Реализация сканера

defs.h:

```
#ifndef __DEFS
#define __DEFS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
using namespace std;
#define MAXTEXT 10000
#define MAXLEX 50
#define MAXKEYW 6
typedef char TypeLex[MAXLEX];
typedef char TypeMod[MAXTEXT];
#define Tid  1
#define Tchar  10
#define Tdo  11
#define Twhile  12
#define Tfloat  13
#define Tvoid 14
#define Tmain 15
#define Tsave  20
#define Tless  21
#define Tmore  22
#define Tsum  23
#define Tsub  24
#define Tmul  25
#define Tdiv  26
#define Tmod  27
#define Teq  28
#define Tneq  29
#define Tsumeq  30
#define Tsubeq  31
#define Tmuleq  32
#define Tdiveq  33
#define Tmodeq  34
#define Tlesseq  35
#define Tmoreeq  36
#define Tinc  37
#define Tdec  38
#define Tconstint  50
#define Tconstsymb  51
#define Tconststr 52
#define Tsemi  60
#define Tapos 61
#define Tcom  62
#define Tquote 63
#define TLBKT  70
#define TRBKT  71
#define TfLBKT  72
#define TfRBKT  73
#define TsLBKT 74
#define TsRBKT 75
#define Terr 1000
#define Teof 2000
#endif
```

scaner.h:

```
#ifndef __SCANER
#define __SCANER
#include "defs.h"
class Tscaner{
private:
	TypeMod t; // исходный текст
	int uk; // указатель текущего положения в тексте
	int line,pos;//позиция ошибки
	void getData(char *fileName);
public:
	void setUk (int value);
	int getUk ();
	void setLine (int value);
	int getLine ();

	void printError(char *error, char *symbol);
	void printSemError(char *error);
	int scaner (TypeLex l);

	Tscaner(char * fileName);
	~Tscaner() {}
};
#endif
```

