IF2211 STRATEGI ALGORITMA

PENYELESAIAN PERMAINAN KARTU 24 DENGAN ALGORITMA BRUTE FORCE

Disusun oleh:

Athif Nirwasito **13521053**

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

DAFTAR ISI

Contents

BAB I

DESKRIPSI MASALAH

Permainan kartu 24 (24 game) adalah sebuah permainan kartu aritmatika. Permainan ini menarik cukup banyak peminat dikarenakan dapat meningkatkan kemampuan berhitung serta mengasah otak agar dapat berpikir dengan cepat dan akurat. Dalam permainan kartu 24, dari satu set kartu remi, pemain mengambil 4 kartu secara acak. Kartu yang tersedia adalah As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King, dengan As bernilai 1, Jack bernilai 11, Queen bernilai 12, dan King bernilai 13. Dari 4 kartu tersebut,pemain harus menyusun sebuah persamaan dengan urutan yang bebas dari kartu-kartu tersebut menggunakan operasi-operasi dasar matematika, yaitu pertambahan (+), pengurangan (−), perkalian (×), pembagian (÷), serta tanda kurung buka(() dan tanda kurung tutup ()) sehingga persamaan yang dihasilkan bernilai 24. Permainan berakhir jika pemain berhasil menemukan solusi persamaan yang menghasilkan nilai 24 tersebut.

BAB II

TEORI SINGKAT

2.1 24 GAME

2.1.1 Cara Main 24 Game

Permainan dimulai dengan pemain mengambil 4 kartu atau moderator memberikan 4 kartu ke pemain dari satu set kartu remi. Empat kartu yang didapatkan di interpretasi sebagai angka dengan As sebagai 1, Jack sebagai 11, Queen sebagai 12, dan King sebagai 13. Pemain harus menyusun 4 kartu tersebut menjadi sebuah persamaan dengan operasi-operasi matematika dasar, yaitu (+), pengurangan (−), perkalian (×), pembagian (÷), serta tanda kurung sehingga hasil persamaan yang dihasilkan bernilai 24. Urutan angka boleh disusun secara sembarang, begitu juga dengan operasi dasar matematika selama persamaan matematika yang dihasilkan valid secara sintaks. Permainan berakhir jika ditemukan solusi yang bernilai sama dengan 24.

2.1.2 Kemungkinan Solusi 24 Game

Secara umum, persamaan yang dihasilkan memiliki format sebagai berikut,

x1 op1 x2 op2 x3 op3 x4

Dengan tanda kurung, didapatkan tambahan sepuluh format solusi, yaitu

(x1 op1 x2) op2 x3 op3 x4

x1 op1 (x2 op2 x3) op3 x4

x1 op1 x2 op2 (x3 op3 x4)

(x1 op1 x2 op2 x3) op3 x4

x1 op1 (x2 op2 x3 op3 x4)

((x1 op1 x2) op2 x3) op3 x4

(x1 op1 (x2 op2 x3)) op3 x4

x1 op1 ((x2 op2 x3) op3 x4)

x1 op1 (x2 op2 (x3 op3 x4))

(x1 op1 x2) op2 (x3 op3 x4)

Kemudian, terdapat 3 operasi matematika dengan setiap operasi memiliki 4 pilihan sehingga terdapat 4^3 banyak cara untuk memilih operator. Dari 4 kartu yang didapatkan, jika keempat kartu unik, maka terdapat 4! cara untuk menyusun 4 kartu. Secara keseluruhan, terdapat 16896 cara untuk calon solusi untuk menyelesaikan permainan kartu 24.

2.2 ALGORITMA BRUTE FORCE

2.2.1 Definisi Algoritma Brute Force

Algoritma brute force adalah sebuah algoritma dengan pendekatan yang lempang (straightforward) untuk menyelesaikan suatu persoalan. Algoritma ini didasarkan pada pernyataan pada persoalan serta definisi/konsep yang dilibatkan. Brute Force memiliki cara pendekatan yang sederhana dan langsung serta penyelesaian yang jelas caranya.

2.2.2 Karakteristik Algoritma Brute Force

Algoritma brute force merupakan pendekatan yang tidak konstruktif atau kreatif dibandingkan strategi pemecahan masalah lainnya. Algoritma brute force umumnya tidak cerdas yang mengakibatkan algoritma yang dibuat tidak efisien sehingga dibutuhkannya komputasi yang besar dan waktu yang lama dalam penyelesaiannya. Namun, algoritma brute force merupakan algoritma yang hampir semua persoalan dapat diselesaikannya. Terkadang, satu-satunya cara untuk menyelesaikan suatu masalah adalah dengan memanfaatkan algoritma brute force. Algoritma brute force sederhana untuk dibuat dan untuk dipahami. Selain itu, algoritma brute force dijadikan sebagai standar untuk beberapa tugas komputasi.

BAB III

IMPLEMENTASI PROGRAM

```
//main
int main(){
    int input;
    int x1,x2,x3,x4;
    boolean run;
    FILE *ptr;
    run = true;
    List l;
    CreateList(&l);
    while(run){
        //take input
        //perbaiki textnya bang
        printf("24 SOLVER\n");
        printf("\n\nKetik:\n");
        printf("(1) Masukan Keyboard\n");
        printf("(2) Random Generate\n");
        printf("(0) Exit\n");
        while(!get_input1(&input)){
            printf("Input tidak valid!\n");
        }

        if(input == 0){
            run = false;
        }else if (input == 1)
        {//ambil masukan keyboard
            printf("Masukan nilai kartu: ");
            while(!get_input2(&x1, &x2, &x3, &x4)){
                printf("Input tidak valid!\n");
                printf("Masukan nilai kartu: ");
            }
        }else if (input==2)
        {
            srand( time(NULL));
            x1 = rand()%13 + 1;
            x2 = rand()%13 + 1;
            x3 = rand()%13 + 1;
            x4 = rand()%13 + 1;
            printf("Kartu yang didapat:\n");
            card(x1); printf(" "); card(x2); printf(" "); card(x3); printf(" "); card(x4); printf("\n");
        }
        if(input == 1 || input ==2){
            // process
            clock_t begin = clock();
            find_solution_alt(&l, x1, x2, x3, x4);
            delete_duplicate(&l);
            clock_t end = clock();
            double time_spent = (double)(end-begin) / CLOCKS_PER_SEC;
            printf("Execution time %.2f\n", time_spent);
            printall(l);

            //ask if output want to be saved
            printf("Simpan Jawaban dalam file?\n");
            printf("(1) YA\n");
            printf("(0) Tidak\n");
            while(!get_input3(&input)){
                printf("Input salah\n");
            }
            if(input==1){
                writeFile(l, x1, x2, x3, x4, time_spent);
            }
            //clear list
            clear_list(&l);
            printf("\n\n\n");
        }


    }
    return 0;
}
```

Gambar 3.1 Main Program

Program pertama-tama menerima input untuk memilih cara pengambilan 4 kartu, apakah acak atau memilih sendiri.

```
void find_solution_alt(List *l, int x1, int x2, int x3, int x4){

    char EQ[MaxEl];
    fraction y1,y2,y3,y4, test;
    int i, j, k, a, b, c, d;
    //start timer

    //transform to frac
    y1 = transformItoF(x1);
    y2 = transformItoF(x2);
    y3 = transformItoF(x3);
    y4 = transformItoF(x4);
    char ops[4] = {'+', '-', '*', '/'};
    fraction y[4] = {y1, y2, y3, y4};
```

```
for(a=0;a<4;a++){
    for(b=0;b<4;b++){
        if(a==b){continue;}
        for(c=0;c<4;c++){
            if(a==c || b==c){continue;}
            d = 6-a-b-c;
                if(a==d || b==d || c==d){continue;}
                for(i=0;i<4;i++){
                    for(j=0;j<4;j++){
                        for(k=0;k<4;k++){
                            fraction temp[4] = {y[a], y[b], y[c], y[d]};
                            //case one by one
                            //bracket mode and its
                            //1. ((a op1 b) op2 c) op3 d
                            sprintf(EQ, "((%d%c%d)%c%d)%c%d\n", 0,ops[i],1,ops[j],2,ops[k],3);
                            if(is24(EQ,temp)){
                                insertFirst(1, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 1,ops[i], ops[j], ops[k]);
                            }

                            //2. (a op1 (b op2 c)) op3 d
                            sprintf(EQ, "(%d%c(%d%c%d))%c%d\n", 0,ops[i],1,ops[j],2,ops[k],3);
                            if(is24(EQ,temp)){
                                insertFirst(1, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 2,ops[i], ops[j], ops[k]);
                            }
```

```
                //3. a op1 ((b op2 c) op3 d)
                sprintf(EQ, "%d%c((%d%c%d)%c%d)\n", 0,ops[i],1,ops[j],2,ops[k],3);
                if(is24(EQ,temp)){
                    insertFirst(l, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 3,ops[i], ops[j], ops[k]);
                }

                //4. a op1(b op2 (c op3 d))
                sprintf(EQ, "%d%c(%d%c(%d%c%d))\n", 0,ops[i],1,ops[j],2,ops[k],3);
                if(is24(EQ,temp)){
                    insertFirst(l, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 4,ops[i], ops[j], ops[k]);
                }

                //case pair by pair
                //5. (a op1 b) op2 (c op3 d)
                sprintf(EQ, "(%d%c%d)%c(%d%c%d)\n", 0,ops[i],1,ops[j],2,ops[k],3);
                if(is24(EQ,temp)){
                    insertFirst(l, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 5,ops[i], ops[j], ops[k]);
                }
                /*One bracket, 3 integers*/
                /*6.(a op1 b op2 c) op3 d*/
                sprintf(EQ, "(%d%c%d%c%d)%c%d\n", 0,ops[i],1,ops[j],2,ops[k],3);
                if(is24(EQ,temp)){
                    insertFirst(l, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 6,ops[i], ops[j], ops[k]);
                }
                /*7.a op1 (b op2 c op3 d)*/
                sprintf(EQ, "%d%c(%d%c%d%c%d)\n", 0,ops[i],1,ops[j],2,ops[k],3);
                if(is24(EQ,temp)){
                    insertFirst(l, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 7,ops[i], ops[j], ops[k]);
                }
                /*one bracket, 2 integers*/
                /*8.(a op1 b) op2 c op3 d*/
                sprintf(EQ, "(%d%c%d)%c%d%c%d\n", 0,ops[i],1,ops[j],2,ops[k],3);
                if(is24(EQ,temp)){
                    insertFirst(l, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 8,ops[i], ops[j], ops[k]);
                }
```

```
                }
                /*9.a op1 (b op2 c) op3 d*/
                sprintf(EQ, "%d%c(%d%c%d)%c%d\n", 0,ops[i],1,ops[j],2,ops[k],3);
                if(is24(EQ,temp)){
                    insertFirst(l, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 9,ops[i], ops[j], ops[k]);
                }
                /*10.a op1 b op2 (c op3 d)*/
                sprintf(EQ, "%d%c%d%c(%d%c%d)\n", 0,ops[i],1,ops[j],2,ops[k],3);
                if(is24(EQ,temp)){
                    insertFirst(l, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 10,ops[i], ops[j], ops[k]);
                }

                /*No brackets*/
                /*11.a op1 b op2 c op3 d*/
                sprintf(EQ, "%d%c%d%c%d%c%d\n", 0,ops[i],1,ops[j],2,ops[k],3);
                if(is24(EQ,temp)){
                    insertFirst(l, NUM(y[a]), NUM(y[b]), NUM(y[c]), NUM(y[d]), 11,ops[i], ops[j], ops[k]);
                }

            }
        }
    }
    }
}
```

Gambar 3.2 Penyelesaian kartu 24 menggunakan brute force

Setelah mendapatkan input, keempat kartu diubah menjadi fraksi. Hal ini dilakukan untuk mempermudah kalkulasi dan mencegah kasus adanya informasi yang hilang ditengah-tengah kalkulasi. Setelah itu, program mulai mencari semua solusi dengan menggunakan pendekatan brute force.

1. Urut operator dan kartu dalam berturut-turut array of char dan array of fraction.

2. Lakukan traversal dari 0 hingga 3, misal pada variable ‘a’. Lakukan traversal juga pada variabel b dan c, namun jika antara a, b dan c ada nilai yang sama, maka lanjut ke kasus selanjutnya. Jika a, b, c saling unik, pilih nilai d dari 0 hingga 3 yang berbeda dari a,b, dan c. Keempat variable tersebut digunakan sebagai urutan angka dengan urutan a op1 b op2 c op3 d, dengan a,b,c,d merupakan indeks yang mengakses array yang menyimpan kartu.
3. Lakukan traversal lagi dari 0 hingga 3 lagi di dalam traversal sebelumnya, misal i,j, dan k. Ketiga variabel tersebut akan mengakses array yang berisi operator. Oleh karena itu, i, j, dan k boleh sama. Dengan ini, semua kemungkinan operator dan permutasi kartu akan teruji.
4. Pada urutan kartu dan susunan operator tersebut, “a i b j c k d”, uji pada 11 kasus tanda kurung yang mungkin. Jika ada persamaan yang menghasilkan 24, simpan hasilnya ke dalam linked list.
5. Lanjutkan hingga loop berakhir yang menandakan semua kasus telah teruji.

Program kemudian akan menampilkan semua kemungkinan solusi beserta juga runtime dari algoritma brute force tersebut.

```
void infixToPost(char *a){
    int i=0;
    Stack s;
    int j= 0;
    char temp;
    char buffer[MaxEl];
    CreateEmpty(&s);
    Push(&s,'\n');
    while(*(a+i) != '\n'){
        if(*(a+i)=='('){
            Push(&s, '(');
        }
        else if(*(a+i) == ')'){
            while(InfoTop(s) != '('){
                Pop(&s,&temp);
                buffer[j] = temp;j++;
            }
            Pop(&s, &temp);
        }
        else if(*(a+i)>=48 && *(a+i)<=51){
            buffer[j] = *(a+i);
            j++;
        }else{
            while(rank(*(a+i))<= rank(InfoTop(s))){Pop(&s, &temp);buffer[j] = temp;j++;}
            Push(&s, *(a+i));
        }
        i++;
    }
    while(!IsEmpty(s)){
        Pop(&s, &temp);
        buffer[j] = temp;j++;
    }
    for(i=0;i<j;i++){
        *(a+i) = buffer[i];
    }
}
```

Gambar 3.3 Algoritma perubah string persamaan infix menjadi suffix

Untuk membandingkan persamaan dengan 24, program memanfaatkan ADT stack. Pertama-tama, buat persamaan dalam bentuk sebuah string secara infix dengan bantuan fungsi sprint. Setelah dibuat persamaan dalam bentuk infix, persamaan akan diubah bentuknya menjadi bentuk suffix dengan bantuan ADT Stack yang dapat menyimpan char.

```
boolean is24(char *EQ, fraction *in){
    //use array indexing
    Stack s;
    StackF SF;
    fraction result;
    fraction frac[4] = {in[0], in[1], in[2], in[3]};
    int i = 0;
    fraction x1, x2;
    CreateEmpty(&s);
    CreateEmptyF(&SF);
    /*change infix*/
    infixToPost(EQ);
    /*solve suffix*/
    while(EQ[i]!='\n'){
        if(isOp(EQ[i])){
            PopF(&SF,&x2);
            PopF(&SF,&x1);
            result = calc(x1, x2, EQ[i]);
            PushF(&SF, result);
        }else{
            if(EQ[i]>='0' && EQ[i]<='3'){

                PushF(&SF,  frac[EQ[i]-48]);
            }

        }
        i++;
    }
    PopF(&SF, &result);
    simplify(&result);
/*    printf("%s",EQ);
    display(result);
    get_input3(&i);*/
    return (NUM(result) == 24) && (DENOM(result)==1);
}
```

Gambar 3.4 Algoritma check persamaan bernilai 24

Setelah diubah menjadi suffix, string persamaan dapat diselesaikan dengan bantuan Stack yang dapat menyimpan fraction dengan cara berikut,

- Lakukan traversal pada string hingga ditemukan ‘\n’.
- Jika angka, maka angka tersebut digunakan sebagai indeks untuk mengakses array of fraction untuk di push ke stack.
- Jika operator, makan akan di pop 2 fraction dan dilakukan operasi sesuai dengan operator tersebut. Hasilnya akan di push kembali ke dalam stack.

Setelah selesai, fraction yang ada di stack akan di sederhanakan dengan cara membagi penyebut dan pembilang masing-masing dengan FPB dari penyebut dan pembilang. Jika

penyebut bernilai 1 dan pembilang bernilai 24, maka persamaan tersebut bernilai sama dengan 24.

