LAPORAN TUGAS KECIL

IF2211 STRATEGI ALGORITMA

Penyelesaian Permainan Kartu 24

dengan Algoritma *Brute Force*



Disusun oleh

Akhmad Setiawan - 13521164 - K2

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Sekolah Teknik Elektro dan Informatika ITB** | Nomor Dokumen | | Halaman |
| *IF2211-TK1-13521164-K2* | | *26* |
| *Revisi* | *1* | *25-01-2023* |

Daftar Isi

[1 Deskripsi Algoritma *Brute Force* yang Diimplementasikan 3](#_Toc125538815)

[*2 Source Program* 4](#_Toc125538816)

[3 Tampilan Program dan Contoh Input/Output 18](#_Toc125538817)

[4 Link Repository 26](#_Toc125538818)

[5 Ceklis 26](#_Toc125538819)

# Deskripsi Algoritma *Brute Force* yang Diimplementasikan

Permainan kartu 24 adalah permainan di mana pemain menghitung 4 buah angka yang direpresentasikan dengan kartu remi. Kartu remi yang digunakan ialah dari As, yang direpresentasikan sebagai angka 1 hingga kartu King yang direpresentasikan dengan angka 13. Pemain diharuskan mencari operasi aritmatika yang menghubungkan keempat angka tersebut sehingga terbentuk sebuah angka 24. Operasi aritmatika yang dapat digunakan dalam permainan tersebut adalah tambah (+), kurang (-), kali (x), dan bagi (/). Angka pada setiap kartu harus digunakan sekali dan hanya boleh digunakan sekali dalam proses operasi perhitungan. Penyelesaian permainan ini dapat menggunakan beberapa pendekatan, salah satunya dengan algoritma *brute force*.

Algoritma *brute force* yang ialah metode penyelesaian persoalan dengan secara lempang (straightforward). Artinya, jika menyelesaikan permainan kartu 24 dengan metode ini, kita akan mencari semua kemungkinan jawaban dari semua kombinasi operasi dan angka yang ada, dan mengumpulkan semua kemungkinan yang hasil nilainya adalah 24. Berikut adalah kemungkinan susunan angka-angka dalam setiap operasi pada 4 kartu tadi.

* Dalam permainan 24, jumlah angka yang dapat digunakan dalam operasi haruslah 4. Maka dari itu, dengan menggunakan permutasi, terdapat maksimal 4! = 24 kemungkinan susunan angka yang mungkin. Namun, ada kemungkinan terdapat duplikat antara keempat angka tersebut, sehingga perlu diperhatikan terkait permutasi dengan unsur yang sama.
* Terdapat 4 buah operator yang dapat digunakan dalam permainan 24, yaitu tambah, kurang, kali, dan bagi. Karena terdapat 4 buah angka dalam permainan ini, maka terdapat 3 tempat di mana operator harus dimasukkan untuk menghasilkan operasi aritmatika, yaitu 1 di antara 2 angka pertama, 1 di antara 2 angka tengah, dan 1 di antara 2 angka terakhir. Karena terdapat 4 operator dan 3 buah tempat yang tersedia, dengan aturan perkalian, maka kita mendapatkan maksimal 4x4x4 = 64 buah kemungkinan susunan operator yang mungkin.
* Penempatan tanda kurung dalam operasi aritmatika berfungsi untuk menentukan urutan

operasi mana yang dikerjakan terlebih dahulu. Dari penempatan tanda kurung tersebut,

terdapat 5 buah jenis solusi yang ada.

1. (a + b) + (c + d)

2. ((a + b) + c) + d

3. (a + (b + c)) + d

4. a + ((b + c) + d)

5. a + (b + (c + d))

Dalam implementasinya ke dalam kode program, pertama-tama yang saya lakukan ialah dengan menelusuri semua kemungkinan angka yang didapat. Terdapat beberapa variasi, yakni:

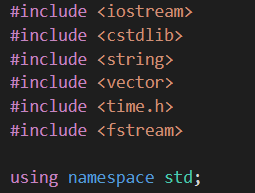
1. Jika semua kartu memiliki angka yang berbeda
2. Jika ada sepasang kartu dengan angka yang sama
3. Jika ada 2 pasang kartu dengan angka yang sama
4. Jika ada 3 kartu dengan angka yang sama, dan
5. Jika semua kartu angkanya sama.

Setiap variasi ini dicari dengan metode *brute force* secara manual satu-per-satu. Masing-masing dari variasi tersebut kemudian dicari lagi secara menyeluruh terkait variasi tanda kurung yang telah disebutkan di atas. Dalam setiap variasi tanda kurung pula, dicari semua kemungkinan dengan mengakses semua variasi operator yang tersedia. Dengan metode ini lah, diperoleh hasil semua perhitungan dari semua variasi angka kartu, variasi tanda kurung, dan variasi operator yang tersedia. Namun, yang hanya kita pedulikan ialah variasi yang menghasilkan angka 24 saja. Untuk fungsi, prosedur, dan variabel yang digunakan dapat dilihat pada sumber kode program yang akan ditampilkan di bawah.

