

**Laporan Tugas Kecil IF2211 Strategi Algoritma  
Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force**

Oleh  
Michael Jonathan Halim - 13521124



**Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
INSTITUT TEKNOLOGI BANDUNG  
2022**

# DAFTAR ISI

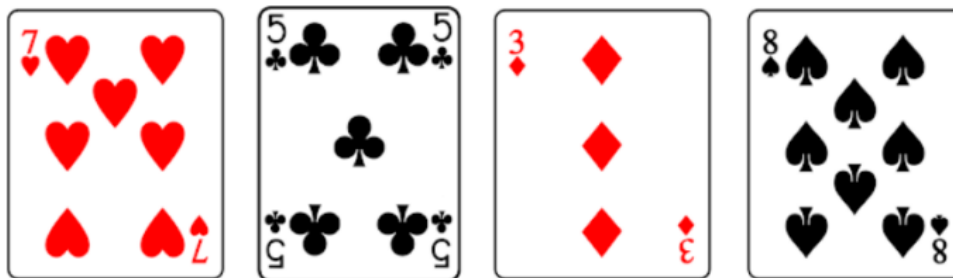
<b>DAFTAR ISI</b>	<b>1</b>
<b>BAB I</b>	<b>2</b>
1.1 Permainan Kartu 24	2
1.2 Spesifikasi Tugas Kecil	3
<b>BAB II</b>	<b>5</b>
2.1 Ide Dasar Algoritma	5
<b>BAB III</b>	<b>7</b>
3.1 Source Code Program	7
<b>BAB IV</b>	<b>18</b>
4.1 User Interface	18
4.2 Contoh Input	19
4.3 Contoh Output	20
4.4 Test Case	21
<b>BAB V</b>	<b>28</b>
5.1. Link Repository Github	28
5.2 Checklist Table Progress	28

# BAB I

## SPESIFIKASI TUGAS KECIL

### 1.1 Permainan Kartu 24

Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. Permainan ini menarik cukup banyak peminat dikarenakan dapat meningkatkan kemampuan berhitung serta mengasah otak agar dapat berpikir dengan cepat dan akurat. Permainan Kartu 24 biasa dimainkan dengan menggunakan kartu remi. Kartu remi terdiri dari 52 kartu yang terbagi menjadi empat suit (sekop, hati, keriting, dan wajik) yang masing-masing terdiri dari 13 kartu (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). Yang perlu diperhatikan hanyalah nilai kartu yang didapat (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). As bernilai 1, Jack bernilai 11, Queen bernilai 12, King bernilai 13, sedangkan kartu bilangan memiliki nilai dari bilangan itu sendiri. Pada awal permainan moderator atau salah satu pemain mengambil 4 kartu dari dek yang sudah dikocok secara random. Permainan berakhir ketika pemain berhasil menemukan solusi untuk membuat kumpulan nilainya menjadi 24. Pengubahan nilai tersebut dapat dilakukan menggunakan operasi dasar matematika penjumlahan (+), pengurangan (-), perkalian ( $\times$ ), divisi ( $/$ ) dan tanda kurung ( ). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas. (Paragraf di atas dikutip dari sini: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah2016/MakalahStima-2016-038.pdf>).



MAKE IT 24

Gambar 1. Permainan Kartu 24

## 1.2 Spesifikasi Tugas Kecil

- Tulislah program sederhana dalam Bahasa C/C++/Java yang mengimplementasikan algoritma Brute Force untuk mencari seluruh solusi permainan kartu 24.
- Input: 4 angka/huruf yang terdiri dari: (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K). Contoh input: A 8 9 Q Selain itu, input juga dapat dilakukan dengan program men-generate 4 angka/hurufnya sendiri secara random. Pengguna dapat memilih apakah program meminta input dari pengguna atau generate sendiri. Apabila masukan tidak sesuai, maka program menampilkan luaran “Masukan tidak sesuai” dan akan meminta ulang.
- Output: 1. Banyaknya solusi yang ditemukan. 2. Solusi dari permainan kartu 24 ditampilkan di layar dan terdapat opsi untuk menyimpan solusi dalam file text. Untuk contoh kasus di atas A 8 9 Q, maka salah satu solusinya adalah:  $((9 + A) - 8) * Q$  atau  $((9 + 1) - 8) * 12$  (Kedua jenis output dibebaskan) Note: Format penulisan output yang dicetak tidak harus persis contoh, yang penting merepresentasikan solusinya sudah cukup. Output apabila tidak ada solusi untuk pasangan kombinasi input, cukup ditampilkan “Tidak ada solusi”. Untuk solusi setiap masukan, perlu dipertimbangkan urutan nilai (x1..x4), urutan operator, dan grouping dengan kurung yang mungkin.