Adapun ADT-ADT yang membantu pembuatan program:

1. ADT fraction
   ADT fraction digunakan untuk mengabaikan potensi adanya masalah dalam kalkulasi, seperti pembagian 0, kesalahan pembulatan, dan lain-lain.

```c
#ifndef FRACTION
#define FRACTION
#include <stdio.h>

typedef struct {
    int num;
    int denom;
} fraction;

#define NUM(f) (f).num
#define DENOM(f) (f).denom
fraction transformItoF(int x){

    fraction y;
    NUM(y) = x;
    DENOM(y) = 1;
    return y;
}
int transformFtoI(fraction x){
    //I.S. Denominator x = 1, Numerator sembarang
    //F.S. terbentuk x dalam bentuk int
    return NUM(x);
}
int gcd(int x1, int x2){
    if(x2 == 0){return x1;
    }else{return gcd(x2, x1%x2);}
}

void simplify(fraction *f){
    int g = gcd(NUM(*f), DENOM(*f));
    if(g==0){
        NUM(*f) = -9999;
        DENOM(*f) = 1;
    }else{
        NUM(*f) = NUM(*f)/g;
        DENOM(*f) = DENOM(*f)/g;
    }
}
```

```c
void simplify(fraction *f){
    int g = gcd(NUM(*f), DENOM(*f));
    if(g==0){
        NUM(*f) = -9999;
        DENOM(*f) = 1;
    }else{
        NUM(*f) = NUM(*f)/g;
        DENOM(*f) = DENOM(*f)/g;
    }
}

fraction add(fraction f1, fraction f2){
    fraction result;
    NUM(result) = NUM(f1)*DENOM(f2) + NUM(f2)*DENOM(f1);
    DENOM(result) = DENOM(f1)*DENOM(f2);
    return result;
}

fraction subtract(fraction f1, fraction f2){
    fraction result;
    NUM(result) = NUM(f1)*DENOM(f2) - NUM(f2)*DENOM(f1);
    DENOM(result) = DENOM(f1)*DENOM(f2);
    return result;
}

fraction multiply(fraction f1, fraction f2){
    fraction result;
    NUM(result) = NUM(f1)*NUM(f2);
    DENOM(result) = DENOM(f1)*DENOM(f2);
    return result;
}
```

```
fraction divide(fraction f1, fraction f2){
    fraction result;
    NUM(result) = NUM(f1)*DENOM(f2);
    DENOM(result) = DENOM(f1)*NUM(f2);
    return result;
}

void display(fraction f1){
    printf("%d/%d\n",NUM(f1),DENOM(f1));
}


fraction calc(fraction x1, fraction x2, char op){
    switch(op){
        case '+':
            return add(x1, x2);
        case '-':
            return subtract(x1,x2);
        case '*':
            return multiply(x1, x2);
        case '/':
            return divide(x1,x2);
    }
}
#endif
```

Gambar 3.5 ADT fraction

2. ADT input
   ADT input adalah ADT mesin kata yang digunakan untuk menerima dan menvalidasi input. Untuk input kartu, input menerima input huruf seperti A atau q, ataupun angka seperti 11 atau 13.

```c
#include "boolean.h"

#define ENTER '\n'
#define BLANK ' '
char currentChar;
boolean endWord;
boolean signal;
#define NMax 2

typedef struct {
    char tab[NMax];
    int length;
} input_type;

typedef struct{
    char tab[13];
} equation;

#define ELMT(a,b) (a).tab[b]
input_type input;

void adv(){
    scanf("%c", &currentChar);
}
void start(){
    signal = true;
    adv();
}
void ignoreBlanks()
{
    while (currentChar == BLANK)
    {
        adv();
    }
}
```

```
void copyWord(){
    input.length = 0;
    while (currentChar != BLANK && currentChar != ENTER){
        if (input.length <  NMax){
            input.tab[input.length++] = currentChar;
            adv();
        }else{
            //input salah
            while(currentChar !=ENTER){
                adv();

            }
            signal = false;
        }
    }
}

void STARTWORD(){
    start();
    ignoreBlanks();
    if (currentChar == ENTER){
        endWord = true;
    }else{
        endWord = false;
        copyWord();
    }

}
```

```
void forceEnd(){
    while(currentChar!=ENTER){
        adv();
    }
}
void ADVWORD()
{

    ignoreBlanks();
    if (currentChar == ENTER)
    {
        endWord = true;
    }
    else
    {
        copyWord();
        ignoreBlanks();
    }
}

void uppercase(input_type *x){
    for(int i = 0; i< x->length; i++){
        if (x->tab[i]>=97 && x->tab[i]<=122){
            x->tab[i] -= 32;
        }
    }
}
```

```
int transformWtoInt(input_type x){
    uppercase(&x);
    switch (x.length)
    {
    case 1:
        if(x.tab[0]=='A'){
            return 1;
        }
        else if(x.tab[0]=='J'){
            return 11;
        }
        else if(x.tab[0]=='Q'){
            return 12;
        }
        else if(x.tab[0]=='K'){
            return 13;
        }
        else if(x.tab[0]>48 && x.tab[0]<58){
            return  x.tab[0]-48;
        }
        else{
            return -1;
        }
    case 2:
        if(x.tab[0]=='1' && x.tab[1]=='0'){
            return 10;
        }else if(x.tab[0]=='1' && x.tab[1]=='1'){
            return 11;
        }else if(x.tab[0]=='1' && x.tab[1]=='2'){
            return 12;
        }else if(x.tab[0]=='1' && x.tab[1]=='3'){
            return 13;
        }
        else{
            return -1;
        }
    default:
        return -1;
```

```
}

boolean get_input1(int *x){
    STARTWORD();
    if(!endWord && input.length==1 && input.tab[0]>=48 && input.tab[0]<=50){
        *x = input.tab[0] - 48;
        ADVWORD();
        if(endWord){
        return true;}
        else{forceEnd();return false;}
    }else{
        forceEnd();
        return false;
    }
}
```

```
boolean get_input2(int *x1, int *x2, int *x3, int *x4){
    STARTWORD();
    if(endWord){return false;}
    *x1 = transformWtoInt(input);
    ADVWORD();
    if(endWord){return false;}
    *x2 = transformWtoInt(input);

    ADVWORD();
    if(endWord){return false;}
    *x3 = transformWtoInt(input);
    ADVWORD();
    if(endWord){ return false;}
    *x4 = transformWtoInt(input);
    ADVWORD();
    if(endWord){
        if(*x1==-1 || *x2==-1||*x3==-1||*x4==-1){
            return false;
        }else{
            return true;
        }
    }else{
        forceEnd();
        return false;
    }
}
```

```
boolean get_input3(int *x){
    STARTWORD();
    if(!endWord && input.length==1 && input.tab[0]>=48 && input.tab[0]<=49){
        *x = input.tab[0] - 48;
        ADVWORD();
        if(endWord){
        return true;}
        else{forceEnd();return false;}
    }else{
        forceEnd();
        return false;
    }
}
```

