RESPONSI

PRAKTIKUM KONSEP PEMROGRAMAN

2024

IDENTITAS

Nama : Rifqia Hani Milatunnisa

NIM : L0124074

Kelas : B

Judul Program :B_Responsi1KP_L0124074_RifqiaHani Milatunnisa_Game.c

Deskripsi Program : Game Tic Tac Toe modifikasi menggunakan bahasa C

Dokumentasi Program

Analisis Kode

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

#Include adalah instruksi prepocessor bahasa C. Tujuannya untuk mengimpor fungsi-fungsi (pustaka) yang sudah didefinisikan pada header file. Header file sendiri merupakan file yang berisi definisi fungsi yang sudah dibuat. Header file bertujuan agar bisa digunakan pada program C yang lain. Beberapa Pustaka yang diimpor adalah:

- stdio.h : Pustaka berisi fungsi-fungsi unutk input output dasar, seperti menampilkan teks ke layar, membaca input dari pengguna, atau bekerja dengan file. Pada kode di atas digunakan untuk printf, scanf, fgetc.

- stdlib.h : Pustaka berisi fungsi-fungsi unutk alokasi memori dinamis, konversi tipe data, pengelolaan program, dan operasi lainnya yang tidak terkait langsung dengan input/output. Pada kode di atas digunakan untuk system("clear"), system("CLS"), rand, dan srand.
- time.h : Pustaka berisi fungsi unutk bekerja dengan waktu dan tanggal, serta fungsi untukmenelola waktu system dan interval waktu. Pada kode di atas digunakan untuk fungsi time(NULL), yang digunakan sebagai seed untuk srand agar menghasilkan nilai acak yang berbeda setiap program dijalankan.

```
#define SIZE 3

// kode warna ANSI
#define RED "\033[0;31m"
#define YELLOW "\033[0;33m"
#define BLUE "\033[0;34m"
#define RESET "\033[0m"

// Variabel global
char board[SIZE][SIZE];
int movesX[3] = {-1, -1, -1}; // Menyimpan posisi langkah X
int movesO[3] = {-1, -1, -1}; // Menyimpan posisi langkah O
int lastRemoved = -1; // Menyimpan posisi langkah yang akan dihapus
char playerX[50], playerO[50]; // Nama pemain
```

- #define SIZE 3 digunakan unutk mendefinisikan sebuah konstanta dalam program. Di sini, SIZE didefinisikan dengan nilai 3, yang menunjukkan ukuran papan permainan 3x3.
- Kode warna ANSI (RED, YELLOW, BLUE, RESET) digunakan unutk memberi warna pada symbol X dan O di papan. RED akan memberi warna merah pada teks, YELLOW akan memberi warna kuning pada teks, BLUE akan memberi warna biru pada teks, dan RESET akan mengembalikan warna teks ke warna default terminal setelah teks yang diberi warna (dalam hal ini berwarna putih).
- char[SIZE][SIZE] merupakan deklarasi variabel global yang menginilisasi ukuran sisi papan permainan menjadi 3. board adalah array 2D cahr untuk menyimpan posisi X atau O di papan.
- int movesX[3] = {-1, -1, -1} merupakan deklarasi variabel global yang menginilisasi nilai awal (posisi awal) X yaitu -1 atau dianggap belum terisi. Kemudian langkah-langkah X berikutnya juga akan tersimpan di dalam variabel ini.

- int movesO[3] = {-1, -1, -1} merupakan deklarasi variabel global yang menginilisasi nilai awal (posisi awal) O yaitu -1 atau dianggap belum terisi. Kemudian langkah-langkah O berikutnya juga akan tersimpan di dalam variabel ini.
- int lastRemoved = -1 merupakan deklarasi variabel global yang menginilisai langkah terakhir yang dihapus yaitu -1 (belum terisi). Variabel ini akan menyimpan posisi Langkah yang dihapus saat pemain mencapai batas tiga langkah.
- char playerX[50] dan player[50], kedua variabel global ini digunakana unutk menyimpan nama pemain X dan pemain O. Setiap nama pemain dapat terdiri hingga 50 karakter.

```
// Fungsi untuk membersihkan layar
void clearScreen() {
    #ifdef _WIN32
        system("CLS");
    #else
        system("clear");
    #endif
}
```

- Fungsi clearscreen() berfungsi untuk membersihkan layar menggunakan perintah system system(“CLS”) untuk Windows dan system(“clear”) unutk UNIX.
- #ifdef adalah direktif preprocessor dalam bahasa C yang memeriksa apakah suatu makro telah didefinisikan sebelumnya.
- #ifdef _WIN32 memeriksa apakah makro _WIN32 didefinisikan. Makro _WIN32 biasanya secara otomatis didefinisikan oleh compiler ketika program dijalankan pada sistem operasi Windows. Ini berarti bahwa jika program berjalan di Windows, maka blok kode dalam #ifdef akan dijalankan.
- #else : Jika makro _WIN32 tidak didefinisikan, artinya program dijalankan pada sistem operasi selain Windows (misalnya Linux atau macOS), maka bagian #else yang berisi perintah system("clear") akan dieksekusi.
- #endif : Direktif preprocessor ini menandai akhir dari pemeriksaan kondisi yang dimulai dengan #ifdef.

```c
// Mencetak banner/awalan permainan
void start() {
    printf("========         SELAMAT DATANG         ========\n");
    printf("======== DI PERMAINAN TIC TAC TOE 3-MAX ========\n");
    printf("========         SELAMAT BERMAIN        ========\n\n");
}
```

- Fungsi start() berfungsi untuk mencetal banner ucapan selamat dating sebelum pemain memainkan permainan Tic Tac Toe 3 -Max.

```c
// Fungsi untuk mencetak aturan permainan yang ada dalam file txt
void printRules() {
    FILE *file_ptr;
    char ch;

    file_ptr = fopen("aturan.txt", "r");
    if (file_ptr == NULL) {
        printf("File aturan tidak bisa dibuka.\n");
        return;
    }

    printf("Aturan Permainan:\n");
    while ((ch = fgetc(file_ptr)) != EOF) {
        printf("%c", ch);
    }

    fclose(file_ptr);
    printf("\n");
}
```

- Fungsi printRules berfungsi unutk membaca dan menampilkan aturan permainan dari file aturan.txt
- FILE *file_ptr: variabel file_ptr adalah pointer yang digunakan untuk merujuk ke file yang akan dibuka. Tipe data FILE adalah tipe data yang disediakan oleh pustakan standar C (stdio.h) unutk menangani file.
- char ch ; variabel ch digunakan unutk menyimpan karakter sementara yang dibaca dari file. Variabel ini akan digunakan dalam proses pembacaan karakter per karakter dari file.
- fopen() digunakan untuk membuka file txt yanag disebutkan (“aturan.txt”). Fungsi ini mengembalikan pointer ke file (FILE *) jika file berhasil dibuka. Jika tidak ditemukan atau ada masalah saat membuka file, maka fopen() akan mengembalikna nilai NUUL.
- “r” adalah mode yang digunakan untuk membuka file dalam mode baca (read) dan tidak mengubahnya.

- Jika file gagal dibuka, program akan mencetak pesan kesalahan : “File aturan tidak bisa dibuka”. Funsi return digunakan unutk keluar dari fungsi printRules, sehingga program tidak melanjutkan ke bagian membaca file.
- Fungsi fgetc() digunakan untuk membaca satu karakter dari file setiap kali dipanggil. Fungsi ini akan mengembalikan karakter berikutnya yang ada dalam file dan memindahkan posisis pointer file ke karakter berikutnya. fgetc(file_ptr) akan membaca karakter dari file yang ditunjuk oleh oleh file_ptr.
- while ((ch = fgetc(file_ptr)) != EOF) ; proses ini akan berulang selama fgetc() tidak mengembalikan EOF (End Of File) -> akhir file.
- Selama proses membaca, karakter yang dibaca disimpan dalam varibel ch, dan kemudian dicetak menggunakan printf(“%c”,ch), yang mencetak satu karakter pada layar.
- Dengan cara ini, seluruh isi file akan dibaca kemudian dicetak ke layar hinga seluruh file selesai dibaca.

```
// Fungsi untuk menginisialisasi nilai awal papan permainan
void initBoard() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            board[i][j] = ' ';
        }
    }
}
```

