Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий Кафедра прикладной математики

Отчет защищен с оценкой \_ Преподаватель (подпись)

« » 2023 г.

Отчет

по лабораторной работе № 4

**«**Лексика языка программирования»

по дисциплине «Программа лексического анализатора»

Студент гр. ПИ-02 Чередов Р.А.

Преподаватель, к.ф.-м.н., доцент,

Крючкова Е. Н.

Барнаул 2023

# Задание

**124 Чередов Роман Алексеевич**

Программа: главная программа языка С++.. Допускается описание классов. Функции не имеют параметров и возвращают void.

Типы данных: double, int, short int.

Операции: все арифметические, сравнения.

Операторы: пустой, составной, присваивания и do {} while.

Операнды: простые переменные, поля классов, именованные константы , константы.

Константы: целые в 10 c/c.

# Решение

Defs.h

/\*

Модуль Defs.h

Общие типы данных и кароопределения

\*/

#pragma once

#ifndef \_\_DEFS

#define \_\_DEFS

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#define MAX\_TEXT 10000 // Максимальная длина текста

#define MAX\_LEX 100 // Максимальная длинна лексемы

#define MAX\_KEYW 10

typedef char TypeLEX[MAX\_LEX]; // Тип лексемы

typedef char TypeIM[MAX\_TEXT]; // Тип исходного модуля

// Ключевые слова

#define TShort 1

#define TDouble 2

#define TInt 3

#define Tdo 4

#define Twhile 5

#define TConst 6

#define TMain 7

#define TReturn 8

#define TClass 9

#define TVoid 10

// Идентификаторы и констаны

#define TIdent 20

#define TconstInt 30

// Специальные знаки

#define TDot 40

#define TSemiColon 41

#define TComma 42

#define TColon 43

#define TLeftBracket 44

#define TRightBracket 45

#define TLeftSquareBracket 46

#define TRightSquareBracket 47

// Знаки операций

#define TPlus 50

#define TMinus 51

#define TMultiply 52

#define TDivide 53

#define TMod 54

#define TSave 55

#define TEqual 56

#define TLess 57

#define TLessOrEqual 58

#define TGreat 59

#define TGreatOrEqual 60

#define TLeftShift 61

#define TRightShift 62

#define TNotEqual 63

#define TIncrement 64

#define TDecrement 65

// Конец исходного модуля

#define TEnd 200

// Ошибочный символ

#define TError 100

#endif // \_\_DEFS

Scanner.h

#pragma once

#ifndef \_\_SCANER

#define \_\_SCANER

#include "Defs.h"

class Scaner

{

private:

TypeIM im; // Исходный модуль

int uk; // Указатель позиции в исходном модуле

public:

Scaner(const char\* FileName);

~Scaner() {};

void PrintError(const char\* err, const char\* c);

void GetData(const char\* FileName);

int Scan(TypeLEX Lex);

};

#endif // \_\_SCANER

Scaner.cpp

#include "Scaner.h"

void Scaner::GetData(const char\* FileName) {

FILE\* filestream = fopen(FileName, "r"); // Открыть файл для чтения

if (filestream == NULL) {

printf("Отсутствует входной файл");

exit(1);

}

int i = 0;

for (i = 0; !feof(filestream); i++) {

if (i < MAX\_TEXT) {

fscanf(filestream, "%c", &im[i]);

}

else {

printf("Превышина максимальная длинна исходного модуля");

break;

}

}

im[--i] = '\0';

}

Scaner::Scaner(const char\* FileName) {

GetData(FileName);

uk = 0;

}

// Ключевые слова

TypeLEX Keyword[MAX\_KEYW] = {

"short", "double", "int",

"do", "while", "const",

"main", "return", "class",

"void",

};

// Типы ключевых слов

int TypeKeyword[MAX\_KEYW] = {

TShort, TDouble, TInt,

Tdo, Twhile, TConst,

TMain, TReturn, TClass,

TVoid,

};

int Scaner::Scan(TypeLEX lex) {

int legth\_lex = 0;

for (legth\_lex = 0; legth\_lex < MAX\_LEX; lex[legth\_lex++] = 0);

legth\_lex = 0;

start:

// ПРОПУСК НЕЗНАЧАЩИХ ЭЛЕМЕНТОВ

while ((im[uk] == ' ')

|| (im[uk] == '\n')

|| (im[uk] == '\t'))

uk++;

if ((im[uk] == '/') && (im[uk + 1] == '/')) {

while (im[uk] != '\n') uk++;

goto start;

}

if (im[uk] == '\0') {

lex[0] = '#';

return TEnd;