Gambar 3.6 ADT input

3. ADT listlinier
   ADT listlinier digunakan untuk menyimpan hasil sementara, menghapus hasil-hasil yang duplikat serta mengoutput hasil pencarian solusi dari kartu24.

```c
#include "stdlib.h"


typedef struct {
    int x[4];
    char op[3];
    int bracket;
} ElType;
typedef struct node* Address;
typedef struct node {
    ElType info;
    Address next;
} Node;
typedef Address List;
#define INFO(p) (p)->info
#define NEXT(p) (p)->next
#define FIRST(l) (l)

// belum implementasi bawah
Address newNode(ElType val){
    Address p = (Address)malloc(sizeof(Node));
    if(p!= NULL){
        INFO(p) = val;
        NEXT(p) = NULL;
    }
    return p;
}

void CreateList(List *l){
/* I.S. sembarang             */
/* F.S. Terbentuk list kosong */
    FIRST(*l) = NULL;
}

boolean isEmpty(List l){return FIRST(l)==NULL;}
```

```c
void insertFirst(List *l, int x1, int x2, int x3, int x4, int b, char op1, char op2, char op3){
    /*KAMUS LOKAL*/
    Address p, k;
    ElType val;
    /*ALGORITMA*/
    val.x[0] = x1;
    val.x[1] = x2;
    val.x[2] = x3;
    val.x[3] = x4;
    val.op[0] = op1;
    val.op[1] = op2;
    val.op[2] = op3;
    val.bracket = b;
    p = newNode(val);
    if(p!=NULL){
        NEXT(p) = FIRST(*l);
        FIRST(*l) = p;
    }
}

void deleteFirst(List *l){
    /*KAMUS LOKAL*/
    Address p;
    /*ALGORITMA*/
    p=FIRST(*l);
    FIRST(*l)=NEXT(p);
    free(p);
}
```

```c
int len(List l){
    int i;
    Address p;
    i=0;
    p=FIRST(l);
    while(p!=NULL){
        i++;
        p = NEXT(p);
    }
    return i;
}
```

```c
void printall(List l){
    Address p;
    ElType exp;
    p = FIRST(l);
    if(len(l)==0){
        printf("Tidak ada solusi!\n");
    }else{
    printf("Banyak Solusi: %d\n", len(l));
    while(p!=NULL){
        exp = INFO(p);
        switch (exp.bracket)
        {
        //1. ((a op1 b) op2 c) op3 d
        //2. (a op1 (b op2 c)) op3 d
        //3. a op1 ((b op2 c) op3 d)
        //4. a op1 (b op2 (c op3 d))
        case 1:
            printf("((%d %c %d) %c %d) %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        case 2:
            printf("(%d %c (%d %c %d)) %c %d\n",exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        case 3:
            printf("%d %c ((%d %c %d) %c %d)\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        case 4:
            printf("%d %c (%d %c (%d %c %d))\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        /*twin case*/
        //5. (a op1 b) op2 (c op3 d)
        case 5:
            printf("(%d %c %d) %c (%d %c %d)\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        /*One bracket, 3 integers*/
        /*6.(a op1 b op2 c) op3 d*/
        case 6:
            printf("(%d %c %d %c %d) %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
```

```c
        /*7.a op1 (b op2 c op3 d)*/
        case 7:
            printf("%d %c (%d %c %d %c %d)\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        /*one bracket, 2 integers*/
        /*8.(a op1 b) op2 c op3 d*/
        case 8:
            printf("(%d %c %d) %c %d %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        /*9.a op1 (b op2 c) op3 d*/
        case 9:
            printf("%d %c (%d %c %d) %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        /*10.a op1 b op2 (c op3 d)*/
        case 10:
            printf("%d %c %d %c (%d %c %d)\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        /*No brackets*/
        /*11.(a op1 b) op2 c op3 d*/
        case 11:
            printf("%d %c %d %c %d %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
            break;
        default:
            printf("case error\n");
            break;
        }
        p = NEXT(p);
    }
    }
}
```

```
void clear_list(List *l){
    Address p;
    while(!isEmpty(*l)){
        deleteFirst(l);
    }
}

boolean sameEQ(ElType eq1, ElType eq2){
    return (eq1.x[0]==eq2.x[0]) && (eq1.x[1]==eq2.x[1]) &&
    (eq1.x[2]==eq2.x[2]) && (eq1.x[3]==eq2.x[3]) && (eq1.op[0]==eq2.op[0]) &&
    (eq1.op[1]==eq2.op[1]) && (eq1.op[2]==eq2.op[2]) && (eq1.bracket==eq2.bracket);
}

void delete_duplicate(List *l){
    Address p, q, r, dump;
    p = FIRST(*l);
    while(p!= NULL){
        q = NEXT(p);
        r = p;
        while(q != NULL){
            if(sameEQ(INFO(p), INFO(q))){
                NEXT(r) = NEXT(q);
                dump = q;
                q = NEXT(q);
                free(dump);

            }else{
            r = q;
            q = NEXT(q);
            }
        }
        p = NEXT(p);
    }
}
```

```c
void card(int d){

    switch(d){
        case 1:
            printf("A");
            break;
        case 11:
            printf("J");
            break;
        case 12:
            printf("Q");
            break;
        case 13:
            printf("K");
            break;
        default:
            printf("%d", d);

    }
}
    void cardfile(int d, FILE *ptr){
    switch(d){
        case 1:
            fprintf(ptr, "A");
            break;
        case 11:
            fprintf(ptr, "J");
            break;
        case 12:
            fprintf(ptr, "Q");
            break;
        case 13:
            fprintf(ptr, "K");
            break;
        default:
            fprintf(ptr, "%d", d);

    }
}
```

```c
void writeFile(List l, int x1, int x2, int x3, int x4, double time){
    FILE *ptr;
    Address p;
    ElType exp;
    p = FIRST(l);
    ptr = fopen("../test/output.txt","w");
    fprintf(ptr, "24 Game\n");
    fprintf(ptr, "Kartu: ");
    cardfile(x1, ptr); fprintf(ptr, " ");
    cardfile(x2, ptr); fprintf(ptr, " ");
    cardfile(x3, ptr); fprintf(ptr, " ");
    cardfile(x4, ptr); fprintf(ptr, "\n");
    fprintf(ptr, "Execution time %.2f\n", time);
    if(len(l)==0){
        fprintf(ptr, "\nTidak ada solusi!");
    }else{
        fprintf(ptr, "\nBanyak Solusi: %d\n", len(l));
        while(p!=NULL){
            exp = INFO(p);
            switch (exp.bracket){
            //1. ((a op1 b) op2 c) op3 d
            //2. (a op1 (b op2 c)) op3 d
            //3. a op1 ((b op2 c) op3 d)
            //4. a op1 (b op2 (c op3 d))
            case 1:
                fprintf(ptr,"((%d %c %d) %c %d) %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            case 2:
                fprintf(ptr,"(%d %c (%d %c %d)) %c %d\n",exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            case 3:
                fprintf(ptr,"%d %c ((%d %c %d) %c %d)\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            case 4:
                fprintf(ptr,"%d %c (%d %c (%d %c %d))\n",exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
```

```c
            /*twin case*/
            //5. (a op1 b) op2 (c op3 d)
            case 5:
                fprintf(ptr,"(%d %c %d) %c (%d %c %d)\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            /*One bracket, 3 integers*/
            /*6.(a op1 b op2 c) op3 d*/
            case 6:
                fprintf(ptr,"(%d %c %d %c %d) %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            /*7.a op1 (b op2 c op3 d)*/
            case 7:
                fprintf(ptr,"%d %c (%d %c %d %c %d)\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            /*one bracket, 2 integers*/
            /*8.(a op1 b) op2 c op3 d*/
            case 8:
                fprintf(ptr,"(%d %c %d) %c %d %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            /*9.a op1 (b op2 c) op3 d*/
            case 9:
                fprintf(ptr,"%d %c (%d %c %d) %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            /*10.a op1 b op2 (c op3 d)*/
            case 10:
                fprintf(ptr,"%d %c %d %c (%d %c %d)\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            /*No brackets*/
            /*11.(a op1 b) op2 c op3 d*/
            case 11:
                fprintf(ptr,"%d %c %d %c %d %c %d\n", exp.x[0], exp.op[0], exp.x[1], exp.op[1], exp.x[2], exp.op[2], exp.x[3]);
                break;
            default:
                printf("case error\n");
                break;
            }
            p = NEXT(p);
        }
    }
```