- Fungsi initBoard() berfungsi untuk menginilisasi papan permainan dengan mengisi semua posisi dengan karakter spasi.
- Fungsi ini menggunakan dua loop for bersarang untuk mengakses setiap elemen dalam matriks board. Loop pertama mengiterasi setiap baris dari matriks papan permainan. Loop kedua mengiterasi setiap kolom dalam setiap baris.
- Di dalam loop bersarang ini board[i][j] di set ke ‘ ‘ (spasi kosong). Ini berarti setiap elemen dalam matriks board diinilisasi dengan karakter kosong(‘ ‘), yang menunjukkan bahwa seluruh posisi papan permainan pada awalnya kosong.

```c
// Fungsi untuk mencetak papan permainan
void displayBoard() {
    int oldestMoveX = movesX[0];
    int oldestMoveO = movesO[0];

    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            int position = i * SIZE + j;
            // Jika yang dicetak simbol X atau O yang akan hilang maka warnanya akan merah
            if (position == oldestMoveX && board[i][j] == 'X') {
                printf(" %sX%s ", RED, RESET);
            }
            else if (position == oldestMoveO && board[i][j] == 'O') {
                printf(" %sO%s ", RED, RESET);
            }
            // Jika yang dicetak simbol X maka warnanya kuning
            else if (board[i][j] == 'X') {
                printf(" %sX%s ", YELLOW, RESET);
            }
            // Jika yang dicetak simbol O maka warnanya biru
            else if (board[i][j] == 'O') {
                printf(" %sO%s ", BLUE, RESET);
            }
            // Jika kotak kosong maka akan berisi spasi
            else {
                printf("   ");
            }

            if (j < SIZE - 1)
            printf("|");
        }

        printf("\n");
        if (i < SIZE - 1) printf("---+---+---\n");
    }
}
```

- Fungsi displayBoard() berfungsi untuk menampilkan papan permainan dan memberi warna pada symbol berdasarkan beberapa kondisi.
- oldestMoveX dan oldestMoveO adalah variabel yang digunakan untuk menyimpan langkah paling awal dari pemain 'X' dan pemain 'O' (diambil dari array movesX[] dan movesO[]). Ini digunakan untuk memberikan warna merah pada simbol 'X' atau 'O' yang paling lama (yang akan hilang).
- Fungsi ini menngunakan dua loop bersarang (for) untuk mencetak seluruh papan yang memiliki ukuran SIZE x SIZE (3x3).

- position = i * SIZE + j: Variabel position dihitung berdasarkan indeks baris (i) dan kolom (j). Ini digunakan untuk menghitung posisi linear pada papan 2D dan membandingkannya dengan posisi oldestMoveX dan oldestMoveO.
- Kemudian pencetakan isi papan mengikuti beberapa kondisi:
    a. if (position == oldestMoveX && board[i][j] == 'X'); jika posisi saat ini (position) adalah langkah pertama (terlama) dari X maka ‘X’ akan dicetak dengan warna RED merah, kemudian RESET digunakan untuk mengembalikan warna teks ke default (putih).
    b. else if (position == oldestMoveO && board[i][j] == 'O'); jika posisi saat ini (position) adalah langkah pertama (terlama) dari O maka ‘O’ akan dicetak dengan warna RED merah, kemudian RESET digunakan untuk mengembalikan warna teks ke default (putih).
    c. else if (board[i][j] == 'X') ; Jika posisi pada papan berisi simbol ‘X’ dan bukan Langkah pertama (terlama) maka simbol X akan dicetak dengan warna kuning.
    d. else if (board[i][j] == 'O') ; Jika posisi pada papan berisi simbol ‘O’ dan bukan Langkah pertama (terlama) maka simbol O akan dicetak dengan warna biru.
    e. Else { printf(“ “); } ; Jika posisi papan kosong, maka akan dicetak spasi yang menunjukkkan kotak kosong.
- if (j < SIZE - 1) printf("|"); berfungsi untuk mencetak pembatas vertical berupa tanda (|) untuk memisahkan kolom apabila kolom yang sedang diproses bukan kolom yang terakhir.
- if (j < SIZE - 1) printf("---+---+---\n"); berfungsi untuk mencetak pembatas horizontal berupa tanda (---+---+---+) untuk memisahkan baris apabila baris yang sedang diproses bukan baris yang terakhir.

```
// Fungsi untuk mengecek jika salah satu pemain sudah menang
int checkWin(char player) {
    for (int i = 0; i < SIZE; i++) {
        if (board[i][0] == player && board[i][1] == player && board[i][2] == player) return 1;
        if (board[0][i] == player && board[1][i] == player && board[2][i] == player) return 1;
    }

    if (board[0][0] == player && board[1][1] == player && board[2][2] == player) return 1;
    if (board[0][2] == player && board[1][1] == player && board[2][0] == player) return 1;
    return 0;
}
```

- Fungsi checkWin berfungsi untuk mengecek apakah salah satu pemain menang dengan memeriksa baris, kolom, atau diagonal.

- Fungsi ini mengembalikan nilai 1 jika pemain yang diberikan (player) telah menang, atau 0 jika belum menang.
- Parameter char player adalah simbol pemain yang sedang diperiksa, yaitu 'X' atau 'O'. Pemain yang sedang diperiksa diberikan ke dalam parameter ini.
- if (board[i][0] == player && board[i][1] == player && board[i][2] == player) return 1; digunakan untuk pengecekan horizontal. Jika ketiga kolom dalam satu baris memiliki nilai yang sama maka pemain tersebut menang.
- if (board[0][i] == player && board[1][i] == player && board[2][i] == player) return 1; digunakan untuk pengecekan vertikal. Jika ketiga baris dalam satu kolom memiliki nilai yang sama maka pemain tersebut menang.
- if (board[0][0] == player && board[1][1] == player && board[2][2] == player) return 1; digunakan untuk pengecekan diagonal (kiri-kanan). Jika ketiga elemen ini memiliki nilai (simbol) yang sama maka pemain tersebut menang.
- if (board[0][2] == player && board[1][1] == player && board[2][0] == player) return 1; digunakan untuk pengecekan diagonal (kanan-kiri). Jika ketiga elemen ini memiliki nilai (simbol) yang sama maka pemain tersebut menang.
- return 0; jika tidak ada kondisi kemenangan yang ditemukan pada baris, kolom, atau diagonal, fungsi akan mengembalikan nilai 0, yang berarti pemain tersebut belum menang.

```
// Fungsi untuk memperbarui langkah yang tersimpan menggunakan pointer
void updateMoves(int *moves, int row, int col) {
    if (moves[2] != -1) {
        int oldRow = moves[0] / SIZE;
        int oldCol = moves[0] % SIZE;
        board[oldRow][oldCol] = ' ';
    }
    else {
        lastRemoved = -1; // Tidak ada langkah yang dihapus
    }

    for (int i = 0; i < 2; i++) {
        moves[i] = moves[i + 1];
    }
    moves[2] = row * SIZE + col;
}
```

- Fungsi updateMoves() berfungsi untuk memperbarui array movesX atau movesO setelah pemain melakukan Langkah.

- Jika pemain sudah melakukan tiga langkah, langkah tertua akan dihapus, dan kotak pada Langkah tersebut dikosongkan (board[oldRow][oldCol] = ‘ ‘).
- ```
  void updateMoves(int *moves, int row, int col) {
  ```

  int *moves adalah parameter berupa pointer ke array yang menyimpan Langkah-langkah pemain. Array ini menyimpan tiga elemen, yang masing-masing menunjukkan Langkah-langkah yang diambil oleh pemain dalam bentuk satu dimensi.

  int row dan int col adalah parameter yang menunjukkan posisi baris dan kolom pada papan permainan yang diinginkan untuk langkah baru.
- ```
  if (moves[2] != -1) {
      int oldRow = moves[0] / SIZE;
      int oldCol = moves[0] % SIZE;
      board[oldRow][oldCol] = ' ';
  }
  else {
      lastRemoved = -1; // Tidak ada langkah yang dihapus
  }
  ```

  moves[2] menyimpan langkah terakhir yang diambil, jika moves[2] == -1 (artinya belum ada 3 langkah), tidak perlu ada Langkah yang harus dihapus.

  Jika moves[2] != 1 proram akan menghitung baris(oldRow) dan kolom(oldCol) dengan cara moves[0]/SIZE untuk oldRow, dan moves[0] % SIZE untuk oldCol.

  Kemudian langkah pada board[oldRow][oldCol] akan dihapus atau = ‘ ‘.
- ```
  for (int i = 0; i < 2; i++) {
      moves[i] = moves[i + 1];
  }
  moves[2] = row * SIZE + col;
  ```

  Looping berfungsi untuk menyesuaikan langkah yang lain, yaitu memindahkan posisi moves[i + 1] ke posisi moves[i]. Dengan cara ini langkah ketiga (moves[2]) dan kedua (moves[1]) akan bergerak satu tempat lebih maju, Langkah 2 (moves[1]) menempati posisi Langkah 1 (moves[0]) yang terhapus, kemudian Langkah 3 (moves[2]) menempati posisi Langkah 2 (moves[1]) .

