Erika Valle-Baird

CIS 310

Program 4

//Program That Reads in a File, Splits the Data Into Hexadecimal Operands and Operator, and Outputs the Calculations

#include <iostream>

#include<string>

#include <fstream>

#include <ostream>

using namespace std;

//Function to Convert Character to Integer

int charToInt(char op) {

int val;

switch (op) {

case '0': {

val = 0;

break; }

case '1': {

val = 1;

break; }

case '2': {

val = 2;

break; }

case '3': {

val = 3;

break; }

case '4': {

val = 4;

break; }

case '5': {

val = 5;

break; }

case '6': {

val = 6;

break; }

case '7': {

val = 7;

break; }

case '8': {

val = 8;

break; }

case '9': {

val = 9;

break; }

case 'A': {

val = 10;

break; }

case 'B': {

val = 11;

break; }

case 'C': {

val = 12;

break; }

case 'D': {

val = 13;

break; }

case 'E': {

val = 14;

break; }

case 'F': {

val = 15;

break; }

}

return val;

}

//Function to Convert Integer Back to Character

char intToChar(int val) {

if (val <= 9)

return((char)('0' + val));

else

return((char)('A' + (val - 10)));

}

//Function to Delete Tens Place Zeros For Single Digit Integers

string deleteExtraZeros(string operand) {

int checkPlace = 0;

string answer = "";

for (checkPlace = 0; checkPlace < operand.length(); checkPlace++) {

if (operand.at(checkPlace) != '0')

break; }

for (int i = checkPlace; i < operand.length(); i++)

answer += operand.at(i);

return answer;

}

//Function to Reverse the Direction of the String For Calculations and Display

string stringReversal(string operand) {

string answer;

int position;

answer = "";

position = operand.length() - 1;

for (position; position >= 0; position--)

answer += operand.at(position);

return answer;

}

//Function to Add Two Operands

string addition(string op1, string op2) {

int op1Length, op2Length, larger, sum;

string calculate, answer;

op1Length = op1.length();

op2Length = op2.length();

if (op1Length > op2Length)

larger = op1Length;

else

larger = op2Length;

op1 = stringReversal(op1);

op2 = stringReversal(op2);

sum = 0;

calculate = "";

answer = "";

for (int i = 0; i < larger; i++) {

if (op1Length > i)

sum += charToInt(op1.at(i));

if (op2Length > i)

sum += charToInt(op2.at(i));

calculate += intToChar(sum % 16);

if (15 >= sum)

sum = 0;

else

sum = 1; }

if (sum == 1)

calculate += '1';

answer = stringReversal(calculate);

return answer;

}

//Function to Subtract to Operands

string subtraction(string op1, string op2) {

int op1Length, op2Length, larger, sum2Comp,makePositive;

string calculate, answer;

op1Length = op1.length();

op2Length = op2.length();

if (op1Length > op2Length)

larger = op1Length;

else

larger = op2Length;

op1 = stringReversal(op1);

op2 = stringReversal(op2);

sum2Comp = 0;

calculate = "";

answer = "";

for (int i = 0; i < larger; i++) {

if (op1Length > i)

sum2Comp += charToInt(op1.at(i));

if (op2Length > i)

sum2Comp -= charToInt(op2.at(i));

if (sum2Comp < 0) {

makePositive = 16 + sum2Comp;

calculate += intToChar(makePositive % 16);

sum2Comp = -1; }

else {

calculate += intToChar(sum2Comp % 16);

sum2Comp = 0; }

}

if (sum2Comp == -1)

calculate += '-';

answer = stringReversal(calculate);

return answer;

}

//Function to Help With Repeated Addition in Multiplication

string multiplyHelper(string op1, char op2) {

int product;

string answer;

product = charToInt(op2);

answer = "";

for (int i = 0; i < product; i++)

answer = addition(op1, answer);

return answer;

}

//Function to Perform Multiplication of Two Operands Via Addition

string multiply(string op1, string op2) {

int op2Length,position;

string answer, calculate;

op2Length = op2.length();

answer = "";

calculate = "";

for (int i = 0; i < op2Length; i++) {

position = op2Length - i - 1;

calculate = multiplyHelper(op1, op2.at(position));

for (int j = 0; j < i; j++)

calculate += '0';

answer = addition(answer, calculate); }

return answer;

}

//Function to Help Division via Repeated Subtraction

bool divisionHelper(string op1, string op2) {

bool answer;

string subCheck;

answer = true;

subCheck = subtraction(op1, op2);

if (subCheck.at(0) == '-')

answer = false;

return answer;

}

//Function to Perform Division on Two Operands Via Repeated Subtraction

string division(string op1, string op2, string& remainder) {

string quotient, tally;

quotient = "";

tally = "1";

remainder = op1;

while (divisionHelper(remainder, op2) == true) {

remainder = subtraction(remainder, op2);

quotient = addition(quotient, tally); }

return quotient;

}

//Function to Calculate Base 16 Position for Exponent in Power Function Using Repeated Addition

long hexPositionCalculator(long op2Position) {

long answer = 0;

for (long i = 0; i < op2Position; i++)

answer += 16;

return answer;

}

//Function to Add Char Amount to Base 16's Position for Exponent Calculation

long powerHelper(string op2) {

long sum, hexNum;

int position, op2Length, op2Num;

sum = 0;

op2Length = op2.length();

for (int i = 0; i < op2Length; i++) {

position = op2Length - i - 1;

hexNum = hexPositionCalculator(position);

op2Num = charToInt(op2.at(i));

sum += (hexNum + op2Num);

}

return sum;

}

//Function to Perform Power Operation of Two Operands Using Repeated Addition

string power(string op1, string op2) {

string answer;

long powerTotal;

answer = "1";

powerTotal = powerHelper(op2);

for (long i = 0; i < powerTotal; i++)

answer = multiply(answer, op1);

return answer;

}

//Function to Pull One Line Out of the File At a Time, Split into Operands and Operator

//Call Necessary Calculations, and Print to Screen

void bufferToOp(string line) {

int length, counter;

string op1, op2,remainder;

char op,current;

length = line.length();

counter = 0;

op1 = "";

op2 = "";

for (int i = 0; i < length; i++) {

current = line.at(i);

if ((current == '+') || (current == '-') || (current == '\*') || (current == '/') || (current == '$')) {

op = current;

counter++;

continue; }

if (current == '=')

break;

if (counter == 0)

op1 += current;

else

op2 += current; }

if ((op1.length() > 40) || (op2.length() > 40)) {

cout << "ERROR INVALID OPERAND LENGTH";

exit(0); }

if (op == '+') {

cout << addition(op1, op2);

}

if (op == '\*') {

cout << multiply(op1, op2);

}

if (op == '-') {

cout << deleteExtraZeros(subtraction(op1, op2));

}

if (op == '/') {

cout << " quotient ";

cout << division(op1, op2, remainder);

cout << ", remainder " << deleteExtraZeros(remainder);

}

if (op == '$') {

cout << power(op1, op2);

}

}

//Main Function to Read Input File and Call Function to Put Data into

int main() {

fstream inFile;

inFile.open("testDataProgram4.txt", ios::in);

int length;

string fileData;

if (!inFile) {

cout << "Error With Input File, Check File or Link!" << endl;

exit(0);

}

while (inFile >> fileData) {

cout << fileData;

bufferToOp(fileData);

cout << endl; }

return(0);

}

Output:

