Київський національний університет
імені Т.Шевченка

Звіт

до лабораторної роботи №3
з дисципліни:
«Системне Програмування»

Студента третього курсу
Групи МІ-32
Факультету комп'ютерних наук
та кібернетики
Федорича Андрія

Київ-2023

Мета

Метою цієї лабораторної роботи є розробка лексичного аналізатора мови програмування з можливістю розрізнення та класифікації різних класів лексем:

- числа (десяткові, з плаваючою крапкою, шістнадцяткові),
- рядкові та символьні константи,
- директиви препроцесора,
- коментарі,
- зарезервовані слова,
- оператори,
- розділові знаки,
- ідентифікатори.

Позначати ситуації з помилками (наприклад, нерозпізнавані символи).

Основні принципи виконання роботи

Варіант: 2

Лексичний аналізатор для мови C.

Мова виконання: C++

Джерело:
Код програми, тестові та результуючі файли можна знайти за посиланням:*https://github.com/StopFuture/KNU_Bachelor_Assignments/tree/main/SystemProgramming/Lab3*

Реалізація:

- Створено два **.c** файли та один **.h** файл-заголовок.

- В файлі **tokenizer.c** створені допоміжну структуру даних **Token**, що дозволяє зберігати пару із regex-патерну та назви для нього.
- Створено `vector<Token> tokens` для таких лексем:
    - Comment (inline / multiline)
    - String/Char
    - Directive
    - Library
    - Reserved
    - DataType
    - Bool (like false and true)
    - HexNumber
    - BinNumber
    - Number
    - Function
    - FunctionCall
    - Variable
    - Delimiter
    - Operator
- Також функція tokenize приймає на вхід текст програми, та повертає, як результат токенізовану програму.
- В основній частині програми наступна логіка:
    - Ініціалізуємо файл **input_<number>** з якого зчитуватимемо код програми, при чому перевіряємо коректність його відкриття.
    - Використовуємо функцію **tokenize** для послідовної обробки
    - Виводимо кожен знайдений токен в консоль, також зберігаємо його в пам’яті.

    - Створюємо відповідний файл **output_<number>** в який записуємо весь результат, якщо файл вже існував перезаписуємо його**.**
- При виявленні тексту-помилки, яку неможливо зчитати програма присвоює їй значення **Uncategorized**

Для демонстрацій роботи було використано код із минулих лабораторних робіт із Системного Програмування написаних мною на C:

1. https://github.com/StopFuture/KNU_Bachelor_Assignments/tree/main/SystemProgramming/Lab1
2. https://github.com/StopFuture/KNU_Bachelor_Assignments/tree/main/SystemProgramming/Lab2

Приклад 1

main.cpp | input_1.txt | tokenizer.cpp | tokenizer.h | input_3.txt

```
//
// Created by Fedorych Andriy on 24/9/23.
//

#ifndef LAB1_DICTIONARY_H
#define LAB1_DICTIONARY_H

typedef struct {
    char* key;
    int value;
} Pair;

/*
Random
long
comment
*/

typedef struct {
    Pair* pairs;
    int size;
    int count;  // Track the current count of elements
} Dictionary;

Dictionary* init(int initial_size);
void add_pair(Dictionary* dict, const char* key, int value);
int get_value(Dictionary* dict, const char* key);
void clear_dictionary(Dictionary* dict);

#endif //LAB1_DICTIONARY_H
```

Lab3

```
/Users/stopfuture/CLionProjects/SystemProgramming/Lab3/
Enter a file name: input_1.txt
Comment:   //
Comment:   // Created by Fedorych Andriy on 24/9/23.
Comment:   //
Directive:   #ifndef
Variable:   LAB1_DICTIONARY_H
Directive:   #define
Variable:   LAB1_DICTIONARY_H
DataType:   typedef
DataType:   struct
Delimiter:   {
DataType:   char
Operator:   *
Variable:   key
Delimiter:   ;
DataType:   int
Variable:   value
Delimiter:   ;
Delimiter:   }
Variable:   Pair
Delimiter:   ;
Comment:   /*
Random
long
comment
*/
DataType:   typedef
DataType:   struct
Delimiter:   {
Variable:   Pair
Operator:   *
Variable:   pairs
```

