|  |
| --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет»  Факультет информационных технологий  Кафедра технологий программирования  **Лабораторная работа №5**  **По дисциплине «Компиляторные технологии»**  **Тема: «Синтаксический анализатор»**  Выполнил: Купаленко А.Д.  Группа 21ИТ1  Проверила: Сыцевич Д.Н.  Преподаватель кафедры ТП  Полоцк 2022 |

**Лабораторная работа № 5**

**“Синтаксический анализатор”**

**Цель работы:** изучение основных принципов генерации компилятором объектного кода, ознакомление с методами оптимизации результирующего объектного кода для линейного участка программы с помощью свертки и исключения лишних операций.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ** (ответы на контрольные вопросы):

В иерархии грамматик Хомского выделено 4 основных группы языков (и описывающих их грамматик). При этом наибольший интерес представляют регулярные и контекстно-свободные (КС) грамматики и языки. Они используются при описании синтаксиса языков программирования. С помощью регулярных грамматик можно описать лексемы языка – идентификаторы, константы, служебные слова и прочие. На основе КС-грамматик строятся более крупные синтаксические конструкции: описания типов и переменных, арифметические и логические выражения, управляющие операторы, и, наконец, полностью вся программа на входном языке.

Входные цепочки регулярных языков распознаются с помощью конечных автоматов (КА). Они лежат в основе сканеров, выполняющих лексический анализ и выделение слов в тексте программы на входном языке. Результатом работы сканера является преобразование исходной программы в список или таблицу лексем. Дальнейшую ее обработку выполняет другая часть компилятора – синтаксический анализатор. Его работа основана на использовании правил КС-грамматики, описывающих конструкции исходного языка.

**Синтаксический анализатор** – это часть компилятора, которая отвечает за выявление и проверку синтаксических конструкций входного языка.

