```cpp
Driver.cpp
//C++ Lexical Analyzer
//Sam Dressler
//CSCI 465
//9/26/2020
//This program takes in an input file and generates a list of lexeme-spelling pairs.
//The program takes the loads the file, generate symbols, and then classifies them into
// their appropriate lexemes.
//The output is generated into "output_lex.txt"
#include "lexical_analyzer.h"
#include "intermediate_code_generator.h"
#include "driver.h"
//Main Function to drive the modules.
int main(int argc, char * argv []){
    //Check to see if the user input a file
    if(argc != 2){
        cout << "Error: Enter the name of the pascal file to parse" << endl;
        exit(0);
    }
    //get the file name from user
    ifstream file(argv[1]);
    //Verify the file is opened.
    //If the file can not be opened, tell the user.
    if(!file.is_open()){
        cout << "Error file: "<<"\""<<argv[1]<<"\""<<" could not be opened"<<endl;
        exit(0);
    }
    //read from the file all of the characters and return to main
    char * file_as_chars = load_file(file);
    //Print raw input file
    cout << "--------------------------------------------------"<<endl;
    cout << "          Contents of: "<<"\""<<argv[1]<<"\""<<endl;
    cout << "--------------------------------------------------"<<endl;
    cout << file_as_chars << endl;
    cout << "--------------------------------------------------"<<endl;
    vector<symbol> lex_symbol_table = vector<symbol>();
    vector<icg_symbol> icg_symbol_table = vector<icg_symbol>();
    vector<string> symbols = vector<string>();
    symbols = generate_symbols(file_as_chars);
    // for(vector<string>::iterator it = symbols.begin(); it < symbols.end(); ++it){
    //     string temp = *it;
    //     cout << "SYM : " << temp << endl;
    // }
    lex_symbol_table = classify_symbols(symbols, lex_symbol_table);
    FILE * output_lex = NULL;
    output_lex = fopen("output_lex.txt","w");
```

```
    for(vector<symbol>::iterator it = lex_symbol_table.begin(); it != lex_symbol_table.end(); ++it)
    {
        symbol temp = *it;
        icg_symbol s;
        s.token_type = temp.token_type;
        s.value = temp.value;
        icg_symbol_table.push_back(s);
        const char * token_t = temp.token_type.c_str();
        const char * token = temp.value.c_str();
        //cout << left << setw(12) << temp.token_type <<  " -->    " << temp.value << endl;
        fprintf(output_lex,"%s %s\n",token_t, token);

    }
    cout << "-------------------------------------------------"<<endl;
    cout << "          Output file for lex created" << endl;
    cout << "-------------------------------------------------"<<endl;
    /*
        beginning of code for version 2
    */
    cout << "-------------------------------------------------"<<endl;
    cout << "      Beginning Intermediate Code Generation " <<endl;
    cout << "-------------------------------------------------"<<endl;

    generate_three_address_code(icg_symbol_table);
    cout << "-------------------------------------------------"<<endl;
    cout << "Output file for intermediate code generation created" << endl;
    cout << "-------------------------------------------------"<<endl;
    cout << "              Symbol table: " << endl;
    cout << "-------------------------------------------------"<<endl;
    print_sym_table();


    fclose(output_lex);

    return 0;
}
```

Driver.h

```
#include <iostream>
#include <fstream>
#include <string>
#include <string.h>
```

```cpp
#include <stdio.h>
#include <vector>
#include <regex>
#include <iomanip>

using namespace std;
```