  Langkah yang baru diambil pemain disimpan di posisi moves[2] dengan rumus row * SIZE + col. Sebagai contoh jika pemain memilih baris 2 dan kolom 1 (yang terbaca array sudah -1) pada papan, maka Langkah tersebut akan tersimpan pada moves[2] sebagai 2*1+3 = 7.
  ```

```c
int makeMove(char player, char *playerName) {
    int row, col;
    printf("%s (%c), masukkan baris dan kolom dipisahkan oleh spasi (1, 2, atau 3)", playerName, player);
    printf("\n>> ");
    scanf("%d %d", &row, &col);

    // Karena pemain diminta input 1-3 maka row dan col harus dikurangi 1 agar sesuai dengan array
    row--; col--;

    if (row < 0 || row >= SIZE || col < 0 || col >= SIZE || board[row][col] != ' ') {
        printf("Langkah tidak valid, coba lagi.\n");
        return 0;
    }

    board[row][col] = player;
    if (player == 'X') {
        updateMoves(movesX, row, col);
    }
    else {
        updateMoves(movesO, row, col);
    }
    return 1;
}
```

- Fungsi makeMove() berfungsi untuk mengambil input dari pemain untuk posisi Langkah yang ingin dimainkan. Kemudian memperbarui Langkah pada papan dan menyimpan posisi langkah baru ke dalam array movesX atau movesO. Fungsi ini akan mengembalikan nilai 1 jika Langkah berhasil dan 0 jika Langkah tidak valid.
- ```c
  int makeMove(char player, char *playerName) {
  ```
  char player adalah parameter yang menunjukkan pemain yang melakukan Langkah. Ini bisa berupa ‘X’ atau ‘O’, yang mewakili pemain dalam permainan.
  char *playerName adalah parameter yang berisi nama pemain yan sedang melakukan langkah. Nama akan digunakan dalam pesan yang meminta pemain untuk memasukkan Langkah mereka.
- ```c
  printf("%s (%c), masukkan baris dan kolom dipisahkan oleh spasi (1, 2, atau 3)", playerName, player);
  printf("\n>> ");
  scanf("%d %d", &row, &col);
  ```
  Pemain diminta memasukkan posisi baris dan kolom dengan format (1,2, atau 3) karena ukuran papan 3x3.
  scanf(“%d %d”, &row, &col); berfungsi unutk mengambil input baris(row) dan kolom (col) dalam bentuk integer.
- row--; col--; berfungsi untuk mengkonversi input yang dimulai dari 1 ke dalam format indeks array yang dimulai dari 0. Program mengurangi nilai row dan col masing-masing sebesar 1. Misal pemain memilih baris 2 dan kolom 3, maka hasil akhir yang tersimpan adalah row = 1 dan col = 2.

```
if (row < 0 || row >= SIZE || col < 0 || col >= SIZE || board[row][col] != ' ') {
    printf("Langkah tidak valid, coba lagi.\n");
    return 0;
}
```

- Potongan pogram tersebut berfungsi sebagai validasi Langkah. Jika row < 0 atau row >= SIZE atau col < 0 atau col >= SIZE atau baris dan kolom tersebut sudah terisi (tidak kosong) maka Langkah dianggap tidak valid, pesan kesalahan akan ditampilkan dan fungsi akan mengembalikan nilai 0.

```
board[row][col] = player;
if (player == 'X') {
    updateMoves(movesX, row, col);
}
else {
    updateMoves(movesO, row, col);
}
return 1;
```

- Potongan program ini berfungsi untuk memperbarui papan permainan dan juga memperbarui Langkah yang tersimpan.

  Jika Langkah valid, maka isi papan (board) akan diperbarui dengan simbol pemain (‘X’ atau ‘O’) pada posisi yang dipilih dengan board[row][col] = player.

  Kemudian fungsi updateMoves() akan dipanggil untuk memperbarui Langkah. Dengan parameter tergantumg dengan player yang sedang bermain. Misal jika player = ‘X’ maka fungsi yang dipanggil adalah updateMoves(movesX, row, col), dengan moves[X] yang dimaksud adalah array movesX. Begitu pula dengan kondisi player bukan ‘X’ atau lebih tepatnya player = ‘O’.

  Jika valid, fungsi akan mengembalikan nilai 1, yang menunjukkan Langkah berhasil dilakukan.

```
// Fungsi utama permainan, mulai dari input pemain sampai salah satu pemain menang
void playGame() {
    char currentPlayer, *currentPlayerName;

    printf("\nMasukkan nama untuk pemain X: ");
    scanf(" %[^\n]", playerX);
    printf("Masukkan nama untuk pemain O: ");
    scanf(" %[^\n]", playerO);
    printf("\n");
```

```
    // Mengacak siapa yang menjadi pemain pertama
    if (rand() % 2 == 0) {
        currentPlayer = 'X';
        currentPlayerName = playerX;
    }
    else {
        currentPlayer = 'O';
        currentPlayerName = playerO;
    }

    initBoard();

    // Permainan akan terus dijalankan sampai salah satu pemain menang
    while (1) {
        printf("\n");
        displayBoard();
        if (makeMove(currentPlayer, currentPlayerName)) {
            if (checkWin(currentPlayer)) {
                printf("\n");
                displayBoard();
                printf("\n%s menang!\n", currentPlayerName);
                break;
            }
            // Pemain menjalankan permainan secara bergantian
            currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
            currentPlayerName = (currentPlayer == 'X') ? playerX : playerO;
        }
    }
}
```

- Fungsi playGame() berfungsi untuk mengelola permainan, mulai dari input nama pemain, mengacak pemain pertama, dan menjalankan giliran secara bergantian. Setiap Langkah dicek apakah sudah berakhir menang dengan memanggil fungsi checkWin(). Jika salah satu pemain menang, maka pemainan berakhir dan papan permainan terakhir ditampilkan dengan nama pemain.
- scanf(" %[^\n]", playerX); Membaca nama pemain untuk X. Format %[^\n] memungkinkan pembacaan seluruh nama pemain, termasuk spasi, hingga mencapai baris baru (enter). Begitu juga dengan input nama pemain playerO.
- Program menggunakan fungsi rand() untuk mengacak giliran pemain pertama. Fungsi rand() % 2 menghasilkan angka 0 dan 1 secara acak. Jika hasilnya 0, pemain X menjadi pemain pertama,

tapi jika hasilnya 1, pemain O menjadi pemain pertama. Kemudian nama pemain (currentPlayerName) akan disesuaikan dengan pemain yang giliran bermain (currentPlayer).

- Pemanggilan fungsi initBoard() berfungsi untuk menginilisasi papan permainan, yaitu semuanya kosong sebelum permainan dimulai.
- Program memasuki looping while(1) yang akan terus dijalankan sampai salah satu pemain menang.
- Fungsi displayBoard dipanggil untuk menampilkan papan permainan pada setiap Langkah, sehingga pemain mampu melihat kondisi terakhir papan.
- Setiap pemain melakukan langkah, program akan memeriksa apakah pemain tersebut menangd dengan memanggil checkWin(currentPlayer). Jika checkWin mengembalikan nilai 0 maka looping while(1) akan terus dijalankan. Namun jika checkWin mengembalikan nilai1 atau true maka program akan menampilkan papan terakhir dan pesan kemenangan, kemudian keluar dari loop dengan break.
- Giliran permainan akan terus bergantian, dengan cara mengambil player lain setelah satu player melakukan langkah dan tidak menang.

```
// Fungsi untuk mengosongkan langkah pemain
void freeBoard() {
    for (int i=0; i<SIZE; i++) {
        movesX[i] = -1;
        movesO[i] = -1;
    }
}
```

- Fungsi freeboard berfungsi untuk mengkosongkan papan (Langkah) dan mengatur array movesX dan movesO ke nilai -1 saat pemain memutuskan untuk bermain lagi. Sehingga papan dapat digunakan kembali untuk permainan baru.
- Dalam perulangan for (int i = 0; i < SIZE; i++), variabel i digunakan untuk mengakses setiap elemen dalam array movesX[] dan movesO[], yang masing-masing menyimpan posisi langkah-langkah yang dilakukan oleh pemain 'X' dan pemain 'O'. movesX[i] = -1 menunjukkan bahwa setiap elemen array movesX[] diatur ulang ke nilai -1. Nilai -1 menunjukkan bahwa Langkah pada posisi tersebut masih kosong atau ke-reset. Begitu juga dengan array movesO[].

