

TUGAS
STRATEGI ALGORITMA
KELAS RB

Kelompok:

Muhammad Hadi Arsa	120140150
Ryan Ernanda	120140154

Program SUDOKU

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI SUMATERA
2022

A. SOURCE CODE

```
/*  
Ryan Ernanda 120140154  
Muhammad Hadi Arsa 120140150  
*/  
  
#include <iostream>  
#include <fstream>  
  
#define N 9  
  
using namespace std;  
  
int grid[N][N];  
  
bool AdadiCol(int col, int num){  
    for (int row = 0; row < N; row++){  
        if (grid[row][col] == num)  
            return true;  
    }  
    return false;  
}  
  
bool AdadiRow(int row, int num){  
    for (int col = 0; col < N; col++){  
        if (grid[row][col] == num)  
            return true;  
    }  
    return false;  
}  
  
bool AdadiBox(int mulairow, int mulaicol, int num){  
    for (int row = 0; row < 3; row++){  
        for (int col = 0; col < 3; col++){  
            if (grid[row+mulairow][col+mulaicol] == num)  
                return true;  
        }  
    }  
    return false;  
}
```

```

}

void SudokuGrid(){

    for (int row = 0; row < N; row++){

        for (int col = 0; col < N; col++){

            if(col == 3 || col == 6)

                cout << " | ";

            cout << grid[row][col] <<" ";

        }

        if(row == 2 || row == 5){

            cout << endl;

            for(int i = 1; i<N; i++){

                cout << "---";

            }

            cout << endl;

        }

    }

}

bool TemukanTempatKosong(int &row, int &col){

    for (row = 0; row < N; row++)

        for (col = 0; col < N; col++)

            if (grid[row][col] == 0)

                return true;

    return false;

}

bool TempatyangBenar(int row, int col, int num){

    return !AdadiRow(row, num) && !AdadiCol(col, num) && !AdadiBox(row - row%3, col - col%3, num);

}

bool SudokuSelesai(){

    int row, col;

```

```

    if (!TemukanTempatKosong(row, col))

        return true;

    for (int num = 1; num <= 9; num++){

        if (TempatyangBenar(row, col, num)){

            grid[row][col] = num;

            if (SudokuSelesai())

                return true;

            grid[row][col] = 0;

        }

    }

    return false;
}

int main() {

    grid[N][N];

    fstream inputfile;

    fstream outputfile;

    cout << "Masukkan Nama File Sudoku: ";

    string fileName, outputFileName;

    cin >> fileName;

    inputfile.open(fileName, ios::in);

    cout << "Masukan Nama File Untuk Menyimpan Hasil Sudoku: ";

    cin >> outputFileName;

    outputfile.open(outputFileName, ios::out);

    if (!inputfile.is_open() || !outputfile.is_open())

    {

        cout << "Error membuka file";

        return 0;
    }
}

```

```

}

for (int row = 0; row < N; row++)
    for (int col = 0; col < N; col++)
        inputfile >> grid[row][col];

if (SudokuSelesai())
    SudokuGrid();

else
    cout << "Tidak ada solusi";

inputfile.close();

outputfile << "Hasil sudoku yang diselesaikan: " << endl;

for (int row = 0; row < N; row++){
    for (int col = 0; col < N; col++){
        if(col == 3 || col == 6)
            outputfile << " | ";

        outputfile << grid[row][col] <<" ";
    }

    if(row == 2 || row == 5){
        outputfile << endl;

        for(int i = 1; i<N; i++)
            outputfile << "---";
    }

    outputfile << endl;
}
}

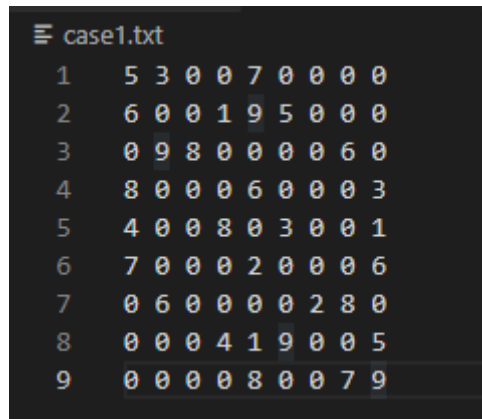
```