**Вариант 11**

*Листинг программы*

***#include <windows.h>***

***#include <iostream>***

***#include <fstream>***

***#include <iostream>***

***#include <stdlib.h>***

***#include <string.h>***

***#include <ctype.h>***

***#include <cstdio>***

***#include <cstring>***

***#include <windows.h>***

***#include <vector>***

***using namespace std;***

***bool isPunctuator(char ch)***

***{***

***if (ch == ' ' || ch == '+' || ch == '-' || ch == '\*' ||***

***ch == '/' || ch == ',' || ch == ';' || ch == '>' ||***

***ch == '<' || ch == '=' || ch == '(' || ch == ')' ||***

***ch == '[' || ch == ']' || ch == '{' || ch == '}' ||***

***ch == '&' || ch == '|')***

***{***

***return true;***

***}***

***return false;***

***}***

***bool validIdentifier(char\* str)***

***{***

***if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||***

***str[0] == '3' || str[0] == '4' || str[0] == '5' ||***

***str[0] == '6' || str[0] == '7' || str[0] == '8' ||***

***str[0] == '9' || isPunctuator(str[0]) == true)***

***{***

***return false;***

***}***

***int i, len = strlen(str);***

***if (len == 1)***

***{***

***return true;***

***}***

***else***

***{***

***for (i = 1; i < len; i++)***

***{***

***if (isPunctuator(str[i]) == true)***

***{***

***return false;***

***}***

***}***

***}***

***return true;***

***}***

***bool isHex(char\* str)***

***{***

***if ((str[0] == '0' || str[0] == '1' || str[0] == '2' ||***

***str[0] == '3' || str[0] == '4' || str[0] == '5' ||***

***str[0] == '6' || str[0] == '7' || str[0] == '8' ||***

***str[0] == '9') &&***

***(strchr(str, 'a') != NULL || strchr(str, 'b') != NULL ||***

***strchr(str, 'c') != NULL || strchr(str, 'd') != NULL ||***

***strchr(str, 'e') != NULL || strchr(str, 'f') != NULL ||***

***isPunctuator(str[0]) == true)) {***

***return true;***

***}***

***else return false;***

***}***

***bool isRavno(char str)***

***{***

***if (str == '=') return true;***

***else return false;***

***}***

***bool isZnak(char str)***

***{***

***if (str == ';') return true;***

***else return false;***

***}***

***bool isDelimetr(char str)***

***{***

***if (str == '(' || str == ')' || str == '{' || str == '}') return true;***

***else return false;***

***}***

***bool isOperator(char ch)***

***{***

***if (ch == '+' || ch == '-' || ch == '\*' ||***

***ch == '/' || ch == '>' || ch == '<' || ch == '|' || ch == '&')***

***{***

***return true;***

***}***

***return false;***

***}***

***bool isKeyword(char\* str)***

***{***

***if (!strcmp(str, "if") || !strcmp(str, "else") ||***

***!strcmp(str, "while") || !strcmp(str, "do") ||***

***!strcmp(str, "break") || !strcmp(str, "continue")***

***|| !strcmp(str, "return") || !strcmp(str, "case")***

***|| !strcmp(str, "typedef") || !strcmp(str, "switch")***

***|| !strcmp(str, "struct") || !strcmp(str, "extern")***

***|| !strcmp(str, "sizeof") || !strcmp(str, "typedef")***

***|| !strcmp(str, "enum") || !strcmp(str, "union")***

***|| !strcmp(str, "cout")) return true;***

***else return false;***

***}***

***bool isType(char\* str)***

***{***

***if (!strcmp(str, "int") || !strcmp(str, "double")***

***|| !strcmp(str, "float") || !strcmp(str, "char")***

***|| !strcmp(str, "long") || !strcmp(str, "short")***

***|| !strcmp(str, "unsigned") || !strcmp(str, "void")***

***|| !strcmp(str, "static") || !strcmp(str, "sizeof")***

***|| !strcmp(str, "long") || !strcmp(str, "volatile")***

***|| !strcmp(str, "enum") || !strcmp(str, "const")***

***|| !strcmp(str, "bool") || !strcmp(str, "const")) return true;***

***else return false;***

***}***

***bool isNumber(char\* str)***

***{***

***int i, len = strlen(str), numOfDecimal = 0;***

***if (len == 0)***

***{***

***return false;***

***}***

***for (i = 0; i < len; i++)***

***{***

***if (numOfDecimal > 1 && str[i] == '.')***

***{***

***return false;***

***}***

***else if (numOfDecimal <= 1)***

***{***

***numOfDecimal++;***

***}***

***if (str[i] != '0' && str[i] != '1' && str[i] != '2'***

***&& str[i] != '3' && str[i] != '4' && str[i] != '5'***

***&& str[i] != '6' && str[i] != '7' && str[i] != '8'***

***&& str[i] != '9' || (str[i] == '-' && i > 0))***

***{***

***return false;***

***}***

***}***

***return true;***

***}***

***char\* subString(char\* realStr, int l, int r)***

***{***

***int i;***

***char\* str = (char\*)malloc(sizeof(char) \* (r - l + 2));***

***for (i = l; i <= r; i++)***

***{***

***str[i - l] = realStr[i];***

***str[r - l + 1] = '\0';***

***}***

***return str;***

***}***

***void parse(char\* str)***

***{***

***SetConsoleCP(1251);***

***SetConsoleOutputCP(1251);***

***string operators = "";***

***string znakov = "";***

***string dels = "";***

***string keys = "";***

***string numbs = "";***

***string idents = "";***

***string hexs = "";***

***string errors = "";***

***string znaks = "";***

***vector<string> syntax;***

***int left = 0, right = 0;***

***int len = strlen(str);***

***while (right <= len && left <= right) {***

***if (isPunctuator(str[right]) == false)***

***{***

***right++;***

***}***

***if (isPunctuator(str[right]) == true && left == right)***

***{***

***if (isZnak(str[right]) == true)***

***{***

***znaks = znaks + str[right] + " ";***

***syntax.push\_back("float number: ");***

***}***

***if (isOperator(str[right]) == true)***

***{***

***operators = operators + str[right] + " ";***

***syntax.push\_back("operator: ");***

***}***

***if (isRavno(str[right]) == true)***

***{***

***znakov = znakov + str[right] + " ";***

***syntax.push\_back("equal sign: ");***

***}***

***if (isDelimetr(str[right]) == true)***

***{***

***dels = dels + str[right] + " ";***

***syntax.push\_back("devider: ");***

***}***

***right++;***

***left = right;***

***}***

***else if (isPunctuator(str[right]) == true && left != right***

***|| (right == len && left != right))***

***{***

***char\* sub = subString(str, left, right - 1);***

***if (isKeyword(sub) == true)***

***{***

***syntax.push\_back("keyword: ");***

***keys = keys + sub + " ";***

***}***

***if (isType(sub) == true)***

***{***

***syntax.push\_back("type: ");***

***}***

***else if (isNumber(sub) == true)***

***{***

***syntax.push\_back("numbers: ");***

***numbs = numbs + sub + " ";***

***}***

***else if (validIdentifier(sub) == true***

***&& isPunctuator(str[right - 1]) == false)***

***{***

***syntax.push\_back("indetifiers: ");***

***idents = idents + sub + " ";***

***}***

***else if (isHex(sub) == true***

***&& isPunctuator(str[right - 1]) == false)***

***{***

***syntax.push\_back("hex nubers: ");***

***hexs = hexs + sub + " ";***

***}***

***else if (validIdentifier(sub) == false***

***&& isPunctuator(str[right - 1]) == false)***

***{***

***errors = errors + sub + " ";***

***}***

***left = right;***

***}***

***}***

***if (znaks != "") {***

***cout << "float numbers: " << znaks << '\n';***

***}***

***if (operators != "") {***

***cout << "operators: " << operators << '\n';***

***}***

***if (znakov != "") {***

***cout << "equals signs: " << znakov << '\n';***

***}***

***if (dels != "") {***

***cout << "deviders: " << dels << '\n';***

***}***

***if (keys != "") {***

***cout << "keywords: " << keys << '\n';***

***}***

***if (numbs != "") {***

***cout << "real numbers: " << numbs << '\n';***

***}***

***if (idents != "") {***

***cout << "intetifiers: " << idents << '\n';***

***}***

***if (hexs != "") {***

***cout << "hex numbers: " << hexs << '\n';***

***}***

***if (errors != "") {***

***cout << "errors: " << errors;***

***}***

***cout << "syntax analiz: " << endl;***

***for (int i = 0; i < syntax.size(); i++)***

***cout << syntax[i] << endl;***

***return;***

***}***

***int main()***

***{***

***SetConsoleCP(1251);***

***SetConsoleOutputCP(1251);***

***struct tree\* derevo;***

***int a, b;***

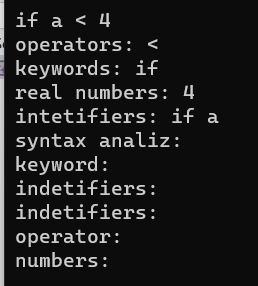
***char str[100];***

***gets\_s(str);***

***parse(str);***

***return 0;***

***}***

**

*Рисунок 1 - Результат работы программы*