# *Source Program*

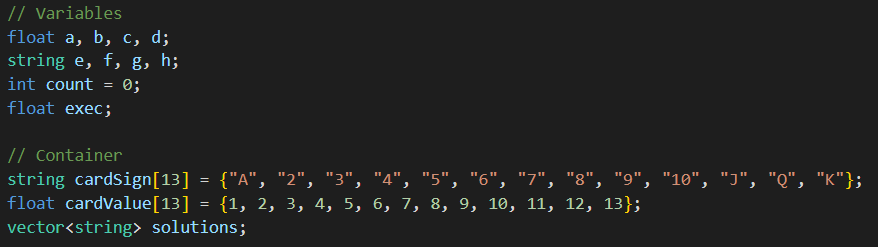
Program ini ditulis dengan bahasa C++ dengan bantuan beberapa *library*:

1. iostream, untuk *handling* *input* dan *output*
2. cstdlib, untuk bantuan men-*generate random number*
3. string, untuk *input* dalam bentuk *string*
4. vector, untuk menyimpan solusi yang banyaknya tidak tentu (dinamis)
5. time.h, untuk menghitung waktu eksekusi dan *generate random number*
6. fstream, untuk *handling save output to file*

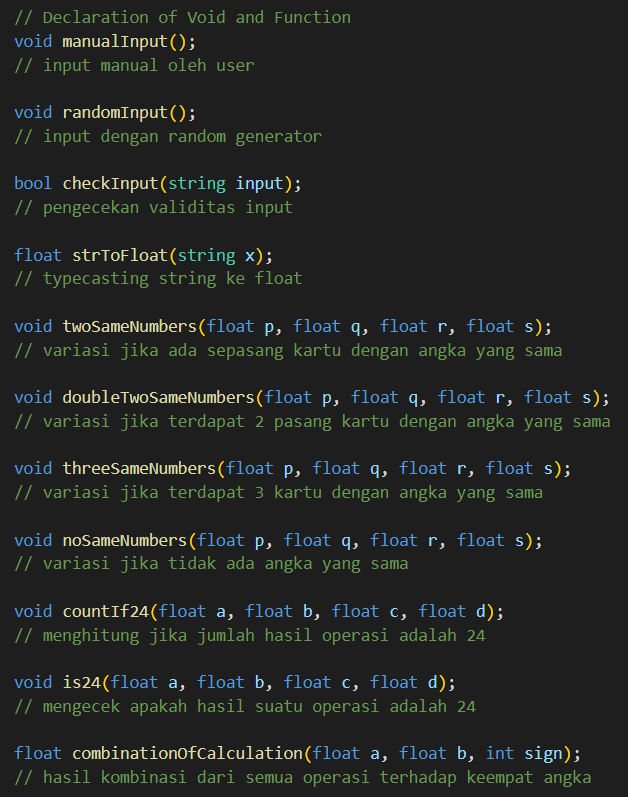


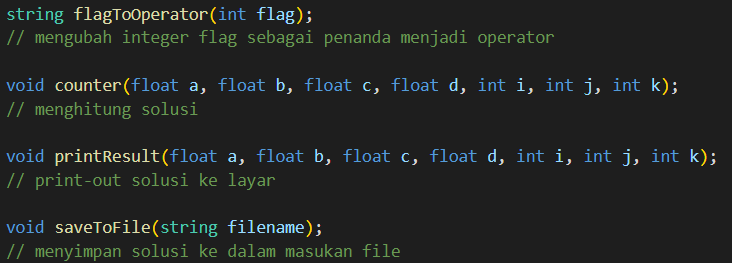
**Gambar 2.1** Daftar *Library yang digunakan*

Adapun beberapa deklarasi variabel, *container*, void, dan function dapat dilihat pada gambar berikut.



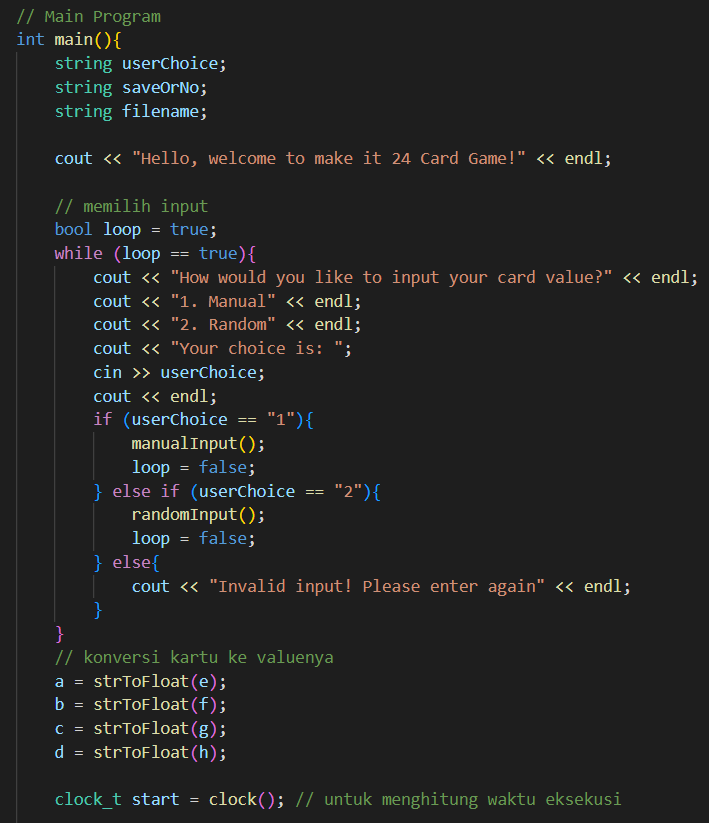
**Gambar 2.2** Deklarasi Variabel Global dan Container