B. INPUT PROGRAM

B.1 INPUT 1

buat file yang berisi permasalahan sudoku dengan extensi contohnya “case1.txt” file tanpa spasi, untuk isinya mengikuti format di gambar

1. permasalahan yang tidak ada nilainya maka berikan nilai tersebut dengan nilai ‘0’,
2. gunakan spasi untuk memisahkan antar nilai,
3. isi dari baris dan kolom mengikuti pada permainan sudoku 9x9



```
case1.txt
1  5 3 0 0 7 0 0 0 0
2  6 0 0 1 9 5 0 0 0
3  0 9 8 0 0 0 0 6 0
4  8 0 0 0 6 0 0 0 3
5  4 0 0 8 0 3 0 0 1
6  7 0 0 0 2 0 0 0 6
7  0 6 0 0 0 0 2 8 0
8  0 0 0 4 1 9 0 0 5
9  0 0 0 0 8 0 0 7 9
```

jika file sudah dibuat maka lanjut menjalankan programnya, di dalam program meminta 2 inputan yaitu :

1. Masukkan nama file sudoku yang sudah di buat contohnya case1.txt
2. Masukkan nama file untuk menyimpan hasil dari output yang terselesaikan, untuk file ini jika belum ada maka program membuat otomatis filenya sesuai nama dan extensi yang kalian berikan, disini saya memberi namanya ‘hasilcase1.txt’

B.2 INPUT 2

buat file yang berisi permasalahan sudoku dengan extensi contohnya “case2.txt” file tanpa spasi, untuk isinya mengikuti format di gambar

4. permasalahan yang tidak ada nilainya maka berikan nilai tersebut dengan nilai ‘0’,
5. gunakan spasi untuk memisahkan antar nilai,
6. isi dari baris dan kolom mengikuti pada permainan sudoku 9x9

```

≡ case2.txt
1  0 7 0 2 3 8 0 0 0
2  0 0 0 7 4 0 8 0 9
3  0 6 8 1 0 9 0 0 2
4  0 3 5 4 0 0 0 0 8
5  6 0 7 8 0 2 5 0 1
6  8 0 0 0 0 5 7 6 0
7  2 0 0 6 0 3 1 9 0
8  7 0 9 0 2 1 0 0 0
9  0 0 0 9 7 4 0 8 0

```

jika file sudah dibuat maka lanjut menjalankan programnya, di dalam program meminta 2 inputan yaitu :

3. Masukan nama file sudoku yang sudah di buat contohnya case2.txt
4. Masukan nama file untuk menyimpan hasil dari output yang terselesaikan, untuk file ini jika belum ada maka program membuat otomatis filenya sesuai nama dan ekstensi yang kalian berikan, disini saya memberi namanya 'hasilcase2.txt'

B.3 INPUT 3

buat file yang berisi permasalahan sudoku dengan ekstensi contohnya "case3.txt" file tanpa spasi, untuk isinya mengikuti format di gambar

7. permasalahan yang tidak ada nilainya maka berikan nilai tersebut dengan nilai '0',
8. gunakan spasi untuk memisahkan antar nilai,
9. isi dari baris dan kolom mengikuti pada permainan sudoku 9x9

```

≡ case3.txt
1  0 7 0 0 4 5 0 0 0
2  5 0 0 8 7 0 0 0 3
3  0 0 4 1 0 0 9 0 0
4  0 0 0 5 0 2 0 6 8
5  0 0 2 6 0 0 0 3 0
6  0 0 1 0 3 0 2 0 0
7  0 0 5 0 1 8 0 9 2
8  1 4 0 0 0 6 0 0 7
9  2 9 0 0 0 0 0 0 1

