RESPONSI

PRAKTIKUM KONSEP PEMROGRAMAN

2023

IDENTITAS

Nama : Mohammad Nazhiif Al-Ghoniy

NIM : L0123084

Kelas : C

Judul Program : C_Responsi1KP_L0123084_Nazhiif_01.c

Deskripsi Program : Game Minesweeper menggunakan bahasa C

Dokumentasi Program

Analisis Kode

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<string.h>
#include<stdbool.h>
```

#include adalah direktif preprosesor yang digunakan untuk mengimpor pustaka atau header file ke dalam program. Dalam kode ini, beberapa pustaka yang diimpor adalah:

- stdio.h: Pustaka standar untuk fungsi input/output standar.
- stdlib.h: Pustaka standar yang digunakan untuk alokasi memori dan fungsi-fungsi terkait. Pada kode di atas digunakan untuk rand().
- time.h: Pustaka untuk mengakses fungsi-fungsi terkait waktu dan random seperti time().
- string.h: Pustaka standar yang berhubungan dengan manipulasi string. Pada kode di atas digunakan untuk memset()
- stdbool.h: Pustaka standar yang menyediakan tipe data bool dan konstanta true serta false.

```c
#define MAXSISI 10
#define MAXBOM 10

int SISI_PAPAN = 9;
int BOM = 10;
```

- #define MAXSISI 10 dan #define MAXBOM 10 adalah direktif preprosesor yang mendefinisikan dua konstanta. MAXSISI digunakan untuk menentukan ukuran maksimal sisi papan permainan yaitu 10 (seharusnya 9, 10 digunakan agar ada sisa alokasi tempat pada array), sedangkan MAXBOM digunakan untuk menentukan jumlah maksimal bom pada papan yaitu 10 bom.
- int SISI_PAPAN = 9; adalah deklarasi variabel global yang menginisialisasi ukuran sisi papan permainan menjadi 9. Ini adalah ukuran papan permainan yang akan digunakan dalam permainan Minesweeper.
- int BOM = 10; adalah deklarasi variabel global yang menginisialisasi jumlah bom pada papan menjadi 10. Ini adalah jumlah bom yang akan ditempatkan di papan permainan.

```c
int main (){
    banner();

    mainGame ();

    return 0;
}
```

- Memanggil fungsi banner() yang mencetak pesan selamat datang di layar untuk memberi pemain informasi awal tentang permainan.
- Memanggil fungsi mainGame() yang mengatur alur permainan Minesweeper.

```c
void banner (){
    printf("========        SELAMAT DATANG        ========\n");
    printf("======== DI PERMAINAN MINESWEEPER 9x9 ========\n");
    printf("========       SELAMAT
BERMAIN        ========\n\n");

    return;
```

```
}
```

- Ini adalah pesan selamat datang yang akan dilihat pemain sebelum mereka mulai bermain Minesweeper.

```
// Fungsi untuk memainkan minesweeper
void mainGame (){
    // Variabel boolean untuk menentukan
    // sudah atau belum berakhirnya game
    bool gameOver = false;

    // char realBoard untuk menyimpan posisi semua bom
    // dan posisi aman
    // playerBoard hanya bisa dilihat pemain
    char realBoard[MAXSISI][MAXSISI],
playerBoard[MAXSISI][MAXSISI];

    // sisa move yang bisa dilakukan
    int movesLeft = SISI_PAPAN * SISI_PAPAN - BOM;
    int x, y;
    int bom[MAXBOM][2]; // menyimpan koordinat semua bom

    // mengisi array
    inisialisasi (realBoard, playerBoard);

    // mengisi posisi bom dengan acak
    placeBom (bom, realBoard);

    int movePertama = 0;

    // selama belum gameOver
    while (gameOver == false){
        cetakPapan (playerBoard);
        pilihPosisi (&x, &y);

        // pilihan pertama dijamin aman
        if (movePertama == 0){
```

```
            if (isBom (x, y, realBoard) == true)
                replaceBom (x, y, realBoard);
        }

        movePertama ++;

        gameOver = Minesweeper (playerBoard, realBoard, bom, x,
y, &movesLeft);

        if ( (gameOver == false) && (movesLeft == 0) ){
            printf ("\nKAMU MENANG!\n\n");
            gameOver = true;
        }
    }
    return;
}
```

Fungsi mainGame() adalah inti dari permainan Minesweeper. Di dalam fungsi ini, semua logika permainan Minesweeper diimplementasikan. Berikut penjelasan singkat tentang bagian-bagian utama dari fungsi ini:

- bool gameOver = false;: Variabel gameOver digunakan untuk menentukan apakah permainan sudah berakhir atau belum. Pada awal permainan, inisialisasi dengan false, yang berarti permainan belum berakhir.
- char realBoard[MAXSISI][MAXSISI], playerBoard[MAXSISI][MAXSISI];: Dua array 2D realBoard dan playerBoard digunakan untuk merepresentasikan papan permainan Minesweeper. realBoard digunakan untuk menyimpan posisi semua bom dan posisi aman, sementara playerBoard hanya dilihat oleh pemain dan menunjukkan sel-sel yang telah diungkap.
- int movesLeft = SISI_PAPAN * SISI_PAPAN - BOM;: Variabel movesLeft menghitung sisa langkah yang dapat dilakukan oleh pemain. Jumlah sisa langkah dihitung dengan mengurangkan jumlah bom (BOM) dari total sel di papan (SISI_PAPAN * SISI_PAPAN).

- int x, y;: Variabel x dan y digunakan untuk menyimpan posisi yang dipilih oleh pemain.
- int bom[MAXBOM][2];: Array bom digunakan untuk menyimpan koordinat dari semua bom pada papan. Setiap baris array ini mewakili koordinat bom pada papan.
- inisialisasi(realBoard, playerBoard);: Fungsi inisialisasi() digunakan untuk mengisi papan permainan dengan karakter awal yaitu - dan melakukan persiapan awal sebelum memulai permainan.
- placeBom(bom, realBoard);: Fungsi placeBom() digunakan untuk menempatkan bom secara acak di papan permainan. Ini akan menentukan posisi bom-bom di papan.
- int movePertama = 0;: Variabel movePertama digunakan untuk melacak apakah pemain sudah membuat langkah pertama atau belum.
- Loop while (gameOver == false) adalah inti permainan Minesweeper. Di dalam loop ini, pemain akan terus bermain hingga permainan berakhir (gameOver menjadi true).
- gameOver = Minesweeper(playerBoard, realBoard, bom, x, y, &movesLeft);: Fungsi Minesweeper() digunakan untuk mengelola langkah pemain, mengungkap sel, dan menghitung hasil permainan. Hasil dari fungsi ini akan mempengaruhi nilai gameOver.
- Setelah loop selesai, terdapat pengecekan apakah permainan berakhir dengan kemenangan atau kalah, dan pesan sesuai dengan hasilnya akan dicetak.

Fungsi mainGame() adalah inti dari permainan Minesweeper dan bertanggung jawab untuk mengontrol seluruh alur permainan.

```
void inisialisasi (char realBoard[MAXSISI][MAXSISI], char
playerBoard[MAXSISI][MAXSISI]){
    // set seed untuk fungsi rand()
    srand(time (NULL));

    // mengisi semua array agar kosong
    for (int i = 0; i < SISI_PAPAN; i++){
        for (int j = 0; j < SISI_PAPAN; j++){
            playerBoard[i][j] = realBoard[i][j] = '-';
        }
    }

    return;
```

```
}
```

- srand(time (NULL));: Fungsi srand() digunakan untuk menginisialisasi generator angka acak dengan nilai yang berubah-ubah berdasarkan waktu saat program dijalankan.
- Nested for loop digunakan untuk mengisi seluruh papan dengan karakter '-'. Variabel i digunakan untuk mengiterasi melalui baris, dan variabel j digunakan untuk mengiterasi melalui kolom papan. Seluruh papan permainan, baik playerBoard maupun realBoard, diinisialisasi dengan karakter '-'.

```
void placeBom(int bom[MAXBOM][2], char
realBoard[MAXSISI][MAXSISI]){

    bool adaBom[MAXSISI*MAXSISI];

    // set isi dari array adaBom dengan nilai false
    memset (adaBom, false, sizeof (adaBom));

    for (int i = 0; i < BOM; ){
        int random = rand() % (SISI_PAPAN*SISI_PAPAN);
        int x = random / SISI_PAPAN;
        int y = random % SISI_PAPAN;

        if (adaBom[random] == false){
            // indeks baris bom
            bom[i][0]= x;
            // indeks kolom bom
            bom[i][1] = y;

            // tempatkan bom pada posisi x,y
            realBoard[bom[i][0]][bom[i][1]] = '*';

            // sekarang sudah terisi bom
            adaBom[random] = true;
            i++;
        }
```

```
    }

    return;
}
```

- bool adaBom[MAXSISI*MAXSISI];: Dalam fungsi ini, sebuah array boolean adaBom digunakan untuk melacak apakah suatu sel di papan sudah berisi bom atau belum. Array ini memiliki ukuran yang cukup untuk mewakili semua sel di papan permainan (yaitu MAXSISI * MAXSISI).
- memset(adaBom, false, sizeof(adaBom));: Fungsi memset digunakan untuk mengisi seluruh array adaBom dengan nilai false. Ini dilakukan untuk memastikan bahwa awalnya tidak ada bom di sel mana pun di papan permainan.
- Loop for dengan variabel i digunakan untuk menempatkan bom sebanyak BOM kali pada papan. Loop ini akan berjalan hingga semua bom ditempatkan dengan aman.
- int random = rand() % (SISI_PAPAN * SISI_PAPAN);: Ini adalah langkah pertama untuk memilih posisi acak untuk menempatkan bom. Fungsi rand() digunakan untuk menghasilkan angka acak, dan operasi modulo (%) digunakan untuk memastikan angka tersebut berada dalam rentang yang sesuai dengan ukuran papan (SISI_PAPAN * SISI_PAPAN).
- int x = random / SISI_PAPAN; dan int y = random % SISI_PAPAN;: Langkah ini digunakan untuk mengonversi angka acak random ke koordinat baris x dan kolom y pada papan.
- if (adaBom[random] == false) { ... }: Fungsi ini memeriksa apakah sel yang dipilih secara acak (random) belum memiliki bom. Jika sel belum memiliki bom, maka bom ditempatkan pada sel tersebut.
- bom[i][0] = x; dan bom[i][1] = y;: Koordinat bom (baris dan kolom) disimpan dalam array bom. Ini digunakan untuk melacak lokasi bom-bom di papan.
- realBoard[bom[i][0]][bom[i][1]] = '*';: Papan realBoard diperbarui untuk menandai bahwa sel tersebut berisi bom.
- adaBom[random] = true;: Array adaBom diperbarui untuk menandai bahwa sel tersebut sekarang sudah berisi bom.

```c
// mencetak keadaan papan ke layar player
void cetakPapan (char playerBoard[MAXSISI][MAXSISI]){

    printf ("  ");

    for (int i = 0; i < SISI_PAPAN; i++)
        printf ("%d ", i);

    printf ("\n");

    for (int i = 0; i < SISI_PAPAN; i++){
        printf ("%d ", i);

        for (int j = 0; j < SISI_PAPAN; j++)
            printf ("%c ", playerBoard[i][j]);

        printf ("\n");
    }
    return;
}
```

- Loop pertama untuk mencetak header kolom papan. Loop ini digunakan untuk mencetak nomor kolom (indeks kolom) di atas papan permainan. Ini membantu pemain untuk mengidentifikasi kolom pada papan.
- printf ("\n");: Ini digunakan untuk mengakhiri baris header kolom dan pindah ke baris sel berikutnya.
- Loop pertama (for (int i = 0; i < SISI_PAPAN; i++)) digunakan untuk mengiterasi melalui baris papan.
- printf ("%d ", i); mencetak nomor baris (indeks baris) di awal setiap baris.
- Loop kedua (for (int j = 0; j < SISI_PAPAN; j++)) digunakan untuk mengiterasi melalui sel-sel pada baris saat ini.
- printf ("%c ", playerBoard[i][j]); mencetak karakter yang sesuai dengan isi sel pada papan playerBoard. Ini bisa berupa karakter yang menunjukkan bom, angka, atau tanda bahwa sel belum terungkap.

- printf ("\n"); digunakan untuk mengakhiri baris saat ini setelah seluruh sel dalam baris telah dicetak.

```
// input posisi oleh player
void pilihPosisi (int *x, int *y){
    printf("Pilih posisi dipisahkan spasi (baris kolom) -> ");
    scanf("%d %d", x, y);
    return;
}
```

- int *x, int *y: Fungsi ini menerima dua argumen dengan tipe data pointer ke integer. Pointer x dan y akan digunakan untuk mengembalikan koordinat yang dimasukkan oleh pemain ke dalam variabel yang ada di luar fungsi.
- printf("Pilih posisi dipisahkan spasi (baris kolom) -> ");: Ini adalah perintah cetak yang digunakan untuk memberikan petunjuk kepada pemain tentang apa yang harus mereka masukkan. Pesan ini meminta pemain untuk memasukkan koordinat baris dan kolom yang dipisahkan oleh spasi.
- scanf("%d %d", x, y);: Ini adalah perintah untuk mengambil input dari pemain. Dengan menggunakan format %d, fungsi scanf akan mengambil dua bilangan bulat yang dipisahkan oleh spasi dari pemain dan menyimpannya ke dalam variabel yang ditunjuk oleh pointer x dan y.
- return;: Ini adalah pernyataan yang mengakhiri eksekusi fungsi. Setelah pemain memasukkan koordinat, fungsi ini selesai dan program kembali ke tempat di mana fungsi ini dipanggil.

```
// fungsi untuk menentukan apakah posisi yang
// dipilih adalah bom
bool isBom (int baris, int kolom, char
realBoard[MAXSISI][MAXSISI]){
    if (realBoard[baris][kolom] == '*')
        return true;
    else
        return false;
}
```

Fungsi isBom digunakan untuk menentukan apakah suatu posisi pada papan adalah bom atau tidak. Fungsi ini menerima tiga argumen:

- int baris: Variabel yang menyimpan indeks baris posisi yang akan diperiksa.
- int kolom: Variabel yang menyimpan indeks kolom posisi yang akan diperiksa.
- char realBoard[MAXSISI][MAXSISI]: Array 2D yang merepresentasikan papan permainan dan berisi informasi tentang lokasi bom.

Fungsi ini bekerja dengan cara memeriksa isi sel pada papan realBoard pada koordinat (baris, kolom). Jika sel tersebut berisi karakter bom ('*'), maka fungsi akan mengembalikan true, yang menunjukkan bahwa posisi tersebut adalah bom. Jika sel tersebut tidak berisi bom, maka fungsi akan mengembalikan false.

```
void replaceBom (int baris, int kolom, char
realBoard[MAXSISI][MAXSISI]){
    for (int i = 0; i < SISI_PAPAN; i++){

        for (int j = 0; j < SISI_PAPAN; j++){

                if (realBoard[i][j] != '*'){
                    realBoard[i][j] = '*';
                    realBoard[baris][kolom] = '-';
                    return;
                }
            }
    }
    return;
}
```

Fungsi replaceBom digunakan untuk mengganti posisi bom yang pertama kali dipilih oleh pemain. Fungsi ini menerima tiga argumen:

- int baris: Variabel yang menyimpan indeks baris dari posisi bom yang akan diganti.
- int kolom: Variabel yang menyimpan indeks kolom dari posisi bom yang akan diganti.
- char realBoard[MAXSISI][MAXSISI]: Array 2D yang merepresentasikan papan permainan dan berisi informasi tentang lokasi bom.

Fungsi ini bekerja dengan cara:

- Menggunakan dua loop bersarang untuk mengiterasi melalui seluruh papan realBoard (seluruh baris dan kolom).
- Memeriksa setiap sel pada papan (if (realBoard[i][j] != '*')), jika sel tersebut bukan bom.
- Jika ditemukan sel yang bukan bom, maka sel tersebut diganti dengan karakter bom ('*'), dan sel yang pertama kali dipilih oleh pemain (realBoard[baris][kolom]) diganti menjadi karakter '-'.
- Setelah penggantian dilakukan, fungsi langsung keluar dengan pernyataan return.

```
bool Minesweeper (char playerBoard[MAXSISI][MAXSISI], char
realBoard[MAXSISI][MAXSISI], int bom[][2], int baris, int
kolom, int *movesLeft){

    // base case ketika posisi sudah terisi bukan bom
    // kembalikan gameOver bernilai false
    if (playerBoard[baris][kolom] != '-')
        return false;

    // jika posisi yang dipilih adalah bom
    if (realBoard[baris][kolom] == '*'){
        playerBoard[baris][kolom]='*';

        for (int i = 0; i < BOM; i++)
            playerBoard[bom[i][0]][bom[i][1]]='*';

        cetakPapan (playerBoard);
        printf ("\nKAMU KALAH!\n\n");
        return true ;
    }

    else
    {
        // hitung banyak bom di sekitar posisi yang dipilih
        int count = hitungBom(baris, kolom, bom, realBoard);
        (*movesLeft)--;
```

```
        playerBoard[baris][kolom] = count + '0';

        // jika di sekitar posisi yang dipilih tidak ada bom
        // maka hitung bom di semua posisi sekitarnya
        if (!count){

            //----------- 1. North ------------

            if (isValid (baris-1, kolom) == true){
                if (isBom (baris-1, kolom, realBoard) == false)
                    Minesweeper(playerBoard, realBoard, bom,
baris-1, kolom, movesLeft);
            }

            //----------- 2. South ------------

            if (isValid (baris+1, kolom) == true){
                if (isBom (baris+1, kolom, realBoard) == false)
                    Minesweeper(playerBoard, realBoard, bom,
baris+1, kolom, movesLeft);
            }

            //----------- 3. East ------------

            if (isValid (baris, kolom+1) == true){
                if (isBom (baris, kolom+1, realBoard) == false)
                    Minesweeper(playerBoard, realBoard, bom,
baris, kolom+1, movesLeft);
            }

            //----------- 4. West ------------

            if (isValid (baris, kolom-1) == true){
                if (isBom (baris, kolom-1, realBoard) == false)
                    Minesweeper(playerBoard, realBoard, bom,
baris, kolom-1, movesLeft);
```

```
            }

            //----------- 5. North-East ------------

            if (isValid (baris-1, kolom+1) == true){
                if (isBom (baris-1, kolom+1, realBoard) ==
false)
                    Minesweeper(playerBoard, realBoard, bom,
baris-1, kolom+1, movesLeft);
            }

            //----------- 6. North-West ------------

            if (isValid (baris-1, kolom-1) == true){
                if (isBom (baris-1, kolom-1, realBoard) ==
false)
                    Minesweeper(playerBoard, realBoard, bom,
baris-1, kolom-1, movesLeft);
            }

            //----------- 7. South-East ------------

            if (isValid (baris+1, kolom+1) == true){
                if (isBom (baris+1, kolom+1, realBoard) ==
false)
                    Minesweeper(playerBoard, realBoard, bom,
baris+1, kolom+1, movesLeft);
            }

            //----------- 8. South-West ------------

            if (isValid (baris+1, kolom-1) == true){
                if (isBom (baris+1, kolom-1, realBoard) ==
false)
                    Minesweeper(playerBoard, realBoard, bom,
baris+1, kolom-1, movesLeft);
            }
```