```c
// Fungsi rekursi jika pemain ingin bermain kembali
void playAgain() {
    char response;
    printf("\n\n\nApakah Anda ingin bermain lagi? (y/n): ");
    printf("\n>> ");
    scanf(" %c", &response);
    if (response == 'y' || response == 'Y') {
        freeBoard();
        clearScreen();
        playGame();
        playAgain();
    }
    else if (response == 'n' || response == 'N') {
        printf("\n\nTerima kasih telah bermain! Jangan lupa datang lagi!\n\n");
    }
    // Jika input salah, fungsi playAgain akan dipanggil di dalam fungsinya sendiri
    else {
        printf("Input salah, masukkan ulang pilihan! \n");
        playAgain();
    }
}
```

- Fungsi playAgain() berfungsi untuk menanyakan kepada pemain apakah ingin bermain lagi. Jika pemain memilih ‘y’ atau ‘Y’, permainan dimulai ulang dengan memanggil fungsi freeboard(), clearScreen, playGame(), dan playAgain() Kembali (rekursif). Jika pemain memilih ‘n’ atau ‘N’, permainan berakhir. Jika input salah, fungsi playAgain() akan dipanggil kembali (rekursif) hingga input valid.
- char response;: Variabel response digunakan untuk menyimpan input yang diberikan pemain, apakah mereka ingin bermain lagi ('y' atau ‘Y’ untuk ya, 'n' atau ‘N’ untuk tidak).
- printf berfungsi untuk menampilkan pesan yang ingin disampaikan kepada pemain.
- scanf(" %c", &response); Format %c digunakan untuk membaca satu karakter (bentuk char). Adanya spasi di depan %c untuk memastikan bahwa input sebelumnya yang mungkin berupa enter atau karakter lainnya tidak mengganggu pembacaan input berikutnya.
- Jika pemain memilih ‘y’ atau ‘Y’ maka:
    a. Memanggil fungsi freeboard() : untuk mengosongkan Langkah-langkah pada permainan sebelumnya dan memastikan bahwa permainan baru akan dimulai dari awal.
    b. Memanggil fungsi clearScreen() : untuk membersihkan layar di terminal
    c. Memanggil fungsi playGame() : untuk menjalankan permainan baru dari awal
    d. Memanggil fungsi playAgain() kembali (rekursif) : untuk menanyakan kembali kepada pemain setelah menyelesaikan permainan selesai apakah ingin menjalankan permainan

lagi. Ini menciptakan proses rekursif, di mana permainan akan terus diulang jika pemain memilih untuk bermain lagi.

- Jika pemain memilih ‘n’ atau ‘N’ maka program akan menampilkan pesan “Terima kasih telah bermain! Jangan lupa dating lagi!” kemudian mengakhiri permainan.
- Jika pemain memberikan input selain ‘y’, ‘Y’, ‘n’, ‘N’, maka program akan memberi tahu pemain bahwainput salah dan meminta pemain memasukkan input yang benar. Fungsi playAgain() dipanggail Kembali secara rekursif untuk meminta pemain memasukkan input yang benar.

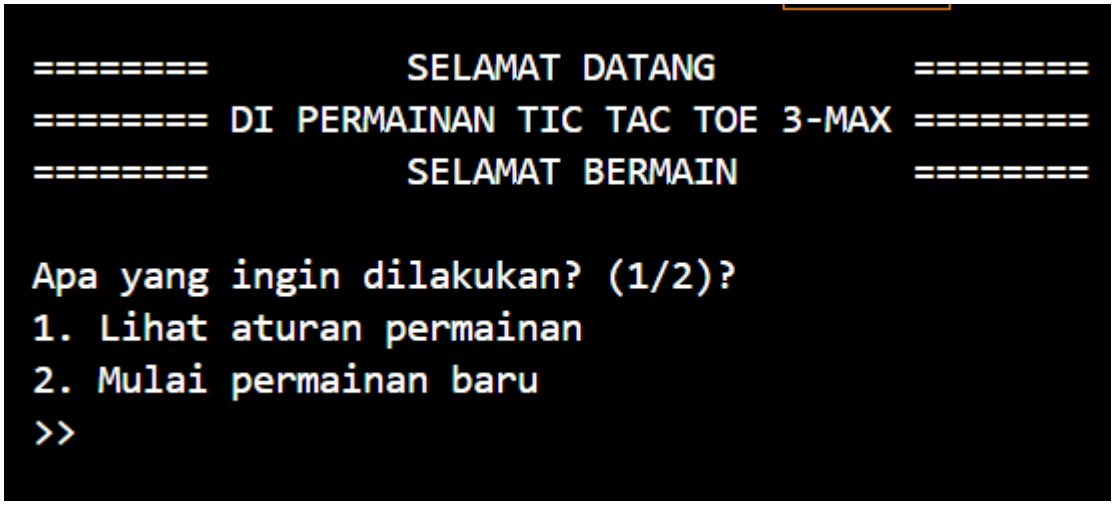
```
// Fungsi untuk menampilkan menu pilihan kepada pemain
void mainMenu() {
    start();
    int menuChoice;
    do {
        printf("Apa yang ingin dilakukan? (1/2)?\n");
        printf("1. Lihat aturan permainan\n");
        printf("2. Mulai permainan baru\n");
        printf(">> ");
        scanf("%d", &menuChoice);

