

LAPORAN TUGAS BESAR
IF2220/ Teori Bahasa Formal dan Automata



Dipersiapkan oleh:
Muhammad Raihan Asyraf Desanto /13517027
M. Rifky I. Bariansyah / 13517081
Gardahadi /13517144

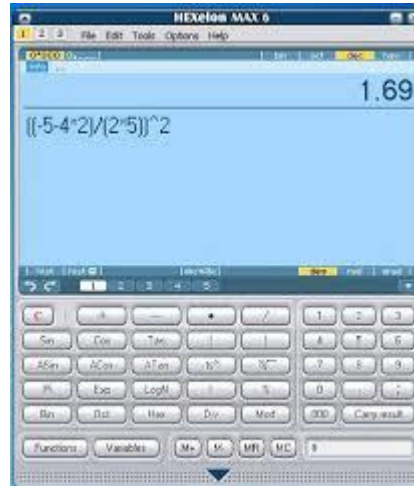
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2018

A. Deskripsi Masalah

Kalkulator adalah alat untuk menghitung dari perhitungan sederhana seperti penjumlahan, pengurangan, perkalian, dan pembagian, sampai kepada kalkulator sains yang dapat menghitung rumus matematika tertentu. Pada perkembangannya saat ini, kalkulator sering dimasukkan sebagai fungsi tambahan dari komputer dan telepon genggam.



Gambar 1.1 Contoh Kalkulator

Pada tugas kedua TBFO ini, penulis diminta untuk membuat sebuah kalkulator sederhana, yang menggunakan implementasi tata bahasa bebas konteks (CFG) dan/atau pushdown automata (PDA). Bila kalkulator diberikan sebuah ekspresi matematika, program harus bisa mengenali apakah ekspresi tersebut valid atau tidak (syntax error). Bila ekspresi tersebut sudah valid, program akan menghitung nilai dari ekspresi tersebut dengan mengubah terlebih dahulu setiap simbol terminal (angka) menjadi nilai numerik yang bersesuaian. Program juga harus dapat mengenali apakah ekspresi tersebut mungkin dihitung atau tidak (math error).

Contoh ekspresi matematika yang valid adalah $(-457.01+1280) * (35.7-11.0233)/(-6.1450)$ (setelah di-enter akan menampilkan hasil perhitungan ekspresi tersebut yaitu -3304.91) . Contoh ekspresi tidak valid adalah $3*+-12/(57)$ (setelah di-enter akan ditampilkan pesan "SYNTAX ERROR"), atau $(-5)^(2/3)$ (setelah di-enter akan ditampilkan pesan "MATH ERROR").

B. Gambaran Umum Solusi

Program ini menggunakan algoritma *Recursive Descent Parser* sebagai algoritma implementasi.

Secara garis besar, program bekerja dengan mengikuti langkah-langkah berikut :

- 1) Program melakukan *looping* meminta masukan berupa string dari pengguna selama masukan bukan “exit”. Didalam program di deklarasikan char *P sebagai pointer yang menunjuk ke karakter pada string masukan.
- 2) Program kemudian menjalankan algoritma *Recursive Descent Parser* berdasarkan string yang dimasukan oleh pengguna. Digunakan fungsi charToDouble untuk mengubah sebuah char menjadi double
- 3) Algoritma akan menjalankan fungsi OperationA(), urutan eksekusi operasi terakhir, yang mengeksekusi penjumlahan dan pengurangan.
- 4) Didalam OperationA() dideklarasikan sebuah variabel double yang menyimpan nilai kembalian dari OperationB(), urutan eksekusi sebelum OperationA(), yang mengeksekusi operasi pembagian dan perkalian.
- 5) Didalam OperationB() dideklarasikan sebuah variabel double yang menyimpan nilai kembalian dari Expn(), urutan eksekusi sebelum OperationB(), yang mengeksekusi operasi perpangkatan.
- 6) Didalam Expn() dideklarasikan sebuah variabel double yang menyimpan nilai kembalian dari CheckNumber(), yang mengeksekusi operasi pembagian dan perkalian.
- 7) Fungsi CheckNumber() mengembalikan Error bila sebelum tanda ‘-’ tidak terdapat ‘(’, mengembalikan -GetNumber() jika terdapat ‘-’, mengembalikan CalcPar() jika terdapat ‘(’, selain itu program akan mengembalikan GetNumber().
- 8) GetNumber() digunakan untuk mengembalikan nilai bilangan double dari masukan. Digunakan variabel f untuk menyimpan nilai yang akan dikembalikan. Pointer bergerak hingga yang ditunjuk bukan angka. f menyimpan nilai dengan mengalikan f dengan sepuluh setiap jumlah angka bertambah dan mengalikan 1/10 bila angka yang bertambah dibelakang ‘.’.
- 9) CalcPar() digunakan untuk menjalankan operasi didalam kurung. CalcPar akan menyimpan kembalian dari OperationA pada sebuah variabel, kemudian mengecek apakah penggunaan kurung. Bila benar maka akan mengembalikan nilai dari OperationA bila tidak akan mengembalikan error.