Приклад 2

main.cpp | input_2.txt | tokenizer.cpp | tokenizer.h | input_3.txt | outp

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "dfa.h"

#define MAX_WORD_LENGTH 30
#define MAX_FILENAME_LENGTH 100
const char* test_directory = "Tests/";
const char* result_directory = "Results/output";


int main() {
    char input_file_path[MAX_FILENAME_LENGTH];
    char file_name[MAX_FILENAME_LENGTH];

    FILE* input_file = NULL;
    while (input_file == NULL) {
        printf("Enter a file name: ");
        scanf("%s", file_name);
        char file_path[MAX_FILENAME_LENGTH];
        strcpy(file_path, test_directory);
        strcat(file_path, file_name);
        input_file = fopen(file_path, "r");

        if (input_file == NULL) printf("Couldn't open the input fi
        else strcpy(input_file_path, file_path);
    }

    DFA dfa;
    initialise_transition_map(&dfa);
    initialise_dfa(&dfa, input_file);
    printf("DFA initialized successfully.\n");
```

Lab3

```
/Users/stopfuture/CLionProjects/SystemProgramming/Lab3/c
Enter a file name: input_2.txt
Directive:   #include
Library:    <stdio.h>
Directive:   #include
Library:    <string.h>
Directive:   #include
Library:    <stdlib.h>
Directive:   #include
Library:    <ctype.h>
Directive:   #include
String:   "dfa.h"
Directive:   #define
Variable:   MAX_WORD_LENGTH
Number:   30
Directive:   #define
Variable:   MAX_FILENAME_LENGTH
Number:   100
Variable:   const
DataType:   char
Operator:   *
Variable:   test_directory
Operator:   =
String:   "Tests/"
Delimiter:   ;
Variable:   const
DataType:   char
Operator:   *
Variable:   result_directory
Operator:   =
String:   "Results/output"
Delimiter:   ;
```

Приклад 3

main.cpp | input_3.txt | tokenizer.cpp | tokenizer.h

```
#include <stdio.h>

int main() {
    int num1, num2, num3, sum;

    num1 = 56;
    num2 = 0b101;
    num3 = 0x10F

    float float_num = 0.1111;

    sum = num1 + num2 + num3;

    // Display the result
    printf("The sum of %d, %d, and %d is:
    printf("рандомні літерки");

    укр_текст

    return 0;
}
```

```
Enter a file name: input_3.txt
Directive:   #include
Library:    <stdio.h>
DataType:   int
Function:   main
Delimiter:   (
Delimiter:   )
Delimiter:   {
DataType:   int
Variable:   num1
Delimiter:   ,
Variable:   num2
Delimiter:   ,
Variable:   num3
Delimiter:   ,
Variable:   sum
Delimiter:   ;
Variable:   num1
Operator:   =
Number:   56
Delimiter:   ;
Variable:   num2
Operator:   =
BinNumber:   0b101
Delimiter:   ;
Variable:   num3
Operator:   =
HexNumber:   0x10F
DataType:   float
Variable:   float_num
Operator:   =
Number:   0.1111
Delimiter:   ;
Variable:   sum
Operator:   =
Variable:   num1
Operator:   +
Variable:   num2
Operator:   +
Variable:   num3
Delimiter:   ;
Comment:   // Display the result
FunctionCall:   printf
Delimiter:   (
String:   "The sum of %d, %d, and
Delimiter:   ,
Variable:   num1
Delimiter:   ,
Variable:   num2
Delimiter:   ,
Variable:   num3
Delimiter:   ,
Variable:   sum
Delimiter:   )
Delimiter:   ;
FunctionCall:   printf
Delimiter:   (
String:   "рандомні літерки"
Delimiter:   )
Delimiter:   ;
Uncategorized: укр_текст
Reserved:   return
Number:   0
```

Висновок

У даній лабораторній роботі була успішно виконана основна умова завдання, цей аналізатор допоможе легко та точно визначити та класифікувати лексеми в тексті програми, що є важливим етапом у розробці компіляторів та інших інструментів для обробки програмного коду.