**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и вычислительной техники

Кафедра ИиСП

**Пояснительная записка к расчетно-графической работе по курсу**

**«Теория автоматов и формальных языков»**

Выполнили:

студенты группы ПС-31

Винокуров Д.В.

Иванов Н.С.

Проверила:

преподаватель

Нехорошкова Л.Г.

г. Йошкар-Ола

2016

# Содержание

[Содержание 2](#_Toc440575088)

[Постановка задачи 3](#_Toc440575089)

[Состав бригады разработчиков и выполненная каждым работа 4](#_Toc440575090)

[Грамматика языка 4](#_Toc440575091)

[Множество терминальных символов 5](#_Toc440575092)

[Определение типа грамматики 5](#_Toc440575093)

[Принцип построения транслятора 5](#_Toc440575094)

[Структура транслятора, информационные потоки 6](#_Toc440575095)

[Используемые таблицы 7](#_Toc440575096)

[Тесты 8](#_Toc440575097)

[Проверочная программа сортировки пузырьком 10](#_Toc440575098)

[Вывод для проверочной программы 10](#_Toc440575099)

[Используемое программное обеспечение 11](#_Toc440575100)

# Постановка задачи

Написать транслятор с языка программирования придуманного Вами в ассемблер.

Язык должен удовлетворять следующим условиям:

|  |  |  |
| --- | --- | --- |
| ТРЕБОВАНИЯ | ЗНАЧЕНИЕ | ПОВЫШЕННЫЕ ТРЕБОВАНИЯ |
| Типы данных | Целое  Константы  Массив | Символьное  Словарь |
| Операторы | Ввода/вывода  Присвоения  (арифм.выражение)  Условия  Один из операторов цикла | Процедуры  Операторы цикла |

Требования к реализации- LL(1) или LR(1). Использование генераторов возможно для проверки грамматики на однозначность.

Написать на своем языке три программы:

Пузырьковая сортировка

Преобразование числа из одной системы исчисления в другую (до 16-ричной)

Перемножение матриц

# Состав бригады разработчиков и выполненная каждым работа

Винокуров Д.В. – синтаксический анализатор, семантический анализатор, генератор кода.

Иванов Н.С. – лексический анализатор, грамматика, таблицы разбора, генератор кода, документация.

# Грамматика языка

<code> -> <func> <other\_code>

<other\_code> -> $ | <code>

<func> -> <return\_type> <id> ( <idlist> ) { <explist> }

<return\_type> -> <var\_type> | void

<var\_type> -> int | string | char | float

<idlist> -> <id\_decl> <other\_decl> | $

<id\_decl> -> <var\_type> <id>

<other\_decl> -> , <id\_decl> <other\_decl> | $

<explist> -> <expression> <explist> | $

<expression> -> <decl\_action> | <op\_action> | <IO\_call> | <cycle> | <condition> | <return\_call>

<decl\_action> -> <var\_decl> <decl\_right> ; | <map\_decl> ;

<var\_decl> -> <var\_type> <id> <array\_part>

<map\_decl> -> map '<' <var\_type> , <var\_type> '>' <id>

<array\_part> -> $ | [ <right\_part> ] <second\_dimension>

<second\_dimension> -> $ | [ <right\_part> ]

<decl\_right> -> = <right\_part> | $

<right\_part> -> <right\_part> <low\_prior\_bin\_op> <mult> | <mult>

<mult> -> <mult> <high\_prior\_bin\_op> <exp\_part> | <exp\_part>

<exp\_part> -> ( <right\_part> ) | - ( <right\_part> ) | <left\_part> | - <left\_part> | <const>

<const> -> <const\_int> | <const\_string> | <const\_float> | <const\_char>

<low\_prior\_bin\_op> -> + | - | '|' | %

<high\_prior\_bin\_op> -> \* | / | &

<op\_action> -> <left\_part\_wo\_functions> = <right\_part> ;

<left\_part\_wo\_functions> -> <id> <array\_part>

<left\_part> -> <id> <next\_part>

<next\_part> -> <array\_part> | ( <func\_list> )

<func\_list> -> $ | <right\_part> <other\_func\_list>

<other\_func\_list> -> $ | , <right\_part> <other\_func\_list>

<IO\_call> -> cin >> <left\_part\_wo\_functions> ; | cout << <right\_part> ;

<cycle> -> while ( <cond\_expr> ) { <explist> } | for ( <var\_decl> = <right\_part> ; <cond\_expr> ; <id> = <right\_part> ) { <explist> }

<cond\_expr> -> <cond\_expr> <bool\_op> <rel\_expr> | <rel\_expr>

<rel\_expr> -> <exp\_cond\_part> <rel\_op> <exp\_cond\_part>

<exp\_cond\_part> -> ( <cond\_expr> ) | ! ( <cond\_expr> ) | <right\_part>

<bool\_op> -> && | '||'