A 8 9 Q
38 solutions found ((1 - 8) + 9) * 12 (1 - (8 - 9)) * 12 (1 * 8) * (12 - 9) .. .. .. (dst.)

- Di akhir, program akan menanyakan “Apakah ingin menyimpan solusi?”. Jika iya, program akan meminta sebuah nama untuk file teks dan semua solusi yang didapat akan disimpan dalam file text tersebut. 3. Waktu eksekusi program (tidak termasuk waktu pembacaan file input). Luaran no 1 cukup ditampilkan pertama sebelum mencetak solusi, sementara luaran no 3 cukup ditampilkan di akhir program (saat program selesai)
- Berkas laporan yang dikumpulkan adalah laporan dalam bentuk PDF yang setidaknya berisi:

1. Algoritma brute force (deskripsi langkah-langkahnya, bukan notasi pseudocode)
  2. Source program dalam bahasa pemrograman yang dipilih
  3. Screenshot yang memperlihatkan input dan output (minimal 6 contoh)
  4. Link to Repository yang berisi kode program
- Contoh solusi untuk kombinasi input dapat diperiksa melalui halaman berikut <http://24solver.us-west-2.elasticbeanstalk.com/>
  - Program dibebaskan memakai antarmuka GUI atau hanya antarmuka teks (tampilan teks).
  - Program disimpan dalam repository yang bernama Tucil1\_NIM. Berikut merupakan struktur dari isi folder tersebut :
    1. Folder src berisi source code
    2. Folder bin berisi executable
    3. Folder test berisi solusi jawaban dari data uji yang digunakan dalam laporan
    4. Folder doc berisi laporan tugas kecil
    5. README untuk tata cara penggunaan yang minimal berisi:
      - a. Deskripsi singkat program yang dibuat
      - b. Requirement program dan instalasi tertentu bila ada
      - c. Cara menggunakan program
      - d. Author / identitas pembuat

## **BAB II**

### **ALGORITMA BRUTE FORCE**

#### **2.1 Ide Dasar Algoritma**

Ide dasar dari algoritma yang diterapkan pada program ini adalah menggunakan permutasi untuk mencari seluruh kemungkinan yang ada. Kita ketahui bahwa bentuk dari solusi terdiri dari 4 angka yang di antara setiap angka terdapat operator seperti tambah (+), kurang (-), kali (x), dan bagi (/), juga diapit oleh tanda kurung yang menandakan urutan operasi yang dilakukan. Contoh, untuk kartu 2, 7, 4, dan 5 terdapat solusi yaitu  $((7 + 5) / 2) * 4$  yang artinya adalah kartu 7 ditambah terlebih dahulu dengan kartu 5 menghasilkan angka 12, lalu dibagi dengan 2 menghasilkan angka 6 dan dilanjutkan perkalian dengan angka 4 sehingga hasil akhir dari operasi keseluruhan adalah 24. Dari bentuk solusi tersebut, kita bisa ketahui dengan jelas urutan operasi kartu karena tanda kurung yang mengapit operasi-operasi angka tersebut. Maka dari itu, Terdapat 5 jenis bentuk solusi yang dapat dihasilkan dari algoritma ini. Misalkan terdapat kartu A, B, C, dan D, maka bentuk-bentuk solusi yang dapat dihasilkan adalah  $((A + B) * C) / D$ ,  $(A+B) * (C / D)$ ,  $(A + (B * C)) / D$ ,  $A + ((B * C) / D)$ , dan  $A + (B * (C / D))$ .

#### **2.2 Algoritma Brute Force 24 Game**

Untuk algoritma utama dari 24 Game ini dengan teknik brute force yang diterapkan adalah sebagai berikut :

1. Lakukan iterasi dari 4 kartu pemain untuk posisi kartu pertama.
2. Di dalam iterasi pertama, lakukan iterasi dari 4 kartu pemain untuk posisi kartu kedua. Lakukan hal ini juga untuk posisi kartu ketiga dan keempat.
3. Di dalam iterasi kartu-kartu, lakukan juga iterasi untuk 3 operator yang terdapat dalam bentuk solusi seperti langkah-langkah sebelumnya sehingga terdapat total loop dalam loop sebanyak 7 layer.
4. Selama looping permutasi dilakukan, kita cari kombinasi-kombinasi solusi yang menghasilkan angka 24. Jangan lupa untuk memeriksa terlebih dahulu apakah pada kombinasi tersebut kartunya berbeda semua atau tidak. Berbeda yang dimaksud dari konteks ini adalah urutan kartu yang dihasilkan dari iterasi adalah 4 kartu yang sudah diinput. Maka dari itu, lakukan pemeriksaan pada permutasi yang dilakukan apakah

kartunya sudah berbeda semua atau tidak. Lalu, kita ketahui bahwa terdapat 5 kemungkinan kombinasi, maka untuk semua hasil permutasi kita uji untuk kelima bentuk solusi tersebut.