        // Jika pemain memilih 1 maka peraturan akan tercetak di layar
        if (menuChoice == 1) {
            clearScreen();
            printRules();
            printf("\nKetik apapun untuk kembali ke halaman utama\n\n>> ");
            getch();
            clearScreen();
            mainMenu();
        }
        // Jika pemain memilih 2 maka permainan akan dimulai
        else if (menuChoice == 2) {
            playGame();
            playAgain();
        }
        else {
            printf("\n");
            printf("Input salah, coba lagi!\n");
        }
    }
    while (menuChoice != 2);
}
```

- Fungsi mainMenu() berfungsi untuk memberi opsi kepada pemain: pilihan 1 untuk melihat aturan permainan, pilihan 2 untuk memulai permainan baru. Jika input tidak valid, pemain diminta mengulangi inputnya.
- Fungsi start() dipanggil sehingga program menampilkan pesan selmat dating dan banner permainan.
- int menuChoice; variabel yang digunkan untuk menyimpan pilihan menu yang diinput pemain.
- do { ... } while(menuChoice != 2): merupakan loop do-while, yang akan terus menampilkan menu dan meminta input pemain hingga pemain memilih untuk memulai permainan baru (menuChoice == 2).
- Printf() dan scanf() berfungsi untuk menampilkan pilihan menu dan mengambil input pemain.
- Jika menuChoice = = 1 maka:
    a. Memanggil fungsi clearScreen() : untuk membersihkan layar (tampilan) terminal
    b. Memanggil fungsi printRules() : untuk menampilkan aturan permainan dari file “aturan.txt”
    c. Fungsi getch() digunakan untuk membaca input karakter yang tidak perlu diproses lebih lanjut. Setelah pemain menekan tombol, maka layar akan kembali ke fungsi mainMenu()
    d. Memanggil fungsi clearScreen()
    e. Memanggil fungsi mainMenu() : agar pemain dapat Kembali ke menu utama setelah melihat aturan.
- Jika menuChoice == 1, maka fungsi playGame() akan dipanil dan memulai permainan baru. Fungsi playAgain() juga dipangil untuk menanyai pemain apakah ingin memulai permainan lagi atau tidak.
- Jika input bernilai selain 1 atau 2 maka program akan memberikan pesan kepada pemain bahwa input tidak valid. Dan meminta input ulang pilihan.
- Looping pada fungsi ini akan terus berjalan sampai pemain memilih untuk memulai permainan baru (menuChoice == 2). Setelah pemain memilih input 2 maka fungsi playGame() dan fungsi playAgain() akan dipanggil.
- Efek rekursi juga akan muncul jika pemain memilih input 1 yang memanggil kembali fungsi mainMenu().

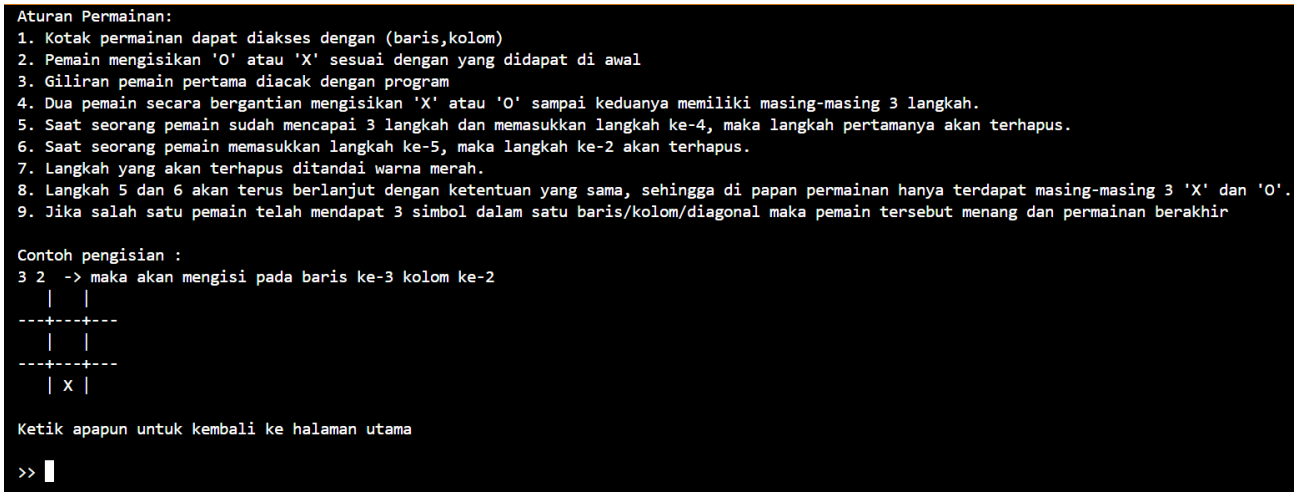
```c
int main() {
    srand(time(NULL));
    clearScreen();
    mainMenu();
    return 0;
}
```

- Fungsi main() berfungsi unutk menginilisasi permainan dengan memanggil clearScreen() dan MainMenu().
- Fungsi srand(time(NULL)) berfungsi untuk menginilisasi generator bilangan acak yang akan digunakan oleh rand(). Disini fungsi ini berperan sebagai pemberi “seed” pada generator acak. time(NULL) mengembalikan nilai waktu saat ini dalam detik sejak Epoch UNIX (1 Januari 1970).
- Pemanggilan fungsi clearScreen() untuk membersihkan layar.
- Pemanggilan fungsi mainMenu() akan menampilkan menu utama permainan.
- Mengakhiri program dengan return 0; menandakan bahwa program tela selesai dijalankan tanpa kesalahn. Nilai 0 menunjukkan bahwa program selesai dengan sukses, sesuai dengan konversi umum di C.
- Fungsi main() menjadi titik masuk utama program. Fungsi ini bertanggung jawab untuk memulai seluruh alur permainan.

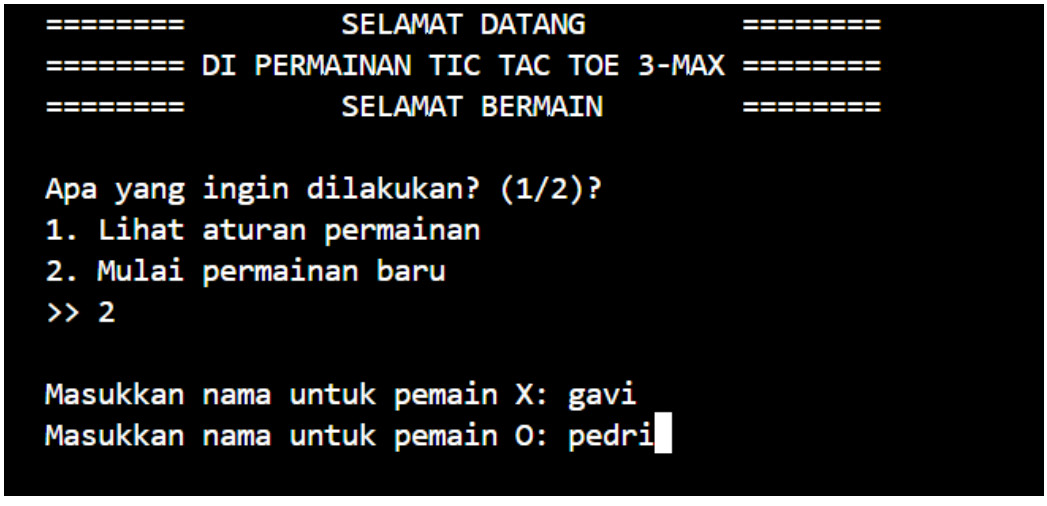
Analisis Program (saat dijalankan)



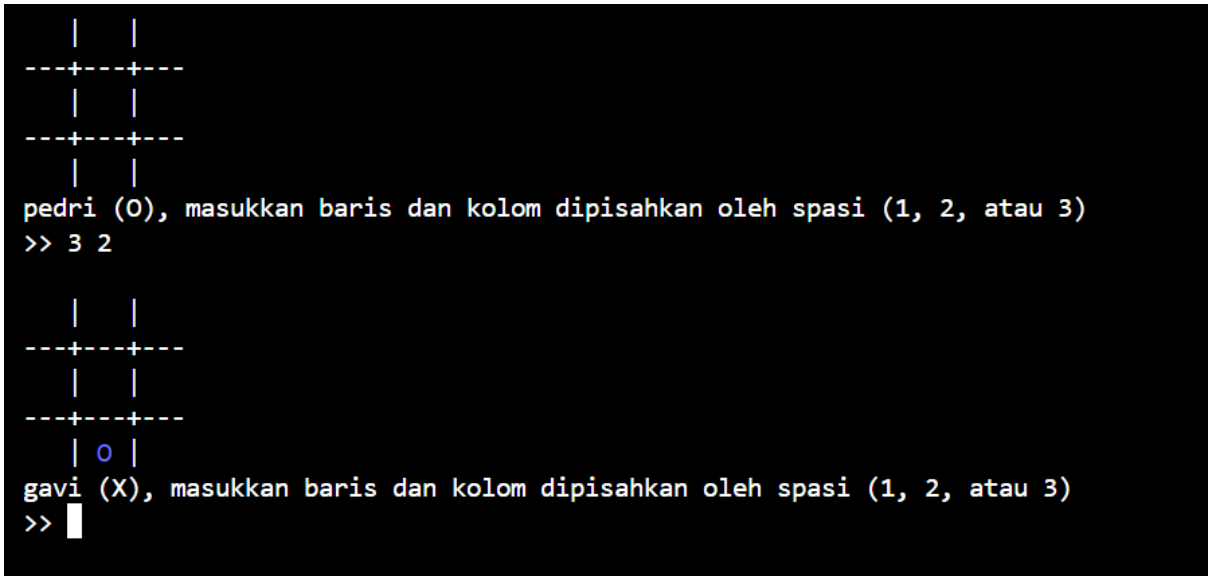
- Saat kode pertama dijalankan, terminal akan menampilkan banner permainan, dan pemain diminta memilih pilihan 1 atau 2 pada menu.



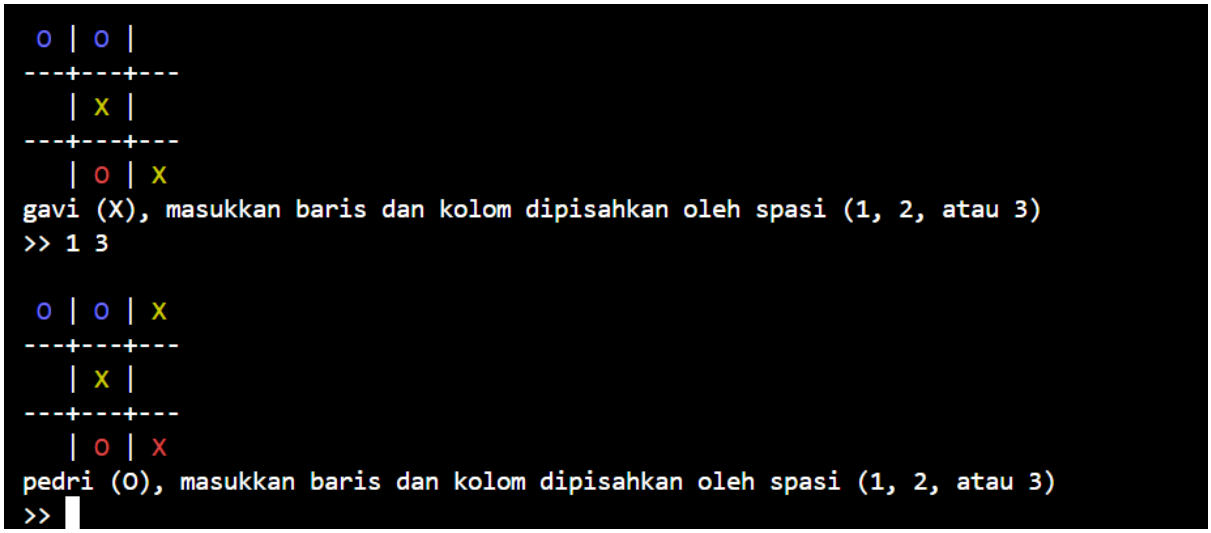
- Ketika pemain memilih menu 1 maka aturan permainan akan tercetak di terminal.
- Kemudian pemain diminta untuk mengetik apapun agar kembali ke halaman utama (menu).



- Ketika pemain memilih menu 2 maka pemain akan memasuki permainan baru dan pemain diminta untuk menginputkan nama untuk masing-masing pemain.
- Kemudian nanti pemain akan mengikuti alur permainan.



- Masing-masing pemain akan diminta menjalankan permainan secara bergantian sampai salah satu menang.
- Urutan pemain pertama menggunakan random pada program.



- Jika pemain sudah memiliki 3 langkah, maka langkah pertamanya akan bewarna merah, menandai bahwa Langkah tersebuat kan terhapus di langkah berikutnya.