<rel\_op> -> == | != | > | < | >= | <=

<condition> -> if ( <cond\_expr> ) { <explist> } <else>

<else> -> $ | else { <explist> }

<return\_call> -> return <right\_part> ;

# Множество терминальных символов

{буквы, цифры, ;, “,”, . , +, -, /, \*, &, |. %, <, >, [, ], (, ), void, int, string, float, char, = , cin >>, cout <<, while, for, if, {, }, else, return, &&, ||, ==, !=, <=, >=, map}

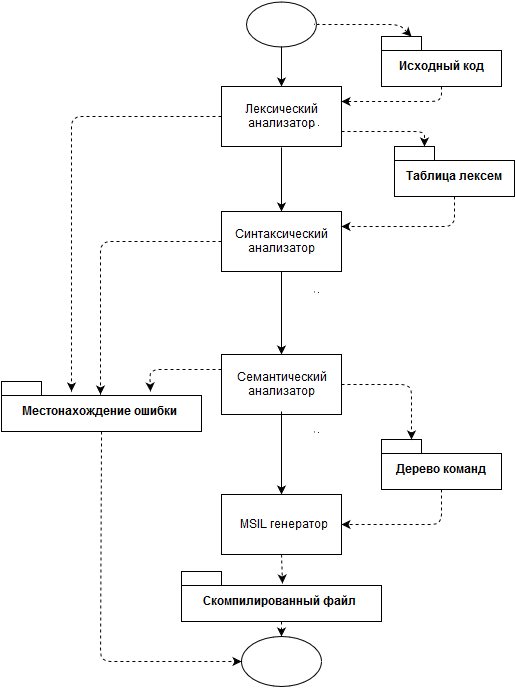
# Определение типа грамматики

Согласно классификации Хомского, наша грамматика относится к контекстно-свободным, так как левые части правил состоят из одиночного нетерминала, а правые – из набора терминалов и нетерминалов.

# Принцип построения транслятора

Исходный код подается на распознавание лексическому анализатору. Он либо возвращает ошибку, либо упорядоченный набор токенов, который использует синтаксический анализатор. Синтаксический анализатор проверяет синтаксическую правильность; семантический – семантическую и выводит в файл дерево команд для генерации. В случае прохождения семантической проверки, запускается генератор кода.

# Структура транслятора, информационные потоки



# Используемые таблицы

* Таблица лексем
* Таблица LL1 разбора
* Таблица SLR разбора
* Дерево команд
* Таблица переменных
* Таблица функций

# Тесты

Перемножение матриц

void main()

{

float a[2][2];

float b[2][2];

float result[2][2];

for (int i = 0; i < 2; i = i + 1)

{

for (int j = 0; j < 2; j = j + 1)

{

cin >> a[i][j];

}

}

for (int i = 0; i < 2; i = i + 1)

{

for (int j = 0; j < 2; j = j + 1)

{

cin >> b[i][j];

}

}

for (int i = 0; i < 2; i = i + 1)

{

for (int j = 0; j < 2; j = j + 1)

{

float res = 0;

for (int p = 0; p < 2; p = p + 1)

{

res = res + a[i][p] \* b[p][j];

}

result[i][j] = res;

}

}

for (int i = 0; i < 2; i = i + 1)

{

for (int j = 0; j < 2; j = j + 1)

{

cout << result[i][j];

cout << " ";

}

cout << '\n';

}

}

Перевод систем счисления

int ToDec(int num, int notation)

{

int result = 0;

int m = 1;

while (num > 0)

{

int rest = num % 10;

result = result + rest \* m;

num = num / 10;

m = m \* notation;

}

return result;

}

int ToNotation(int num, int notation)

{

int result = 0;

int m = 1;

while (num > 0)

{

int rest = num % notation;

result = result + rest \* m;

num = num / notation;

m = m \* 10;

}

return result;

}

void main()

{

int firstNum;

int notation;

int notationTo = -1;

cout << "Number: ";

cin >> firstNum;

cout << "Current notation: ";

cin >> notation;

cout << "Destination notation: ";

cin >> notationTo;

cout << ToNotation(ToDec(firstNum, notation), notationTo);

}

# Проверочная программа сортировки пузырьком

void main()

{

int size = 5;

int a[size];

for (int i = 0; i < size; i = i + 1)

{

cin >> a[i];

}

for (int i = 0; i < size; i = i + 1)

{

for (int j = i + 1; j < size; j = j + 1)

{

if (a[i] > a[j])

{

int t = a[i];

a[i] = a[j];

a[j] = t;

}

}

}

for (int i = 0; i < size; i = i + 1)

{

cout << a[i];

cout << " ";

}

}

# Вывод для проверочной программы

Вход: 1 6 9 7 -6666

Выход: -6666 1 6 7 9

# Используемое программное обеспечение

Microsoft Visual Studio 2013

Генератор написан на C# + MSIL, остальное – C++11.