

LAPORAN TUGAS KECIL 01

IF2211 STRATEGI ALGORITMA

“Penyelesaian Permainan Kartu 24 dengan Algoritma *Brute Force*”



Disusun oleh:

Alexander Jason K-02 13521100

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

DAFTAR ISI	2
BAB 1 DESKRIPSI MASALAH	3
BAB 2 TEORI SINGKAT	6
BAB 3 IMPLEMENTASI PROGRAM	12
BAB 4 EKSPERIMEN	21
BAB 5 PENUTUP	54
DAFTAR REFERENSI	56

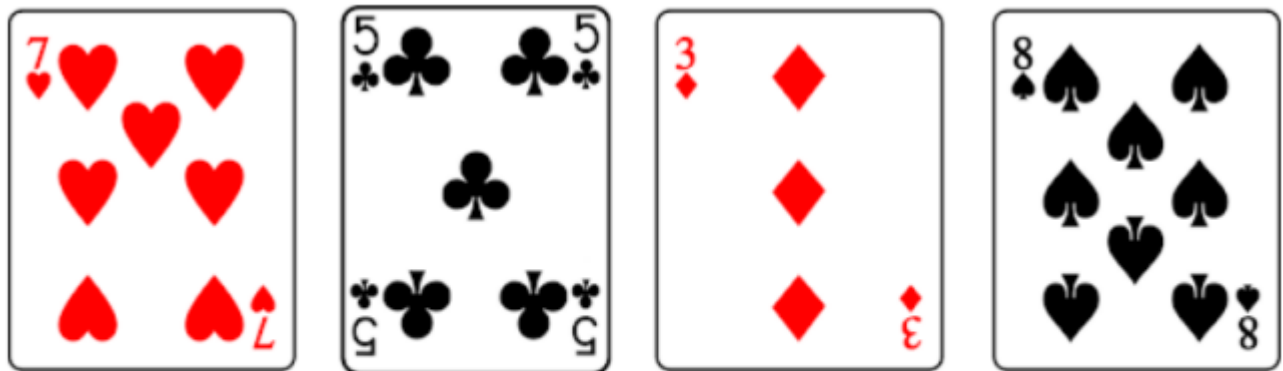
BAB 1

DESKRIPSI MASALAH

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

BAB 2

TEORI SINGKAT

2.1 Algoritma Brute Force

Algoritma yang digunakan untuk menyelesaikan permasalahan ‘permainan 24’ adalah metode *Brute Force*. Program akan mengecek semua jenis kemungkinan susunan kartu dari 4 kartu yang sudah tersedia.

Langkah-langkah algoritma yang digunakan dalam menyelesaikan permainan tersebut adalah sebagai berikut:

1. Permutasi susunan kartu

Program menghasilkan semua kombinasi susunan dari 4 kartu yang ada. Kombinasi dari 4 kartu tersebut akan dimasukkan ke sebuah *set of array*. Jika terdapat kombinasi kartu yang sama, maka kombinasi tersebut tidak akan dimasukkan ke dalam set. Jika 4 kartu berbeda, maka akan terdapat 24 kombinasi, 3 kartu berbeda maka akan terdapat 12 kombinasi, 2 kartu berbeda maka akan terdapat 4 kombinasi, serta jika semua kartu sama maka akan terdapat 1 kombinasi saja.

2. Permutasi susunan operasi aritmatika dasar

Program menghasilkan semua kombinasi peletakkan operator yang legal (+, -, x, ÷). Terdapat 64 susunan yang dihasilkan.

3. Permutasi peletakkan tanda kurung ()

Terdapat 5 jenis susunan peletakkan tanda kurung, yaitu:

$(a \text{ op } b) \text{ op } (c \text{ op } d)$

$((a \text{ op } b) \text{ op } c) \text{ op } d$

$(a \text{ op } (b \text{ op } c)) \text{ op } d$

$a \text{ op } ((b \text{ op } c) \text{ op } d)$

$a \text{ op } (b \text{ op } (c \text{ op } d))$

dengan (a, b, c, d) sebagai angka dari kartu, serta (op) sebagai operator aritmatika. Program lalu akan memproses dan menghitung semua jenis susunan permutasi yaitu maksimal sebanyak $24 \times 64 \times 5 = 7680$ kali.

4. Solusi

Terakhir, program akan memeriksa semua kombinasi tersebut. Kombinasi yang menghasilkan angka 24 akan disimpan sebagai solusi dari jawaban tersebut. Solusi-solusi yang dihasilkan

program ini jika bersifat komutatif, asosiatif, maupun distributif akan dianggap berbeda dengan solusi lain. Contoh: $(4 \times 3) + (6 \times 2)$ akan dianggap berbeda dengan $(3 \times 4) + (6 \times 2)$

BAB 3

IMPLEMENTASI PROGRAM

3.1 FOLDER PRIMITIF (src)

a. main.cpp

file *main.cpp* memuat ascii art yang menjadi *splash screen* dari program utama. Program ini juga menjalankan program utama.



```
1  #include <bits/stdc++.h>
2  #include "menu.h"
3  #include "combination.h"
4  #include "solver.h"
5  using namespace std;
6
7  int main(){
8      std::string inp, choice;
9      bool valid=false;
10     //displaying ascii art
11     string fileline;
12     ifstream file("../src/ascii.txt");
13     while(getline(file, fileline)){
14         cout << fileline << endl;
15     }
16     file.close();
17     printf("Welcome to 24 Solver!\n");
18     mainmenu();
19     return 0;
20 }
```