Gambar 3.7 Source code ADT listlinier

4. ADT stack

Terdapat dua ADT stack pada program. Yang pertama adalah stack of char yang berfungsi untuk membantu proses perubahan persamaan infix menjadi postfix. Kemudian ada juga stack of fraction yang berfungsi untuk menyelesaikan persamaan dengan format postfix.

```c
#include "boolean.h"
#include "fraction.c"
#define Nil -1
#define MaxEl 15
/* Nil adalah stack dengan elemen kosong . */
typedef char infotype;
typedef int address;
typedef struct {
  infotype T[MaxEl]; /* tabel penyimpan elemen */
  address TOP;  /* alamat TOP: elemen puncak */
} Stack;

/* Definisi akses dengan Selektor : Set dan Get */
#define Top(S) (S).TOP
#define InfoTop(S) (S).T[(S).TOP]

/* ************ Prototype ************ */
/* *** Konstruktor/Kreator *** */
void CreateEmpty(Stack *S){
Top(*S) = Nil;
}
/* ************ Predikat Untuk test keadaan KOLEKSI ************ */
boolean IsEmpty(Stack S){
return Top(S) == Nil;
}
/* ************ Menambahkan sebuah elemen ke Stack ************ */
void Push(Stack * S, infotype X){
/* Menambahkan X sebagai elemen Stack S. */
/* I.S. S mungkin kosong, tabel penampung elemen stack TIDAK penuh */
/* F.S. X menjadi TOP yang baru,TOP bertambah 1 */
if(IsEmpty(*S)){
    Top(*S) = 0;
}
else{
    Top(*S) +=1;
}
InfoTop(*S) = X;
}
```

```c
/* ************ Menghapus sebuah elemen Stack ************ */
void Pop(Stack * S, infotype* X){
/* Menghapus X dari Stack S. */
/* I.S. S  tidak mungkin kosong */
/* F.S. X adalah nilai elemen TOP yang lama, TOP berkurang 1 */
*X = InfoTop(*S);
if(Top(*S) == 0){
    Top(*S) = Nil;
}
else{
    Top(*S) -=1;
}
}

int rank(char c){
    switch (c){
        case '(':
            return 0;
        case '+':
            return 1;
        case '-':
            return 1;
        case '*':
            return 2;
        case '/':
            return 2;
        default:
            return 0;
    }
}
```