- 10) Setelah menyimpan keluaran dari CheckNumber, maka Expn() akan menggunakan CalcExp() untuk membantu perhitungan pangkat.
- 11) CalcExp() menghitung pangkat secara rekursif hingga tidak terdapat operasi '^' lagi.
- 12) Bila pada OperationB(), Expn() tidak sama dengan Error maka akan dilakukan perkalian atau pembagian dengan mengecek nilai Expn() pada karakter setelah '/' atau '*'. Bila pembilang pada sebuah operasi pembagian sama dengan 0 maka akan dikembalikan nilai MathError,
- 13) Bila pada OperationA(), OperationB() tidak sama dengan Error maka akan dilakukan penjumlahan atau pengurangan dengan mengecek nilai OperationB() pada karakter setelah '+' atau '-'.
- 14) Program main akan menampilkan nilai keluaran dari OperationA(), bila keluaran Error maka akan menampilkan pesan ("SYNTAX ERROR") sementara bila keluaran MathError akan menampilkan pesan ("MATH ERROR").

C. CFG yang Digunakan

Secar formal, sebuah *Context Free Grammar* (G) di definisikan dengan *4-tuple* sebagai berikut :

$$G = (V, \Sigma, R, S)$$

Dengan V adalah himpunan dari simbol *non-terminal*, Σ adalah himpunan dari *terminal*, R adalah aturan produksi dari CFG yang memetakan V ke $(V \cup \Sigma)^*$, dan S adalah *Start Symbol* yang merupakan anggota dari V. Berikut adalah notasi formal dari CFG yang kami implementasikan dalam program :

$$G = (\{A,B,C,D,N\}, \{+,-,\wedge,*,/,(,),\wedge,1,2,3,4,5,6,7,8,9,0\}, R, A)$$

Berikut adalah aturan produksi R dari CFG kami :

Aturan Produksi	Fungsi implentasi
$A \text{ (start)} \rightarrow N + N \mid N+A \mid N - N \mid N - A \mid N \mid B$	OperationA ()
$B \rightarrow N * N \mid N * B \mid N / N \mid N / B \mid C$	OperationB ()
$C \rightarrow N \wedge N \mid N \wedge C \mid D$	OperationC()
$D \rightarrow (A) \mid (-A) \mid D+A \mid N \mid D-A \mid D^A \mid D/A \mid D*A$	CheckSymbol ()
$N \rightarrow 1N \mid 2N \mid 3N \mid 4N \mid 5N \mid 6N \mid 7N \mid 8N \mid 9N \mid 0$	GetNumber {}

D. Source Code

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "rgbcalculator.c"
#include "boolean.h"

char *P;
boolean OpExist;
int main ()
{
    system("clear");
    printf("===== [ A simple arithmetic expression parser and calculator ] =====\n");
    printf("\n");
    printf("Welcome to RGB Calculator, please input a string (use 'exit' to close): ");
    scanf("%s", InputString);
    while(strcmp(InputString, "exit")){
        P = InputString;
        double result = OperationA();
        if(OpExist){
            if (result == MathError || *P != '\0')
            {
                printf("MATH ERROR\n");
                printf("Please try again : ");
            }
            else if (result == Error || *P != '\0')
            {
                printf("SYNTAX ERROR\n");
                printf("Please try again : ");
            }
        }
        else {
            printf("Result : %.2f\n", result);
            printf("Way to go! Write another one : ");
        }
    }
    else {
        printf("Result : %s\n", InputString);
        printf("Way to go! Write another one (This time use an operator!) : ");
    }

    scanf("%s", InputString);
}
}
```

boolean.h

```
#ifndef _BOOLEAN_h
#define _BOOLEAN_h

#define boolean unsigned char
#define true 1
#define false 0

#endif
```

rgbcalculator.h

```
#include "boolean.h"