5. Untuk pengujian, perlu divalidasi apakah operasi tersebut merupakan operasi yang valid atau tidak secara matematika karena bisa saja terdapat division by zero. Oleh karena itu, periksa terlebih dahulu untuk setiap perhitungan apakah terdapat division by zero atau tidak. Jika iya, maka kita tidak perlu kalkulasi untuk menghindari error.
6. Apabila pengujian menghasilkan angka 24, simpanlah hasil bentuk solusi tersebut dalam tipe data string dan tambahkan juga jumlah solusi untuk sekarang.
7. Untuk pengujian berikutnya, sebelum hasil solusi disimpan dan jumlah solusi ditambah, periksa terlebih dahulu apakah kita sudah memiliki solusi yang sama untuk menghindari solusi duplikat. Jika terdapat solusi yang sama, lewati pengujian tersebut dan lanjut ke pengujian berikutnya.
8. Lakukan langkah 5-7 hingga bentuk solusi kelima.
9. Lakukan langkah 4-7 hingga semua kemungkinan permutasi diuji.

## BAB III

### SOURCE CODE PROGRAM

#### 3.1 Source Code Program

```
#include <iostream>
#include <string.h>
#include <vector>
#include <fstream>
#include <ctime>

using namespace std;

int converterInput(string x){
    // Fungsi untuk mengkonversi input kartu dari user
    if(x == "A"){
        return 1;
    } else if(x.length() == 1 && x[0] - '0' > 1 && x[0] - '0' < 11){
        return stoi(x);
    } else if(x == "10"){
        return 10;
    } else if(x == "J"){
        return 11;
    } else if(x == "Q"){
        return 12;
    } else if(x == "K"){
        return 13;
    } else {
        return -1;
    }
}

double calTwoNumber(double a, double b, char x) {
    // Fungsi untuk melakukan operasi pada dua kartu
    if(x == '+'){
        return a + b;
    } else if (x == '-'){
        return a - b;
    } else if (x == '*'){
        return a * b;
    } else if (x == '/'){
        double aTemp = a;
        double bTemp = b;
        return aTemp / bTemp;
    } else {
        return 0;
    }
}
```



```

}

bool validCalculation(double a, double b, char x){
    // Fungsi untuk mengvalidasi operasi pada dua kartu
    if(x == '/'){
        if(b == 0){
            return false;
        }
    }

    return true;
}

bool find(vector<string> list, string s){
    // Fungsi untuk menentukan apakah string s ada pada list atau tidak
    bool exist = false;
    for(int i = 0; i < list.size(); i++){
        if(list[i] == s){
            exist = true;
            break;
        }
    }
    return exist;
}

bool validInputCard(string s){
    // Fungsi untuk memeriksa apakah input kartu valid atau tidak
    int count = 0;
    for(int i = 0; i < s.length(); i++){
        if(s[i] == ' '){
            count++;
        }
    }
    if(count == 3){
        return true;
    } else {
        return false;
    }
}

void processCard(string s, string *a, string *b, string *c, string *d){
    // Fungsi untuk memroses kartu yang diinput
    string temp = "";
    int i = 0;
    int count = 0;
    string collection[4];
    while(i != s.length()){
        if(i == s.length() - 1){
            temp += s[i];

```





```

    } else if (action == "2") {
        // DECLARATION ARRAY OF CARDS
        string cards[13] = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"};

        // RANDOMIZE
        int index1 = rand() % 13;
        int index2 = rand() % 13;
        int index3 = rand() % 13;
        int index4 = rand() % 13;

        // CONVERSION TO INTEGER
        firstNumber = converterInput(cards[index1]);
        secondNumber = converterInput(cards[index2]);
        thirdNumber = converterInput(cards[index3]);
        fourthNumber = converterInput(cards[index4]);

        cout << "\nYour cards : " << cards[index1] << " " << cards[index2] << " " << cards[index3] << " " <<
cards[index4] << endl;
        valid = true;
    } else if (action == "3") {
        printf("\nThank you for playing!\n");
        return 0;
    } else {
        cout << "\nWRONG INPUT! Please input correctly!\n" << endl;
    }
}

// MAIN ALGORITHM
// DECLARATION VARIABLES FOR MAIN ALGORITHM
int num[4] = {firstNumber, secondNumber, thirdNumber, fourthNumber};
char operators[4] = {'+', '-', '*', '/'};
vector<string> ans;
int count = 0;

// Initiation time
double timeConsumed = clock();

// Looping Permutation
for(int firstIdx = 0; firstIdx < 4; firstIdx++){
    for(int secondIdx = 0; secondIdx < 4; secondIdx++){
        for(int thirdIdx = 0; thirdIdx < 4; thirdIdx++){
            for(int i = 0 ; i < 4; i++){
                for(int j = 0; j < 4; j++){
                    for(int k = 0; k < 4; k++){
                        for(int l = 0; l < 4; l++){
                            if(i != j && i != k && i != l && j != k && j != l && k != l){
                                string answer;
                                // FIRST COMBINATION
                                // Validation for each calculation of cards