Gambar 3.8 Source code ADT stack untuk char

```c
typedef struct {
  fraction T[MaxEl]; /* tabel penyimpan elemen */
  address TOP;  /* alamat TOP: elemen puncak */
} StackF;
/* ************ Prototype ************ */
/* *** Konstruktor/Kreator *** */
void CreateEmptyF(StackF *S){
Top(*S) = Nil;
}
/* ************ Predikat Untuk test keadaan KOLEKSI ************ */
boolean IsEmptyF(StackF S){
return Top(S) == Nil;
}
/* ************ Menambahkan sebuah elemen ke Stack ************ */
void PushF(StackF * S, fraction X){
/* Menambahkan X sebagai elemen Stack S. */
/* I.S. S mungkin kosong, tabel penampung elemen stack TIDAK penuh */
/* F.S. X menjadi TOP yang baru,TOP bertambah 1 */
if(IsEmptyF(*S)){
    Top(*S) = 0;
}
else{
    Top(*S) +=1;
}
    NUM(InfoTop(*S)) = NUM(X);
    DENOM(InfoTop(*S)) = DENOM(X);
}
```

```
/* ************ Menambahkan sebuah elemen ke Stack ************ */
void PushF(StackF * S, fraction X){
/* Menambahkan X sebagai elemen Stack S. */
/* I.S. S mungkin kosong, tabel penampung elemen stack TIDAK penuh */
/* F.S. X menjadi TOP yang baru,TOP bertambah 1 */
if(IsEmptyF(*S)){
    Top(*S) = 0;
}
else{
    Top(*S) +=1;
}
    NUM(InfoTop(*S)) = NUM(X);
    DENOM(InfoTop(*S)) = DENOM(X);
}
/* ************ Menghapus sebuah elemen Stack ************ */
void PopF(StackF * S, fraction* X){
/* Menghapus X dari Stack S. */
/* I.S. S  tidak mungkin kosong */
/* F.S. X adalah nilai elemen TOP yang lama, TOP berkurang 1 */
NUM(*X)= NUM(InfoTop(*S));
DENOM(*X) = DENOM(InfoTop(*S));
if(Top(*S) == 0){
    Top(*S) = Nil;
}
else{
    Top(*S) -=1;
}
}
```

Gambar 3.9 Source code ADT stack untuk fraction

5. ADT Boolean

```
/* Definisi type boolean */

#ifndef _BOOLEAN_h
#define _BOOLEAN_h

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

#endif
```

Gambar 3.10 Source code ADT stack untuk boolean

BAB IV

EKSPERIMEN

4.1 Kasus 1

```
24 SOLVER


Ketik:
(1) Masukan Keyboard
(2) Random Generate
(0) Exit
2
Kartu yang didapat:
10 10 Q 3
```

Gambar 4.1.1 Input Kasus 1

```
24 Game
Kartu: 10 10 Q 3
Execution time 0.01

Banyak Solusi: 34
3 * (10 - (12 - 10))
3 * (10 - 12 + 10)
3 * ((10 - 12) + 10)
3 * (10 + 10 - 12)
3 * (10 + (10 - 12))
3 * ((10 + 10) - 12)
(3 - 10 / 10) * 12
(3 - (10 / 10)) * 12
12 / 3 + 10 + 10
12 / 3 + (10 + 10)
(12 / 3) + 10 + 10
(12 / 3 + 10) + 10
(12 / 3) + (10 + 10)
((12 / 3) + 10) + 10
12 * (3 - 10 / 10)
12 * (3 - (10 / 10))
10 + 12 / 3 + 10
10 + (12 / 3) + 10
10 + (12 / 3 + 10)
(10 + 12 / 3) + 10
10 + ((12 / 3) + 10)
(10 + (12 / 3)) + 10
(10 - (12 - 10)) * 3
(10 - 12 + 10) * 3
((10 - 12) + 10) * 3
(10 + 10 - 12) * 3
(10 + (10 - 12)) * 3
((10 + 10) - 12) * 3
10 + 10 + 12 / 3
10 + 10 + (12 / 3)
(10 + 10) + 12 / 3
10 + (10 + 12 / 3)
(10 + 10) + (12 / 3)
10 + (10 + (12 / 3))
```

Gambar 4.1.2 Output Kasus 1

4.2 Kasus 2

```
24 SOLVER


Ketik:
(1) Masukan Keyboard
(2) Random Generate
(0) Exit
1
Masukan nilai kartu: K Q J A
```

Gambar 4.2.1 Input Kasus 2

```
24 Game
Kartu: K Q J A
Execution time 0.01

Banyak Solusi: 54
1 * 12 * (13 - 11)
(1 * 12) * (13 - 11)
1 * (12 * (13 - 11))
1 * (13 - 11) * 12
(1 * 13 - 11) * 12
1 * ((13 - 11) * 12)
(1 * (13 - 11)) * 12
((1 * 13) - 11) * 12
12 / (1 / (13 - 11))
12 / 1 * (13 - 11)
(12 / 1) * (13 - 11)
12 * 1 * (13 - 11)
12 * (1 * 13 - 11)
(12 * 1) * (13 - 11)
12 * (1 * (13 - 11))
12 * ((1 * 13) - 11)
12 * (13 / 1 - 11)
12 * ((13 / 1) - 11)
12 * (13 * 1 - 11)
12 * ((13 * 1) - 11)
12 * (13 - 1 * 11)
12 * (13 - (1 * 11))
12 * (13 - 11) / 1
12 * (13 - 11 / 1)
12 * (13 - (11 / 1))
12 * ((13 - 11) / 1)
(12 * (13 - 11)) / 1
12 * (13 - 11) * 1
12 * (13 - 11 * 1)
12 * (13 - (11 * 1))
12 * ((13 - 11) * 1)
(12 * (13 - 11)) * 1
(13 / 1 - 11) * 12
```

```
39   ((13 / 1) - 11) * 12
40   (13 * 1 - 11) * 12
41   ((13 * 1) - 11) * 12
42   (13 - 1 * 11) * 12
43   (13 - (1 * 11)) * 12
44   (13 - 11) / (1 / 12)
45   (13 - 11) / 1 * 12
46   (13 - 11 / 1) * 12
47   (13 - (11 / 1)) * 12
48   ((13 - 11) / 1) * 12
49   (13 - 11) * 1 * 12
50   (13 - 11 * 1) * 12
51   (13 - 11) * (1 * 12)
52   (13 - (11 * 1)) * 12
53   ((13 - 11) * 1) * 12
54   (13 - 11) * 12 / 1
55   (13 - 11) * (12 / 1)
56   ((13 - 11) * 12) / 1
57   (13 - 11) * 12 * 1
58   (13 - 11) * (12 * 1)
59   ((13 - 11) * 12) * 1
```

