Ain Shams University
Faculty of Engineering

User Manual

Name	Mahmoud Hamdy Abd El-Gawwad
Section	3
Name	Doaa Kamel Abdelmoneim Ali
Section	2

Cairo, 2018

Introduction

- Project is implemented in C++ under Windows OS using Visual Studio IDE

How to run?

- Make sure that our test file named Tiny_program.txt exists in the same directory as the .exe file.

- Program starts with operating on the TINY code previously provided in the lecture slides.

- If you want to use another test for a tiny program, edit the existing provided test file.

Snapshots

Output:

C:\WINDOWS\system32\cmd.exe

```
Code is scanned Successfully and output file is generated
Press any key to continue . . .
```

output.txt - Notepad

File Edit Format View Help

```
read, reserved word
x, Identifier
;, Symbol
if, reserved word
0, Number
<, Symbol
x, Identifier
then, reserved word
fact, Identifier
:=, Symbol
1, Number
;, Symbol
repeat, reserved word
fact, Identifier
:=, Symbol
fact, Identifier
*, Symbol
x, Identifier
;, Symbol
x, Identifier
:=, Symbol
x, Identifier
-, Symbol
1, Number
until, reserved word
x, Identifier
=, Symbol
0, Number
;, Symbol
write, reserved word
fact, Identifier
end, reserved word
```

Code:

main.cpp

main.cpp scanner.cpp scanner.h

TINY_Scanner (Global Scope)

```cpp
#include <iostream>
#include<string.h>
#include <fstream>
#include "scanner.h"
using namespace std;
int main(void)
{
    string x;
    string a;
    ifstream my_prog;
    my_prog.open("Tiny_program.txt");
    if (my_prog.is_open())
        while (!my_prog.eof())
        {
            getline(my_prog, a);
            x += a;
        }
    if (x[x.length() - 1] != '\0')
        x += '\0';

    get_token(x);
    system("pause");
}
```

scanner.h

main.cpp scanner.cpp scanner.h

TINY_Scanner (Global Scope)

```cpp
#pragma once
#include <string>
#include <stdlib.h>
#include <iostream>
#include <fstream>

#define START       100
#define INCOMMENT   101
#define INNUM       102
#define INID        103
#define INASSIGN    104
#define DONE        105

typedef enum
{
    IF, THEN, ELSE, END, REPEAT, UNTIL, READ, WRITE, //RESERVED WORD
    PLUS, MINUS, MULTIPLY, DIVISION, EQUAL, LESS_THAN,
    LEFT_BRACKET, RIGHT_BRACKET, SEMI_COLON, ASSIGN,
    NUM, ID
}TokenType;

typedef struct
{
    TokenType token_val;
    std::string lexeme;
    int num_val;
}TokenRecord;

void get_token(std::string s);
```

scanner.cpp

main.cpp | scanner.cpp | scanner.h

TINY_Scanner | (Global Scope)

```cpp
#include "scanner.h"
using namespace std;
void get_token(std::string s)
{
    ofstream out_file("output.txt");
    TokenRecord token;
    bool lookahead = false;
    int state = START;
    int i = 0;
    bool reach_end_flag = false;
    char input = s[i];
    int length = s.length();

    while (s[i] != NULL)
    {
        input = s[i];
        if (reach_end_flag == true)
        {
            if (input != ' ')
                state = DONE;

            i++;
        }
        switch (state)
        {
```

main.cpp | scanner.cpp | scanner.h

TINY_Scanner | (Global Scope)

```cpp
case START:
    if (input == ' ')
        state = START;
    else if (input == '{')
        state = INCOMMENT;
    else if (int(input) >= int('0') && int(input) <= int('9'))
    {
        token.token_val = NUM;
        token.lexeme = input;
        token.num_val = int(input);
        state = INNUM;
    }
    else if (int(input) >= int('A') && int(input) <= int('z'))
    {
        token.token_val = ID;
        token.lexeme = input;
        state = INID;
    }
    else if (input == ':')
    {
        token.token_val = ASSIGN;
        token.lexeme = input;
        state = INASSIGN;
    }
```

main.cpp | scanner.cpp | scanner.h

TINY_Scanner | (Global Scope)

```
            else
            {
                state = DONE;
                token.lexeme = input;

                if (input == '+')
                    token.token_val = PLUS;

                else if (input == '-')
                    token.token_val = MINUS;

                else if (input == '*')
                    token.token_val = MULTIPLY;

                else if (input == '/')
                    token.token_val = DIVISION;

                else if (input == '=')
                    token.token_val = EQUAL;

                else if (input == '<')
                    token.token_val = LESS_THAN;

                else if (input == '(')
                    token.token_val = LEFT_BRACKET;

                else if (input == ')')
                    token.token_val = RIGHT_BRACKET;

```

main.cpp | scanner.cpp | scanner.h

TINY_Scanner | (Global Scope)

```

                else if (input == ';')
                    token.token_val = SEMI_COLON;
            }
            break;

        case INCOMMENT:
            if (input == '}')
                state = START;

            else
                state = INCOMMENT;
            break;

```

main.cpp | scanner.cpp | scanner.h

TINY_Scanner | (Global Scope)

```cpp

        case INNUM:
            if (int(input) >= int('0') && int(input) <= int('9'))
            {
                token.lexeme += input;
                token.num_val = stoi(token.lexeme);
            }
            else
            {
                lookahead = true;
                state = DONE;
            }
            break;

        case INID:
            if (int(input) >= int('A') && int(input) <= int('z'))
            {
                token.lexeme += input;
                state = INID;
            }
            else
            {
                lookahead = true;
                state = DONE;
            }

            break;
```

main.cpp | scanner.cpp | scanner.h

TINY_Scanner | (Global Scope)

```cpp

        case INASSIGN:
            if (input == '=')
            {
                token.lexeme += input;
                state = DONE;
            }
            else
            {
                lookahead = true;
                state = DONE;
            }
            break;

```

main.cpp scanner.cpp scanner.h

TINY_Scanner (Global Scope)

```cpp
        case DONE:
            state = START;
            out_file << token.lexeme << ", ";
            switch (token.token_val)
            {
            case ID:
                if (token.lexeme == "if" || token.lexeme == "else" || token.lexeme == "then" ||
                    token.lexeme == "end" || token.lexeme == "repeat" || token.lexeme == "until" ||
                    token.lexeme == "read" || token.lexeme == "write")
                {
                    out_file << "reserved word";
                }
                else
                {
                    out_file << "Identifier";
                }
                break;

            case NUM:
                out_file << "Number";
                break;

            default:
                out_file << "Symbol";
                break;


            }
            out_file << endl;
```

main.cpp scanner.cpp scanner.h

TINY_Scanner (Global Scope)

```cpp

                if (lookahead == true)
                {
                    i--;
                    lookahead = false;
                }

                break;
            }

            if (i == (length - 1))
            {
                reach_end_flag = true;
            }
            if (state != DONE && i != (length - 1) && i != length)
                i++;

        }
        cout << "Code is scanned Successfully and output file is generated" << endl;
    }
```