Gambar 2 Program *main.cpp*

b. combination.cpp

File ini memuat semua proses permutasi dari langkah-langkah yang sudah dijelaskan pada bab 2.

```

src > combination.cpp > ...
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  /**
6   * @brief evaluate a and b using operator op
7   *
8   * @param op
9   * @param a
10  * @param b
11  * @return float
12  */
13 float evaluate(int op, float a, float b){
14     switch (op){
15         case 0:
16             return a + b;
17         case 1:
18             return a - b;
19         case 2:
20             return a * b;
21         case 3:
22             return a / b;
23     }
24 };
25
26 /**
27  * @brief return operator to string
28  *
29  * @param op
30  * @return string
31  */
32 string opToStr(int op){
33     switch (op){
34         case 0:
35             return "+";
36         case 1:
37             return "-";
38         case 2:
39             return "x";
40         case 3:
41             return "/";
42     }
43 };
44
45 /**
46  * @brief permutation of input cards
47  *
48  * @param cards
49  * @param combination
50  * @param usedPermute
51  */
52 void permutationCards(vector<int> &cards, vector<int> &combination, set<vector<int>> &usedPermute){
53     int i, j, k, l;
54     for (int i = 0; i < 4; i++){
55         for (int j = 0; j < 4; j++){
56             for (int k = 0; k < 4; k++){
57                 for (int l = 0; l < 4; l++){
58                     if (i!=j && i!=k && i!=l && j!=k && j!=l && k!=l){
59                         combination.clear();
60                         combination.push_back(cards[i]);
61                         combination.push_back(cards[j]);
62                         combination.push_back(cards[k]);
63                         combination.push_back(cards[l]);
64                         usedPermute.insert(combination);
65                     }
66                 }
67             }
68         }
69     }
70 };
71
72 /**

```

Gambar 3 Program *combination.cpp* (1)

```

/**
 * @brief permutation operators
 *
 * @param combinationOp
 */
void permutationOps(vector<vector<int>> &combinationOp){
    vector<int> temp(3);
    for (int i = 0; i < 4; i++){
        for (int j = 0; j < 4; j++){
            for (int k = 0; k < 4; k++){
                temp[0] = i;
                temp[1] = j;
                temp[2] = k;
                combinationOp.push_back(temp);
            }
        }
    }
}

```

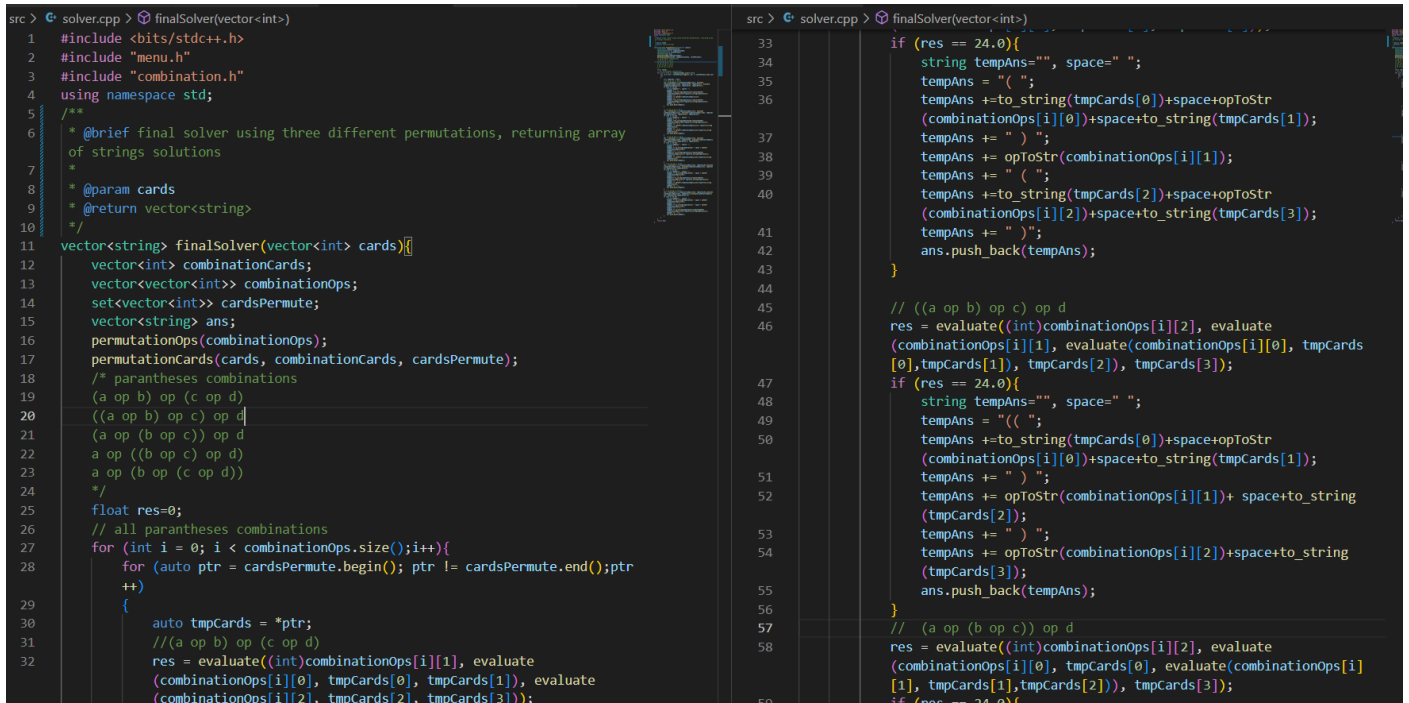
Gambar 4 Program *combination.cpp* (2)

c. solver.cpp

File ini memuat program yang mengeksekusi semua proses permutasi

- `finalSolver()`: program mula-mula menginisiasi array yang berisi kumpulan permutasi kartu dan operator. Kemudian program akan melakukan *looping* terhadap 2 permutasi tersebut dengan 5 kombinasi peletakkan tanda kurung

yang berbeda-beda. Disinilah algoritma brute force digunakan. Jika ada kombinasi kartu yang menghasilkan 24, maka kombinasi tersebut akan disimpan dalam bentuk string.

