наименование структурного типа данных>::=<Буква>|<Буква><цифра>

<Элемент структуры> ::= <массив>|<поле>

<Поле>::=<Тип данных> <Идентификатор>

<Идентификатор структуры> ::= <Буква> <;>| <Буква><цифра><;>

<Индекс> ::= <цифра>

<Буква> ::= [A..z]

<Цифра> ::= [0..9]



**Рис.1 – Схема алгоритм**

**Синтаксическая диаграмма**



**Код программы**

#include "stdafx.h"

#include <iostream>

#include <cstring>

#include <conio.h>

#include <cstdio>

#include <string>

#include <cstdlib>

#include <regex>

using namespace std;

int main()

{

setlocale(LC\_CTYPE,"Russian");

string s;

while (s != "end")

{

cout << "Введите строку для распознавания" << endl;

getline(cin, s);

cout << "Изначальная строка:" << s << endl;

if (s == "end") {}

else {

//regex\_match

cmatch result;

regex regular\_expression("((int)||(float)||(char)||(unsigned char))"

"( )"

"(([A-Za-z]+)||([A-Za-z]+[\\digit]+))"

"(\\[)"

"([\\digit]+)"

"(\\];)"

);

regex regular\_struct(

"(struct )"

"([A-Za-z]+)"

"(( +\\{)||( +\\{ +)||(\\{)||(\\{ +))"

"(((int)||(float)||(char)||(unsigned char))( )([A-Za-z]+)(;||(\\[([\\digit]+)\\];))){1,}"

"(( +\\})||( +\\} +)||(\\})||(\\} +))"

"((([A-Za-z]+)||([A-Za-z]+[0-9]+))||((([A-Za-z]+)||([A-Za-z]+[0-9]+))(,[A-Za-z]+)||(,[A-Za-z]+[0-9]+))){1,}"

"(\\;)"

);

if (regex\_match(s.c\_str(), regular\_expression))

cout << "True" << endl;

else

{

if (regex\_match(s.c\_str(), regular\_struct))

cout << "True" << endl;

else

cout << "False" << endl;

}

// for (int i = 0; i < result.size(); i++)

// cout << result[i] << endl;

}

}

cout << "Завершение работы программы" << endl;

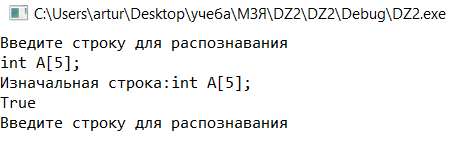
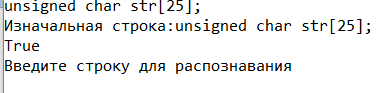
system("pause");

return 0;

}

**Тесты программы**

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| int A[5]; | True | True |
| unsigned char str[25]; | True | True |
| struct student{ char name[25]; char family[50]; int old;} st1,st2,st3; | True | True |
| struct namestruct{ int a[20]; float b[1];}zdat; | True | True |

**Рис.2 – Работа программы**

**Вывод**

Была разработана грамматика и синтаксис заданных выражений. Была разработана программа — анализатор, позволяющая проверить грамматику и синтаксис выражений.