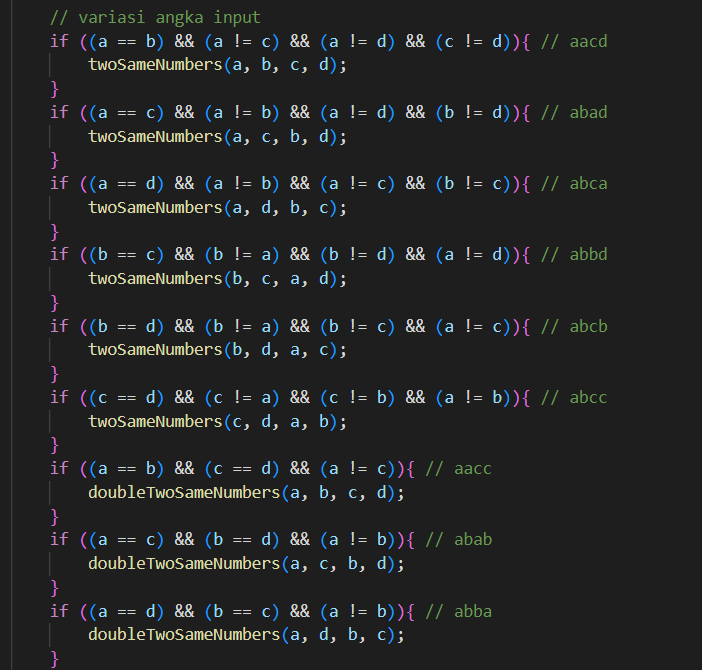


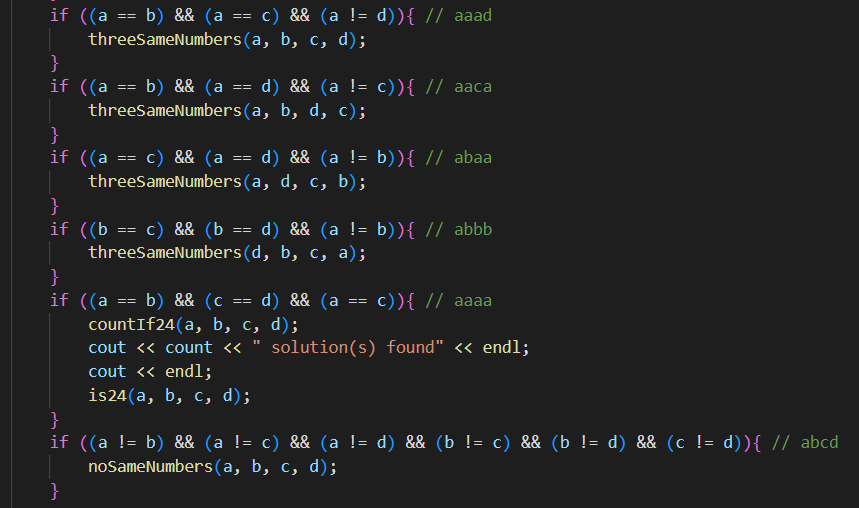


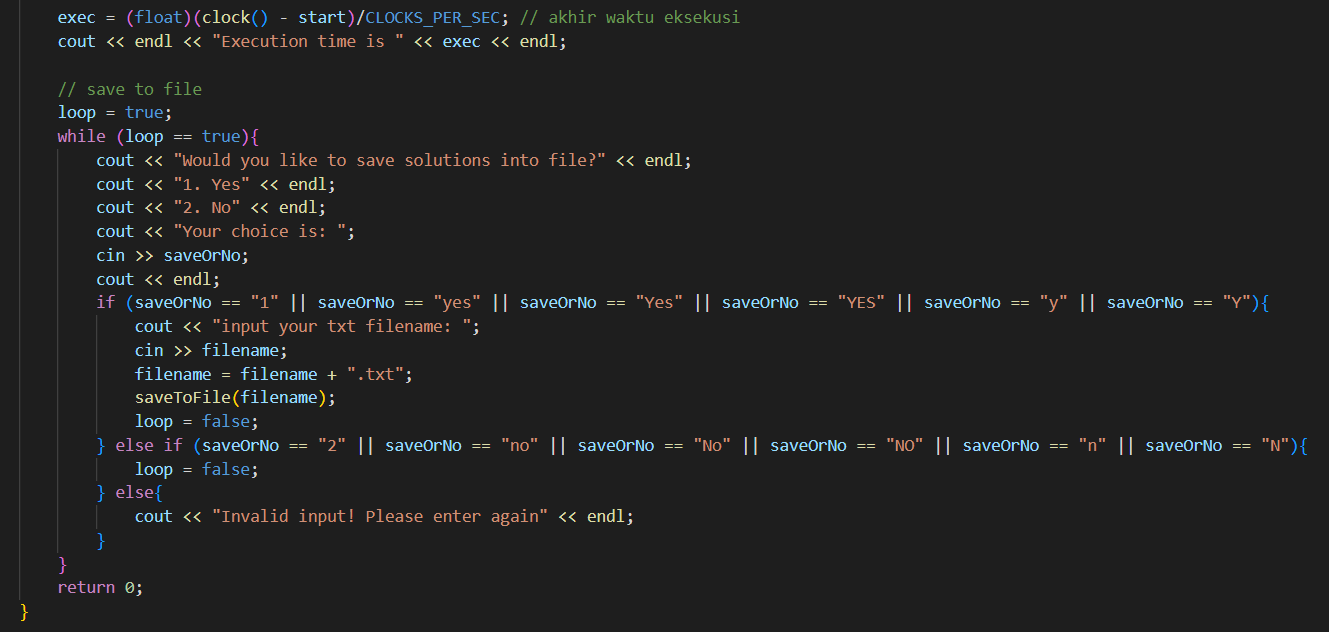
**Gambar 2.3** Deklarasi Void dan Function

Setelah deklarasi void dan function, selanjutnya terdapat implementasi program utama yang dijalankan untuk mendapatkan solusi. Implementasinya dapat dilihat seperti gambar berikut.