#ifndef RGBCALCULATOR_H_INCLUDED
#define RGBCALCULATOR_H_INCLUDED

#define Error -99999.99
#define MathError -99999.98

extern char *P; // Akan digunakan sebagai penunjuk character
extern boolean OpExist; //Menandakan apakah terdapat operator dalam ekspresi

boolean IsNumber(char p);
// Menghasilkan true jika karakter merupakan angka [0..9]

double CheckSymbol(char *p);
// Mengecek apakah bilangan berada dalam tanda kurung, bernilai negatif atau tidak keduanya

double GetNumber();
// Mengubah string angka menjadi double

double OperationA();
// Memproses dan menghasilkan hasil dari penjumlahan atau pengurangan pada bilangan

double OperationB();
// Memproses dan menghasilkan hasil dari perkalian atau pembagian pada bilangan

double CalcExp(double f);
// Memproses perpangkatan pada bilangan

double Expn();
// Menghasilkan hasil perpangkatan bilangan

double CalcPar();
// Memproses dan menghasilkan hasil dari operasi bilangan dalam tanda kurung

void ExpImg(double a, double b);
//mencetak nilai akar bilangan minus

#endif
```

rgbcalculator.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "boolean.h"
#include "rgbcalculator.h"

double charToDouble(char c)
// Char to double
{
    return (c - '0');
}

boolean IsNumber(char p)
{
    return(p >= '0' && p <= '9');
}

double CheckSymbol(char *p)
{
    float check;
    char *prev;

    if (*P == '-')
    {
        prev = p-1;
        // Mengecek apakah sebelum bilangan negatif ada tanda kurung atau tidak
        if (*prev != '(')
        {
            return(Error);
        }
    }
    else
    {
        P++;
        check = GetNumber();
        if (check != Error)
        {
            return ((-1) * check);
        }
    }
    else
    {
        return (Error);
    }
}

}
else if (*P == '(')
{
    P++;
    return (CalcPar());
}
else
{
    return (GetNumber());
}
}

double GetNumber()
{
    double f = 0;
    if (IsNumber(*P))
    {
        while (IsNumber(*P))
        {
            f = (f*10) + charToDouble(*P);
            P++;
        }
    }
}
```



```

    if (*P == '.')
    {
        P++;
        double mul = 0.1;
        if (!IsNumber(*P))
        {
            return (Error);
        }
        else
        {
            while (IsNumber(*P))
            {
                f = f + charToDouble(*P)*mul;
                mul /= 10;
                P++;
            }
        }
    }
    return (f);
}
else
{
    return (Error);
}
}

```

```

double OperationA()
//Addition and Subtraction Operation
{
    double a;
    a = OperationB();

    if (a == Error)
    {
        return (Error);
    }
    else if (a == MathError)
    {
        return (MathError);
    }
    else
    {
        while ((*P == '+') || (*P == '-'))
        {
            OpExist = true;
            char Op;
            Op = *P;
            P++;
            double b;
            b = OperationB();
            if (b == Error)
            {
                return (Error);
            }
            else if (b == MathError)
            {
                return (MathError);
            }

            if (Op == '+')
            {
                a = a + b;
            }
            else if (Op == '-')
            {
                a = a - b;
            }
        }
        return a;
    }
}

```

```

double OperationB()
//Order of Multiplication and Division Operation
{
    double a;

    a = Expn();

    if (a == Error)
    {
        return (Error);
    }
    else if (a == MathError)
    {
        return (MathError);
    }

    while ((*P == '*' || (*P == '/'))
    {
        OpExist = true;
        char Op;
        Op = *P;
        P++;

        double b;

        b = Expn();

        if (b == Error)
        {
            return (Error);
        }
        else if (b == MathError)
        {
            return (MathError);
        }

        if (Op == '*')
        {
            a = b * a;
        }
        else if (Op == '/')
        {
            if(b == 0) // if divided by zero
            {
                return MathError;
            }
            else
            {
                a = a / b;
            }
        }
    }
    return a;
}

double CalcPar()
//Parentheses calculation
{
    double a;
    a = OperationA();
    if (*P == ')')
    {
        P++;
        return a;
    }
    else
    {
        return (Error);
    }
}