```

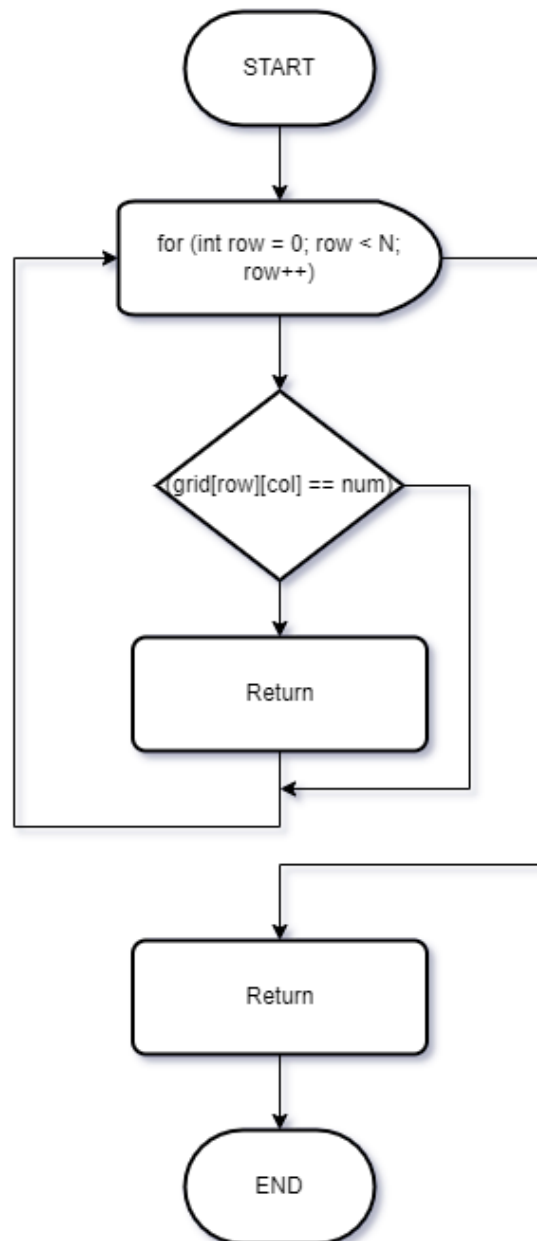
jika file sudah dibuat maka lanjut menjalankan programnya, di dalam program meminta 2 inputan yaitu :

5. Masukan nama file sudoku yang sudah di buat contohnya case3.txt
6. Masukan nama file untuk menyimpan hasil dari output yang terselesaikan, untuk file ini jika belum ada maka program membuat otomatis filenya sesuai nama dan ekstensi yang kalian berikan, disini saya memberi namanya 'hasilcase3.txt'