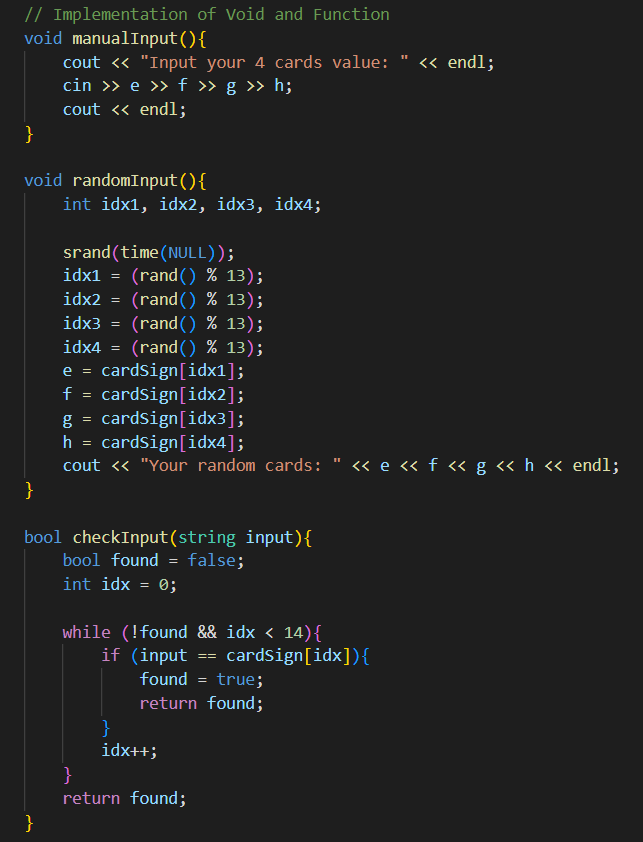


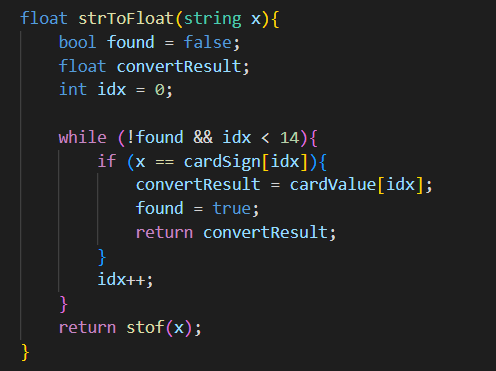


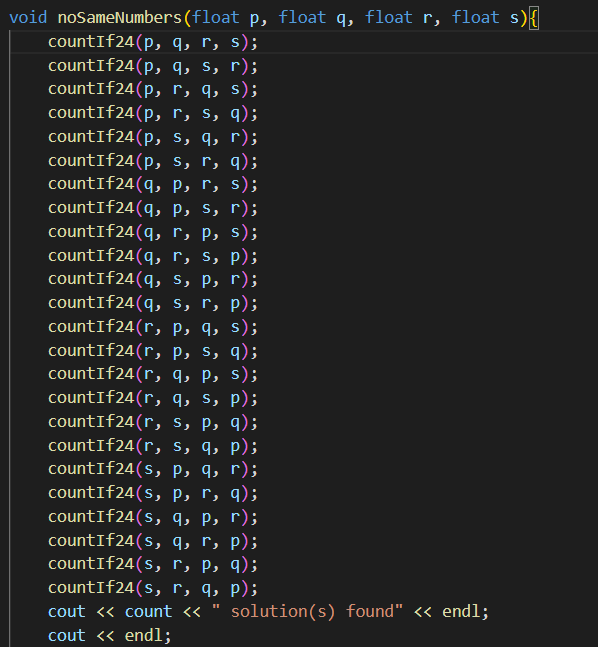


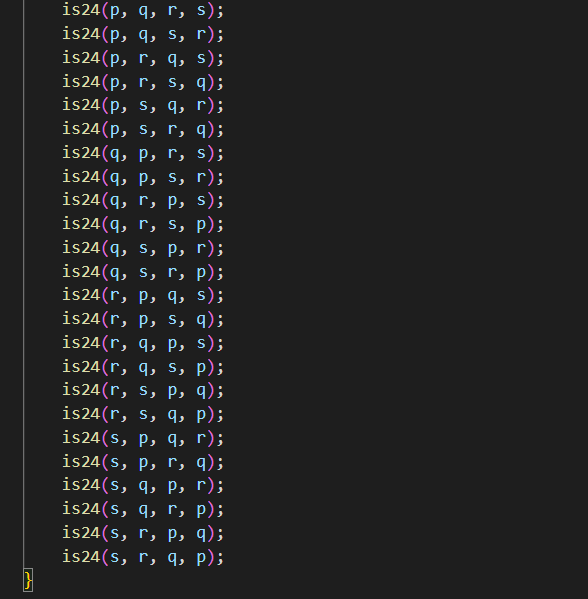
**Gambar 2.4** Implementasi *Main Program*

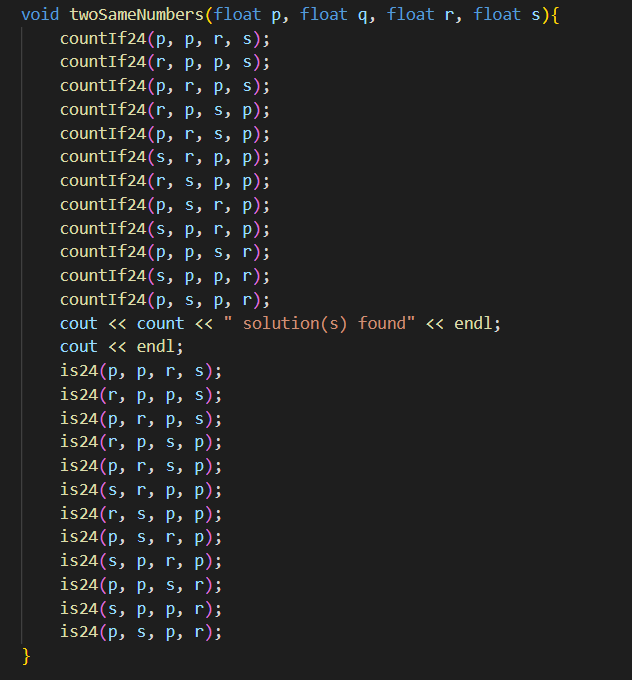
Setelah implementasi *main program*, kode dilanjutkan dengan implementasi void dan function yang sudah dideklarasikan di awal tadi, seperti gambar berikut.

