TUGAS AKHIR PRAKTIKUM ALGORITMA PEMROGRAMAN

“MEMBUAT GAME MAZE”

Dosen pengampu :

Randi Proska Sandi, M.Sc

Disusun oleh :

Nama	**: Fattan Naufan Islami**
NIM	**: 23343037**
Prodi	**: Informatika**
Kode Kelas	**: INF1.62.1008**

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2023

A. Latar Belakang Pembuatan Program

Saya selalu terpesona oleh labirin dan teka-teki sejak saya masih kecil. Saya senang menantang diri saya sendiri dan menemukan cara terbaik untuk keluar dari labirin atau maze yang rumit. Saya juga senang bermain video game yang melibatkan maze ini, seperti The Legend of Zelda, dan Portal. Yang terlebih lagi di game Portal. Ialah suatu game yang sangat kreatif dan menakjubkan dimana player harus menggunakan suatu alat yang bisa menghasilkan teleportasi dari dinding ke dinding.

Saya memilih C sebagai bahasa untuk game saya karena merupakan bahasa pertama yang saya pelajari di universitas, jadi saya ingin mengasah kemampuan saya dan belajar lebih banyak. Saya ingin menerapkan keterampilan dan konsep yang saya pelajari dalam kursus C, seperti array, pointer, struktur, fungsi, dan penanganan file.

Saya mencari beberapa inspirasi untuk game labirin saya di internet. Saya menemukan beberapa artikel dan tutorial yang berguna tentang cara membuat dan menyelesaikan labirin menggunakan algoritma yang berbeda, seperti algoritma backtracking.

Proses pembuatan game maze ini melibatkan beberapa langkah, yaitu: merancang alur dan mekanik game, membuat struktur data dan fungsi-fungsi yang dibutuhkan, membuat tampilan grafis dan suara game, membuat level-level dan maze-maze yang berbeda, membuat fitur papan peringkat dan penyimpanan data, dan melakukan pengujian dan perbaikan game.

Kendala yang saya hadapi dalam pembuatan game maze ini adalah seperti menentukan algoritma yang tepat untuk membuat maze-maze yang acak dan bervariasi, mengatur warna dan ukuran teks dan simbol yang sesuai dengan konsol, membuat kode yang rapi dan mudah dibaca, dan menemukan sumber daya yang relevan dan terpercaya untuk belajar dan mengembangkan game.

Saya sangat bangga dan puas ketika menyelesaikan permainan labirin saya. Saya mengujinya beberapa kali dan menikmati memainkannya. Saya belajar banyak dari proyek ini dan meningkatkan keterampilan pemrograman C saya. Saya juga bersenang-senang di-kala membuat dan juga memecahkan labirin yang saya buat.

B. Unsur Pemograman

1) Looping

1.
```
for (int i = 0; i < UKURAN * 2; i++) {
    printf(WARNA_HIJAU"-" WARNA_RESET);

}
```

Di sini, variabel penghitung adalah i, yang diinisialisasi dengan nilai 0. Kondisi berhenti adalah i < UKURAN * 2, yang berarti looping akan berhenti jika i sudah mencapai atau melebihi nilai UKURAN * 2. Perubahan nilai variabel adalah i++, yang berarti nilai i akan bertambah satu setiap kali looping berlangsung. Di dalam looping, kode yang diulang adalah printf(WARNA_HIJAU"-" WARNA_RESET), yang akan mencetak karakter "-" dengan warna hijau sebanyak UKURAN * 2 kali.

2.
```
for (int i = 0; i < UKURAN; i++) {
     printf("\n" WARNA_HIJAU "|" WARNA_RESET);
     for (int j = 0; j < UKURAN; j++) {
         int Jarak = abs(PemainX - j) + abs(PemainY - i);
         if (Jarak == 0 || Jarak <= 4) {
             if (i == PemainY && j == PemainX) {
                 printf(WARNA_KUNING "P " WARNA_RESET);
             } else if (maze[i][j] == 1) {
                 printf(WARNA_SIAN"# "WARNA_RESET);
             } else {
                 printf(" ");
             }
         } else {
             printf(" ");
         }
     }
     printf(WARNA_HIJAU "|" WARNA_RESET);
```

Di sini, variabel penghitung untuk for loop pertama adalah i, yang diinisialisasi dengan nilai 0. Kondisi berhenti adalah i < UKURAN, yang berarti looping akan berhenti jika i sudah mencapai atau melebihi nilai UKURAN. Perubahan nilai variabel adalah i++, yang berarti nilai i akan bertambah satu setiap kali looping berlangsung. Di dalam looping, kode yang diulang adalah printf("\n" WARNA_HIJAU "|" WARNA_RESET), yang akan mencetak karakter "|" dengan warna hijau di awal setiap baris, dan for loop kedua, yang akan mengatur kolom maze.

Variabel penghitung untuk for loop kedua adalah j, yang diinisialisasi dengan nilai 0. Kondisi berhenti adalah j < UKURAN, yang berarti looping akan berhenti jika j sudah mencapai atau melebihi nilai UKURAN. Perubahan nilai variabel adalah j++, yang berarti nilai j akan bertambah satu setiap kali looping berlangsung.

Kode ini akan mencetak karakter "#" dengan warna sian jika posisi dinding adalah 1, karakter "P" dengan warna kuning jika posisi pemain adalah 0, atau spasi jika posisi dinding adalah 0. Kode ini juga akan menyembunyikan dinding yang berada di luar jangkauan penglihatan pemain, yaitu jika jarak antara pemain dan dinding lebih dari 4.