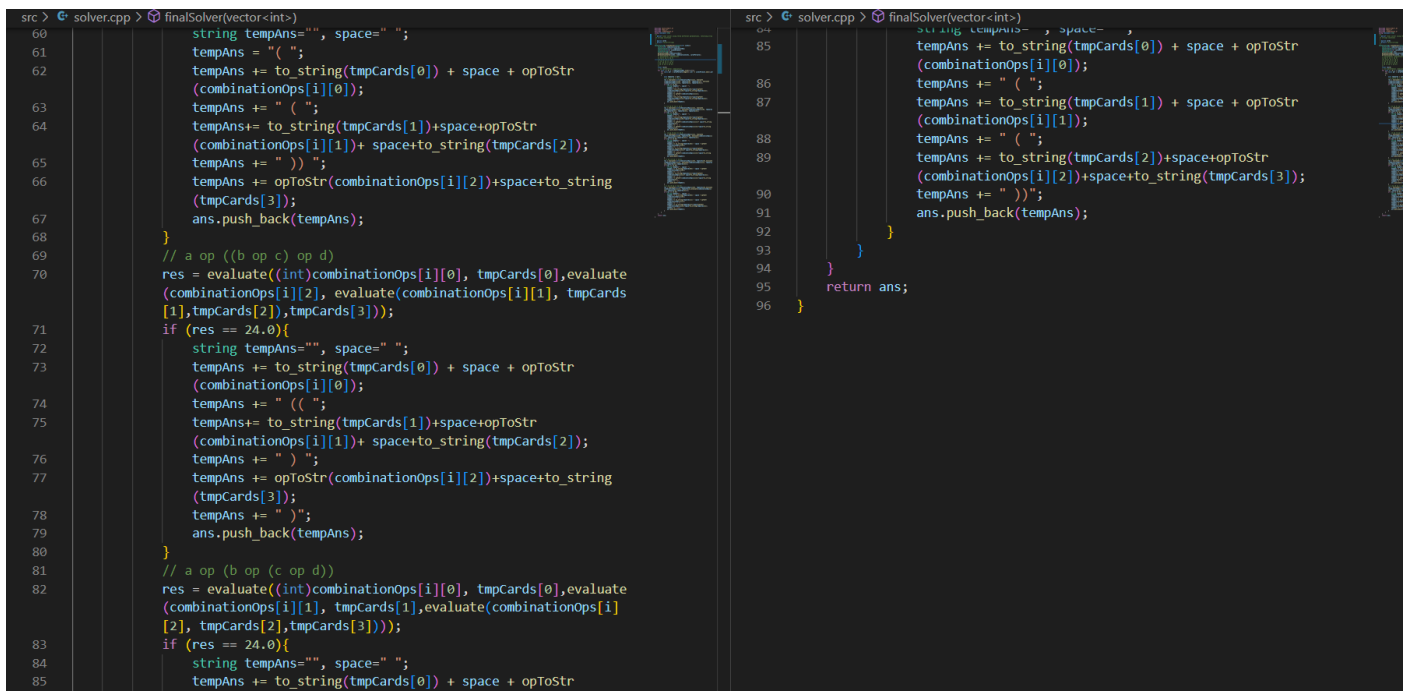


```

src > solver.cpp > finalSolver(vector<int>)
1  #include <bits/stdc++.h>
2  #include "menu.h"
3  #include "combination.h"
4  using namespace std;
5  /**
6   * @brief final solver using three different permutations, returning array
7   * of strings solutions
8   *
9   * @param cards
10  * @return vector<string>
11  */
12  vector<string> finalSolver(vector<int> cards){
13      vector<int> combinationCards;
14      vector<vector<int>>> combinationOps;
15      set<vector<int>>> cardsPermute;
16      vector<string> ans;
17      permutationOps(combinationOps);
18      permutationCards(cards, combinationCards, cardsPermute);
19      /* parentheses combinations
20      (a op b) op (c op d)
21      ((a op b) op c) op d
22      (a op ((b op c) op d))
23      a op ((b op c) op d)
24      a op (b op (c op d))
25      */
26      float res=0;
27      // all parentheses combinations
28      for (int i = 0; i < combinationOps.size();i++){
29          for (auto ptr = cardsPermute.begin(); ptr != cardsPermute.end();ptr
30              ++){
31              auto tmpCards = *ptr;
32              //((a op b) op (c op d))
33              res = evaluate((int)combinationOps[i][1], evaluate
34                  (combinationOps[i][0], tmpCards[0], tmpCards[1]), evaluate
35                  (combinationOps[i][2], tmpCards[2], tmpCards[3]));
36              if (res == 24.0){
37                  string tempAns="", space=" ";
38                  tempAns += "(";
39                  tempAns += to_string(tmpCards[0])+space+opToStr
40                      (combinationOps[i][0])+space+to_string(tmpCards[1]);
41                  tempAns += " )";
42                  tempAns += opToStr(combinationOps[i][1]);
43                  tempAns += " (";
44                  tempAns += to_string(tmpCards[2])+space+opToStr
45                      (combinationOps[i][2])+space+to_string(tmpCards[3]);
46                  tempAns += " )";
47                  ans.push_back(tempAns);
48              }
49              // ((a op b) op c) op d
50              res = evaluate((int)combinationOps[i][2], evaluate
51                  (combinationOps[i][1], evaluate(combinationOps[i][0], tmpCards
52                      [0], tmpCards[1]), tmpCards[2]), tmpCards[3]);
53              if (res == 24.0){
54                  string tempAns="", space=" ";
55                  tempAns += "(";
56                  tempAns += to_string(tmpCards[0])+space+opToStr
57                      (combinationOps[i][0])+space+to_string(tmpCards[1]);
58                  tempAns += " )";
59                  tempAns += opToStr(combinationOps[i][1])+ space+to_string
60                      (tmpCards[2]);
61                  tempAns += " )";
62                  tempAns += opToStr(combinationOps[i][2])+space+to_string
63                      (tmpCards[3]);
64                  ans.push_back(tempAns);
65              }
66              // (a op (b op c) op d)
67              res = evaluate((int)combinationOps[i][2], evaluate
68                  (combinationOps[i][0], tmpCards[0], evaluate(combinationOps[i]
69                      [1], tmpCards[1], tmpCards[2])), tmpCards[3]);
70              if (res == 24.0){
71                  string tempAns="", space=" ";
72                  tempAns += to_string(tmpCards[0]) + space + opToStr
73                      (combinationOps[i][0]);
74                  tempAns += " (";
75                  tempAns += to_string(tmpCards[1])+space+opToStr
76                      (combinationOps[i][1])+ space+to_string(tmpCards[2]);
77                  tempAns += " )";
78                  tempAns += opToStr(combinationOps[i][2])+space+to_string
79                      (tmpCards[3]);
80                  ans.push_back(tempAns);
81              }
82              // a op ((b op c) op d)
83              res = evaluate((int)combinationOps[i][0], tmpCards[0],evaluate
84                  (combinationOps[i][2], evaluate(combinationOps[i][1], tmpCards
85                      [1], tmpCards[2]), tmpCards[3]));
86              if (res == 24.0){
87                  string tempAns="", space=" ";
88                  tempAns += to_string(tmpCards[0]) + space + opToStr
89                      (combinationOps[i][0]);
90                  tempAns += " (";
91                  tempAns += to_string(tmpCards[1])+space+opToStr
92                      (combinationOps[i][1])+ space+to_string(tmpCards[2]);
93                  tempAns += " )";
94                  tempAns += opToStr(combinationOps[i][2])+space+to_string
95                      (tmpCards[3]);
96                  ans.push_back(tempAns);
97              }
98          }
99      }
100      return ans;
101  }