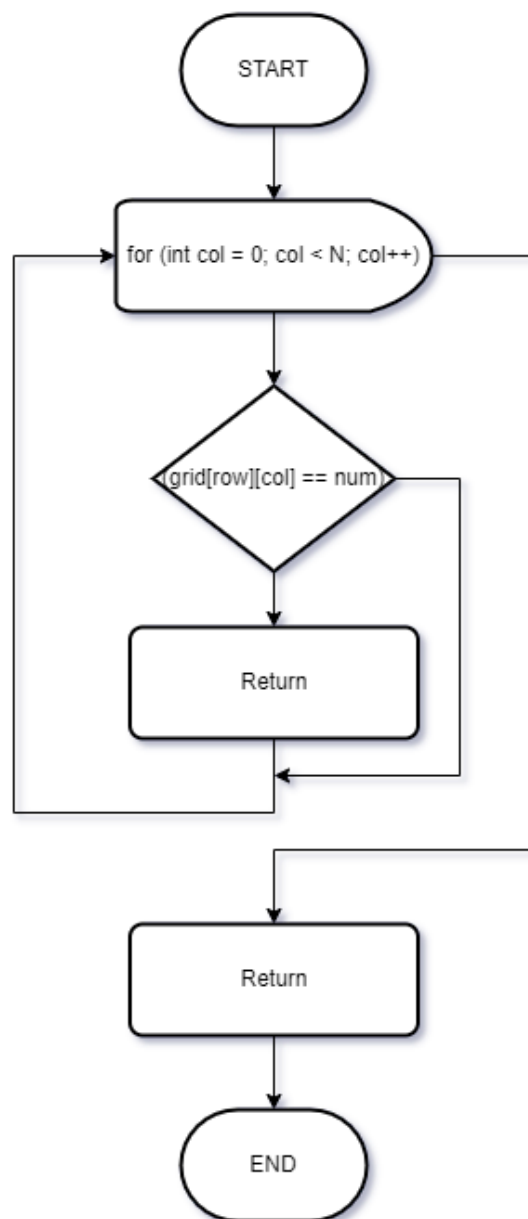
C. PROSES

1. Diagram Alir Code Program

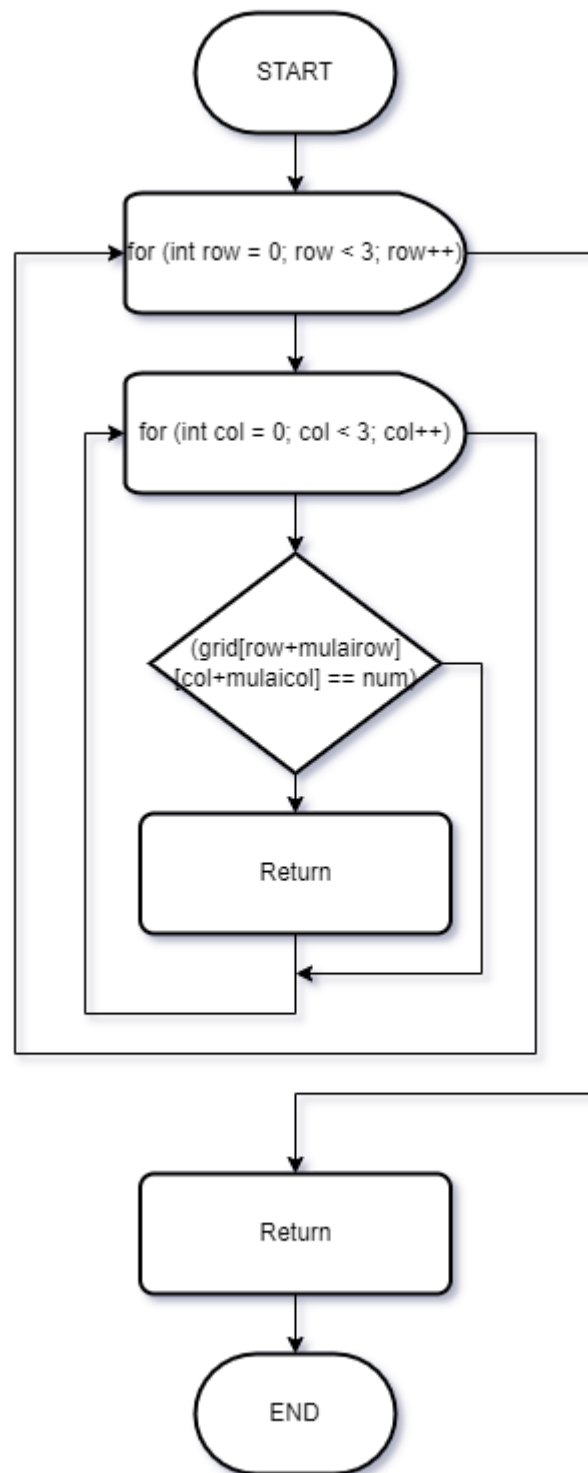
a. boolAdadiCol



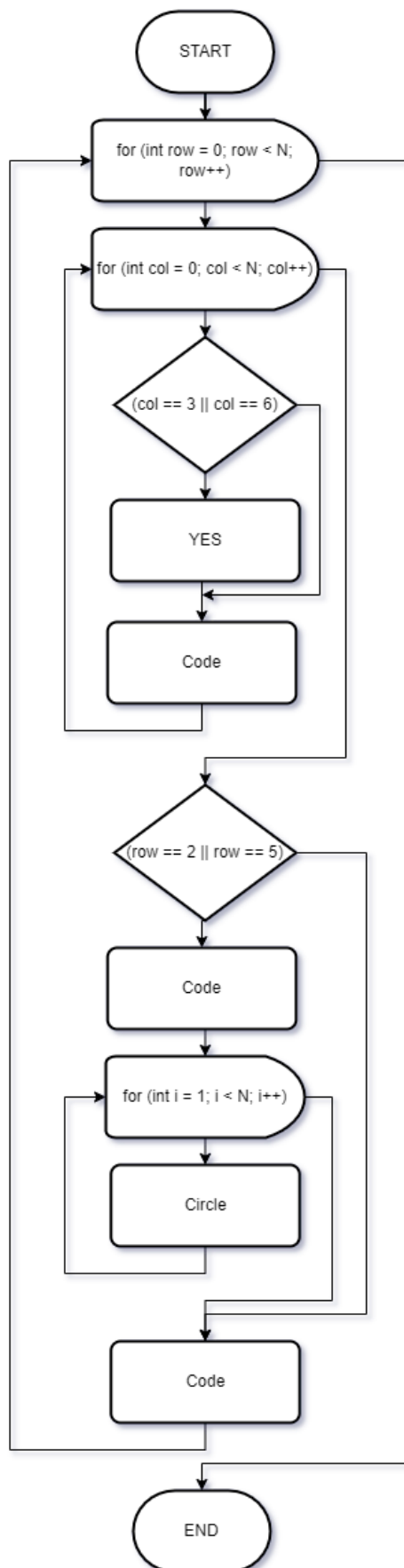
b. boolAdadiRow



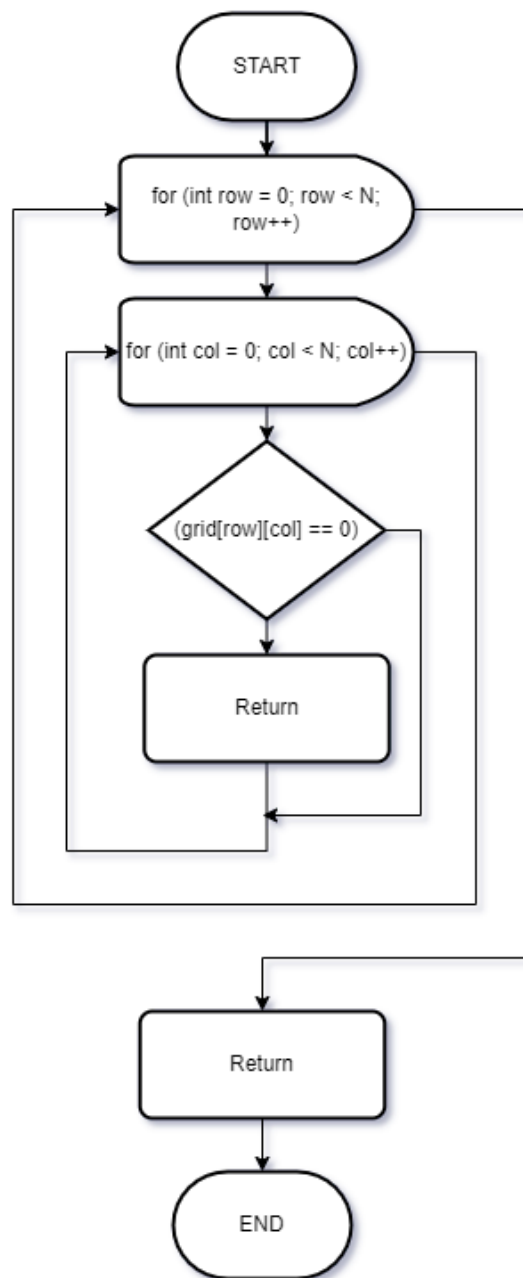
c. boolAdadiBox



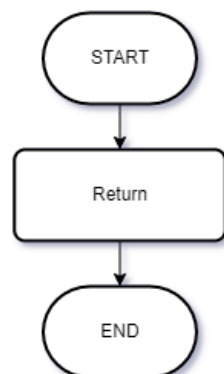
d. void SudokuGrid



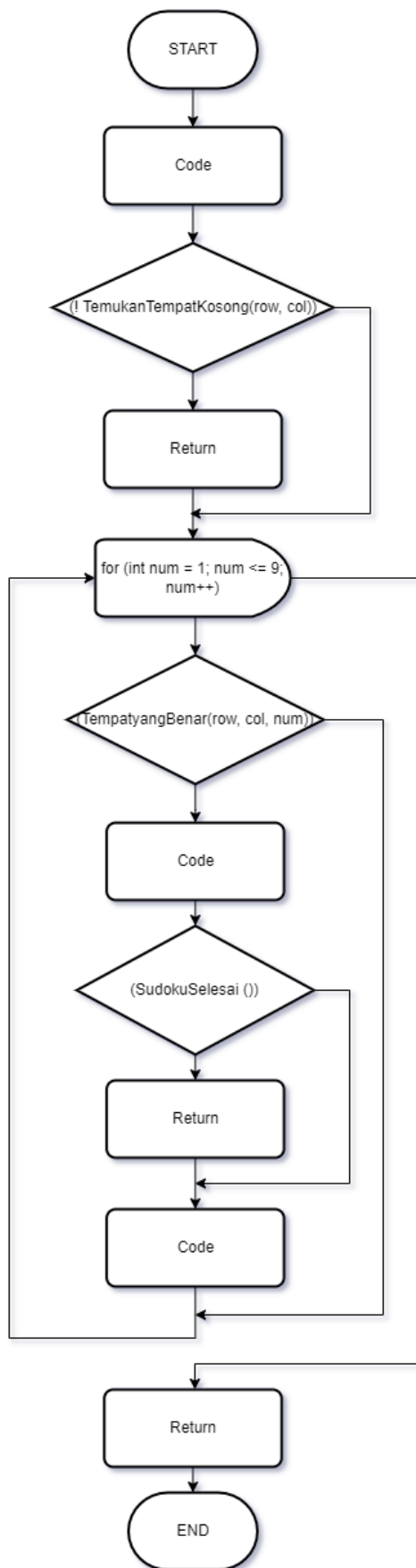
e. `bool` TemukanTempatKosong



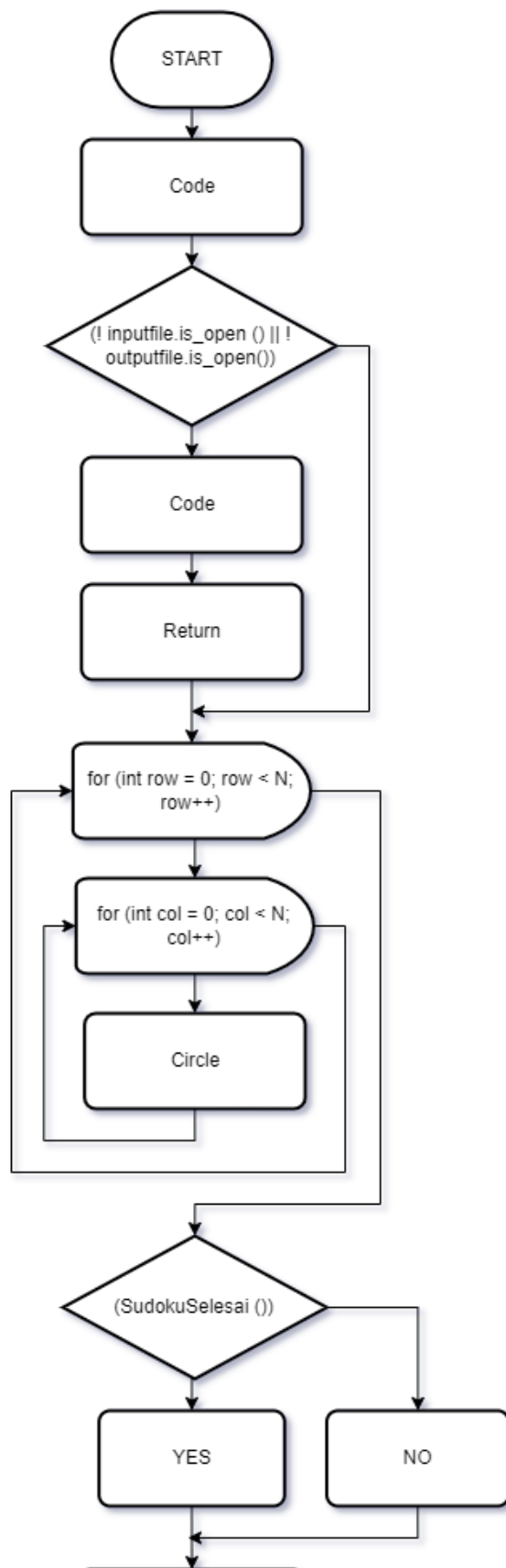
f. `bool` TempatyangBenar

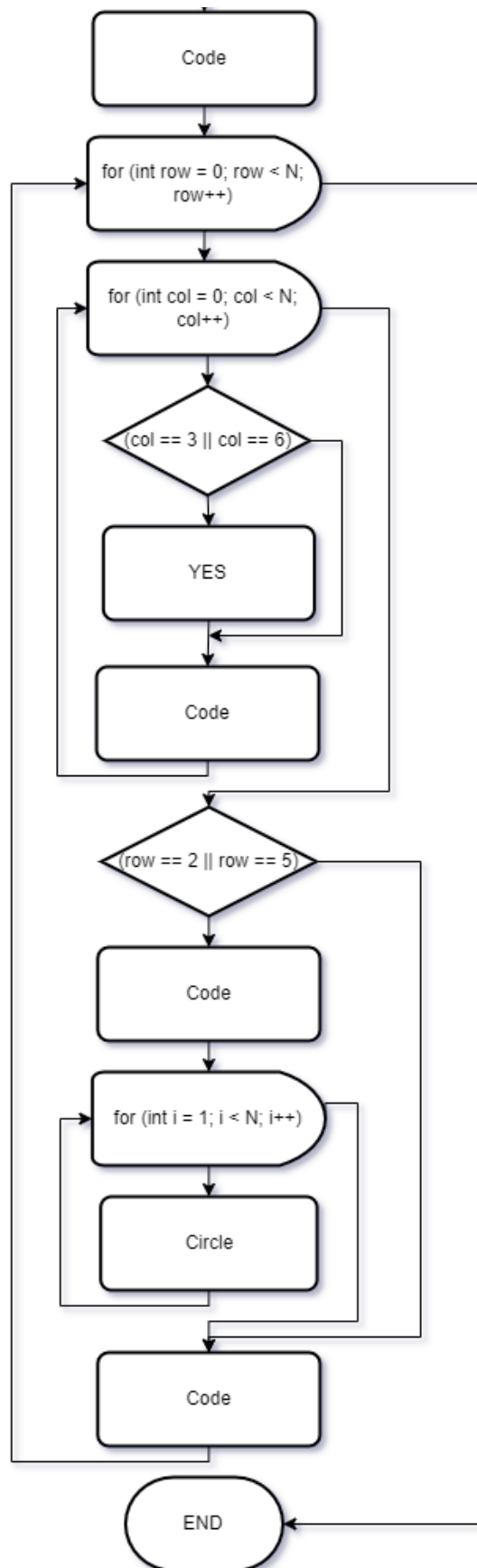


g. `bool SudokuSelesai`



h. int main





2. Penjelasan Setiap Blok Program

```
/*
Ryan Ernanda 120140154
Muhammad Hadi Arsa 120140150
*/

#include <iostream>