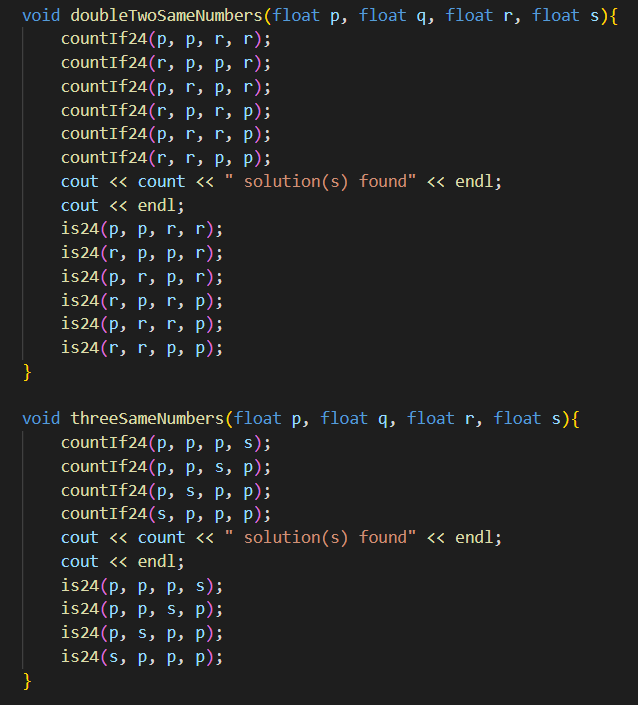


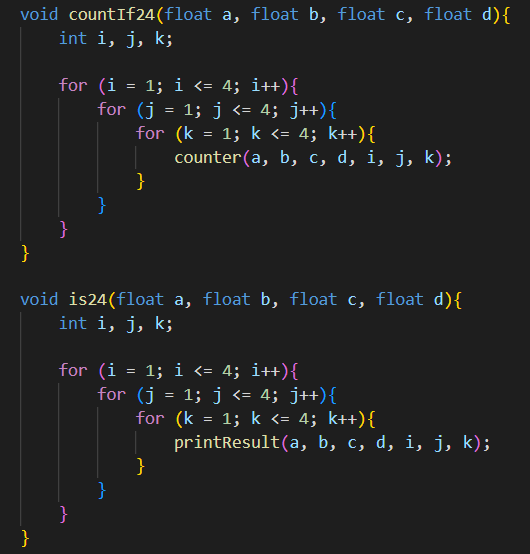




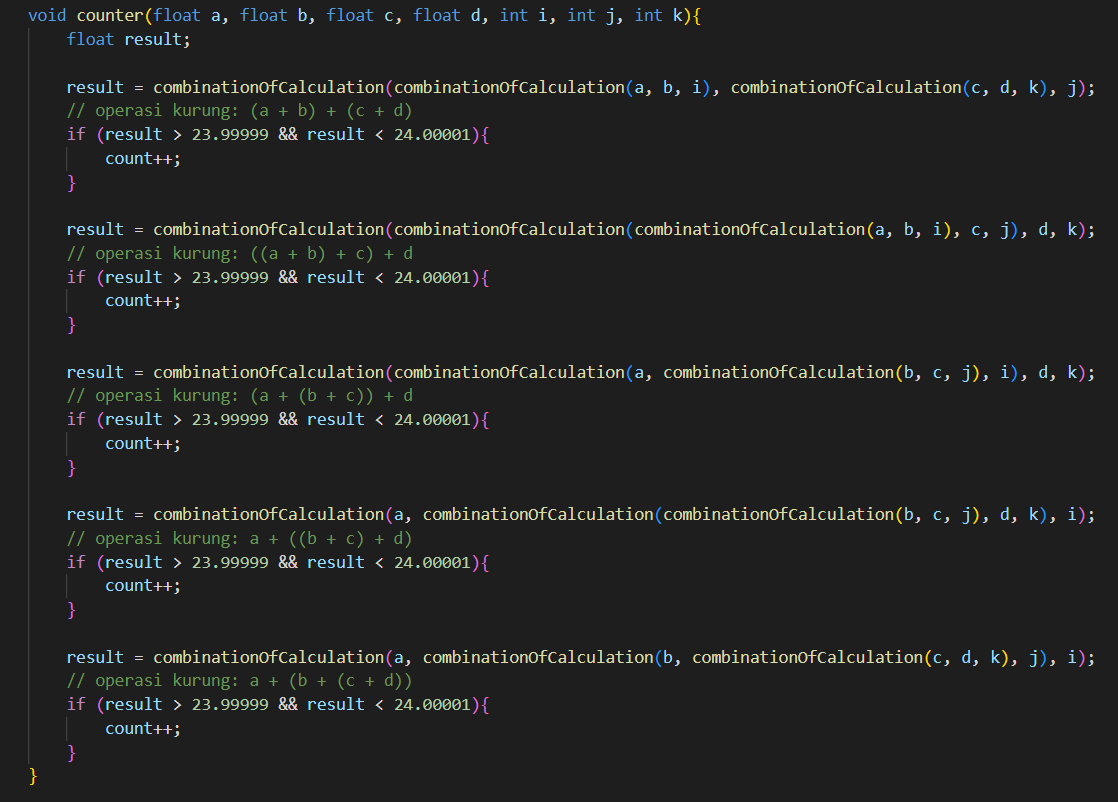


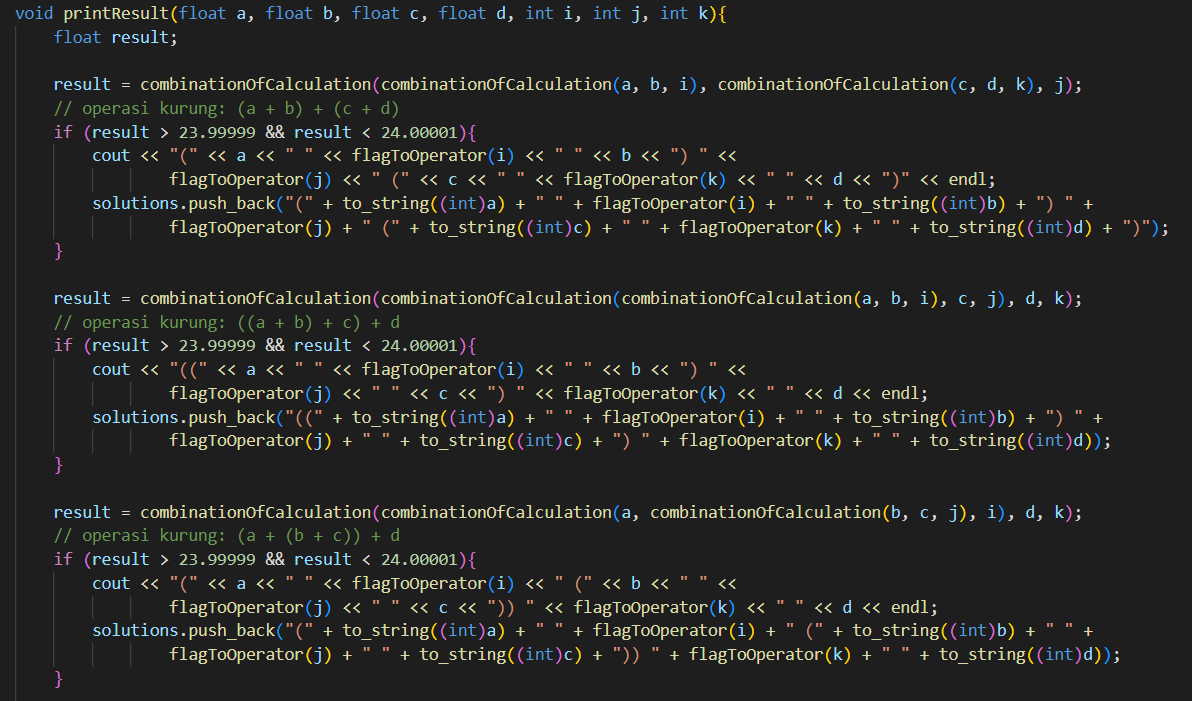


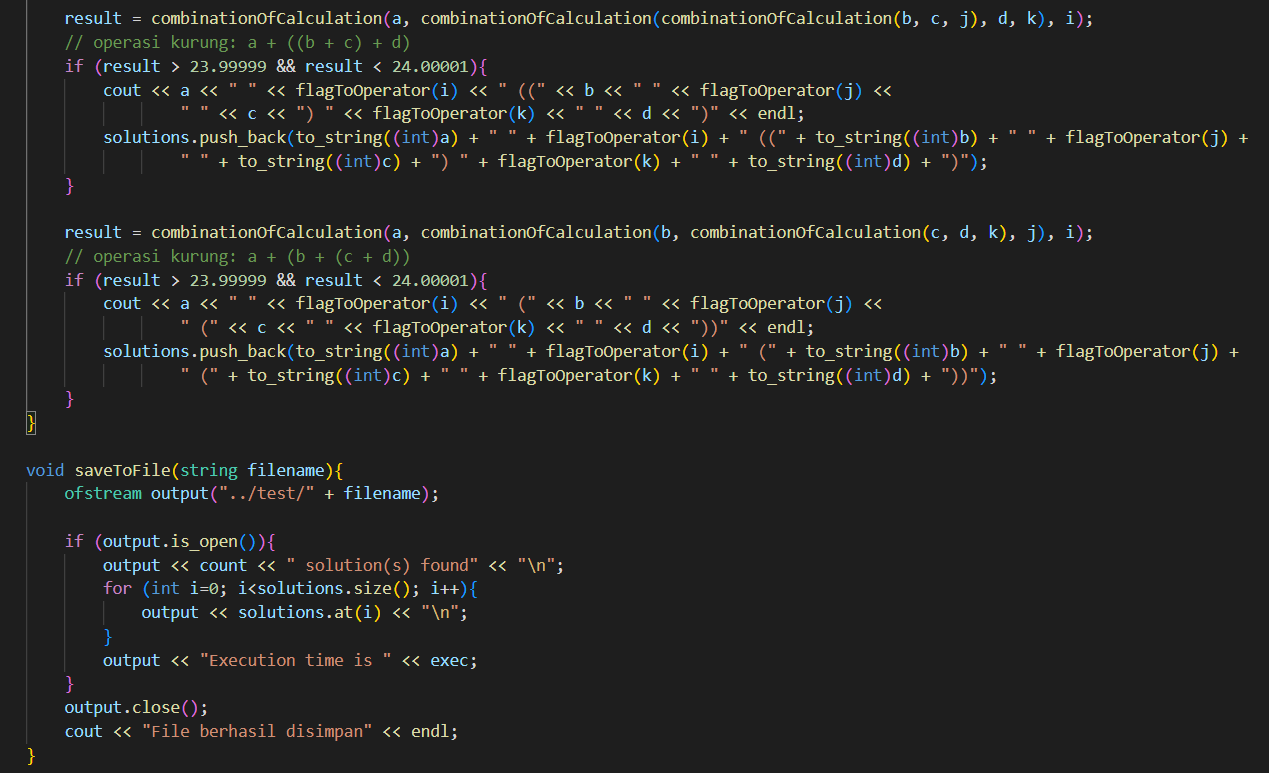








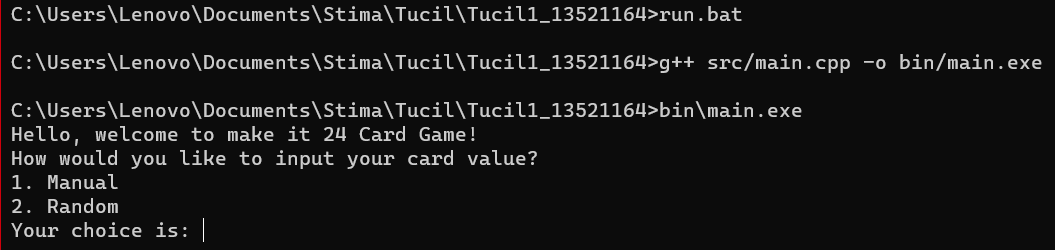




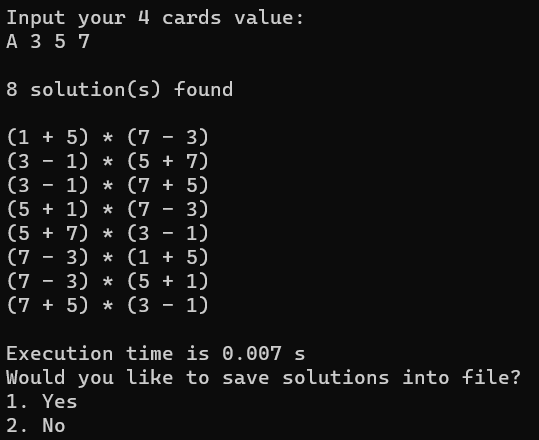
