컴파일러 과제-1

20192800 권대현

1. 과제 내용

* 이번 컴파일러 과제는 1장에서 설명한 Recursive-Descent Parsing 방식을 이용하여 수식이 문법에 맞는지를 검사하고 문법이 맞으면, 수식의 값을 계산하는 프로그램을 완성하는 것이다.
* 입력으로 주어지는 수식은 +, \*, (, ), 정수, 실수 들로 구성된다.
* 수식의 에러에 따라 에러 메시지를 출력한다.

1. 해결 방법

* 먼저 프로그램 실행 환경은 cygwin으로 설정하였다.

스크린샷, 텍스트, 소프트웨어, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

* 기본적인 프로그램 구조는 ‘컴파일러-1장 강의노트.pdf’의 1-18 페이지를 참고하여 작성하였다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* 다만, 과제를 수행하기 위해, 위 구조에 정수 또는 실수를 검출하는 함수 digit()을 추가했고, 실수 계산을 위해 함수들의 반환형을 double형으로 변경했다.
  + 위 구조대로라면, 한 자리 수의 숫자밖에 검사할 수 없다. 따라서 두 자리 이상의 정수나 실수를 검사하기 위해 digit()를 추가했다.
  + Digit()에서는 정수 뿐만 아니라, token이 POINT라면 숫자가 실수임을 판단하여 double형으로 값을 반환해준다.
* 열거형 TOKEN에는 실수의 소수점을 판별하기 위한 POINT 변수를 추가했다.
* Error()에서는 매개변수 i에 따라 switch문에서 error에 대한 상세한 메시지를 출력하도록 구현했다.
* 처음엔 실수 계산을 위해 반환형을 float으로 설정했었다. 그러나, float으로 계산할 경우 5 자리가 넘어가는 긴 자리 수 숫자들의 계산에서 결과값이 미묘하게 틀리는 경우가 발생했다.
  + 이는 float과 int간 형변환에서 데이터가 소실되는 결과라고 판단하였고, 이를 줄이기 위해 float보다 데이터가 큰 double형으로 result를 반환해줬더니 해결되었다.

1. 결론

* 수식을 사용자로부터 입력 받은 뒤 수식의 문법이 틀리면 상세한 에러 메시지와 함께 에러를 출력하고 종료를, 수식의 문법이 맞다면 수식의 값을 계산하여 출력하고 종료를 수행하는 프로그램을 완성했다.
* 프로그램 실행 결과와 원시프로그램은 아래와 같다.
* 프로그램 실행결과
  + 정수형 수식

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + 실수형 수식

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + 괄호가 포함된 수식

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + 문법이 잘못된 수식
    - Error 1

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + - Error 2

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + - Error 3

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* 원시프로그램

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define MAX 100

enum TOKEN {

    NONE = 0,

    PLUS,

    STAR,

    NUMBER,

    POINT,

    LPAREN,

    RPAREN,

    END

} token;

int exprIndex = -1;

int startIndex;

int endIndex;

char ch;

char inputExpression[MAX];

void get\_token();

double expression();

double term();

double factor();

double digit();

void error(int i);

void main()

{

    double result = 0;

    int k = 0;

    char str[MAX];

    scanf("%[^\n]s", inputExpression);

    for (int i = 0; i < strlen(inputExpression); i++)

    {

        if (inputExpression[i] == NULL)

            break;

        if (inputExpression[i] != ' ')

            str[k++] = inputExpression[i];

    }

    str[k] = NULL;

    strcpy(inputExpression, str);

    get\_token();

    result = expression();

    if (token != END)

        error(3);

    else

    {

        if (result - (int)result == 0)

            printf("\n%d", (int)result);

        else

            printf("\n%f", result);

    }

}

void get\_token()

{

    ch = inputExpression[++exprIndex];

    if (ch == NULL)

        token = END;

    else if (ch == '+')

        token = PLUS;

    else if (ch == '\*')

        token = STAR;

    else if (ch == '(')

        token = LPAREN;

    else if (ch == ')')

        token = RPAREN;

    else if (ch == '.')

        token = POINT;

    else if (isdigit(ch))

        token = NUMBER;

    else

        token = NONE;

}

double expression()

{

    double result = 0;

    result = term();

    while (token == PLUS)

    {

        get\_token();

        result = result + term();

    }

    if (result - (int)result == 0)

        return (int)result;

    else

        return result;

}

double term()

{

    double result = 0;

    result = factor();

    while (token == STAR)

    {

        get\_token();

        result = result \* factor();

    }

    if (result - (int)result == 0)

        return (int)result;

    else

        return result;

}

double factor()

{

    double result = 0;

    if (token == NUMBER)

        result = digit();

    else if (token == LPAREN)

    {

        get\_token();

        result = expression();

        if (token == RPAREN)

            get\_token();

        else

            error(2);

    }

    if (result - (int)result == 0)

        return (int)result;

    else

        return result;

}

double digit()

{

    double result = 0;

    char ch[MAX];

    startIndex = exprIndex;

    //integer

    while (token == NUMBER)

    {

        get\_token();

    }

    endIndex = exprIndex;

    strncpy(ch, inputExpression + startIndex, exprIndex);

    ch[exprIndex + 1] = NULL;

    result = (float)atoi(ch);

    if (token == POINT)

    {

        //float

        get\_token();

        while (token == NUMBER)

        {

            get\_token();

        }

        strncpy(ch, inputExpression + startIndex, exprIndex);

        ch[exprIndex + 1] = NULL;

        result = atof(ch);

    }

    if (result - (int)result == 0)

        return (int)result;

    else

        return result;

}

void error(int i)

{

    printf("ERROR(%d): ", i);

    switch (i)

    {

    case 1:

        printf("Factor Error\n");

        break;

    case 2:

        printf("RightParen Error\n");

        break;

    case 3:

        printf("End Error\n");

        break;

    }

    exit(1);

}