|  |  |
| --- | --- |
| **Author Identification Block** | |
| **Author:** | Chris Graff |
| **Student ID:** | \*20274911 |
| **E-Mail:** | [cgraff@uco.edu](mailto:cgraff@uco.edu) |
| **Course:** | CMSC 2613 – Programming 2 |
| **CRN:** | 21641, Spring 2012 |
| **Project:** | p03 |
| **Due:** | February 3, 2012 |
| **Account:** | tt025 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Scoring Block** | | | |
| **Component** | **Available** | **Earned** | **Explanation** |
| Compilation |  |  |  |
| Submission Instructions | 2 |  |  |
| Author Identification | 1 |  |  |
| Modularity | 3 |  |  |
| Command Line | 3 |  |  |
| Input file | 3 |  |  |
| Output file | 3 |  |  |
| Execution | 10 |  |  |
| **Total** | **25** |  |  |

#-------------------------------------------------------------------------

# File p03make creates executable file p03.

#-------------------------------------------------------------------------

# Author: Thomas R. Turner

# E-Mail: tturner@ucok.edu

# Date: September, 2002

#-------------------------------------------------------------------------

#object files

#-------------------------------------------------------------------------

obj = p03.o Scan03.o Stack03.o

#-------------------------------------------------------------------------

# Bind p03.o, Scan03.o and Stack03.o

#-------------------------------------------------------------------------

p03: ${obj}

g++ -o p03 ${obj} -ll

#-------------------------------------------------------------------------

# Compile p03.cpp

#-------------------------------------------------------------------------

p03.o: p03.cpp Scan03.h Stack03.h

g++ -g -c p03.cpp

#-------------------------------------------------------------------------

# Compile Scan03.l. First translate the lex specification, then compile

#-------------------------------------------------------------------------

Scan03.o: Scan03.cpp Scan03.h

g++ -g -c Scan03.cpp

Scan03.cpp: Scan03.l Scan03.h

lex Scan03.l

mv lex.yy.c Scan03.cpp

#-------------------------------------------------------------------------

# Compile Stack03.cpp

#-------------------------------------------------------------------------

Stack03.o: Stack03.cpp Stack03.h

g++ -g -c Stack03.cpp

#include <cstdlib>

#include <cstdio>

#include <iostream>

#include <math.h>

#include <string.h>

#include <iosfwd>

#include <iomanip>

#include <fstream>

#include <string>

#include "Scan03.h"

#include "Stack03.h"

#include <ios>

using namespace std;

//----------------------------------------------------------------

//Author: Chris Graff

//StudentID#: \*20274911

//Email: cgraff@uco.edu

//Course: CMSC2613 Programming II

//CRN: 21641, Spring 2012

//Project: p04

//Due: February 10th, 2012

//Account: tt025

//----------------------------------------------------------------

struct CommandLineException

{

CommandLineException (int max, int actual)

{

cout <<endl <<"Too many command line arguements." <<endl;

cout <<"A maximum of " <<max <<" arguements are permitted." <<endl;

cout <<actual <<" arguements were entered." <<endl;

}

};

struct FileException

{

FileException (char\* filename)

{

cout <<endl <<"File " <<filename <<" could not be opened or doesn't exist" <<endl;

}

};

void postfixmgr(FILE\* i, ostream& o)

{

Scan L(i);

Stack S;

for(;;)

{

int t = L.Lex();

int h = 0;

if(t <=0) break;

switch(t)

{

case INTLIT:

S.push(L.Intlit());

break;

case PLUS:

S.push(S.pop() + S.pop());

break;

case MINUS:

h = S.pop();

S.push(S.pop() - h);

break;

case SLASH:

h = S.pop();

S.push(S.pop() / h);

break;

case STAR:

S.push(S.pop() \* S.pop());

break;

default:

cout << endl << "YOUR CRAP IS BROKE" << endl;

break;

}

}

S.print(o);

}

int main (int argc, char\* argv[])

{

try

{

char iFileName[255], oFileName[255];

switch (argc)

{

case 1:

cout <<"Enter the input file name:";

cin >> iFileName;

cout <<"Enter the output file name:";

cin >> oFileName;

break;

case 2:

strcpy(iFileName, argv[1]);

cout <<"Enter the output file name:";

cin >> oFileName;

break;

case 3:

strcpy(iFileName, argv[1]);

strcpy(oFileName, argv[2]);

break;

default:

throw CommandLineException (2, argc-1);

break;

}

FILE\* i=fopen(iFileName, "r");

if (!i)

throw FileException(iFileName);

ofstream o(oFileName);

if (!o)

throw FileException(oFileName);

postfixmgr(i, o);

fclose(i);

o.close();

}

catch (...)

{

cout <<"Program terminated." <<endl;

exit(EXIT\_FAILURE);

}

}

#ifndef Scan03\_h

#define Scan03\_h 1

//--------------------------------------------------------------------

// File: Scan03.h

// Description:

// Recognizes integers and arithmetic operators for project 3 in

// Programming II.

//--------------------------------------------------------------------

// Author: Thomas R. Turner

// E-Mail: trturner.ucok.edu

// Date: September, 2002

//--------------------------------------------------------------------

// Copyright September, 2002 by Thomas R. Turner

// Do not reproduce without permission from Thomas R. Turner.

//--------------------------------------------------------------------

//--------------------------------------------------------------------

// Standard C and C++ include files

//--------------------------------------------------------------------

#include <cstdio>

#include <fstream>

#include <iostream>

using namespace std;

//--------------------------------------------------------------------

//Token code definitions

//--------------------------------------------------------------------

#define INTLIT 1

#define PLUS 2

#define MINUS 3

#define STAR 4

#define SLASH 5

//--------------------------------------------------------------------

//Function: yylex

//Function yylex is the Scanner. Function yylex returns an integer

//token code as defined above or 0 if end-of-file has been

//reached.

//--------------------------------------------------------------------

#ifdef \_\_cplusplus

extern "C"

#endif

int yylex (void);

//--------------------------------------------------------------------

//Class Scan defines the attributes of a Scanner

//--------------------------------------------------------------------

class Scan {

int tokencode; //Code for the most recent token found

public:

Scan(FILE\* i); //Redirect the input source from the

//keyboard to input file i.

int Lex(void); //Call the scanner yylex and return the code

//found by yylex

int FetchTokenCode(void); //Return the code of the most recent token

void StoreTokenCode(int T); //Store the token code.

char\* FetchSpelling(void); //Return the spelling of the most recent

//token

int Intlit(void); //Return the most recent integer literal

};

#endif

%{

//--------------------------------------------------------------------

// File: Scan03.l

// Description:

// Contains the most elementary example use of lex for the purpose of

// building a scanner.

//-------------------------------------------------------------------

// Author: Thomas R. Turner

// E-Mail: trturner@ucok.edu

// Date: September, 2002

//--------------------------------------------------------------------

//Copyright September, 2002 by Thomas R. Turner.

//Do not reproduce without permission from Thomas R. Turner

//--------------------------------------------------------------------

//--------------------------------------------------------------------

// Standard C and C++ Library Include Files

//--------------------------------------------------------------------

#include <cstdio>

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

//--------------------------------------------------------------------

// Application Includes

//--------------------------------------------------------------------

#include "Scan03.h"

//--------------------------------------------------------------------

//Function prototypes

//--------------------------------------------------------------------

int TokenMgr(int T);

//--------------------------------------------------------------------

//Global Variables

//--------------------------------------------------------------------

%}

%%

[ \t\n]+ ;

[+-]?[0-9]+ {

return(TokenMgr(INTLIT));

}

"+" {

return(TokenMgr(PLUS));

}

"-" {

return(TokenMgr(MINUS));

}

"\*" {

return(TokenMgr(STAR));

}

"/" {

return(TokenMgr(SLASH));

}

%%

//--------------------------------------------------------------------

int TokenMgr(int T)

{ return T;

}

//--------------------------------------------------------------------

//Class Scan implementation

//--------------------------------------------------------------------

//Constructor Scan is used to redirect the input file stream from the

//keyboard to input file stream i.

//--------------------------------------------------------------------

Scan::Scan(FILE\* i)

{ yyin=i;

}

//--------------------------------------------------------------------

//Function Lex calls yylex

//--------------------------------------------------------------------

int Scan::Lex(void)

{ return tokencode=yylex();

}

//--------------------------------------------------------------------

//Function FetchSpelling returns a pointer to the spelling of the most

//recent token.

//--------------------------------------------------------------------

char\* Scan::FetchSpelling(void)

{ return (char\*)yytext;

}

//--------------------------------------------------------------------

//Function FetchTokenCode returns the code of the most recent token

//--------------------------------------------------------------------

int Scan::FetchTokenCode(void)

{ return tokencode;

}

//--------------------------------------------------------------------

//Function StoreTokenCode records the most recent token code

//--------------------------------------------------------------------

void Scan::StoreTokenCode(int T)

{ tokencode=T;

}

//--------------------------------------------------------------------

//Function Intlit returns the most recent integer literal

//--------------------------------------------------------------------

int Scan::Intlit(void)

{ int i;

sscanf(yytext,"%d",&i);

return i;

}

//-----------------------End of Lex Definition------------------------

#ifndef Stack03\_h

#define Stack03\_h

#include <cstdlib>

#include <iostream>

using namespace std;

class Stack

{

struct Element

{

Element\* prev;

int data;

Element(Element\* p, int v):prev(p), data(v)

{}

};

Element\* tos;

void Kill(Element\* e);

public:

Stack():tos(NULL){};

~Stack(){Kill(tos);}

bool IsEmpty();

bool IsFull();

void push(int v);

int pop();

void print(ostream& o);

};

#endif

#include "Stack03.h"

#include <cstdlib>

#include <iostream>

using namespace std;

struct stackex

{

stackex(string m)

{

cout << endl << "The Stack Is ";

cout << m << "." << endl;

}

};

bool Stack::IsEmpty()

{ return (tos == NULL); }

bool Stack::IsFull()

{ return false; }

void Stack::push(int v)

{

if(IsFull())

throw stackex("full");

Element\* e = new Element(tos, v);

tos = e;

}

int Stack::pop()

{

if(IsEmpty())

throw stackex("empty");

Element\* e = tos;

int v = e -> data;

tos = e -> prev;

delete e;

return v;

}

void Stack::Kill(Element\* e)

{

while(e)

{

Element\* p = e;

e = e -> prev;

delete p;

}

}

void Stack::print(ostream& o)

{

Element\* e=tos;

for(int a=0;e;--a, e=e->prev)

{

o << endl << "Stack[tos";

if(a<0) o << a;

o << "]= " << e->data;

}

o << endl;

}