```

Gambar 5 Program solver.cpp (1)



```

src > solver.cpp > finalSolver(vector<int>)
60  string tempAns="", space=" ";
61  tempAns += "(";
62  tempAns += to_string(tmpCards[0]) + space + opToStr
63      (combinationOps[i][0]);
64  tempAns += " (";
65  tempAns += to_string(tmpCards[1])+space+opToStr
66      (combinationOps[i][1])+ space+to_string(tmpCards[2]);
67  tempAns += " )";
68  tempAns += opToStr(combinationOps[i][2])+space+to_string
69      (tmpCards[3]);
70  ans.push_back(tempAns);
71  }
72  // a op ((b op c) op d)
73  res = evaluate((int)combinationOps[i][0], tmpCards[0],evaluate
74      (combinationOps[i][2], evaluate(combinationOps[i][1], tmpCards
75          [1],tmpCards[2]),tmpCards[3]));
76  if (res == 24.0){
77      string tempAns="", space=" ";
78      tempAns += to_string(tmpCards[0]) + space + opToStr
79          (combinationOps[i][0]);
80      tempAns += " (";
81      tempAns += to_string(tmpCards[1])+space+opToStr
82          (combinationOps[i][1])+ space+to_string(tmpCards[2]);
83      tempAns += " )";
84      tempAns += opToStr(combinationOps[i][2])+space+to_string
85          (tmpCards[3]);
86      ans.push_back(tempAns);
87  }
88  // a op (b op (c op d))
89  res = evaluate((int)combinationOps[i][0], tmpCards[0],evaluate
90      (combinationOps[i][1], tmpCards[1],evaluate(combinationOps[i]
91          [2], tmpCards[2],tmpCards[3])));
92  if (res == 24.0){
93      string tempAns="", space=" ";
94      tempAns += to_string(tmpCards[0]) + space + opToStr
95          (combinationOps[i][0]);
96      tempAns += " (";
97      tempAns += to_string(tmpCards[1])+space+opToStr
98          (combinationOps[i][1])+ space+to_string(tmpCards[2]);
99      tempAns += " )";
100      tempAns += opToStr(combinationOps[i][2])+space+to_string
101          (tmpCards[3]);
102      ans.push_back(tempAns);
103  }
104  }
105  return ans;
106  }

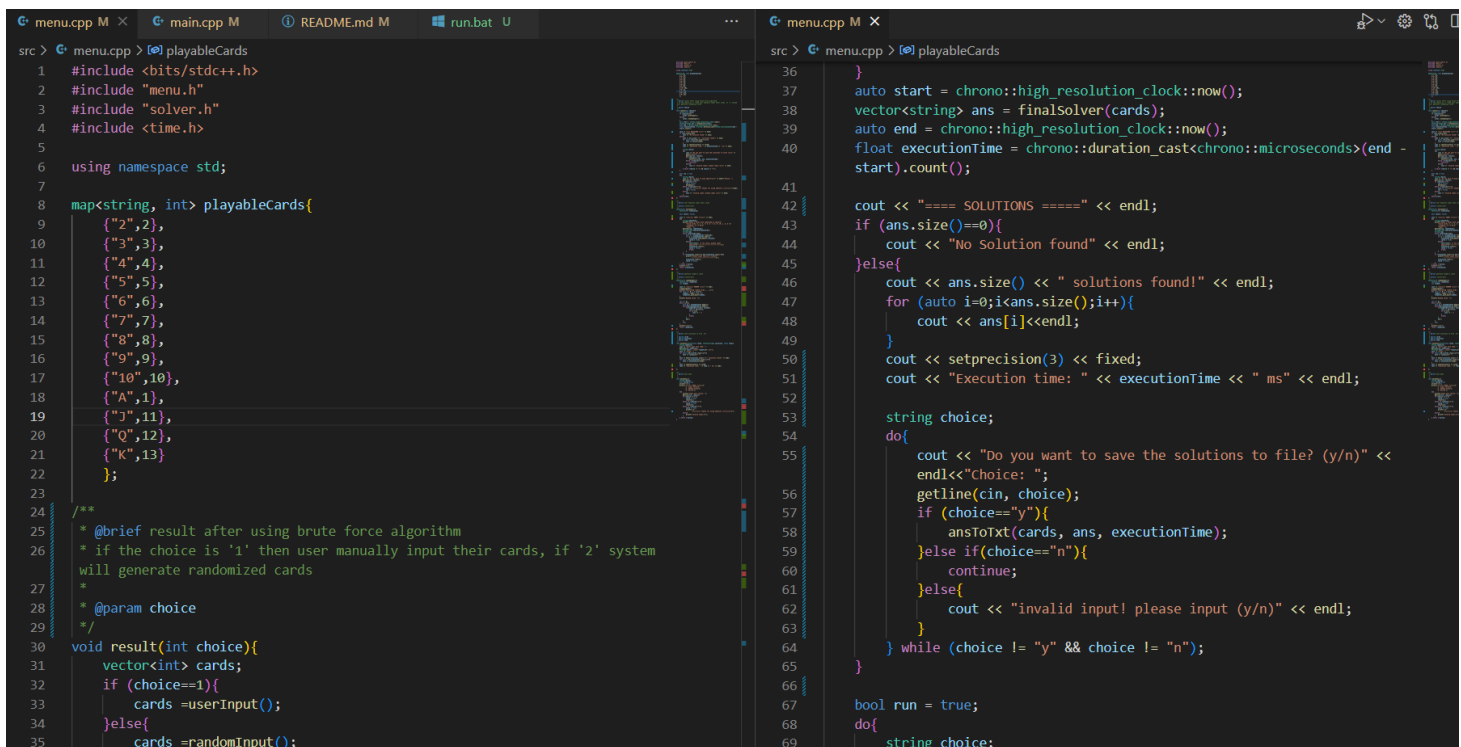
```

Gambar 6 Program solver.cpp (2)

d. menu.cpp

Terdapat variabel global yang menyimpan jenis kartu yang valid beserta nilainya. File ini memuat menu apa saja yang digunakan dalam program utama. Seperti:

- `mainmenu()`: menu utama
- `result(int choice)`: hasil akhir dari permainan. Menampilkan jumlah solusi serta waktu eksekusi program.
- `userInput()`: user dapat memasukkan 4 kartu yang diinginkan secara manual. Jika input tidak valid atau tidak berjumlah 4 buah, maka program akan meminta user untuk melakukan input ulang.
- `randomInput()`: program akan menghasilkan 4 kartu secara random
- `ansToTxt()`: program akan menyimpan solusi ke dalam file .txt



```
src > menu.cpp M x main.cpp M README.md M run.bat U ... menu.cpp M x
src > menu.cpp > playableCards
1 #include <bits/stdc++.h>
2 #include "menu.h"
3 #include "solver.h"
4 #include <time.h>
5
6 using namespace std;
7
8 map<string, int> playableCards{
9     {"2",2},
10    {"3",3},
11    {"4",4},
12    {"5",5},
13    {"6",6},
14    {"7",7},
15    {"8",8},
16    {"9",9},
17    {"10",10},
18    {"A",1},
19    {"J",11},
20    {"Q",12},
21    {"K",13}
22 };
23
24 /**
25  * @brief result after using brute force algorithm
26  * if the choice is '1' then user manually input their cards, if '2' system
27  * will generate randomized cards
28  *
29  * @param choice
30  */
31 void result(int choice){
32     vector<int> cards;
33     if (choice==1){
34         cards =userInput();
35     }else{
36         cards =randomInput();
37     }
38     auto start = chrono::high_resolution_clock::now();
39     vector<string> ans = finalSolver(cards);
40     auto end = chrono::high_resolution_clock::now();
41     float executionTime = chrono::duration_cast<chrono::microseconds>(end - start).count();
42
43     cout << "==== SOLUTIONS =====< endl;
44     if (ans.size()==0){
45         cout << "No Solution found" << endl;
46     }else{
47         cout << ans.size() << " solutions found!" << endl;
48         for (auto i=0;i<ans.size();i++){
49             cout << ans[i]<<endl;
50         }
51         cout << setprecision(3) << fixed;
52         cout << "Execution time: " << executionTime << " ms" << endl;
53
54         string choice;
55         do{
56             cout << "Do you want to save the solutions to file? (y/n)" <<
57             endl<<"Choice: ";
58             getline(cin, choice);
59             if (choice=="y"){
60                 ansToTxt(cards, ans, executionTime);
61             }else if(choice=="n"){
62                 continue;
63             }else{
64                 cout << "invalid input! please input (y/n)" << endl;
65             }
66         } while (choice != "y" && choice != "n");
67     }
68
69     bool run = true;
70     do{
71         string choice;
```

Gambar 7 Program *menu.cpp* (1)

```

70     cout << "Do you want to play again?(y/n)" << endl<<"Choice: ";
71     getline(cin, choice);
72     if (choice=="y"){
73         run = false;
74         mainmenu();
75     }else if(choice=="n"){
76         cout << "===== Thanks for using 24Solver =====<<endl;
77         run = false;
78     }else{
79         cout << "invalid input! please input (y/n)" << endl;
80     }
81 }while(run);
82 };
83
84 /**
85  * @brief user manually input their cards
86  *
87  * @return vector<int>
88  */
89 vector<int> userInput(){
90     vector<int> finalCards;
91
92     bool valid = false;
93
94     cout << "\n===== INPUT =====<< endl;
95     do{
96         string inputCards;
97         printf("Input 4 Cards each separated by space\n"
98             "valid cards: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K \n"
99             "(example: A 2 3 Q)\n"
100             "Input: ");
101         getline(cin, inputCards);
102         stringstream charCard(inputCards);
103         string inp;
104         while (charCard>>inp){
105             auto i = playableCards.find(inp);

```

```

106         if (i != playableCards.end()){
107             finalCards.push_back(i->second);
108             valid = true;
109         }else{
110             cout<<inp<<" is not valid, please input
111             again\n"<<"-----\n";
112             finalCards.clear();
113             valid = false;
114             break;
115         }
116         if (finalCards.size()!=4 and finalCards.size()!=0){
117             printf("Please input exactly 4 cards\n"
118                 "-----\n");
119             finalCards.clear();
120             valid = false;
121         }
122     } while (!valid);
123     printf("\n");
124     return finalCards;
125 };
126
127 /**
128  * @brief generate random 4 cards
129  *
130  * @return vector<int>
131  */
132 vector<int> randomInput(){
133     vector<int> randCards;
134     int random;
135
136     cout << "\n===== RANDOM =====<< endl;
137     srand(time(0));
138     printf("Generating random cards.....\n");
139     for (int i = 0; i < 4;i++){
140         random = rand() % 13 + 1;

```

Gambar 8 Program *menu.cpp* (2)

```

141     randCards.push_back(random);
142 }
143 printf("Random Cards: ");
144
145 int i = 0;
146 while (i<4){
147     auto op = playableCards.begin();
148     while(op!=playableCards.end()){
149         if(randCards[i]==op->second){
150             cout << op->first;
151             if (i!=3){
152                 cout << " ";
153             }
154             break;
155         }
156         op++;
157     }
158     i++;
159 }
160 printf("\n\n");
161 return randCards;
162 };
163
164 /**
165  * @brief save solutions to file .txt
166  *
167  * @param cards
168  * @param solutions
169  * @param time
170  */
171 void ansToTxt(vector<int> cards, vector<string> solutions, float time){
172     string namafile;
173     cout << "Enter your file name: ";
174     getline(cin, namafile);
175     ofstream text("../test/"+namafile+".txt");
176     text << "Cards: ";

```

```

177     for (auto j=0;j<cards.size();j++){
178         text << cards[j]<<" ";
179     }
180     text << endl<<solutions.size() << " solutions found!" << endl;
181     for (auto i=0;i<solutions.size();i++){
182         text << solutions[i]<<endl;
183     }
184     text << setprecision(3) << fixed;
185     text << "Execution time: " << time << " ms" << endl;
186
187 };
188
189 /**
190  * @brief main menu
191  *
192  */
193 void mainmenu(){
194     string choice;
195     bool valid=false;
196     printf("\n");
197     printf("===== MENU =====\n"
198         "1. Input Cards\n"
199         "2. Random Cards\n"
200         "3. Exit\n");
201
202     do{
203         printf("Input your choice: ");
204         getline(cin, choice);
205         if (choice=="1"){
206             valid = true;
207             result(1);
208         }else if (choice=="2"){
209             valid = true;
210             result(2);
211         }else if (choice=="3"){
212             valid = true;
213             printf("\n"