```
 o | o | x
---+---+---
   | x |
---+---+---
   | o | x
pedri (O), masukkan baris dan kolom dipisahkan oleh spasi (1, 2, atau 3)
>> 2 1

 o | o | x
---+---+---
 o | x |
---+---+---
   |   | x
gavi (X), masukkan baris dan kolom dipisahkan oleh spasi (1, 2, atau 3)
>> 2 3

 o | o | x
---+---+---
 o | x | x
---+---+---
   |   |
pedri (O), masukkan baris dan kolom dipisahkan oleh spasi (1, 2, atau 3)
>>
```

- Ketika pemain memasukkan Langkah ke-4, maka langkah pertamanya akan terhapus.
- Program terus dijalankan sampai salah satu pemain menang.

```
   | o | x
---+---+---
 o |   | x
---+---+---
 o |   | x

gavi menang!



Apakah Anda ingin bermain lagi? (y/n):
>>
```

- Tampilan program Ketika salah satu pemain sudah menang, papan permainan terakhir di cetak Bersama dengan pernyataan menang pemain.
- Di sini Gavi (X) menang karena sudah terdapat 3 simbol ‘X’ secara vertikal yang terletak dalam satu kolom.

- Kemudian pemain akan ditanyai apakahingin bermain lagi atau tidak.

```
Masukkan nama untuk pemain X: balde
Masukkan nama untuk pemain O: yamal


   |   |
---+---+---
   |   |
---+---+---
   |   |
yamal (O), masukkan baris dan kolom dipisahkan oleh spasi (1, 2, atau 3)
>>
```

- Jika input y/Y maka permainan baru akan dimainkan dan pemain disuruh untuk menginputkan ulang nama dan mengikuti alur permainan sampai salah satu pemain menang.

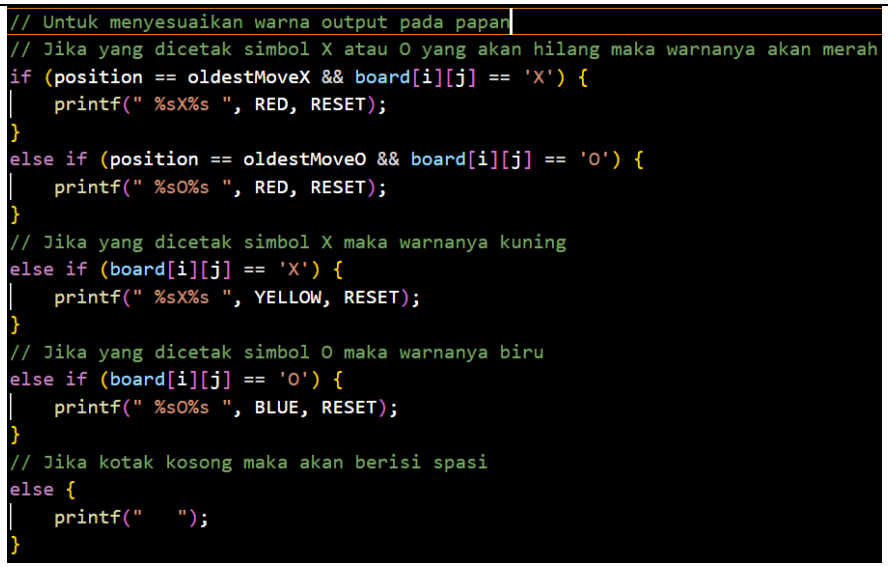
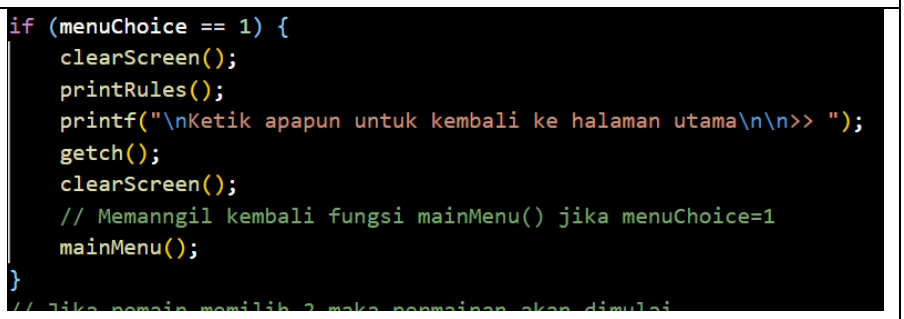
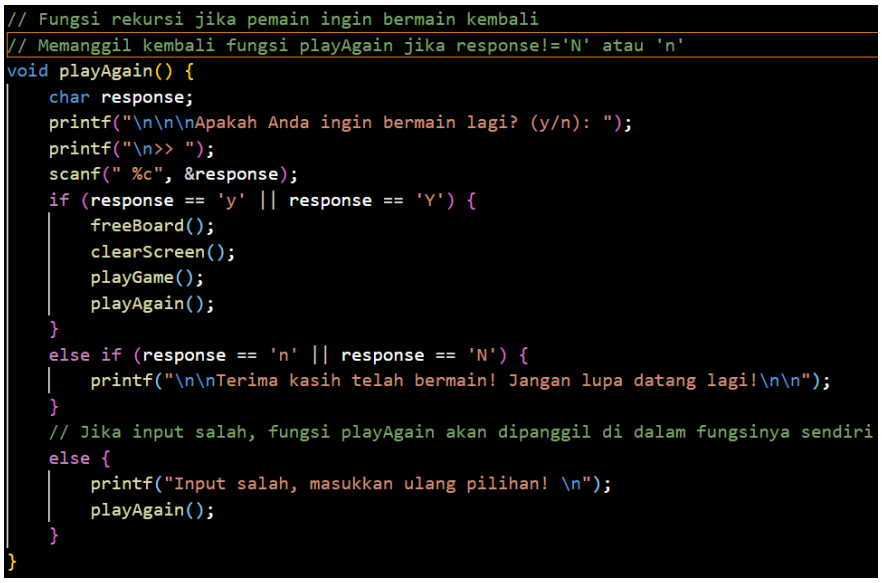
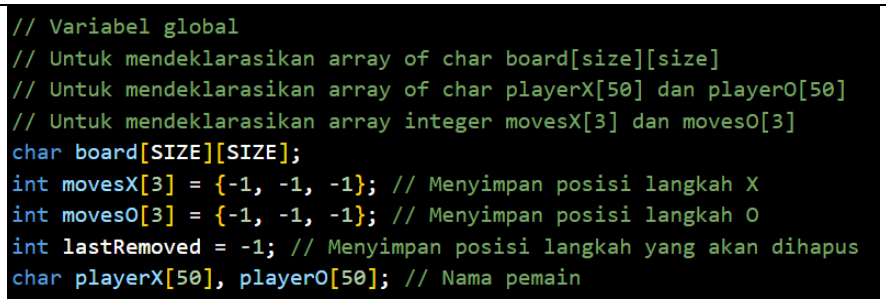
```
Apakah Anda ingin bermain lagi? (y/n):
>> n