```

```

if(validCalculation(num[i],
                    num[j],
                    operators[firstIdx]
                    ) &&
    validCalculation(calTwoNumber(num[i],
                                   num[j],
                                   operators[firstIdx]
                                   ),
                    num[k],
                    operators[secondIdx]
                    ) &&
    validCalculation(calTwoNumber(calTwoNumber(num[i],
                                                num[j],
                                                operators[firstIdx]
                                                ),
                                   num[k],
                                   operators[secondIdx]
                                   ),
                    num[l],
                    operators[thirdIdx]
                    )
){
    // If valid, then calculate the result
    double cal = calTwoNumber(
        calTwoNumber(
            calTwoNumber(num[i], num[j], operators[firstIdx]), num[k],
            operators[secondIdx]
        ),
        num[l], operators[thirdIdx]
    );

    // If the result equals to 24, save the answer
    if(cal == 24){
        answer = "(" + to_string(num[i]) + " " +
            operators[firstIdx] + " " + to_string(num[j]) + " " +
            operators[secondIdx] + " " + to_string(num[k]) + " " +
            operators[thirdIdx] + " " + to_string(num[l]);
        // Check if solution has already saved or not
        if(!find(ans, answer)){
            count++;
            ans.push_back(answer);
        }
    }
}

// Do the same steps in first combination for the rest combination

// SECOND COMBINATION
if(validCalculation(num[i],

```

```

        num[j],
        operators[firstIdx]
    ) &&
    validCalculation(num[k],
        num[l],
        operators[thirdIdx]
    ) &&
    validCalculation(calTwoNumber(num[i],
        num[j],
        operators[firstIdx]
    ),
        calTwoNumber(num[k],
            num[l],
            operators[thirdIdx]
        ),
        operators[secondIdx])
) {
    double hasil1 = calTwoNumber(num[i], num[j], operators[firstIdx]);
    double hasil2 = calTwoNumber(num[k], num[l], operators[thirdIdx]);
    double cal = calTwoNumber(
        hasil1,
        hasil2,
        operators[secondIdx]
    );
    if(cal == 24){
        answer = "(" + to_string(num[i]) + " " +
            operators[firstIdx] + " " + to_string(num[j]) + ")" +
            operators[secondIdx] + " (" + to_string(num[k]) + " " +
            operators[thirdIdx] + " " + to_string(num[l]) + ")";
        if(!find(ans, answer)){
            count++;
            ans.push_back(answer);
        }
    }
}
}

```

// THIRD COMBINATION

```

if(validCalculation(num[j],
    num[k],
    operators[secondIdx]
) &&
    validCalculation(num[i],
        calTwoNumber(num[j],
            num[k],
            operators[secondIdx]
        ),
        operators[firstIdx]
    ) &&
    validCalculation(calTwoNumber(num[i],

```

```

        calTwoNumber(num[j],
                    num[k],
                    operators[secondIdx]
                ),
        operators[firstIdx]
    ),
    num[l],
    operators[thirdIdx]
)
) {
    double cal = calTwoNumber(
        calTwoNumber(num[i],
                    calTwoNumber(num[j], num[k], operators[secondIdx]),
                    operators[firstIdx]),
        num[l],
        operators[thirdIdx]
    );
    if(cal == 24){
        answer = "(" + to_string(num[i]) + " " +
            operators[firstIdx] + " (" + to_string(num[j]) + " " +
            operators[secondIdx] + " " + to_string(num[k]) + ") " +
            operators[thirdIdx] + " " + to_string(num[l]);
        if(!find(ans, answer)){
            count++;
            ans.push_back(answer);
        }
    }
}

// FOURTH COMBINATION
if(validCalculation(num[j],
    num[k],
    operators[secondIdx]
) &&
validCalculation(calTwoNumber(num[j],
    num[k],
    operators[secondIdx]
),
    num[l],
    operators[thirdIdx]
) &&
validCalculation(num[i],
    calTwoNumber(calTwoNumber(num[j],
        num[k],
        operators[secondIdx]
    ),
        num[l],
        operators[thirdIdx]
    ),