Gambar 4.2.2 Output Kasus 2

4.3 Kasus 3

```
24 SOLVER


Ketik:
(1) Masukan Keyboard
(2) Random Generate
(0) Exit
2
Kartu yang didapat:
K 10 6 A
```

Gambar 4.3.1 Input Kasus 3

```
24 Game
Kartu: K 10 6 A
Execution time 0.01

Banyak Solusi: 24
(1 - (10 - 13)) * 6
(1 - 10 + 13) * 6
((1 - 10) + 13) * 6
(1 + 13 - 10) * 6
(1 + (13 - 10)) * 6
((1 + 13) - 10) * 6
6 * (1 - (10 - 13))
6 * (1 - 10 + 13)
6 * ((1 - 10) + 13)
6 * (1 + 13 - 10)
6 * (1 + (13 - 10))
6 * ((1 + 13) - 10)
6 * (13 + 1 - 10)
6 * (13 + (1 - 10))
6 * ((13 + 1) - 10)
6 * (13 - (10 - 1))
6 * (13 - 10 + 1)
6 * ((13 - 10) + 1)
(13 + 1 - 10) * 6
(13 + (1 - 10)) * 6
((13 + 1) - 10) * 6
(13 - (10 - 1)) * 6
(13 - 10 + 1) * 6
((13 - 10) + 1) * 6
```

Gambar 4.3.2 Output Kasus 3

4.4 Kasus 4

```
Masukan nilai kartu: 1 1 1 1
```

Gambar 4.4.1 Input Kasus 4

```
24 Game
Kartu: A A A A
Execution time 0.01

Tidak ada solusi!
```

Gambar 4.4.2 Output Kasus 4

4.5 Kasus 5

```
24 SOLVER


Ketik:
(1) Masukan Keyboard
(2) Random Generate
(0) Exit
2
```

Gambar 4.5.1 Input Kasus 5

```
24 Game
Kartu: 5 2 J J
Execution time 0.01

Tidak ada solusi!
```

Gambar 4.5.2 Output Kasus 5

4.6 Kasus 6

```
24 SOLVER


Ketik:
(1) Masukan Keyboard
(2) Random Generate
(0) Exit
1
Masukan nilai kartu: 6 6 6 6
```

Gambar 4.6.1 Input Kasus 6

```
24 Game
Kartu: 6 6 6 6
Execution time 0.01

Banyak Solusi: 17
6 * 6 - 6 - 6
(6 * 6) - 6 - 6
(6 * 6 - 6) - 6
((6 * 6) - 6) - 6
6 * 6 - (6 + 6)
(6 * 6) - (6 + 6)
6 + 6 + 6 + 6
6 + 6 + (6 + 6)
6 + (6 + 6) + 6
(6 + 6) + 6 + 6
6 + (6 + 6 + 6)
(6 + 6 + 6) + 6
(6 + 6) + (6 + 6)
6 + (6 + (6 + 6))
6 + ((6 + 6) + 6)
(6 + (6 + 6)) + 6
((6 + 6) + 6) + 6
```

Gambar 4.6.2 Output Kasus 6

4.7 Kasus 7

```
24 SOLVER


Ketik:
(1) Masukan Keyboard
(2) Random Generate
(0) Exit
2
Kartu yang didapat:
10 6 A 4
```

Gambar 4.7.1 Input Kasus 7

```
24 Game
Kartu: 10 6 A 4
Execution time 0.01

Banyak Solusi: 4
(4 - 1) * 10 - 6
((4 - 1) * 10) - 6
10 * (4 - 1) - 6
(10 * (4 - 1)) - 6
```

Gambar 4.7.2 Output Kasus 7

4.8 Kasus 8

```
24 SOLVER


Ketik:
(1) Masukan Keyboard
(2) Random Generate
(0) Exit
2
Kartu yang didapat:
6 5 7 9
```

Gambar 4.8.1 Input Kasus 8

```
24 Game
Kartu: 6 5 7 9
Execution time 0.01

Banyak Solusi: 12
9 * (7 - 5) + 6
(9 * (7 - 5)) + 6
(7 - 5) * 9 + 6
((7 - 5) * 9) + 6
6 + 9 * (7 - 5)
6 + (9 * (7 - 5))
6 - 9 * (5 - 7)
6 - (9 * (5 - 7))
6 + (7 - 5) * 9
6 + ((7 - 5) * 9)
6 - (5 - 7) * 9
6 - ((5 - 7) * 9)
```

Gambar 4.8.2 Input Kasus 8

Poin	Ya		Tidak
1. Program berhasil dikompilasi tanpa kesalahan	V		
2. Program berhasil running	V		
3. Program dapat membaca input / generate sendiri dan memberikan luaran	V		
4. Solusi yang diberkan program memenuhi (berhasil mencapai 24)	V		
5. Program dapat menyimpan solusi dalam file teks	V		

LAMPIRAN

Link Repository Github: https://github.com/Onyxcodeotto/TUCIL1_13521053

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tucil1-Stima-2023.pdf, diakses pada tanggal 24/01/2023

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf , diakses pada tanggal 24/01/2023