Terima kasih telah bermain! Jangan lupa datang lagi!
```

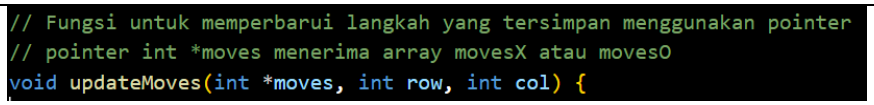
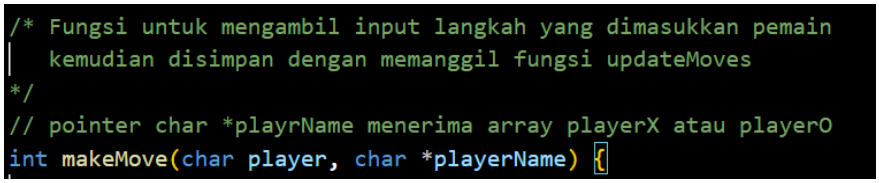
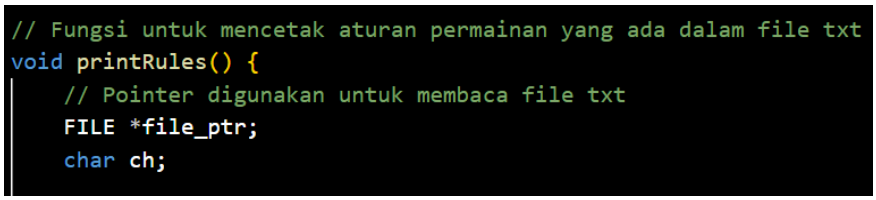
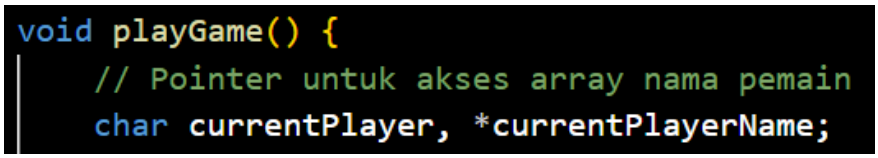
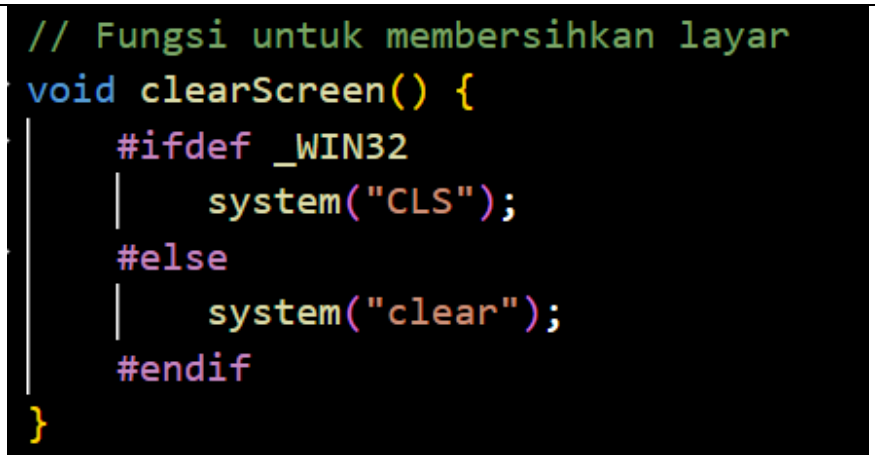
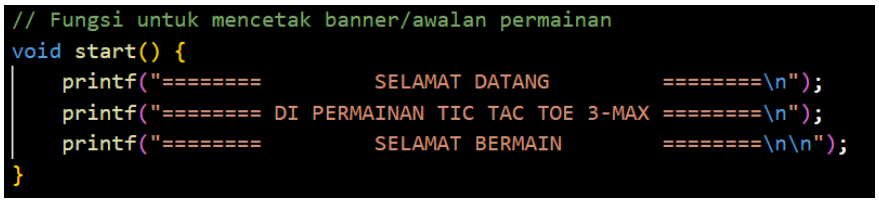
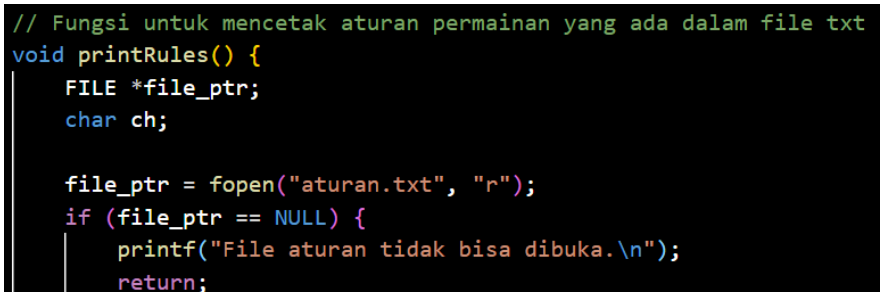
- Jika pemain memilih opsi n/N maka ucapan terimakasih akan tercetak di terminal dan program pun berakhir.

Tabel pengimplementasian materi :

No.	MATERI	POIN	PERAN DALAM PROGRAM
1	Tipe Data	5	`// Variabel global` `// Char -> untuk mendeklarasikan variabel board, playerX, dan playerO` `char board[SIZE][SIZE];` `// INT` `// Untuk mendeklarasikan variabel movesX, movesO, dan lastRemoved` `int movesX[3] = {-1, -1, -1}; // Menyimpan posisi langkah X` `int movesO[3] = {-1, -1, -1}; // Menyimpan posisi langkah O` `int lastRemoved = -1; // Menyimpan posisi langkah yang akan dihapus` `char playerX[50], playerO[50]; // Nama pemain`
2	I/O	5	`// Untuk input nama pemain` `printf("\nMasukkan nama untuk pemain X: ");` `scanf(" %[^\n]", playerX);` `printf("Masukkan nama untuk pemain O: ");` `scanf(" %[^\n]", playerO);` `printf("\n");` `// Untuk mencetak papan permainan permainan di fungsi` `// Fungsi untuk mencetak papan permainan` `void displayBoard() {` `// Untuk mencetak banner permainan di fungsi` `// Mencetak banner/awalan permainan` `void start() {` `printf("========  SELAMAT DATANG  ========\n");` `printf("======== DI PERMAINAN TIC TAC TOE 3-MAX ========\n");` `printf("========  SELAMAT BERMAIN  ========\n\n");` `}`
3	Looping	15	`// Fungsi untuk menginisialisasi nilai awal papan permainan` `void initBoard() {` `// Looping untuk menginilisasi nilai awal pada papan` `for (int i = 0; i < SIZE; i++) {` `for (int j = 0; j < SIZE; j++) {` `board[i][j] = ' ';` `}` `}` `}`

			`` // Permainan akan terus dijalankan sampai salah satu pemain menang // Looping permainan sampai salah satu pemain menang while (1) { printf("\n"); displayBoard(); if (makeMove(currentPlayer, currentPlayerName)) { if (checkWin(currentPlayer)) { printf("\n"); displayBoard(); printf("\n%s menang!\n", currentPlayerName); break; } // Pemain menjalankan permainan secara bergantian currentPlayer = (currentPlayer == 'X') ? 'O' : 'X'; currentPlayerName = (currentPlayer == 'X') ? playerX : playerO; } } ``` ``` // Looping sampai input menuChoice = 2 do { printf("Apa yang ingin dilakukan? (1/2)?\n"); printf("1. Lihat aturan permainan\n"); printf("2. Mulai permainan baru\n"); printf(">> "); scanf(" %d", &menuChoice); // Jika pemain memilih 1 maka peraturan akan tercetak di layar if (menuChoice == 1) { clearScreen(); printRules(); printf("\nKetik apapun untuk kembali ke halaman utama\n\n>> "); getch(); clearScreen(); mainMenu(); } // Jika pemain memilih 2 maka permainan akan dimulai else if (menuChoice == 2) { playGame(); playAgain(); } else { printf("\n"); printf("Input salah, coba lagi!\n"); } } while (menuChoice != 2); ```
4	Conditional	15	``` // Untuk mengecek apakah pemain sudah memenuhi kondisi menang for (int i = 0; i < SIZE; i++) { if (board[i][0] == player && board[i][1] == player && board[i][2] == player) return 1; if (board[0][i] == player && board[1][i] == player && board[2][i] == player) return 1; } if (board[0][0] == player && board[1][1] == player && board[2][2] == player) return 1; if (board[0][2] == player && board[1][1] == player && board[2][0] == player) return 1; return 0; ```

			
5	Rekursi	5	 
6	Array	15	

7	Pointer	10	   
8	Function	10	  

```
// Fungsi untuk memperbarui langkah yang tersimpan menggunakan pointer
// pointer int *moves menerima array movesX atau movesO
void updateMoves(int *moves, int row, int col) {
```

```
/* Fungsi untuk mengambil input langkah yang dimasukkan pemain
   kemudian disimpan dengan memanggil fungsi updateMoves
*/
// pointer char *playrName menerima array playerX atau playerO
int makeMove(char player, char *playerName) {
```

```
// Fungsi untuk mencetak aturan permainan yang ada dalam file txt
void printRules() {
    // Pointer digunakan untuk membaca file txt
    FILE *file_ptr;
    char ch;
```

```
void playGame() {
    // Pointer untuk akses array nama pemain
    char currentPlayer, *currentPlayerName;
```