```

```

        operators[firstIdx]
    )
) {
    double cal = calTwoNumber(num[i],
        calTwoNumber(
            calTwoNumber(num[j], num[k], operators[secondIdx]),
            num[l],
            operators[thirdIdx]
        ),
        operators[firstIdx]
    );
    if(cal == 24){
        answer = to_string(num[i]) + " " +
            operators[firstIdx] + " (" + to_string(num[j]) + " " +
            operators[secondIdx] + " " + to_string(num[k]) + ") " +
            operators[thirdIdx] + " " + to_string(num[l]) + ")";
        if(!find(ans, answer)){
            count++;
            ans.push_back(answer);
        }
    }
}

// FIFTH COMBINATION
if(validCalculation(num[k],
    num[l],
    operators[thirdIdx]
) &&
validCalculation(num[j],
    calTwoNumber(num[k],
        num[l],
        operators[thirdIdx]
    ),
    operators[thirdIdx]
) &&
validCalculation(num[i],
    calTwoNumber(num[j],
        calTwoNumber(num[k],
            num[l],
            operators[thirdIdx]
        ),
        operators[secondIdx]
    ),
    operators[firstIdx]
)
) {
    double cal = calTwoNumber(num[i],
        calTwoNumber(num[j],
            calTwoNumber(num[k], num[l], operators[thirdIdx]),

```



```

        operators[secondIdx]
    ),
    operators[firstIdx]
);
if(cal == 24){
    answer = to_string(num[i]) + " " +
        operators[firstIdx] + " (" + to_string(num[j]) + " " +
        operators[secondIdx] + " (" + to_string(num[k]) + " " +
        operators[thirdIdx] + " " + to_string(num[l]) + ")"));
    if(!find(ans, answer)){
        count++;
        ans.push_back(answer);
    }
}
}
} else {
    continue;
}
}
}
}
}
}
}

// Calculate time execution main algorithm
timeConsumed = clock() - timeConsumed;

// OUTPUT
if(count == 0){
    printf("\nTidak ada solusi\n");
} else {
    printf("\n%d solutions found\n", count);
    for(int i = 0; i < ans.size(); i++){
        cout << ans[i] << endl;
    }
}

// TIME EXECUTION
cout << "Total time required = " << (double)timeConsumed/CLOCKS_PER_SEC << " seconds" << endl;

// OPTION TO SAVE FILE
while(true){
    printf("\nDo you want to save the answer? (y/n) : ");
    getline(cin >> ws, action);
    if(action == "y"){
        string nameFile;
        while(true){

```

```

printf("\nPlease input new name file : ");
getline(cin >> ws, nameFile);
nameFile += ".txt";
file.open("./test/" + nameFile);
if(file){
    printf("\nFile already exist! Please input a new name file!\n");
} else {
    outdata.open("./test/" + nameFile);
    if(count != 0){
        outdata << count << " solutions found" << endl;
        for(int i = 0; i < ans.size(); i++){
            outdata << ans[i] << endl;
        }
    } else {
        outdata << "Tidak ada solusi" << endl;
    }
    outdata << "Total time required = " << (double)timeConsumed/CLOCKS_PER_SEC << " seconds"
<< endl;
    outdata.close();
    cout << "\nFile " << nameFile << " has been created" << endl;
    break;
}
}
break;
} else if(action == "n"){
    break;
} else {
    printf("\nWRONG INPUT! Please input correctly!\n");
}
}

// OPTION TO PLAY AGAIN OR EXIT
while(true){
    printf("\nDo you want to play again? (y/n) : ");
    getline(cin >> ws, action);
    if(action == "y"){
        break;
    } else if (action == "n"){
        printf("\nThank you for playing!\n");
        done = true;
        break;
    } else {
        printf("\nWRONG INPUT! Please input correctly!\n");
    }
}
}
}
}