3.
```
for (int level = 0; level < JMLH_LVL; level++) {
    // Kode untuk menampilkan dan memainkan level
}
```

   Di sini, variabel penghitung adalah level, yang diinisialisasi dengan nilai 0. Kondisi berhenti adalah level < JMLH_LVL, yang berarti looping akan berhenti jika level sudah mencapai atau melebihi nilai JMLH_LVL. Perubahan nilai variabel adalah level++, yang berarti nilai level akan bertambah satu setiap kali looping berlangsung. Di dalam looping, kode yang diulang adalah kode untuk menampilkan dan memainkan level.

4.
```
while (PemainX != KeluarX || PemainY != KeluarY) {
    // Kode untuk mengambil input dan memindahkan pemain
}
```

   Di sini, kondisi berlanjut adalah PemainX != KeluarX || PemainY != KeluarY, yang berarti looping akan berlanjut selama posisi pemain belum sama dengan posisi keluar. Di dalam looping, kode yang diulang adalah kode untuk mengambil input dan memindahkan pemain.

5.
```
for (int i = SKORPemain - 1; i > SelipanPosisi; i--) {
```

   Digunakan untuk menggeser skor yang ada di papan peringkat. Looping ini menggunakan for loop, yang mengulang kode sebanyak SKORPemain - 1 - SelipanPosisi kali. For loop ini memiliki tiga komponen: variabel penghitung, kondisi berhenti, dan perubahan nilai variabel.

2) Decision Making

1.
```
char Gerak = getch();
int PosisiX = PemainX, PosisiY = PemainY;
switch (Gerak) {
    case 'W':
    case 'w':
        PosisiY--;
        break;
    case 'S':
    case 's':
        PosisiY++;
        break;
    case 'A':
    case 'a':
        PosisiX--;
        break;
    case 'D':
    case 'd':
        PosisiX++;
        break;
    case 'K':
    case 'k':
```

```
            printf(WARNA_PATMA"Mengakhiri Permainan...\n"WARNA_RESET);
            getch();
            return;
        default:
            continue;
    }
```

Untuk menentukan gerakan pemain, kode ini menggunakan switch case untuk memeriksa input dari user berdasarkan tombol yang ditekan. Jika user menekan W, S, A, atau D, maka kode ini akan mengubah posisi pemain sesuai dengan arah yang dipilih. Jika user menekan K, maka kode ini akan mengakhiri permainan. Jika user menekan tombol lain, maka kode ini akan mengabaikan input tersebut.

2.
```
printf(WARNA_HIJAU"Masukkan pilihanmu (1-3): "WARNA_RESET);
scanf("%d", &pilihan);
fflush(stdin);
switch (pilihan) {
    case 1:
        system("cls");
        PermainanMaze(level, &totalskor, papanperingkat);
        TampilanPapanPeringkat(papanperingkat);
        getch();
        SimpanPapanPeringkat(papanperingkat);
        system("cls");
        break;
    case 2:
        TampilanPapanPeringkat(papanperingkat);
        printf("\nTekan tombol apapun untuk kembali ke menu utama...");
        getch();
        break;
    case 3:
        printf(WARNA_PATMA"Keluar dari permainan...\n"WARNA_RESET);
        break;
    default:
        printf("Pilihan salah. Masukkanlah angka 1 - 3.\n");
        break;
}
```

Untuk menampilkan menu utama, kode ini menggunakan if, else if, dan else untuk memeriksa pilihan dari user. Jika user memilih 1, maka kode ini akan memulai permainan. Jika user memilih 2, maka kode ini akan menampilkan papan peringkat. Jika user memilih 3, maka kode ini akan keluar dari permainan. Jika user memilih angka lain, maka kode ini akan menampilkan pesan error.

3.
```
// Cari posisi untuk memasukkan skor pemain saat ini
int SelipanPosisi = -1;
for (int i = SKORPemain - 1; i >= 0; i--) {
    if (*totalskor > papanperingkat[i].skor) {
        SelipanPosisi = i;
    }
}
// Geser skor yang ada untuk memberi ruang bagi skor pemain saat ini
if (SelipanPosisi != -1) {
    for (int i = SKORPemain - 1; i > SelipanPosisi; i--) {
```

```
        strcpy(papanperingkat[i].NamaPemain,       papanperingkat[i       -
1].NamaPemain);
        papanperingkat[i].skor = papanperingkat[i - 1].skor;
    }
    strcpy(papanperingkat[SelipanPosisi].NamaPemain, NamaPemain);
    papanperingkat[SelipanPosisi].skor = *totalskor;
}
```

Untuk menyimpan papan peringkat, kode ini menggunakan for loop dan if untuk mencari posisi untuk memasukkan skor pemain saat ini. Jika skor pemain saat ini lebih besar dari skor pemain di posisi tertentu, maka kode ini akan menggeser skor yang ada untuk memberi ruang bagi skor pemain saat ini. Jika tidak, maka kode ini akan mengabaikan skor pemain saat ini.

3) Fungsi

1. Fungsi TampilanUtama adalah fungsi untuk membuat tampilan utama dari permainan, yang berisi menu pilihan, judul permainan, dan nama pembuat. Fungsi ini menggunakan perintah system("cls") untuk membersihkan layar, printf untuk mencetak teks dengan warna yang berbeda, dan #define untuk mendefinisikan kode warna ANSI.

2. Fungsi TampilanMaze adalah fungsi untuk membuat tampilan maze dengan keterbatasan penglihatan, yang berisi maze, pemain, skor, waktu, dan instruksi. Fungsi ini menggunakan parameter maze, PemainX, PemainY, skor, dan WaktuMulai untuk mengambil data dari fungsi lain. Fungsi ini juga menggunakan perintah clock untuk menghitung waktu, abs untuk menghitung jarak, printf untuk mencetak teks dengan warna yang berbeda, dan #define untuk mendefinisikan kode warna ANSI.

3. Fungsi PermainanMaze adalah fungsi untuk bermain game, yang berisi logika permainan, perhitungan skor, dan penyimpanan papan peringkat. Fungsi ini menggunakan parameter level, totalskor, dan papanperingkat untuk mengambil data dari fungsi lain. Fungsi ini juga menggunakan perintah getch untuk mengambil input dari user, switch untuk menentukan gerakan pemain, strcpy dan fgets untuk mengambil dan menyimpan nama pemain, PlaySound untuk memainkan suara, dan Sleep untuk menunda waktu.

4. Fungsi TampilanPapanPeringkat adalah fungsi untuk menampilkan papan peringkat, yang berisi peringkat, nama pemain, dan skor. Fungsi ini menggunakan parameter papanperingkat untuk mengambil data dari fungsi lain. Fungsi ini juga menggunakan perintah printf untuk mencetak teks dengan warna yang berbeda, dan #define untuk mendefinisikan kode warna ANSI.

5. Fungsi SimpanPapanPeringkat adalah fungsi untuk menyimpan papan peringkat ke file, yang berisi operasi file. Fungsi ini menggunakan parameter papanperingkat untuk mengambil data dari fungsi lain. Fungsi ini juga menggunakan perintah FILE, fopen, fprintf, fclose untuk membuka, menulis, dan menutup file.

6. Fungsi MemuatPapanPeringkat adalah fungsi untuk memuat papan peringkat dari file, yang berisi operasi file. Fungsi ini menggunakan parameter papanperingkat untuk mengambil data dari fungsi lain. Fungsi ini juga menggunakan perintah FILE, fopen, fscanf, fclose untuk membuka, membaca, dan menutup file.

7. Fungsi main adalah fungsi utama, yang berisi inisialisasi variabel, array, dan file, serta pemanggilan fungsi lain. Fungsi ini juga menggunakan perintah system("cls") untuk membersihkan layar, printf untuk mencetak teks dengan warna yang berbeda, scanf untuk mengambil input dari user, fflush untuk membersihkan input buffer, switch untuk menentukan pilihan menu, do-while untuk membuat loop, dan return untuk mengembalikan nilai.

4) Array

1. `int Level[JMLH_LVL][UKURAN][UKURAN]` Adalah deklarasi array berdimensi dua dengan tipe data int, nama array Level, jumlah baris JMLH_LVL, dan jumlah kolom UKURAN.

2. `int Level[JMLH_LVL][UKURAN][UKURAN] = { { {0, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {0, 0, 1, 0, 0, 0, 0, 0, 0, 1}, ..., {1, 1, 1, 1, 1, 1, 1, 0, 0, 0} }, { {0, 0, 1, 1, 1, 1, 1, 1, 1, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, ..., {1, 1, 1, 1, 1, 1, 1, 1, 1, 0} }, ..., { {0, 0, 1, 1, 1, 1, 1, 1, 1, 1}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 1}, ..., {1, 1, 1, 1, 1, 1, 1, 1, 1, 0} } }`
   Adalah inisialisasi array berdimensi dua dengan nama Level, yang memiliki JMLH_LVL baris dan UKURAN kolom, dan setiap elemen array berisi nilai 0 atau 1.

3. `Level[Tingkat][PosisiY][PosisiX]` Adalah akses elemen array Level dengan indeks baris Tingkat, indeks kolom PosisiY dan PosisiX. Indeks array dimulai dari 0 hingga jumlah baris atau kolom dikurangi 1.

5) Struct

1.
```
typedef struct {
    // Struktur data untuk menyimpan nama pemain dan skor pemain
    char NamaPemain [30]; // Variabel untuk menyimpan nama pemain
    int skor; // Variabel untuk menyimpan skor pemain
} Pemain; // Nama dari struktur data
```

   Di kode ini, struct Pemain digunakan untuk menyimpan nama dan skor dari pemain yang bermain permainan maze.

2. `Pemain papanperingkat[SKORPemain] = { 0 }; // Inisiialisasi papan peringka`

   Di kode ini, variabel papanperingkat adalah array dari tipe struct Pemain yang berukuran SKORPemain, yaitu 20. Array ini digunakan untuk menyimpan nama dan skor dari 20 pemain terbaik yang bermain permainan maze.

3. ```
   strcpy(papanperingkat[i].NamaPemain, papanperingkat[i - 1].NamaPemain); // Menggeser nama pemain
   papanperingkat[i].skor = papanperingkat[i - 1].skor; // Menggeser skor pemain
   ```

   Di kode ini, fungsi strcpy digunakan untuk menyalin string dari satu variabel ke variabel lain. Fungsi ini digunakan untuk menggeser nama pemain yang ada di papan peringkat, agar dapat memberi ruang bagi pemain yang memiliki skor lebih tinggi. Demikian juga, operator = digunakan untuk menggeser skor pemain yang ada di papan peringkat.

4. ```
   FILE *file; // Inisiialisasi file
   file = fopen("papanperingkat.txt", "w"); // Membuka file untuk menyimpan data papan peringkat
   if (file != NULL) { // Jika file tidak sama dengan NULL
       for (int i = 0; i < SKORPemain; i++) { // Looping untuk menyimpan data papan peringkat ke file
           fprintf(file, "%s %d\n", papanperingkat[i].NamaPemain, papanperingkat[i].skor); // Menyimpan data papan peringkat ke file
       }
       fclose(file); // Menutup file
   }
   ```

   Di kode ini, fungsi fopen digunakan untuk membuka file dengan nama "papanperingkat.txt" dan mode "w", yang berarti write atau menulis. Jika file berhasil dibuka, maka file akan ditunjuk oleh pointer file, yang merupakan variabel dari tipe FILE. Jika file gagal dibuka, maka pointer file akan bernilai NULL, yang menandakan adanya error.

6) File Handling

1. `file = fopen("papanperingkat.txt", "w");`

   Berarti membuka file papanperingkat.txt dengan mode menulis dan menyimpan pointer ke file dalam variabel file.

2. `fclose(file);`

   Berarti menutup file yang ditunjuk oleh variabel file.

3. ```
   fprintf(file, "%s %d\n", papanperingkat[i].NamaPemain, papanperingkat[i].skor);
   ```

   Berarti menulis nama pemain dan skor ke file dengan format tertentu dan baris baru.
   ```

4. `fscanf(file, "%s %d\n", papanperingkat[i].NamaPemain, &papanperingkat[i].skor);`

   Berarti membaca nama pemain dan skor dari file dengan format tertentu dan baris baru.

5. `PlaySound(TEXT("welcome.wav"), NULL, SND_FILENAME);`

   Berarti memainkan suara dari file welcome.wav dengan flag SND_FILENAME yang menunjukkan bahwa argumen pertama adalah nama file.

C. Penjelasan Aplikasi

1) Konstanta: Program ini mendefinisikan beberapa konstanta seperti UKURAN, JMLH_LVL, SKORPemain, dan WARNA_*. Konstanta ini digunakan untuk menentukan ukuran maze, jumlah level, jumlah pemain di papan peringkat, dan kode warna ANSI untuk menampilkan output yang berwarna.

2) Struktur data: Program ini mendefinisikan sebuah struktur data bernama Pemain, yang memiliki dua anggota: NamaPemain dan skor. Struktur data ini digunakan untuk menyimpan nama dan skor dari setiap pemain yang bermain permainan ini.

3) Fungsi: Program ini menggunakan beberapa fungsi untuk melakukan berbagai tugas, seperti:
   - TampilanUtama: Fungsi ini digunakan untuk menampilkan menu utama dari permainan, yang memiliki tiga pilihan: Mulai Permainan, Liat Papan Peringkat, dan Keluar.
   - TampilanMaze: Fungsi ini digunakan untuk menampilkan maze, pemain, skor, dan waktu yang telah berlalu. Fungsi ini juga menampilkan instruksi cara bermain dan batasan penglihatan pemain.
   - PermainanMaze: Fungsi ini digunakan untuk bermain permainan maze. Fungsi ini mengambil array level, skor total, dan papan peringkat sebagai parameter. Fungsi ini menggunakan loop untuk setiap level dan gerakan pemain. Fungsi ini juga menghitung skor, waktu, dan gerakan pemain. Fungsi ini juga meminta nama pemain dan memasukkannya ke papan peringkat jika skornya cukup tinggi.
   - TampilanPapanPeringkat: Fungsi ini digunakan untuk menampilkan papan peringkat, yang berisi nama dan skor dari pemain terbaik.
   - SimpanPapanPeringkat: Fungsi ini digunakan untuk menyimpan papan peringkat ke file bernama papanperingkat.txt. Fungsi ini menggunakan file pointer dan fprintf untuk menulis data ke file.
   - MemuatPapanPeringkat: Fungsi ini digunakan untuk memuat papan peringkat dari file bernama papanperingkat.txt. Fungsi ini menggunakan file pointer dan fscanf untuk membaca data dari file.
   - main: Fungsi ini adalah fungsi utama dari program. Fungsi ini menginisialisasi skor total, papan peringkat, dan array level. Fungsi ini juga memuat papan peringkat sebelumnya dari file. Fungsi ini juga menggunakan loop untuk menampilkan menu utama dan switch case untuk setiap pilihan menu. Fungsi ini juga memainkan suara dan musik saat permainan dimulai dan selesai.

4) Array: Program ini menggunakan array dua dimensi untuk menyimpan maze. Setiap elemen array memiliki nilai 0 atau 1, yang menunjukkan ruang kosong atau dinding. Program ini juga menggunakan array satu dimensi untuk menyimpan papan peringkat, yang berisi struktur data Pemain.

D. Langkah – Langkah Pembuatan Aplikasi

1) Buatlah struktur data untuk menyimpan maze

Struktur data yang digunakan untuk menyimpan maze adalah array 3 dimensi. Setiap dimensi array mewakili satu baris, kolom, atau karakter dari maze.

```
261     int Level[JMLH_LVL][UKURAN][UKURAN] = { // Inisialisasi maze
```

2) Buatlah fungsi untuk menampilkan menu utama

Fungsi ini akan menampilkan menu utama permainan dan meminta user untuk memilih menu.

```
41  // Fungsi membuat Tampilan Utama dari Permainan
42  void TampilanUtama() {
43      system("cls"); // Membersihkan konsol atau tampilan Command Promt (CMD)
44      printf(WARNA_SIAN"=====================================\n");
45      printf("|              MAZE GAME            |\n");
46      printf("=====================================\n");
47      printf("|1. Mulai Permainan                 |\n");
48      printf("|2. Liat Papan Peringkat            |\n");
49      printf("|3. Keluar                          |\n" );
50      printf("=====================================\n");
51      printf("|"WARNA_RESET);
52      printf(WARNA_MERAH"Dibuat oleh: Fattan Naufan Islami  "WARNA_RESET);
53      printf(WARNA_SIAN"|\n");
54      printf("=====================================\n\n"WARNA_RESET);
55
56  }
```

3) Buatlah fungsi untuk menampilkan maze

Fungsi ini akan menerima array maze, posisi pemain, dan skor sebagai parameter. Fungsi ini akan menampilkan maze dengan karakter 'X' untuk pemain dan karakter ' ' untuk ruang kosong.

```c
// Sebuah fungsi untuk membuat maze dengan keterbatasan penglihatan
void TampilanMaze(int maze[UKURAN][UKURAN], int PemainX, int PemainY, int skor, clock_t WaktuMulai) { // Mengambil maze, posisi pemain, skor,
    system("cls"); // Membersihkan konsol atau tampilan Command Promt (CMD)
    clock_t WaktuSekarang = clock(); // Mengambil waktu saat ini
    double WaktuBerlalu = ((double)(WaktuSekarang - WaktuMulai)) / CLOCKS_PER_SEC; // Menghitung waktu yang telah berlalu

    // Membuat batasan bagian atas dari kotak
    printf(WARNA_MERAH "+" WARNA_RESET); // Bagian batasan kiri dari kotak
    for (int i = 0; i < UKURAN * 2; i++) { // Looping untuk membuat batasan atas dari kotak
        printf(WARNA_HIJAU"-" WARNA_RESET); // Bagian batasan atas dari kotak
    }
    printf(WARNA_MERAH"+" WARNA_RESET); // Bagian batasan kanan dari kotak
    printf(WARNA_SIAN " Skor: %d" WARNA_RESET, skor); // Menampilkan skor

    // Membuat maze dan pemain di sekitar kotak
    for (int i = 0; i < UKURAN; i++) { // Looping untuk membuat maze dan pemain
        printf("\n" WARNA_HIJAU "|" WARNA_RESET); // Bagian batasan kiri dari kotak
        for (int j = 0; j < UKURAN; j++) { // Looping untuk membuat maze dan pemain
            int Jarak = abs(PemainX - j) + abs(PemainY - i); // Menghitung jarak antara pemain dan dinding
            if (Jarak == 0 || Jarak <= 4) { // Jika jarak antara pemain dan dinding adalah 0 atau kurang dari sama dengan 4
                if (i == PemainY && j == PemainX) { // Jika posisi pemain sama dengan posisi dinding
                    printf(WARNA_KUNING "P " WARNA_RESET); // Wujud dari pemain
                } else if (maze[i][j] == 1) { // Jika posisi dinding sama dengan 1
                    printf(WARNA_SIAN"# "WARNA_RESET); // Dinding yang bakal ditampilin ketika berada di jangkauan yang bisa dilihat
                } else { // Jika posisi dinding sama dengan 0
                    printf("  "); // Ruang kosong
                }
            } else { // Jika jarak antara pemain dan dinding lebih dari 4
                printf("  "); // Menyembunyikan dinding yang berada diluar jangkauan penglihatan
            }
        }
        printf(WARNA_HIJAU "|" WARNA_RESET); // Bagian batasan kanan dari kotak
    }

    // Bagian batasan bawah dari kotak
    printf("\n" WARNA_MERAH "+" WARNA_RESET); // Bagian batasan kiri dari kotak
    for (int i = 0; i < UKURAN * 2; i++) { // Looping untuk membuat batasan bawah dari kotak
        printf(WARNA_HIJAU"-"WARNA_RESET); // Bagian batasan bawah dari kotak
    }
    printf(WARNA_MERAH"+" WARNA_RESET); // Bagian batasan kanan dari kotak
    printf(WARNA_SIAN " \nWaktu: %.2f Detik" WARNA_RESET, WaktuBerlalu); // Menampilkan waktu yang telah berlalu di bawah layar permainan
```

```c

    // Menampilkan instruksi cara bermain di bawah layar permainan
    printf(WARNA_PATMA "\n\nCara Bermain:\n");
    printf("W - Maju keatas\n");
    printf("S - Maju Kebawah\n");
    printf("A - Maju kekiri\n");
    printf("D - Maju kekanan\n");
    printf("K - Keluar\n\n" WARNA_RESET);\
}
```

4) Buatlah fungsi untuk memainkan permainan

Fungsi ini akan menerima array maze dan skor pemain sebagai parameter. Fungsi ini akan menampilkan maze dan meminta user untuk memasukkan gerakan. Fungsi ini juga akan menghitung skor pemain dan menyimpannya ke dalam variabel global.

```
// Fungsi untuk bermain game
void PermainanMaze(int Level[JMLH_LVL][UKURAN][UKURAN], int* totalskor, Pemain papanperingkat[]) {
    int skor = 0; // Inisiialisasi skor untuk Tingkat saat ini
    clock_t WaktuMulai, endTime; // Inisiialisasi waktu mulai dan waktu selesai
    double WaktuBerlalu; // Inisialisasi waktu yang dibutuhkan untuk menyelesaikan Tingkat saat ini

    for (int Tingkat = 0; Tingkat < JMLH_LVL; Tingkat++) { // Looping untuk setiap Tingkat
        int PemainX = 0, PemainY = 0; // Inisialisasi posisi pemain
        int KeluarX = UKURAN - 1, KeluarY = UKURAN - 1; // Inisiialisasi posisi Keluar
        int Gerakan = 0; // Variabel untuk menghitung gerakan pemain
        clock_t WaktuMulai = clock(); // Memulai waktu untuk Tingkat

        while (PemainX != KeluarX || PemainY != KeluarY) { // Looping untuk setiap gerakan pemain
            TampilanMaze(Level[Tingkat], PemainX, PemainY, *totalskor, WaktuMulai); // Menampilkan maze, pemain, dan skor
            char Gerak = getch(); // Mengambil input dari user berdasarkan tombol yang ditekan
            int PosisiX = PemainX, PosisiY = PemainY; // Inisiialisasi posisi baru pemain

            switch (Gerak) { // Switch case untuk setiap gerakan pemain
                case 'W':
                case 'w':
                    PosisiY--;
                    break;
                case 'S':
                case 's':
                    PosisiY++;
                    break;
                case 'A':
                case 'a':
                    PosisiX--;
                    break;
                case 'D':
                case 'd':
                    PosisiX++;
                    break;
                case 'K':
                case 'k':
                    printf(WARNA_PATMA"Mengakhiri Permainan...\n"WARNA_RESET);
                    getch();
                    return;
                default:
```

```
                    continue;
            }

            if (PosisiX >= 0 && PosisiX < UKURAN && PosisiY >= 0 && PosisiY < UKURAN && Level[Tingkat][PosisiY][PosisiX] ==
                PemainX = PosisiX;  // Memindahkan pemain ke posisi baru
                PemainY = PosisiY;  // Memindahkan pemain ke posisi baru
                Gerakan++; // Menambahkan hitungan gerakan yang valid
            }
        }

        clock_t endTime = clock(); // Menghentikan waktu untuk Level
        WaktuBerlalu = ((double)(endTime - WaktuMulai)) / CLOCKS_PER_SEC; // Kalkulasikan waktu yang dibutuhkan untuk menyel

        // Mengkalkulasi skor berdasarkan waktu dan gerakan (rumus sederhana)
        skor = 1000 - (Gerakan * 10) - (int)(WaktuBerlalu * 10);
        if (skor < 0) { // Jika skor negatif
            skor = 0; // Memastiin skor tidak negatif
        }

        // Menambah skor pemain saat ini ke skor total
        *totalskor += skor;

        // Mengoper skor ke fungsi TampilanMaze
        TampilanMaze(Level[Tingkat], PemainX, PemainY, *totalskor, WaktuMulai);

        printf(WARNA_KUNING "Selamat! Kamu sudah mencapai jalan keluar dari Level %d!\n" WARNA_RESET, Tingkat + 1); // Menam
        printf(WARNA_SIAN"Jumlah Gerakan : %d\n", Gerakan); // Menampilkan jumlah gerakan yang dibutuhkan untuk menyelesaika
        printf("Waktu dibutuhkan : %.2f seconds\n", WaktuBerlalu); // Menampilkan waktu yang dibutuhkan untuk menyelesaikan
        printf("Skor kamu di Level %d ialah : %d\n"WARNA_RESET, Tingkat + 1, skor); // Menampilkan skor pemain saat ini
        printf(WARNA_HIJAU"Tekan tombol apapun untuk melanjutkan ke Level selanjutnya...\n"WARNA_RESET); // Menampilkan pesa
        getch();

        system("cls");
    }

    printf(WARNA_SIAN "Selamat! Kamu sudah menamatkan semua tantangan\n" WARNA_RESET);
    printf(WARNA_MERAH"Total skor dari semua Level : %d\n"WARNA_RESET, *totalskor); // Menampilkan skor total
    PlaySound(TEXT("Welldone.wav"), NULL, SND_FILENAME); // Memainkan suara saat menyelesaikan permainan
    PlaySound(TEXT("Victory.wav"), NULL, SND_FILENAME); // Memainkan suara saat menyelesaikan permainan

    printf(WARNA_HIJAU"Masukkan namamu yang akan ditampilkan di papan peringkat: "WARNA_RESET); // Meminta nama pemain
    char NamaPemain[50]; // Inisialisasi nama pemain
```

```
    fgets(NamaPemain, sizeof(NamaPemain), stdin); // Mengambil input nama pemain
    NamaPemain[strcspn(NamaPemain, "\n")] = '\0'; // Membuang karakter newline dari NamaPemain

    // Cari posisi untuk memasukkan skor pemain saat ini
    int SelipanPosisi = -1; // Inisialisasi posisi untuk memasukkan skor pemain saat ini
    for (int i = SKORPemain - 1; i >= 0; i--) { // Looping untuk mencari posisi untuk memasukkan skor pemain saat ini
        if (*totalskor > papanperingkat[i].skor) { // Jika skor pemain saat ini lebih besar dari skor pemain di posisi i
            SelipanPosisi = i; // Mengatur posisi untuk memasukkan skor pemain saat ini
        }
    }

    // Geser skor yang ada untuk memberi ruang bagi skor pemain saat ini
    if (SelipanPosisi != -1) { // Jika posisi untuk memasukkan skor pemain saat ini tidak sama dengan -1
        for (int i = SKORPemain - 1; i > SelipanPosisi; i--) { // Looping untuk menggeser skor yang ada untuk memberi ruang
            strcpy(papanperingkat[i].NamaPemain, papanperingkat[i - 1].NamaPemain); // Menggeser nama pemain
            papanperingkat[i].skor = papanperingkat[i - 1].skor; // Menggeser skor pemain
        }
        strcpy(papanperingkat[SelipanPosisi].NamaPemain, NamaPemain); // Memasukkan nama pemain saat ini ke papanperingkat
        papanperingkat[SelipanPosisi].skor = *totalskor; // Memasukkan skor pemain saat ini ke papanperingkat
    }

}
```

5) Buatlah fungsi untuk menampilkan papan peringkat

Fungsi ini akan menerima array papan peringkat sebagai parameter. Fungsi ini akan menampilkan papan peringkat dengan nama pemain dan skornya.

```
// Fungsi untuk menampilkan papan peringkat
void TampilanPapanPeringkat(Pemain papanperingkat[]) { // Mengambil papan peringkat sebagai parameter
    system("cls"); // Membersihkan konsol atau tampilan Command Promt (CMD)
    printf(WARNA_KUNING "\t*** Papan Peringkat ***\n\n"); // Menampilkan judul papan peringkat
    printf("Peringkat\tNama Pemain\tSkor\n"); // Menampilkan header papan peringkat

    for (int i = 0; i < SKORPemain; i++) { // Looping untuk menampilkan papan peringkat
        printf("%d.\t\t%s\t\t%d\n", i + 1, papanperingkat[i].NamaPemain, papanperingkat[i].skor); // Menampilkan papan peringkat
    }
}
```

6) Buatlah fungsi untuk menyimpan papan peringkat ke file

Fungsi ini akan menerima array papan peringkat sebagai parameter. Fungsi ini akan menyimpan papan peringkat ke file dengan nama "papanperingkat.txt".

```
// Fungsi untuk menyimpan papan peringkat ke file
void SimpanPapanPeringkat(Pemain papanperingkat[]) { // Mengambil papan peringkat sebagai parameter
    FILE *Berkas; // Inisiialisasi file
    Berkas = fopen("papanperingkat.txt", "w"); // Membuka file untuk menyimpan data papan peringkat
    if (Berkas != NULL) { // Jika file tidak sama dengan NULL
        for (int i = 0; i < SKORPemain; i++) { // Looping untuk menyimpan data papan peringkat ke file
            fprintf(Berkas, "%s %d\n", papanperingkat[i].NamaPemain, papanperingkat[i].skor); // Menyimpan data papan peringkat ke file
        }
        fclose(Berkas); // Menutup file
    } else { // Jika file tidak ada
        printf("Gagal membuka file untuk menyimpan data papan peringkat\n"); // Menampilkan pesan error
        printf("Coba buat file papanperingkat.txt di direktori yang sama dengan program MazeInd.exe\n"); // Menampilkan pesan error
        getch(); // Menunggu user untuk menekan tombol apapun
    }
}
```

7) Buatlah fungsi untuk membaca papan peringkat dari file

Fungsi ini akan menerima array papan peringkat sebagai parameter. Fungsi ini akan membaca papan peringkat dari file dengan nama "papanperingkat.txt".

```
// Fungsi untuk memuat papan peringkat dari file
void MemuatPapanPeringkat(Pemain papanperingkat[]) { // Mengambil papan peringkat sebagai parameter
    FILE *file; // Inisialisasi file
    file = fopen("papanperingkat.txt", "r"); // Membuka file untuk memuat data papan peringkat
    if (file != NULL) { // Jika file ada
        for (int i = 0; i < SKORPemain; i++) { // Looping untuk memuat data papan peringkat dari file
            fscanf(file, "%s %d\n", papanperingkat[i].NamaPemain, &papanperingkat[i].skor); // Memuat data papan peringkat dari file
        }
        fclose(file);  // Menutup file
    } else { // Jika file tidak ada
        printf("File tidak ada atau tidak bisa dibuka\n"); // Menampilkan pesan error
    }
}
```