scaner.cpp:

```
#include "defs.h"
#include "scaner.h"
TypeLex keyword[MAXKEYW]={ "do","while","char","float", "void", "main"};
int indexKeyword[MAXKEYW] = {Tdo,Twhile,Tchar,Tfloat,Tvoid,Tmain};
Tscaner::Tscaner(char* fileName){
	getData(fileName);
	setUk(0);
	setLine(0);
}
void Tscaner::setUk(int value){
	uk = value;
}
int Tscaner::getUk(){
	return uk;
}
void Tscaner::setLine(int value){
	line = value;
}
int Tscaner::getLine(){
	return line;
}
void Tscaner::getData(char * fileName) {
	char aa;
	FILE * in = fopen(fileName,"r");
	if (in==NULL) {
		t[0] = '\0';
		printError("Отсутствует входной файл","");
		return;
	}
	int i=0;
	while(!feof(in)){
		fscanf(in,"%c",&aa);
		if (!feof(in)) {
			t[i++]=aa;
		}
```

```
		if (i>=MAXTEXT-1){
		printError("Превышен размер входного файла","");
		break;
		}
	}
	t[i]='\0';
	fclose(in);
}
void Tscaner::printError(char *error, char *symbol){
	if (symbol[0]=='\0')
		printf("Ошибка: %s %s\n",error,symbol);
	else
		printf("Ошибка: %s. Неверный символ %s Строка %d\n",error,symbol,line+1);
	exit(1);
}
void Tscaner::printSemError(char *error){
	printf("Ошибка: %s. Строка %d\n",error, line+1);
	exit(1);
}
int Tscaner::scaner(TypeLex l) {
	int i; // текущая длина лексемы

	for (i=0;i<MAXLEX;i++) {
		l[i]=0;
	}
	i=0; // лексема заполняется с позиции i
	while(true){
		while((t[uk]==' ') || (t[uk]=='\n') || (t[uk]=='\t')){
			if(t[uk] == '\n') setLine(getLine()+1);
			uk++;
		}
		// пропуск незначащих элементов
		if ( (t[uk]=='/') && (t[uk+1]=='/') ){ // начался комментарий, надо пропустить текст
до ’\n’
			uk=uk+2;
			while ( (t[uk]!='\n')&&(t[uk]!='\0')&&(t[uk]!='#')) {
				uk++;
			}
			if(t[uk] == '\n'){
				setLine(getLine()+1);

			}
			continue;
		}
		break;
	}
	//проверяем на окончание файла

	if (t[uk]=='\0') {
		l[0]='#';

		return Teof;
	}
	if(((t[uk]>='a')&&(t[uk]<='z'))||((t[uk]>='A')&&(t[uk]<='Z'))){
		l[i++] = t[uk++];
	while(((t[uk]>='a')&&(t[uk]<='z'))||((t[uk]>='A')&&(t[uk]<='Z'))||((t[uk]>='0')&&(t[uk]<='
9'))){
			if(i < MAXLEX-1){
				l[i++] = t[uk++];
			}
			else{
				uk++;
			}
		}
		for(int j = 0;j < MAXKEYW;j++){
			if(strcmp(l,keyword[j]) == 0){
				return indexKeyword[j];
```

```
			}
		}
		return Tid;
	}
	else if((t[uk]>='0')&&(t[uk]<='9')){
			l[i++] = t[uk++];
		bool flag = false;//ошибка длины константы
		while((t[uk]>='0')&&(t[uk]<='9')){
			if(i < MAXLEX-1){
				l[i++] = t[uk++];
			}
			else{
				flag = true;
				uk++;
			}
		}
		if(flag){
			printError("Слишком длинная константа",l);
			return Terr;
		}
		return Tconstint;
	}

	else if(t[uk] =='<'){
		l[i++] = t[uk++];

		if(t[uk] == '='){
			l[i++] = t[uk++];
			return Tlesseq;
		}
		return Tless;
	}
	else if(t[uk] =='>'){
		l[i++] = t[uk++];

		if(t[uk] == '='){
			l[i++] = t[uk++];
			return Tmoreeq;
		}

		return Tmore;
	}
	else if(t[uk] =='='){
		l[i++] = t[uk++];

		if(t[uk] == '='){
			l[i++] = t[uk++];
			return Teq;
		}
		return Tsave;
	}
	else if(t[uk] =='*'){
		l[i++] = t[uk++];

		if(t[uk] == '='){
			l[i++] = t[uk++];
			return Tmuleq;
		}
		return Tmul;
	}
	else if(t[uk] =='/'){
		l[i++] = t[uk++];

		if(t[uk] == '='){
			l[i++] = t[uk++];
			return Tdiveq;
		}
```

```
		return Tdiv;
	}
	else if(t[uk] =='%'){
		l[i++] = t[uk++];

		if(t[uk] == '='){
			l[i++] = t[uk++];
			return Tmodeq;
		}
		return Tmod;
	}
	else if(t[uk] =='!'){
		l[i++] = t[uk++];

		if(t[uk] == '='){
			l[i++] = t[uk++];
			return Tneq;
		}
		else{
			l[0]='!';
			printError("Неверный символ",l);
			return Terr;
		}
	}
	else if(t[uk] =='-'){
		l[i++] = t[uk++];

		if(t[uk] == '='){
			l[i++] = t[uk++];
			return Tsubeq;
		}
		if(t[uk] == '-'){
			l[i++] = t[uk++];
			return Tdec;
		}

		return Tsub;
	}
	else if(t[uk] =='+'){
		l[i++] = t[uk++];

		if(t[uk] == '='){
			l[i++] = t[uk++];
			return Tsumeq;
		}
		if(t[uk] == '+'){
			l[i++] = t[uk++];
			return Tinc;
		}
		return Tsum;
	}
	else if(t[uk] =='\''){
		l[i++] = t[uk++];
		bool flag = false;
		if(t[uk]=='\'')
		{
			l[i++]=t[uk++];
			flag = true;
		}
		else if(t[uk+1]=='\'')
		{
			l[i++]=t[uk++];
			l[i++]=t[uk++];
			flag = true;
		}
		if(flag) return Tconstsymb;
		else{
```

```
                l[0]='\'';
                printError("Неверный символ",l);
                return Terr;
        }
}
else if(t[uk] =='\"'){
        l[i++] = t[uk++];
        bool flag = false;
        bool f11 = false;
        int xxx = 0;
        while(t[uk+xxx]!='\n'){
                if(t[uk+xxx] == '\"')
                {
                        flag = true;
                }
                xxx++;
        }
        if(flag)
        {
                while(t[uk]!='\"')
                {
                        l[i++] = t[uk++];
                }
                l[i++] = t[uk++];
                return Tconststr;
        }
        else{
                l[0]='\"';
                printError("Неверный символ",l);
                return Terr;
        }}
else if(t[uk] ==','){
        l[i++] = t[uk++];
        return Tcom;
}
else if(t[uk] ==';'){
        l[i++] = t[uk++];
        return Tsemi;
}
else if(t[uk] =='('){
        l[i++] = t[uk++];
        return TLBKT;
}
else if(t[uk] ==')'){
        l[i++] = t[uk++];
        return TRBKT;
}
else if(t[uk] =='{'){
        l[i++] = t[uk++];
        return TfLBKT;
}
else if(t[uk] =='}'){
        l[i++] = t[uk++];
        return TfRBKT;
}
else if(t[uk] =='['){
        l[i++] = t[uk++];
        return TsLBKT;
}
else if(t[uk] ==']'){
        l[i++] = t[uk++];
        return TsRBKT;
}
else {
        l[0]=t[uk];
        printError("Неверный символ",l);
        uk++;
```

```
        return Terr;
}}
```