#include <fstream>

#define N 9 // sudoku 9x9

using namespace std;

int grid[N][N]; // sudoku grid

bool AdadiCol(int col, int num){ //periksa apakah num ada dalam col
atau tidak

    for (int row = 0; row < N; row++) // periksa setiap baris

        if (grid[row][col] == num) // jika num ada dalam baris

            return true;

    return false;
}

bool AdadiRow(int row, int num){ //periksa apakah num ada di baris atau
tidak

    for (int col = 0; col < N; col++) // periksa setiap kolom

        if (grid[row][col] == num) // jika num ada di kolom

            return true;

    return false;
}

bool AdadiBox(int mulairow, int mulaicol, int num){ //periksa apakah num
ada di kotak 3x3 atau tidak

    for (int row = 0; row < 3; row++) // periksa setiap baris

        for (int col = 0; col < 3; col++) // periksa setiap kolom
```



```

        if (grid[row+mulairow][col+mulaicol] == num)// jika num ada di
kotak

            return true;

        return false;
    }

void SudokuGrid(){ //cetak isi sudoku setelah dipecahkan

    for (int row = 0; row < N; row++){// periksa setiap baris

        for (int col = 0; col < N; col++){// periksa setiap kolom

            if(col == 3 || col == 6)// jika kolom 3 atau 6

                cout << " | ";

            cout << grid[row][col] <<" ";

        }

        if(row == 2 || row == 5){// jika baris 2 atau 5

            cout << endl;

            for(int i = 1; i<N; i++)

                cout << "---";

        }

        cout << endl;

    }

}

bool TemukanTempatKosong(int &row, int &col){ //dapatkan lokasi kosong
dan perbarui baris dan kolom

    for (row = 0; row < N; row++){// periksa setiap baris

        for (col = 0; col < N; col++){// periksa setiap kolom

            if (grid[row][col] == 0) //ditandai 0 dengan nilai kosong

                return true;

            return false;

        }

    }

}

bool TempatyangBenar(int row, int col, int num){//periksa apakah num
dapat dimasukkan ke sudoku