```

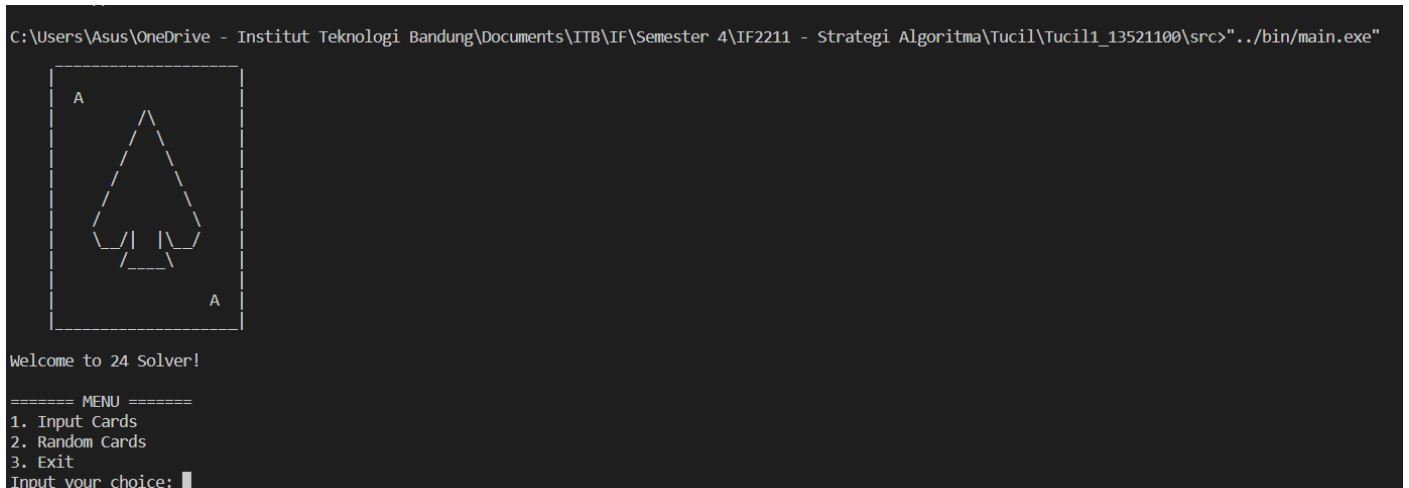
Gambar 9 Program *menu.cpp* (3)

BAB 4

EKSPERIMEN

Initialize Program

```
C:\Users\Asus\OneDrive - Institut Teknologi Bandung\Documents\ITB\IF\Semester 4\IF2211 - Strategi Algoritma\Tucil\Tucil1_13521100\src>"../bin/main.exe"
```



```
Welcome to 24 Solver!
```

```
===== MENU =====
```

```
1. Input Cards
```

```
2. Random Cards
```

```
3. Exit
```

```
Input your choice: |
```

Gambar 10 tampilan awal program

```
Welcome to 24 Solver!
```

```
===== MENU =====
```

```
1. Input Cards
```

```
2. Random Cards
```

```
3. Exit
```

```
Input your choice: 4
```

```
Invalid Input!
```

```
Input your choice: |
```

Gambar 11 tampilan input tidak valid

Input Manual dari User (Opsi 1)

```

===== MENU =====
1. Input Cards
2. Random Cards
3. Exit
Input your choice: 1

===== INPUT =====
Input 4 Cards each separated by space
valid cards: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
(example: A 2 3 Q)
Input: 2 2 K 6

```

Gambar 12 tampilan manual input

```

===== INPUT =====
Input 4 Cards each separated by space
valid cards: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
(example: A 2 3 Q)
Input: A 1 2 3
1 is not valid, please input again
-----
Input 4 Cards each separated by space
valid cards: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
(example: A 2 3 Q)
Input: 10 100 A 2
100 is not valid, please input again
-----
Input 4 Cards each separated by space
valid cards: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
(example: A 2 3 Q)
Input: A 2 3 4 4
Please input exactly 4 cards
-----
Input 4 Cards each separated by space
valid cards: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
(example: A 2 3 Q)
Input: █

```

Gambar 13 tampilan invalid manual input

Program Memilih 4 Kartu Secara Random (Opsi 2)

```
===== MENU =====
1. Input Cards
2. Random Cards
3. Exit
Input your choice: 2

===== RANDOM =====
Generating random cards.....
Random Cards: 8 3 8 2
```

Gambar 14 tampilan randomized cards

Keluar Dari Program (Opsi 3)

```
Welcome to 24 Solver!

===== MENU =====
1. Input Cards
2. Random Cards
3. Exit
Input your choice: 3

===== Thanks for using 24Solver =====
```

Gambar 15 tampilan keluar dari program

Menyimpan Hasil Solusi ke Dalam File

```
===== RANDOM =====
Generating random cards.....
Random Cards: Q K 10 Q

===== SOLUTIONS =====
14 solutions found!
( 10 + 13 ) + ( 12 / 12 )
10 + ( 13 + ( 12 / 12 ))
( 13 + 10 ) + ( 12 / 12 )
13 + ( 10 + ( 12 / 12 ))
( 10 + ( 12 / 12 )) + 13
10 + (( 12 / 12 ) + 13 )
( 13 + ( 12 / 12 )) + 10
13 + (( 12 / 12 ) + 10 )
(( 13 - 10 ) x 12 ) - 12
( 12 x ( 13 - 10 )) - 12
( 12 / 12 ) + ( 10 + 13 )
(( 12 / 12 ) + 10 ) + 13
( 12 / 12 ) + ( 13 + 10 )
(( 12 / 12 ) + 13 ) + 10
Execution time: 992.000 ms
Do you want to save the solutions to file? (y/n)
Choice: y
Enter your file name: testfile
Do you want to play again?(y/n)
Choice:

1 Cards: 12 13 10 12
2 14 solutions found!
3 ( 10 + 13 ) + ( 12 / 12 )
4 10 + ( 13 + ( 12 / 12 ))
5 ( 13 + 10 ) + ( 12 / 12 )
6 13 + ( 10 + ( 12 / 12 ))
7 ( 10 + ( 12 / 12 )) + 13
8 10 + (( 12 / 12 ) + 13 )
9 ( 13 + ( 12 / 12 )) + 10
10 13 + (( 12 / 12 ) + 10 )
11 (( 13 - 10 ) x 12 ) - 12
12 ( 12 x ( 13 - 10 )) - 12
13 ( 12 / 12 ) + ( 10 + 13 )
14 (( 12 / 12 ) + 10 ) + 13
15 ( 12 / 12 ) + ( 13 + 10 )
16 (( 12 / 12 ) + 13 ) + 10
17 Execution time: 992.000 ms
18
```

Gambar 16 tampilan menyimpan solusi pada file

Contoh 1 (A Q 7 Q)

```
===== MENU =====
1. Input Cards
2. Random Cards
3. Exit
Input your choice: 2

===== RANDOM =====
Generating random cards.....
Random Cards: A Q 7 Q

===== SOLUTIONS =====
4 solutions found!
( 12 x 12 ) / ( 7 - 1 )
12 x ( 12 / ( 7 - 1 ))
( 12 / ( 7 - 1 )) x 12
12 / (( 7 - 1 ) / 12 )
Execution time: 0.000 ms
Do you want to save the solutions to file? (y/n)
Choice: n
```

Gambar 17 contoh *test case* 1

Contoh 2 (A A A A)

```
===== MENU =====
1. Input Cards
2. Random Cards
3. Exit
Input your choice: 1

===== INPUT =====
Input 4 Cards each separated by space
valid cards: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
(example: A 2 3 Q)
Input: A A A A

===== SOLUTIONS =====
No Solution found
Do you want to play again?(y/n)
Choice: █
```

Gambar 18 contoh *test case* 2

Contoh 3 (K J A 2)

```
Input: K J A 2

==== SOLUTIONS ====
24 solutions found!
11 + (( 2 - 1 ) x 13 )
13 + (( 2 - 1 ) x 11 )
( 11 + 13 ) x ( 2 - 1 )
11 + ( 13 x ( 2 - 1 ))
( 13 + 11 ) x ( 2 - 1 )
13 + ( 11 x ( 2 - 1 ))
( 11 + 13 ) / ( 2 - 1 )
11 + ( 13 / ( 2 - 1 ))
( 13 + 11 ) / ( 2 - 1 )
13 + ( 11 / ( 2 - 1 ))
11 - (( 1 - 2 ) x 13 )
13 - (( 1 - 2 ) x 11 )
( 2 - 1 ) x ( 11 + 13 )
(( 2 - 1 ) x 11 ) + 13
( 2 - 1 ) x ( 13 + 11 )
(( 2 - 1 ) x 13 ) + 11
11 - ( 13 x ( 1 - 2 ))
13 - ( 11 x ( 1 - 2 ))
11 - ( 13 / ( 1 - 2 ))
13 - ( 11 / ( 1 - 2 ))
( 11 x ( 2 - 1 )) + 13
( 13 x ( 2 - 1 )) + 11
( 11 / ( 2 - 1 )) + 13
( 13 / ( 2 - 1 )) + 11
Execution time: 2000.000 ms
Do you want to save the solutions to file? (y/n)
Choice: 
```

Gambar 19 contoh *test case* 3

Contoh 4 (8 9 8 7)

```
Input your choice: 2

===== RANDOM =====
Generating random cards.....
Random Cards: 8 9 8 7

==== SOLUTIONS ====
6 solutions found!
8 + (( 9 - 7 ) x 8 )
8 + ( 8 x ( 9 - 7 ))
8 - (( 7 - 9 ) x 8 )
(( 9 - 7 ) x 8 ) + 8
8 - ( 8 x ( 7 - 9 ))
( 8 x ( 9 - 7 )) + 8
Execution time: 0.000 ms
```

Gambar 20 contoh *test case* 4

Contoh 5 (2 J 2 3)

```
Input your choice: 2

===== RANDOM =====
Generating random cards.....
Random Cards: 2 J 2 3

===== SOLUTIONS =====
16 solutions found!
(( 3 + 11 ) - 2 ) x 2
( 3 + ( 11 - 2 ) ) x 2
(( 11 + 3 ) - 2 ) x 2
( 11 + ( 3 - 2 ) ) x 2
(( 3 - 2 ) + 11 ) x 2
(( 11 - 2 ) + 3 ) x 2
( 3 - ( 2 - 11 ) ) x 2
( 11 - ( 2 - 3 ) ) x 2
2 x (( 3 + 11 ) - 2 )
2 x ( 3 + ( 11 - 2 ) )
2 x (( 11 + 3 ) - 2 )
2 x ( 11 + ( 3 - 2 ) )
2 x (( 3 - 2 ) + 11 )
2 x (( 11 - 2 ) + 3 )
2 x ( 3 - ( 2 - 11 ) )
2 x ( 11 - ( 2 - 3 ) )
Execution time: 1999.000 ms
```

Gambar 21 contoh *test case* 5

Contoh 6 (Q Q A 7)

```
===== MENU =====
1. Input Cards
2. Random Cards
3. Exit
Input your choice: 2

===== RANDOM =====
Generating random cards.....
Random Cards: Q Q A 7

===== SOLUTIONS =====
4 solutions found!
( 12 x 12 ) / ( 7 - 1 )
12 x ( 12 / ( 7 - 1 ) )
( 12 / ( 7 - 1 ) ) x 12
12 / (( 7 - 1 ) / 12 )
Execution time: 0.000 ms
```

Gambar 22 contoh *test case* 6

Contoh 7 (9 K 5 Q)

```
34 Exit
Input your choice: 2

===== RANDOM =====
Generating random cards.....
Random Cards: 9 K 5 Q

===== SOLUTIONS =====
20 solutions found!
( 5 + 13 ) x ( 12 / 9 )
(( 5 + 13 ) x 12 ) / 9
( 13 + 5 ) x ( 12 / 9 )
(( 13 + 5 ) x 12 ) / 9
(( 5 + 13 ) / 9 ) x 12
(( 13 + 5 ) / 9 ) x 12
( 5 + 13 ) / ( 9 / 12 )
( 13 + 5 ) / ( 9 / 12 )
( 5 - 13 ) x ( 9 - 12 )
( 9 - 12 ) x ( 5 - 13 )
( 12 - 9 ) x ( 13 - 5 )
( 13 - 5 ) x ( 12 - 9 )
( 12 x ( 5 + 13 )) / 9
12 x (( 5 + 13 ) / 9 )
( 12 x ( 13 + 5 )) / 9
12 x (( 13 + 5 ) / 9 )
( 12 / 9 ) x ( 5 + 13 )
( 12 / 9 ) x ( 13 + 5 )
12 / ( 9 / ( 5 + 13 ))
12 / ( 9 / ( 13 + 5 ))
Execution time: 2002.000 ms
```