```

## BAB IV

### HASIL SCREENSHOT

#### 4.1 User Interface

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@#7^::::::::::::::::::::::::::::::::::::::::::::::::::::::::^7#@@
@@!                                                                    !@@
@@~                                                                    ~@@
@@~                                                                    ~@@
@@~                                                                    ~@@
@@~      .:~::~      .:~::~      .:~::~      ~@@
@@~      .!J5PPPPY!    !555P?    ?555PJ    ~@@
@@~      75PP55555PB~  YP55PY    YP55PP    ~@@
@@~      ?P555PP5555B!  .Y555PJ    JP55PP    ~@@
@@~      75PP5!^Y5555B7 .Y555GJ    ?P55PP.   ~@@
@@~      .~.~. .Y555PG: .Y555PP??Y555PP.   ~@@
@@~      ?P555G!      JP5555PP5555PP.   ~@@
@@~      7P555G?      ?PPPPPP5555PP.   ~@@
@@~      ?P555P?      :7?????5555PP.   ~@@
@@~      .JP555P?      ?P55PP.   ~@@
@@~      .JP555P7      JP55PP    ~@@
@@~      .JP555G7 ..... JP55PP    ~@@
@@~      7P5555P55555G~ Y555P5    ~@@
@@~      JP555555PP555B~ .5P55G5    ~@@
@@~      ?5555555555Y5~ JY5YY!    ~@@
@@~      .....      ...      ~@@
@@~      .....      ...      ~@@
@@!                                                                    !@@
@@#7^::::::::::::::::::::::::::::::::::::::::::::::::::::::::^7#@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Welcome to the 24 Game!
Please choose which type of input do you want:
1. Input from user
2. Generate input
3. Exit
Choose (1, 2, or 3) : █
```

Gambar 4.1.1 User Interface Opening

```
Do you want to save the answer? (y/n) : y
Please input new name file : hasil_1
File hasil_1.txt has been created
```

Gambar 4.1.2 User Interface Save Solution

```
Do you want to play again? (y/n) : n  
Thank you for playing!
```

Gambar 4.1.3 User Interface Exit Program

## 4.2 Contoh Input

```
Welcome to the 24 Game!  
Please choose which type of input do you want:  
1. Input from user  
2. Generate input  
3. Exit  
Choose (1, 2, or 3) : 1  
  
Your cards (ex: A 10 Q J): J Q K A
```

Gambar 4.2.1 Input From User

```
Welcome to the 24 Game!  
Please choose which type of input do you want:  
1. Input from user  
2. Generate input  
3. Exit  
Choose (1, 2, or 3) : 2  
  
Your cards : 10 K 4 7
```

Gambar 4.2.3 Generate Input

```
Welcome to the 24 Game!
Please choose which type of input do you want:
1. Input from user
2. Generate input
3. Exit
Choose (1, 2, or 3) : salah

WRONG INPUT! Please input correctly!

Please choose which type of input do you want:
1. Input from user
2. Generate input
3. Exit
Choose (1, 2, or 3) : █
```

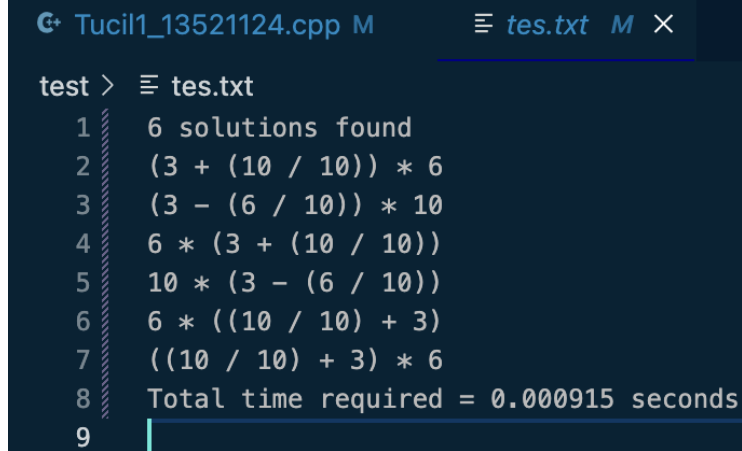
Gambar 4.2.3 Contoh Validasi Input

#### 4.3 Contoh Output

```
Your cards : 10 10 3 6

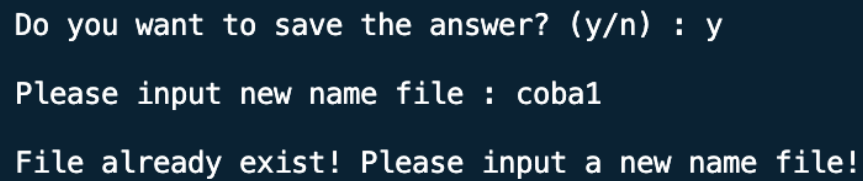
6 solutions found
(3 + (10 / 10)) * 6
(3 - (6 / 10)) * 10
6 * (3 + (10 / 10))
10 * (3 - (6 / 10))
6 * ((10 / 10) + 3)
((10 / 10) + 3) * 6
Total time required = 0.000915 seconds
```

Gambar 4.3.1 Contoh Output di Terminal