**Gambar 2.5** Implementasi Void dan Function

# Tampilan Program dan Contoh Input/Output

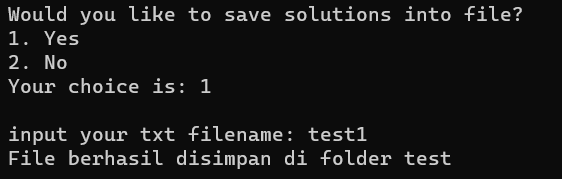
Program dibuat dengan CLI (*Command Line Interface*). Pengguna dapat menjalankan program run.bat setelah berada di folder directory Tucil1\_13521164, untuk proses compiling dan menjalankan program. File hasil compile akan tersimpan di folder bin. Program dijalankan dengan memasukkan input sesuai dengan instruksi yang ada pada terminal.



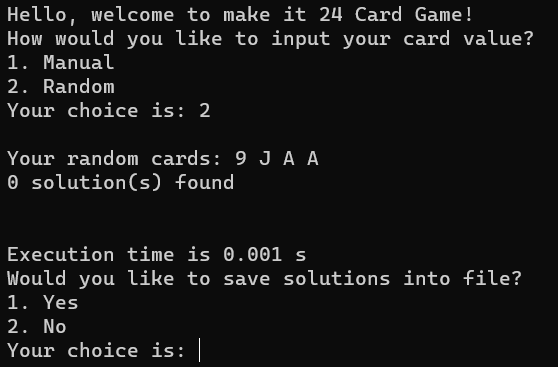
**Gambar 3.1** Tampilan awal ketika program dijalankan



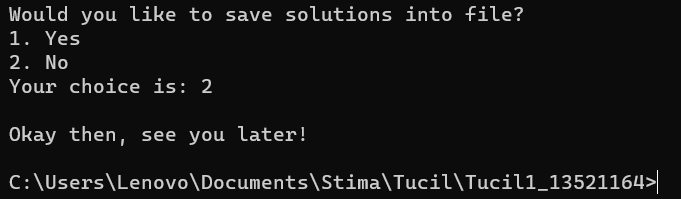
Gambar 3.2 Pengguna Menginput Manual Nilai Kartu