```

```

double Expn()
{
    double a;

    a = CheckSymbol(P);

    if ((*P == '^') && (a != Error))
    {
        OpExist = true;
        P++;
        a = CalcExp(a);
    }
    return (a);
}

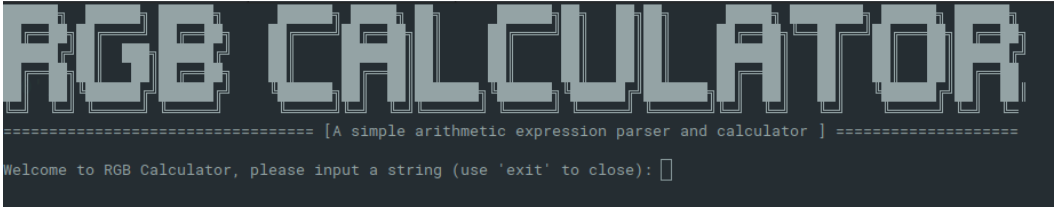
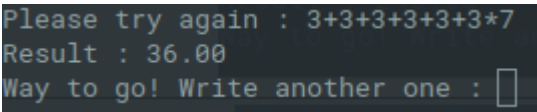
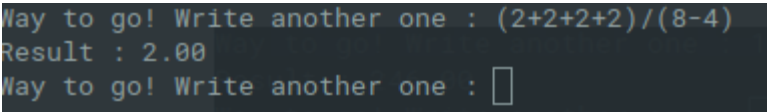
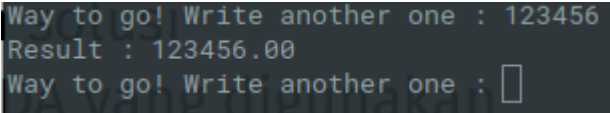
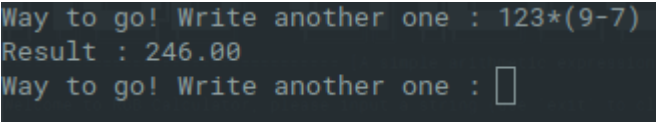
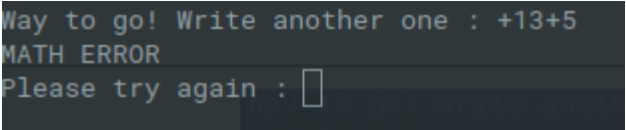
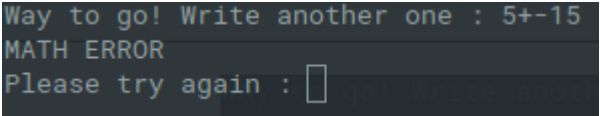
double CalcExp (double a)
{
    double b;

    b = CheckSymbol(P);
    // printf("pada fungsi calcExp, b sekrang bernilai : %f\n",b);
    if (b == Error)
    {
        return (Error);
    }
    else if (*P != '^')
    {
        if (((0<b && b<1) || (-1<b && b<0)) && (a <0))
        {
            return MathError;
        }
        return (pow(a,b));
    }
    else //Recurrence
    {
        P++;
        double c;

        c = CalcExp(b);
        if (c == Error)
        {
            return (Error);
        }
        else if (c == MathError)
        {
            return (MathError);
        }
        else
        {
            return (pow(a,c));
        }
    }
}

```

E. Contoh Masukkan dan Keluaran yang Diterima Program

Masukan	Hasil Keluaran di layar
*Tampilan Awal *	
3+3+3+3+3+3*7 VALID	
(2+2+2+2)/(8-4) VALID	
12345 VALID	
123*(9-7) VALID	
+13+5 INVALID	
5+-15 INVALID	

5+(-15) VALID	Way to go! Write another one : 5+(-15) Result : -10.00 Way to go! Write another one : <input type="text"/>
2^2^3 VALID	Way to go! Write another one : 2^2^3 Result : 256.00 Way to go! Write another one : <input type="text"/>
4^(0.5) VALID	Way to go! Write another one : 4^(0.5) Result : 2.00 Way to go! Write another one : <input type="text"/>
4^(-1) VALID	Way to go! Write another one : 4^(-1) Result : 0.25 Way to go! Write another one : <input type="text"/>
(3+2 INVALID	Way to go! Write another one : (3+2 SYNTAX ERROR Please try again : <input type="text"/>
3++++2 INVALID	Please try again : 3++++2 MATH ERROR Please try again : <input type="text"/>
3* INVALID	Please try again : 3* SYNTAX ERROR Please try again : <input type="text"/>
((((4+3))) VALID	Please try again : (((4+3))) Result : 7.00 Way to go! Write another one : <input type="text"/>