```
Tucil1_13521124.cpp M tes.txt M X
test > tes.txt
1 6 solutions found
2 (3 + (10 / 10)) * 6
3 (3 - (6 / 10)) * 10
4 6 * (3 + (10 / 10))
5 10 * (3 - (6 / 10))
6 6 * ((10 / 10) + 3)
7 ((10 / 10) + 3) * 6
8 Total time required = 0.000915 seconds
9
```

Gambar 4.3.2 Contoh Output di File .txt



```
Do you want to save the answer? (y/n) : y
Please input new name file : coba1
File already exist! Please input a new name file!
```

Gambar 4.2.4 Contoh Validasi File Output

#### 4.4 Test Case

```
28 solutions found
4 + ((7 + 3) * 2)
4 + ((3 + 7) * 2)
((7 + 2) - 3) * 4
(7 + (2 - 3)) * 4
((2 + 7) - 3) * 4
(2 + (7 - 3)) * 4
((7 + 3) * 2) + 4
4 + (2 * (7 + 3))
4 + (2 * (3 + 7))
((3 + 7) * 2) + 4
(2 + 4) * (7 - 3)
(4 + 2) * (7 - 3)
((7 - 3) + 2) * 4
((2 - 3) + 7) * 4
(7 - (3 - 2)) * 4
(2 - (3 - 7)) * 4
(7 - 3) * (2 + 4)
(7 - 3) * (4 + 2)
(2 * (7 + 3)) + 4
(2 * (3 + 7)) + 4
4 * ((7 + 2) - 3)
4 * (7 + (2 - 3))
4 * ((2 + 7) - 3)
4 * (2 + (7 - 3))
4 * ((7 - 3) + 2)
4 * ((2 - 3) + 7)
4 * (7 - (3 - 2))
4 * (2 - (3 - 7))
Total time required = 0.001566 seconds
```

Gambar 4.4.1 Test Case 1 dengan Input User

```
Your cards (ex: A 10 Q J): K K K K
Tidak ada solusi
Total time required = 0.001256 seconds
```

Gambar 4.4.2 Test Case 2 dengan Input User

```

Your cards (ex: A 10 Q J): A K A K

15 solutions found
(13 + 13) - (1 + 1)
13 + (13 - (1 + 1))
((13 + 13) - 1) - 1
(13 + (13 - 1)) - 1
13 + ((13 - 1) - 1)
(13 - (1 + 1)) + 13
13 - ((1 + 1) - 13)
13 - (1 + (1 - 13))
((13 - 1) + 13) - 1
(13 - 1) + (13 - 1)
((13 - 1) - 1) + 13
13 - ((1 - 13) + 1)
(13 - 1) - (1 - 13)
(13 - (1 - 13)) - 1
13 - (1 - (13 - 1))
Total time required = 0.001637 seconds

```

Gambar 4.4.3 Test Case 3 dengan Input User

```

Your cards (ex: A 10 Q J): 1 2 3 4

242 solutions found
((1 + 2) + 3) * 4
(1 + (2 + 3)) * 4
((1 + 3) + 2) * 4
(1 + (3 + 2)) * 4
((2 + 1) + 3) * 4
(2 + (1 + 3)) * 4
((2 + 3) + 1) * 4
(2 + (3 + 1)) * 4
((3 + 1) + 2) * 4
(3 + (1 + 2)) * 4
((3 + 2) + 1) * 4
(3 + (2 + 1)) * 4

```

Gambar 4.4.4 Test Case 4 dengan Input User

```

4 / (1 / (3 * 2))
2 / ((1 / 3) / 4)
2 / ((1 / 4) / 3)
3 / ((1 / 2) / 4)
3 / ((1 / 4) / 2)
4 / ((1 / 2) / 3)
4 / ((1 / 3) / 2)
Total time required = 0.008193 seconds

```

Gambar 4.4.5 Lanjutan Test Case 4



```

Your cards (ex: A 10 Q J): A 10 Q J

20 solutions found
((1 + 11) - 10) * 12
(1 + (11 - 10)) * 12
((11 + 1) - 10) * 12
(11 + (1 - 10)) * 12
(1 + 11) * (12 - 10)
(11 + 1) * (12 - 10)
((1 - 10) + 11) * 12
((11 - 10) + 1) * 12
(1 - (10 - 11)) * 12
(11 - (10 - 1)) * 12
(12 - 10) * (1 + 11)
(12 - 10) * (11 + 1)
12 * ((1 + 11) - 10)
12 * (1 + (11 - 10))
12 * ((11 + 1) - 10)
12 * (11 + (1 - 10))
12 * ((1 - 10) + 11)
12 * ((11 - 10) + 1)
12 * (1 - (10 - 11))
12 * (11 - (10 - 1))
Total time required = 0.00147 seconds

```

Gambar 4.4.6 Test Case 5 dengan Input User