8) Buatlah fungsi utama untuk permainan

Fungsi ini akan memanggil fungsi-fungsi lainnya untuk menjalankan permainan juga menjalankan musik, suara, membuat level menggunakan 0,1 yang bisa dirujuki balik ke fungsi untuk menampilkan maze untuk cara kerjanya, menggunakan sleep di bagian loading juga membuat pilihan menu.

```
int main() { // Fungsi utama
    int totalskor = 0; // Inisialisasi skor total
    Pemain papanperingkat[SKORPemain] = { 0 }; // Inisialisasi papan peringkat
    MemuatPapanPeringkat(papanperingkat); // Memuat papan peringkat sebelumnya dari f

    int Level[JMLH_LVL][UKURAN][UKURAN] = { // Inisialisasi maze
                    { // Level 1
                    {0, 1, 1, 1, 1, 1, 1, 1, 1, 1},
                    {0, 0, 1, 0, 0, 0, 0, 0, 0, 1},
                    {1, 0, 0, 0, 1, 1, 1, 1, 0, 1},
                    {1, 1, 1, 0, 1, 0, 0, 0, 0, 1},
                    {1, 0, 0, 0, 0, 0, 1, 1, 1, 1},
                    {1, 1, 1, 1, 1, 0, 1, 0, 0, 1},
                    {1, 0, 0, 0, 0, 0, 0, 0, 1, 1},
                    {1, 1, 1, 1, 1, 1, 1, 0, 1, 0},
                    {1, 0, 0, 0, 0, 0, 0, 0, 1, 0},
                    {1, 1, 1, 1, 1, 1, 1, 0, 0, 0}
                },
                    { // Level 2
                    {0, 1, 1, 1, 1, 1, 1, 1, 1, 1},
                    {0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
                    {1, 1, 1, 1, 1, 0, 1, 1, 0, 1},
                    {1, 0, 0, 0, 0, 0, 0, 1, 0, 1},
                    {1, 0, 1, 1, 1, 1, 0, 1, 0, 1},
                    {1, 0, 1, 0, 0, 0, 0, 1, 1, 1},
                    {1, 1, 1, 0, 1, 1, 1, 1, 0, 1},
                    {1, 0, 0, 0, 0, 0, 0, 0, 0, 1},
                    {1, 1, 0, 1, 1, 1, 0, 1, 1, 1},
                    {1, 0, 0, 1, 1, 1, 0, 0, 0, 0}
                },
                    { // Level 3
                    {0, 1, 1, 1, 1, 1, 1, 1, 1, 1},
                    {0, 1, 0, 0, 0, 0, 0, 0, 0, 1},
                    {0, 1, 1, 1, 1, 1, 1, 1, 0, 1},
                    {0, 0, 0, 0, 0, 0, 0, 1, 0, 1},
                    {0, 1, 1, 1, 1, 1, 0, 1, 0, 1},
                    {0, 0, 0, 0, 0, 0, 0, 1, 0, 1},
                    {0, 1, 1, 0, 1, 1, 1, 1, 0, 1},
                    {1, 0, 0, 0, 0, 0, 0, 0, 0, 1},
                    {1, 1, 0, 0, 1, 1, 1, 1, 1, 1},
                    {1, 1, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
                },
                    { // Level 4
                    {0, 1, 1, 1, 1, 1, 1, 1, 1},
                    {0, 1, 0, 0, 0, 0, 0, 1, 1},
                    {0, 1, 1, 1, 1, 1, 1, 1, 1},
                    {0, 0, 0, 0, 0, 0, 0, 1, 1},
                    {0, 0, 1, 1, 0, 1, 0, 1, 1},
                    {1, 0, 1, 0, 0, 0, 0, 1, 1},
                    {1, 1, 1, 0, 0, 1, 1, 1, 1},
                    {1, 0, 0, 0, 0, 0, 0, 1, 1},
                    {1, 1, 1, 1, 0, 1, 1, 1, 1},
                    {1, 1, 1, 1, 0, 0, 0, 0, 0}
                },
                    { // Level 5
                    {0, 0, 0, 0, 0, 0, 1, 1, 1},
                    {1, 1, 0, 1, 0, 0, 1, 0, 1},
                    {1, 1, 0, 1, 1, 1, 1, 0, 1},
                    {1, 0, 0, 0, 0, 0, 0, 1, 1},
                    {1, 0, 0, 0, 1, 1, 0, 1, 1},
                    {1, 0, 1, 0, 0, 0, 0, 1, 1},
                    {1, 1, 1, 0, 1, 1, 1, 1, 1},
                    {1, 0, 0, 0, 0, 0, 0, 0, 1},
                    {1, 1, 1, 0, 1, 0, 1, 1, 1},
                    {1, 1, 0, 0, 1, 0, 0, 1, 0}
                },
            };

        PlaySound(TEXT("welcome.wav"), NULL, SND_FILENAME); // Memainkan suara saat memulai permainan
        system("cls"); // Membersihkan konsol atau tampilan Command Promt (CMD)
        printf(WARNA_MERAH "Now Loading" WARNA_RESET); // Menampilkan tulisan Now Loading
        Sleep(500); // Menunggu 0.5 detik
        printf(WARNA_KUNING ". " WARNA_RESET); // Menampilkan titik
        Sleep(500); // Menunggu 0.5 detik
        printf(WARNA_HIJAU ". " WARNA_RESET); // Menampilkan titik
        Sleep(500); // Menunggu 0.5 detik
        printf(WARNA_PATMA ". " WARNA_RESET); // Menampilkan titik
        Sleep(500); // Menunggu 0.5 detik
        printf(WARNA_BIRU ". " WARNA_RESET); // Menampilkan titik
        Sleep(500); // Menunggu 0.5 detik
        printf(".\n"); // Menampilkan titik

        system("cls"); // Membersihkan konsol atau tampilan Command Promt (CMD)
```

```
    printf(WARNA_MERAH "Selamat datang di permainan Maze\n" WARNA_RESET); // Menampilkan judul permainan
    printf(WARNA_HIJAU "Tekan tombol apapun untuk memulai permainan..." WARNA_RESET); // Menampilkan pesan untuk memulai permainan

    getch(); // Menunggu user untuk menekan tombol apapun
    system("cls"); // Membersihkan konsol atau tampilan Command Promt (CMD)
    PlaySound(TEXT("Start.wav"), NULL, SND_FILENAME); // Memainkan suara saat memulai permainan
    PlaySound(TEXT("music.wav"), NULL, SND_FILENAME | SND_ASYNC | SND_LOOP); // Memainkan musik saat permainan dimulai

    int pilihan; // Inisialisasi pilihan menu

    do { // Looping untuk menampilkan menu utama
        TampilanUtama(); // Menampilkan menu utama
        printf(WARNA_HIJAU"Masukkan pilihanmu (1-3): "WARNA_RESET); // Menampilkan pesan untuk memilih menu
        scanf("%d", &pilihan); // Mengambil input dari user
        fflush(stdin); // Membersihkan input buffer

        switch (pilihan) { // Switch case untuk setiap pilihan menu
            case 1:
                system("cls"); // Membersihkan konsol atau tampilan Command Promt (CMD)
                PermainanMaze(Level, &totalskor, papanperingkat); // Memberi array ke fungsi PermainanMaze
                TampilanPapanPeringkat(papanperingkat); // Melihatkan papan peringkat setelah permainan selesai
                getch(); // Menunggu user untuk menekan tombol apapun
                SimpanPapanPeringkat(papanperingkat); // Menyimpan papan peringkat yang telah diperbarui ke file
                system("cls"); // Membersihkan konsol atau tampilan Command Promt (CMD)
                break; // Keluar dari switch case
            case 2: // Jika user memilih untuk melihat papan peringkat
                TampilanPapanPeringkat(papanperingkat); // Melihatkan papan peringkat
                printf("\nTekan tombol apapun untuk kembali ke menu utama..."); // Menampilkan pesan untuk kembali ke menu utama
                getch(); // Menunggu user untuk menekan tombol apapun
                break; // Keluar dari switch case
            case 3: // Jika user memilih untuk keluar dari permainan
                printf(WARNA_PATMA"Keluar dari permainan...\n"WARNA_RESET); // Menampilkan pesan keluar dari permainan
                break; // Keluar dari switch case
            default: // Jika input tidak valid
                printf(WARNA_PATMA"Pilihan salah. Masukkanlah angka 1 - 3.\n"WARNA_RESET); // Menampilkan pesan error
                printf(WARNA_MERAH"Tekan tombol apapun untuk balik ke menu..." WARNA_RESET);
                getch();
                break; // Keluar dari switch case
        }
    } while (pilihan != 3); // Looping untuk menampilkan menu utama

    return 0; // Mengembalikan nilai 0
}
```

E. Referensi

- Sirait, R. B. (2013). Perancangan Aplikasi Game Labirin Dengan Menggunakan Algoritma Backtracking. Volume: V, Nomor, 2.