Gambar 3.3 Pengguna Menyimpan Output ke Dalam File test1.txt

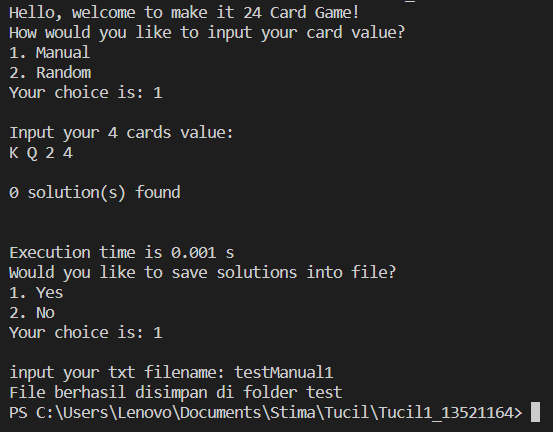


Gambar 3.4 Pengguna Menggunakan Generate Random Card

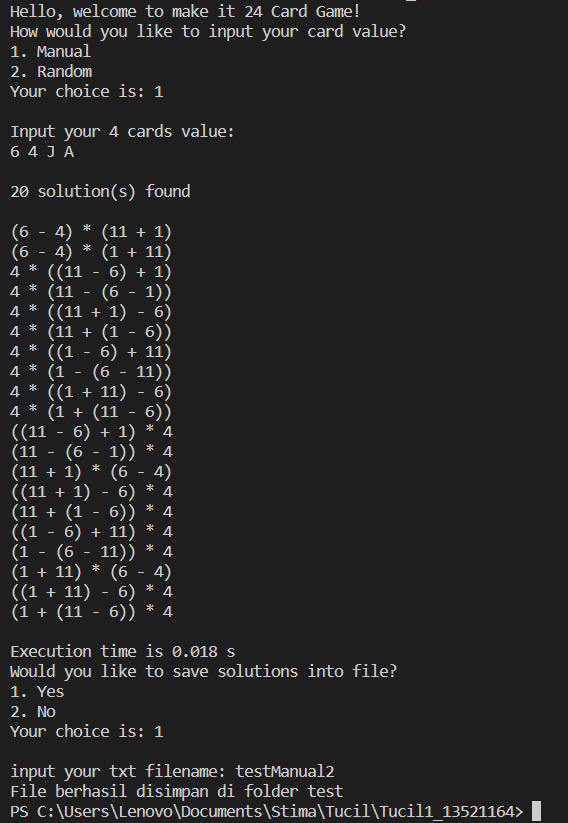


**Gambar 3.5** Pengguna Tidak Ingin Menyimpan Output ke Dalam File

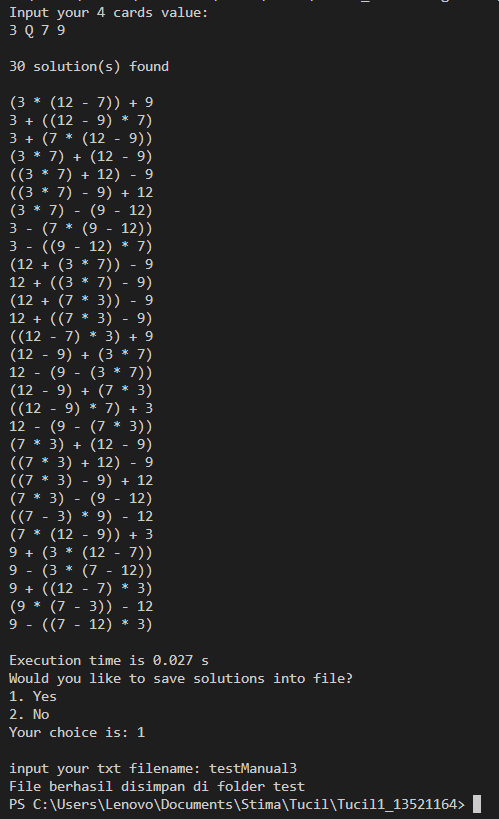
Berikut adalah 3 contoh input secara manual dan 3 contoh input secara random beserta outputnya.



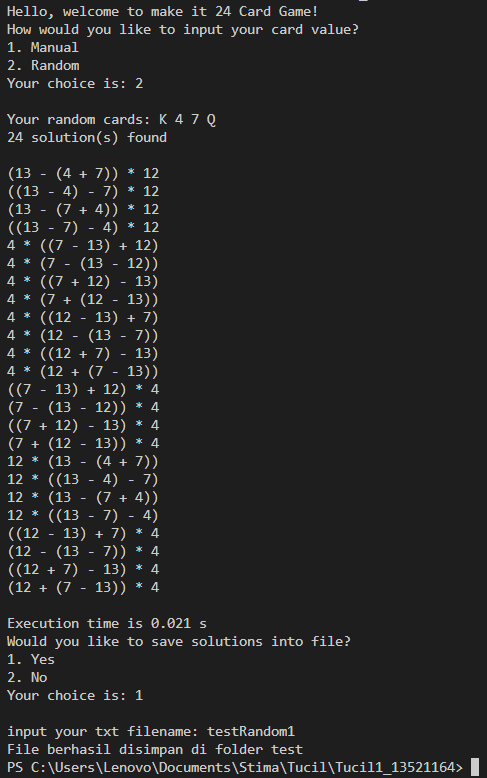
**Gambar 3.6** Input Manual Kartu K Q 2 4



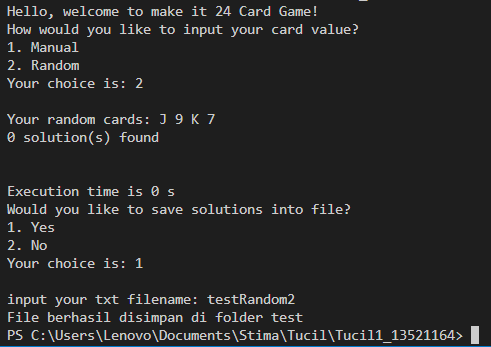
**Gambar 3.7** Input Manual Kartu 6 4 J A



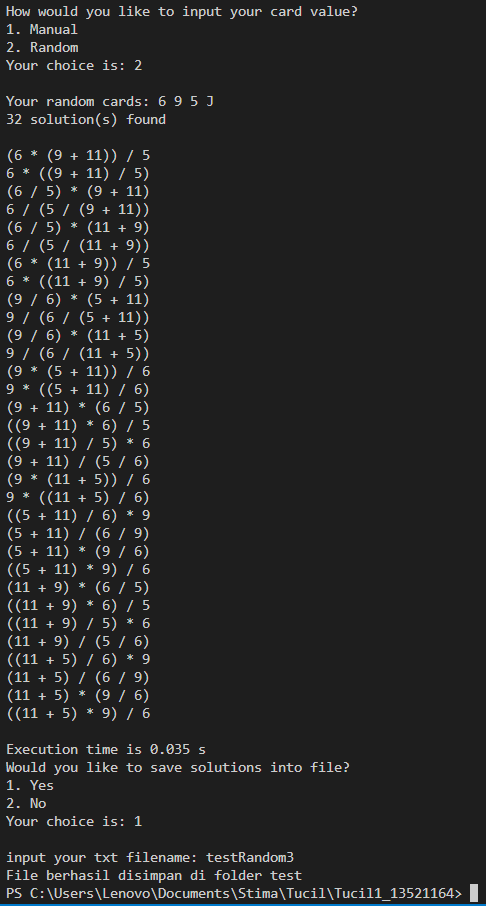
**Gambar 3.8** Input Manual 3 Q 7 9



**Gambar 3.9** Input Random dan Memperoleh Kartu K 4 7 Q



**Gambar 3.10** Input Random dan Memperoleh Kartu J 9 K 7



**Gambar 3.11** Input Random dan Memperoleh Kartu 6 9 5 J

# Link Repository

Repository Github dapat diakses pada: <https://github.com/blixa-rd/Tucil1_13521164>

# Ceklis

|  |  |  |  |
| --- | --- | --- | --- |
| No | 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 (mencapai 24) |  |  |
| 5 | Program dapat menyimpan solusi dalam file teks |  |  |