```

Your cards : A 9 5 10

180 solutions found
(9 + 5) + (1 * 10)
9 + (5 + (1 * 10))
((9 + 5) + 10) * 1
(9 + 5) + (10 * 1)
(9 + (5 + 10)) * 1
9 + ((5 + 10) * 1)
9 + (5 + (10 * 1))
(9 + 10) + (1 * 5)
9 + (10 + (1 * 5))
((9 + 10) + 5) * 1
(9 + 10) + (5 * 1)
(9 + (10 + 5)) * 1

```

Gambar 4.4.7 Test Case 6 dengan Generate Input

```

((5 / 1) + 10) + 9
(5 / 1) + (10 + 9)
((10 / 1) + 9) + 5
(10 / 1) + (9 + 5)
((10 / 1) + 5) + 9
(10 / 1) + (5 + 9)
Total time required = 0.005765 seconds

```

Gambar 4.4.8 Lanjutan Test Case 6

Your cards : 6 A 10 8

```

189 solutions found
(6 + 10) + (1 * 8)
6 + (10 + (1 * 8))
((6 + 10) + 8) * 1
(6 + 10) + (8 * 1)
(6 + (10 + 8)) * 1
6 + ((10 + 8) * 1)
6 + (10 + (8 * 1))
(6 + 8) + (1 * 10)
6 + (8 + (1 * 10))
((6 + 8) + 10) * 1
(6 + 8) + (10 * 1)

```

Gambar 4.4.9 Test Case 7 dengan Generate Input

```

(10 / 1) + (8 + 6)
((8 / 1) + 6) + 10
(8 / 1) + (6 + 10)
((8 / 1) + 10) + 6
(8 / 1) + (10 + 6)
6 / ((10 / 8) - 1)
Total time required = 0.006434 seconds

```

Gambar 4.4.10 Lanjutan Test Case 7

```

Your cards : 10 5 Q 8

16 solutions found
((10 + 5) - 12) * 8
(10 + (5 - 12)) * 8
((5 + 10) - 12) * 8
(5 + (10 - 12)) * 8
((10 - 12) + 5) * 8
((5 - 12) + 10) * 8
(10 - (12 - 5)) * 8
(5 - (12 - 10)) * 8
8 * ((10 + 5) - 12)
8 * (10 + (5 - 12))
8 * ((5 + 10) - 12)
8 * (5 + (10 - 12))
8 * ((10 - 12) + 5)
8 * ((5 - 12) + 10)
8 * (10 - (12 - 5))
8 * (5 - (12 - 10))
Total time required = 0.000924 seconds

```

Gambar 4.4.11 Test Case 8 dengan Generate Input

```

Your cards : A Q 3 3

20 solutions found
12 + ((1 + 3) * 3)
12 + ((3 + 1) * 3)
((1 + 3) * 3) + 12
12 + (3 * (1 + 3))
12 + (3 * (3 + 1))
((3 + 1) * 3) + 12
(1 + (3 / 3)) * 12
(12 - (1 + 3)) * 3
(12 - (3 + 1)) * 3
((12 - 1) - 3) * 3
((12 - 3) - 1) * 3
(3 * (1 + 3)) + 12
(3 * (3 + 1)) + 12
12 * (1 + (3 / 3))
3 * (12 - (1 + 3))
3 * (12 - (3 + 1))
3 * ((12 - 1) - 3)
3 * ((12 - 3) - 1)
12 * ((3 / 3) + 1)
((3 / 3) + 1) * 12
Total time required = 0.001449 seconds

```

Gambar 4.4.12 Test Case 9 dengan Generate Input

```
Your cards : 7 9 2 5  
  
8 solutions found  
(7 * 5) - (9 + 2)  
(7 * 5) - (2 + 9)  
(5 * 7) - (9 + 2)  
(5 * 7) - (2 + 9)  
((7 * 5) - 9) - 2  
((7 * 5) - 2) - 9  
((5 * 7) - 9) - 2  
((5 * 7) - 2) - 9  
Total time required = 0.00088 seconds
```

Gambar 4.4.13 Test Case 10 dengan Generate Input

## BAB V

### LAMPIRAN

#### 5.1. Link Repository Github

Link repository github :

[https://github.com/maikeljh/Tucil1\\_13521124](https://github.com/maikeljh/Tucil1_13521124)

#### 5.2 Checklist Table Progress

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