}

// ЦЕЛОЕ ДЕСЯТИЧНОЕ ЧИСЛО

if ((im[uk] >= '0') && (im[uk] <= '9')) {

while ((im[uk] >= '0') && (im[uk] <= '9')) {

lex[legth\_lex++] = im[uk++];

}

return TconstInt;

}

// ИДЕНТИФИКАТОР

else if ((im[uk] >= 'a') && (im[uk] <= 'z')

|| (im[uk] >= 'A') && (im[uk] <= 'Z')

|| (im[uk] == '\_') && (im[uk + 1] == '\_') && (im[uk + 2] != '\_')) {

while ((im[uk] >= 'a') && (im[uk] <= 'z')

|| (im[uk] >= 'A') && (im[uk] <= 'Z')

|| (im[uk] == '\_')

|| (im[uk] >= '0') && (im[uk] <= '9')) {

// Обрезаем длинный индентификатор

if (legth\_lex < MAX\_LEX - 1) {

lex[legth\_lex++] = im[uk++];

}

else {

uk++;

}

// проверка на ключевое слово

for (int index\_key = 0; index\_key < MAX\_KEYW; index\_key++) {

if (strcmp(lex, Keyword[index\_key]) == 0) {

return TypeKeyword[index\_key];

}

}

}

return TIdent;

}

// СПЕЦИАЛЬНЫЕ ЗНАКИ

else if (im[uk] == '.') { lex[legth\_lex++] = im[uk++]; return TDot; }

else if (im[uk] == ',') { lex[legth\_lex++] = im[uk++]; return TComma; }

else if (im[uk] == '(') { lex[legth\_lex++] = im[uk++]; return TLeftBracket; }

else if (im[uk] == ')') { lex[legth\_lex++] = im[uk++]; return TRightBracket; }

else if (im[uk] == '{') { lex[legth\_lex++] = im[uk++]; return TLeftSquareBracket; }

else if (im[uk] == '}') { lex[legth\_lex++] = im[uk++]; return TRightSquareBracket; }

else if (im[uk] == ';') { lex[legth\_lex++] = im[uk++]; return TSemiColon; }

else if (im[uk] == ':') { lex[legth\_lex++] = im[uk++]; return TColon;

}

// ЗНАКИ ОПЕРАЦИЙ

else if (im[uk] == '=') {

lex[legth\_lex++] = im[uk++];

if (im[uk] == '=') {

lex[legth\_lex++] = im[uk++];

return TEqual;

}

return TSave;

}

else if (im[uk] == '<') {

lex[legth\_lex++] = im[uk++];

if (im[uk] == '=') {

lex[legth\_lex++] = im[uk++];

return TLessOrEqual;

}

return TLess;

}

else if (im[uk] == '>') {

lex[legth\_lex++] = im[uk++];

if (im[uk] == '=') {

lex[legth\_lex++] = im[uk++];

return TGreatOrEqual;

}

return TGreat;

}

else if (im[uk] == '+') { lex[legth\_lex++] = im[uk++]; return TPlus; }

else if (im[uk] == '-') { lex[legth\_lex++] = im[uk++]; return TMinus; }

else if (im[uk] == '\*') { lex[legth\_lex++] = im[uk++]; return TMultiply; }

else if (im[uk] == '/') { lex[legth\_lex++] = im[uk++]; return TDivide; }

else if (im[uk] == '%') { lex[legth\_lex++] = im[uk++]; return TMod; }

else if ((im[uk] == '!') && (im[uk + 1] == '=')) {

lex[legth\_lex++] = im[uk++];

lex[legth\_lex++] = im[uk++];

return TNotEqual;

}

while((im[uk] != ' ') && (im[uk] != '\n') && (im[uk] != '\t')

&& im[uk] != '/' && im[uk + 1] != '/')

lex[legth\_lex++] = im[uk++];

return TError;

Lexical\_Scanner.cpp

#include "Scaner.h"

#include <Windows.h>

#include <iostream>

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Scaner\* scan = new Scaner("Example.txt");

int type;

TypeLEX lex;

do {

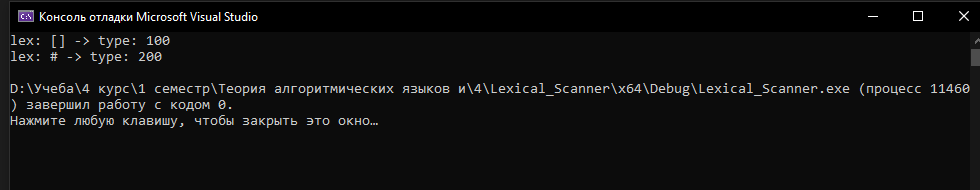
type = scan->Scan(lex);

printf("lex: %s -> type: %d\n", lex, type);

} while (type != TEnd);

}

}

Примеры ошибок: 

Пример обработки