```
// Fungsi untuk membersihkan layar
void clearScreen() {
    #ifdef _WIN32
        system("CLS");
    #else
        system("clear");
    #endif
}
```

```
// Fungsi untuk mencetak banner/awalan permainan
void start() {
    printf("========         SELAMAT DATANG         ========\n");
    printf("======== DI PERMAINAN TIC TAC TOE 3-MAX ========\n");
    printf("========         SELAMAT BERMAIN        ========\n\n");
}
```

```
// Fungsi untuk mencetak aturan permainan yang ada dalam file txt
void printRules() {
    FILE *file_ptr;
    char ch;

    file_ptr = fopen("aturan.txt", "r");
    if (file_ptr == NULL) {
        printf("File aturan tidak bisa dibuka.\n");
        return;
```

```c
// Fungsi untuk menginisialisasi nilai awal papan permainan
void initBoard() {
    // Looping untuk menginilisasi nilai awal pada papan
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            board[i][j] = ' ';
        }
    }
}
```

```c
// Fungsi untuk mencetak papan permainan
void displayBoard() {
    int oldestMoveX = movesX[0];
    int oldestMoveO = movesO[0];

    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            int position = i * SIZE + j;
            // Jika yang dicetak simbol X atau O y
            if (position == oldestMoveX && board[i
                printf(" %sX%s ", RED, RESET);
```

```c
// Fungsi untuk mengecek jika salah satu pemain sudah menang
int checkWin(char player) {
    for (int i = 0; i < SIZE; i++) {
        if (board[i][0] == player && board[i][1] == player && board[i][2] ==
        if (board[0][i] == player && board[1][i] == player && board[2][i] ==
    }

    if (board[0][0] == player && board[1][1] == player && board[2][2] == play
    if (board[0][2] == player && board[1][1] == player && board[2][0] == play
    return 0;
```

```c
// Fungsi untuk memperbarui langkah yang tersimpan menggunakan pointer
void updateMoves(int *moves, int row, int col) {
    if (moves[2] != -1) {
        int oldRow = moves[0] / SIZE;
        int oldCol = moves[0] % SIZE;
        board[oldRow][oldCol] = ' ';
    }
```

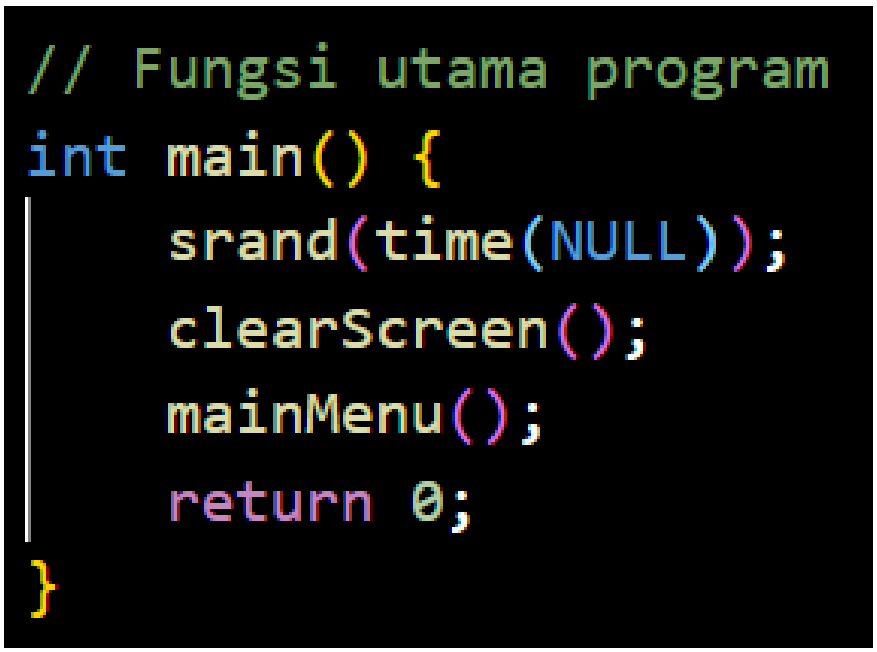
```c
/* Fungsi untuk mengambil input langkah yang dimasukkan pemain
   kemudian disimpan dengan memanggil fungsi updateMoves
*/
int makeMove(char player, char *playerName) {
    int row, col;
    printf("%s (%c), masukkan baris dan kolom dipisahkan oleh spasi (1, 2, atau
    printf("\n>> ");
    scanf("%d %d", &row, &col);

    // Karena pemain diminta input 1-3 maka row dan col harus dikurangi 1 agar s
    row--; col--;

    if (row < 0 || row >= SIZE || col < 0 || col >= SIZE || board[row][col] != '
        printf("Langkah tidak valid, coba lagi.\n");
        return 0;
```

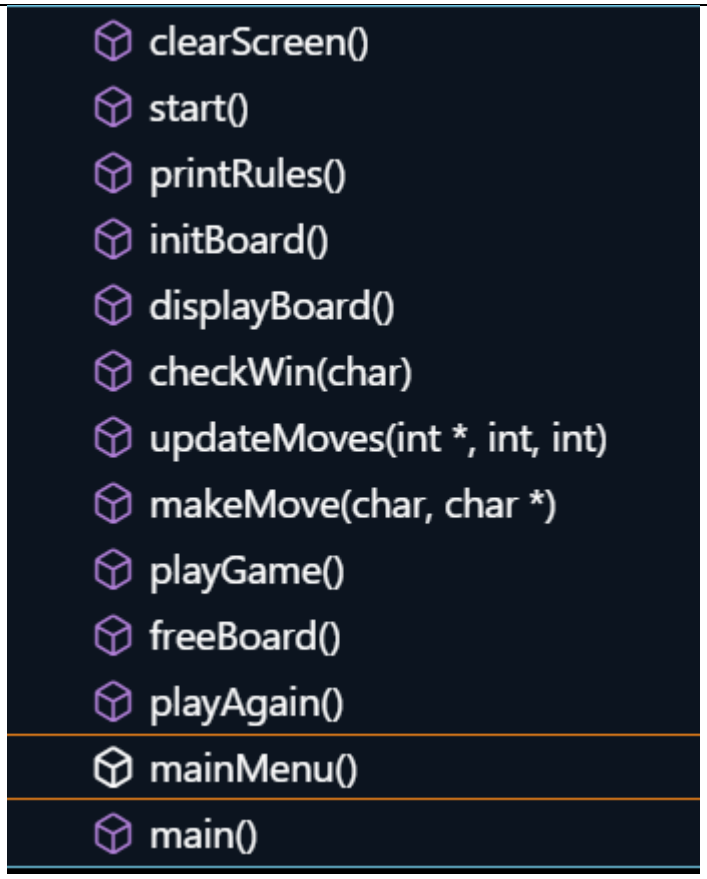
```c
// Fungsi utama permainan, mulai dari input pemain sampai salah satu pemain menang
void playGame() {
    char currentPlayer, *currentPlayerName;

    printf("\nMasukkan nama untuk pemain X: ");
    scanf(" %[^\n]", playerX);
    printf("Masukkan nama untuk pemain O: ");
    scanf(" %[^\n]", playerO);
    printf("\n");

    // Mengacak siapa yang menjadi pemain pertama
    if (rand() % 2 == 0) {
```

```c
// Fungsi untuk mengosongkan langkah pemain
void freeBoard() {
    for (int i=0; i<SIZE; i++) {
        movesX[i] = -1;
        movesO[i] = -1;
    }
}
```

```c
// Fungsi rekursi jika pemain ingin bermain kembali
void playAgain() {
    char response;
    printf("\n\n\nApakah Anda ingin bermain lagi? (y/n): ");
    printf("\n>> ");
    scanf(" %c", &response);
    if (response == 'y' || response == 'Y') {
        freeBoard();
        clearScreen();
        playGame();
        playAgain();
    }
    else if (response == 'n' || response == 'N') {
        printf("\n\nTerima kasih telah bermain! Jangan lupa
    }
    // Jika input salah, fungsi playAgain akan dipanggil di
    else {
```

```c
// Fungsi untuk menampilkan menu pilihan kepada pemain
void mainMenu() {
    start();
    int menuChoice;
    // Looping sampai input menuChoice = 2
    do {
        printf("Apa yang ingin dilakukan? (1/2)?\n");
        printf("1. Lihat aturan permainan\n");
        printf("2. Mulai permainan baru\n");
        printf(">> ");
        scanf(" %d", &menuChoice);

        // Jika pemain memilih 1 maka penatunan akan ter
```



			
Total : 80			

NB : +10 poin kerapian dan kompleksitas, +10 poin kelengkapan laporan