```

```

        return !AdadiRow(row, num) && !AdadiCol(col, num) && !AdadiBox(row -
row%3, col - col%3, num); // jika num tidak ada di baris, kolom, dan
kotak 3x3
    }

bool SudokuSelesai() { //sudoku diselesaikan dengan cara mencari lokasi
kosong dan mencari nilai yang valid untuk lokasi kosong

    int row, col;

    if (!TemukanTempatKosong(row, col)) // jika tidak ada lokasi kosong

        return true; //ketika semua tempat terisi

    for (int num = 1; num <= 9; num++){ //nomor yang valid adalah 1 - 9

        if (TempatyangBenar(row, col, num)){ //periksa validasi, jika ya,
masukkan nomornya ke dalam kotak

            grid[row][col] = num; // masukkan nomor ke dalam kotak

            if (SudokuSelesai()) //secara rekursif pergi ke kotak lain di
grid

                return true;

            grid[row][col] = 0; //beralih ke kotak yang tidak ditetapkan
ketika kondisinya tidak terpenuhi

        }

    }

    return false;
}

int main() {

    grid[N][N]; // sudoku 9x9

    fstream inputfile; // file input

    fstream outputfile; // file output

    cout << "Masukkan nama file input: ";

    string fileName, outputFileName; // nama file input dan output

    cin >> fileName; // masukkan nama file input

    inputfile.open(fileName, ios::in); // membuka file input

```

```

cout << "Masukkan nama file output: ";

cin >> outputFileFileName; // masukkan nama file output

outputfile.open(outputFileName, ios::out); // membuka file output

if (!inputfile.is_open() || !outputfile.is_open()) // jika file tidak
dapat dibuka
{
    cout << "Error membuka file";

    return 0;
}

for (int row = 0; row < N; row++) // periksa setiap baris
    for (int col = 0; col < N; col++) // periksa setiap kolom
        inputfile >> grid[row][col]; // masukkan isi file ke dalam
grid

if (SudokuSelesai()) // jika sudoku dapat diselesaikan
    SudokuGrid();
else
    cout << "Tidak ada solusi"; // jika tidak dapat diselesaikan

inputfile.close();

outputfile << "Hasil sudoku yang diselesaikan: " << endl; //
menuliskan isi sudoku ke file output

for (int row = 0; row < N; row++) { // periksa setiap baris
    for (int col = 0; col < N; col++) { // periksa setiap kolom
        if (col == 3 || col == 6) // jika kolom 3 atau 6
            outputfile << " | ";

        outputfile << grid[row][col] << " "; // menuliskan isi
sudoku ke file output
    }
}

```

```

        if(row == 2 || row == 5){// jika baris 2 atau 5

            outputfile << endl;

            for(int i = 1; i<N; i++){// menuliskan "---"

                outputfile << "---";

            }

            outputfile << endl;

        }

    }
}

```

D. OUTPUT PROGRAM

saat pertama kali dijalankan program meng output teks untuk meminta input seperti gambar dibawah ini

```

Masukkan Nama File Sudoku: case1.txt
Masukan Nama File Untuk Menyimpan Hasil Sudoku: hasilcase1.txt

```

jika sudah maka output yang dikeluarkan adalah hasil dari sudoku yang sudah diselesaikan seperti di gambar dibawah ini

```

Sudoku Berhasil Diselesaikan
5 3 4 | 6 7 8 | 9 1 2
6 7 2 | 1 9 5 | 3 4 8
1 9 8 | 3 4 2 | 5 6 7
-----
8 5 9 | 7 6 1 | 4 2 3
4 2 6 | 8 5 3 | 7 9 1
7 1 3 | 9 2 4 | 8 5 6
-----
9 6 1 | 5 3 7 | 2 8 4
2 8 7 | 4 1 9 | 6 3 5
3 4 5 | 2 8 6 | 1 7 9

```

selain output yang dikeluarkan, program juga menghasilkan file hasil output sesuai dengan nama yang diberikan saat meminta masukkan nama file untuk menyimpan hasil sudoku seperti di gambar bawah ini

```

hasilcase1.txt
1 Hasil sudoku yang diselesaikan:
2 5 3 4 | 6 7 8 | 9 1 2
3 6 7 2 | 1 9 5 | 3 4 8
4 1 9 8 | 3 4 2 | 5 6 7
5 -----
6 8 5 9 | 7 6 1 | 4 2 3
7 4 2 6 | 8 5 3 | 7 9 1
8 7 1 3 | 9 2 4 | 8 5 6
9 -----
10 9 6 1 | 5 3 7 | 2 8 4
11 2 8 7 | 4 1 9 | 6 3 5
12 3 4 5 | 2 8 6 | 1 7 9
13

```