# Laboratory 2

Title of the Laboratory Exercise: Program to find the longest word in a given string.

1. Introduction and Purpose of Experiment

Students learn to use Lex program to find out the longest word in a given string.

1. Aim and Objectives

Aim

* To write a program to fine the longest word in a given string

Objectives

At the end of this lab, the student will be able to

* Define regular expression for words
* Find the longest word in a given string

1. Experimental Procedure

Students are required to carry out the following steps:

* Algorithm
* Write the Lex program
* Compile and execute the program (steps)
* Complete the documentation for the given problem

1. Presentation of Results

**tokens.l**

%{

#include <iostream>

*extern* *int* maxlen;

*extern* std::string maxstring;

%}

**%option** noyywrap

%%

[a-zA-Z0-9\_\+\\*\/]**+**   {

                std::string curr\_string(yytext);

                std::cout << "[" << curr\_string << " : " << curr\_string.length() << "]\n";

                // if the maxstring length is less than the curr\_string then store this

                // curr\_string

                if ( maxstring.length() **<=** curr\_string.length() ) {

                    maxstring = curr\_string;

                }

            }

[ \t\r]     {   }

"\n"        { return 1; }

.           { std::cout << "[UNRECOGNIZED]\n"; }

**main.cpp**

#include <iostream>

#include <string>

*extern* *int* yylex();

std**::**string maxstring **=** "";

*auto* main(*int* **argc**, *char\** **argv**[]) -> *int* {

    std**::**cout **<<** "enter stream of characters" **<<** "\n";

    yylex();

    std**::**cout **<<** "max string: " **<<** maxstring **<<** ", len: " **<<** maxstring.length() **<<** "\n";

**return** 0;

}

1. Analysis and Discussions

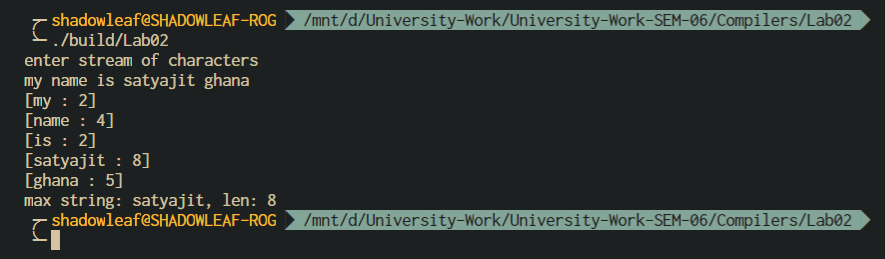


Figure 0‑1 OUTPUT

**Algorithm:**

find­­­\_maxstring():

* + - 1. max\_string = “”
      2. for each character string in stdin
      3. match regex [a-zA-Z0-9\_\+\\*\/]+
      4. curr\_string = yytext
      5. if (max\_string.length < curr\_string.length)
      6. max\_string = curr\_string
      7. display max\_string

1. Conclusions

**Lex Regular Expressions.**

A LEX regular expression is a word made of text characters (letters of the alphabet, digits, ...)

operators : " \ { } [ ] ^ $ < > ? . \* + | () /

A Character Class is a class of characters specified using the operator pair [ ]. The expression

[ab] matches the string a or b.

Within square brackets most operators are ignored except the three special characters \ - ^ are which used as follows

(a) the escape character \ as above,

(b) the minus character - which is used for ranges like in digit [0-9]

(c) the *hat* character ^ as first character after the opening square bracket, it is used for complemented matches like in

NOTabc [^abc]

1. Comments

a. Limitations of Experiments

The experiment does not define the characters that are included in a string; hence a few assumptions are made, alphabet + numbers + some special characters are included in the string

b. Limitations of Results

The solution program written takes input from stdin, which is not generalized, i.e. the program should also take input from file and parse the file to find the max string.

Dynamic resizing of the input buffer is slow, as it entails rescanning all the text matched so far by the current (generally huge) token. Due to both buffering of input and read-ahead, you cannot intermix calls to <stdio.h> routines, such as, getchar(), with flex rules and expect it to work.

c. Learning happened

We learnt how to define regular expression for words and find the longest word in a given string

d. Recommendations

The characters that consist a string should be well defined in the experiment.

Try to use standard C++ string functions for operations with yytext to avoid input buffer resizing problems with <stdio.h>.

|  |  |  |
| --- | --- | --- |
| **Component** | **Max Marks** | **Marks Obtained** |
| **Viva** | **6** |  |
| **Results** | **7** |  |
| **Documentation** | **7** |  |
| **Total** | **20** |  |