Gambar 23 contoh *test case* 7

Contoh 8 (4 K 10 8)

```
===== RANDOM =====
Generating random cards.....
Random Cards: 4 K 10 8

===== SOLUTIONS =====
No Solution found
Do you want to play again?(y/n)
Choice: █
```

Gambar 24 contoh *test case* 8

Contoh 9 (6 6 6 6)

```
===== INPUT =====
Input 4 Cards each separated by space
valid cards: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
(example: A 2 3 Q)
Input: 6 6 6 6

===== SOLUTIONS =====
7 solutions found!
( 6 + 6 ) + ( 6 + 6 )
(( 6 + 6 ) + 6 ) + 6
( 6 + ( 6 + 6 )) + 6
6 + (( 6 + 6 ) + 6 )
6 + ( 6 + ( 6 + 6 ))
( 6 x 6 ) - ( 6 + 6 )
(( 6 x 6 ) - 6 ) - 6
Execution time: 0.000 ms
```

Gambar 24 contoh *test case* 9

Contoh 10 (8 5 10 J)

```
===== RANDOM =====
Generating random cards.....
Random Cards: 8 5 10 J

===== SOLUTIONS =====
90 solutions found!
( 8 + 10 ) + ( 11 - 5 )
(( 8 + 10 ) + 11 ) - 5
( 8 + ( 10 + 11 )) - 5
8 + (( 10 + 11 ) - 5 )
8 + ( 10 + ( 11 - 5 ))
( 8 + 11 ) + ( 10 - 5 )
(( 8 + 11 ) + 10 ) - 5
( 8 + ( 11 + 10 )) - 5
8 + (( 11 + 10 ) - 5 )
8 + ( 11 + ( 10 - 5 ))
( 10 + 8 ) + ( 11 - 5 )
(( 10 + 8 ) + 11 ) - 5
( 10 + ( 8 + 11 )) - 5
10 + (( 8 + 11 ) - 5 )
10 + ( 8 + ( 11 - 5 ))
( 10 + 11 ) + ( 8 - 5 )
(( 10 + 11 ) + 8 ) - 5
( 10 + ( 11 + 8 )) - 5
10 + (( 11 + 8 ) - 5 )
10 + ( 11 + ( 8 - 5 ))
( 11 + 8 ) + ( 10 - 5 )
(( 11 + 8 ) + 10 ) - 5
( 11 + ( 8 + 10 )) - 5
11 + (( 8 + 10 ) - 5 )
11 + ( 8 + ( 10 - 5 ))
( 11 + 10 ) + ( 8 - 5 )
(( 11 + 10 ) + 8 ) - 5
( 11 + ( 10 + 8 )) - 5
11 + (( 10 + 8 ) - 5 )
11 + ( 10 + ( 8 - 5 ))
(( 8 + 10 ) - 5 ) + 11
( 8 + ( 10 - 5 )) + 11
( 10 + 8 ) - ( 5 - 11 )
10 + ( 8 - ( 5 - 11 ))
( 10 + 11 ) - ( 5 - 8 )
10 + ( 11 - ( 5 - 8 ))
( 11 + 8 ) - ( 5 - 10 )
11 + ( 8 - ( 5 - 10 ))
( 11 + 10 ) - ( 5 - 8 )
11 + ( 10 - ( 5 - 8 ))
( 8 - 5 ) + ( 10 + 11 )
(( 8 - 5 ) + 10 ) + 11
( 8 - 5 ) + ( 11 + 10 )
(( 8 - 5 ) + 11 ) + 10
( 10 - 5 ) + ( 8 + 11 )
(( 10 - 5 ) + 8 ) + 11
( 10 - 5 ) + ( 11 + 8 )
(( 10 - 5 ) + 11 ) + 8
( 11 - 5 ) + ( 8 + 10 )
(( 11 - 5 ) + 8 ) + 10
( 11 - 5 ) + ( 10 + 8 )
(( 11 - 5 ) + 10 ) + 8
( 8 - ( 5 - 10 )) + 11
8 - ( 5 - ( 10 + 11 ))
( 8 - ( 5 - 11 )) + 10
8 - ( 5 - ( 11 + 10 ))
( 10 - ( 5 - 8 )) + 11
10 - ( 5 - ( 8 + 11 ))
( 10 - ( 5 - 11 )) + 8
10 - ( 5 - ( 11 + 8 ))
( 11 - ( 5 - 8 )) + 10
11 - ( 5 - ( 8 + 10 ))
( 11 - ( 5 - 10 )) + 8
11 - ( 5 - ( 10 + 8 ))
8 - (( 5 - 10 ) - 11 )
8 - (( 5 - 11 ) - 10 )
10 - (( 5 - 8 ) - 11 )
10 - (( 5 - 11 ) - 8 )
11 - (( 5 - 8 ) - 10 )
11 - (( 5 - 10 ) - 8 )
Execution time: 1997.000 ms
```

Gambar 25 contoh *test case* 10

BAB 5

PENUTUP

5.1 Kesimpulan

Melalui tugas besar ini, saya menjadi belajar banyak hal terkait library dan bahasa pemrograman C++. Algoritma *brute force* dapat menyelesaikan hampir segala macam persoalan algoritma, namun tidak efisien.

5.2 Saran

- jangan mengerjakan h-1

5.3 Link Repository

Link repository untuk tugas kecil 1 mata kuliah IF2211 Strategi Algoritma adalah sebagai berikut

Link : https://github.com/AJason36/Tucil1_13521100

5.4 Tabel Checkpoint Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil <i>running</i>	✓	
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	✓	

DAFTAR REFERENSI

[Algoritma Brute Force](#)

[Solving '24' Card Game Using Combinatorics](#)

[Vector in C++ STL - GeeksforGeeks](#)