```
        }

        return false;
    }
}
```

Fungsi Minesweeper adalah inti dari permainan Minesweeper. Fungsi ini mengimplementasikan logika permainan dan bertanggung jawab untuk mengungkap sel pada papan permainan dan melakukan pemeriksaan yang diperlukan untuk menentukan apakah pemain menang, kalah, atau permainan masih berlanjut. Berikut adalah penjelasan bagian-bagian dari fungsi Minesweeper:

- bool Minesweeper(char playerBoard[MAXSISI][MAXSISI], char realBoard[MAXSISI][MAXSISI], int bom[][2], int baris, int kolom, int *movesLeft): Ini adalah deklarasi fungsi dengan tipe kembalian bool yang menerima beberapa argumen sebagai berikut:
- playerBoard[MAXSISI][MAXSISI]: Array 2D yang merepresentasikan papan permainan yang dilihat oleh pemain dan akan diubah sesuai dengan langkah-langkah permainan.
- realBoard[MAXSISI][MAXSISI]: Array 2D yang merepresentasikan papan permainan yang berisi informasi tentang lokasi bom.
- bom[][2]: Array 2D yang berisi koordinat bom pada papan.
- baris dan kolom: Koordinat yang menunjukkan posisi yang dipilih oleh pemain untuk diungkap.
- movesLeft: Pointer ke variabel yang menyimpan jumlah langkah yang tersisa dalam permainan.
- if (playerBoard[baris][kolom] != '-'): Ini adalah blok kondisional pertama yang memeriksa apakah sel pada playerBoard yang dipilih oleh pemain (baris dan kolom) sudah diungkap atau belum. Jika sel tersebut bukan karakter -, maka sel tersebut sudah diungkap dan fungsi mengembalikan false.
- if (realBoard[baris][kolom] == '*'): Blok kondisional kedua yang memeriksa apakah sel pada realBoard yang dipilih oleh pemain (baris dan kolom) adalah bom. Jika iya,

maka sel pada playerBoard pada posisi tersebut ditandai sebagai bom ('*') dan seluruh bom di playerBoard juga ditampilkan. Pesan "KAMU KALAH!" dicetak, dan fungsi mengembalikan true, yang menunjukkan bahwa pemain kalah.

- Jika sel yang dipilih oleh pemain bukan bom dan belum diungkap, maka sel tersebut akan diungkap. Jumlah bom di sekitar sel tersebut dihitung menggunakan fungsi hitungBom dan jumlahnya disimpan dalam variabel count. Sel pada playerBoard diubah menjadi angka yang menunjukkan jumlah bom di sekitarnya (count + '0').
- Kemudian, pointer movesLeft dikurangi satu karena satu langkah telah digunakan.
- Selanjutnya, fungsi memeriksa apakah sel yang dipilih tidak memiliki bom di sekitarnya (jika count adalah 0). Jika ya, maka fungsi akan memeriksa dan mengungkap sel-sel di sekitarnya dengan melakukan rekursi pada semua delapan arah yang mungkin (Utara, Selatan, Timur, Barat, dan arah diagonal). Ini dilakukan dengan memanggil fungsi Minesweeper untuk setiap arah yang memenuhi kriteria tertentu (valid dan tidak ada bom).
- Fungsi mengembalikan false jika pemain belum kalah dan permainan masih berlanjut.

```
// hitung bom di 8 posisi sekeliling posisi yang dipilih
int hitungBom(int baris, int kolom, int bom[MAXBOM][2], char realBoard[MAXSISI][MAXSISI]){

    int jmlBom = 0;

    /*

              N.W N N.E
                \ | /
           W----POS----E
                / | \
              S.W S S.E

        Pos --> Posisi sekarang (baris, kolom)
        N --> North  (baris-1, kolom)
        S --> South  (baris+1, kolom)
        E --> East       (baris, kolom+1)
        W --> West       (baris, kolom-1)
```

```
        N.E--> North-East (baris-1, kolom+1)
        N.W--> North-West (baris-1, kolom-1)
        S.E--> South-East (baris+1, kolom+1)
        S.W--> South-West (baris+1, kolom-1)
    */

    //----------- 1. North ------------

        if (isValid (baris-1, kolom) == true){
            if (isBom (baris-1, kolom, realBoard) == true)
                jmlBom++;
        }

    //----------- 2. South ------------

        if (isValid (baris+1, kolom) == true){
            if (isBom (baris+1, kolom, realBoard) == true)
                jmlBom++;
        }

    //----------- 3. East ------------

        if (isValid (baris, kolom+1) == true){
            if (isBom (baris, kolom+1, realBoard) == true)
                jmlBom++;
        }

    //----------- 4. West ------------

        if (isValid (baris, kolom-1) == true){
            if (isBom (baris, kolom-1, realBoard) == true)
                jmlBom++;
        }

    //----------- 5. North-East ------------

        if (isValid (baris-1, kolom+1) == true){
```

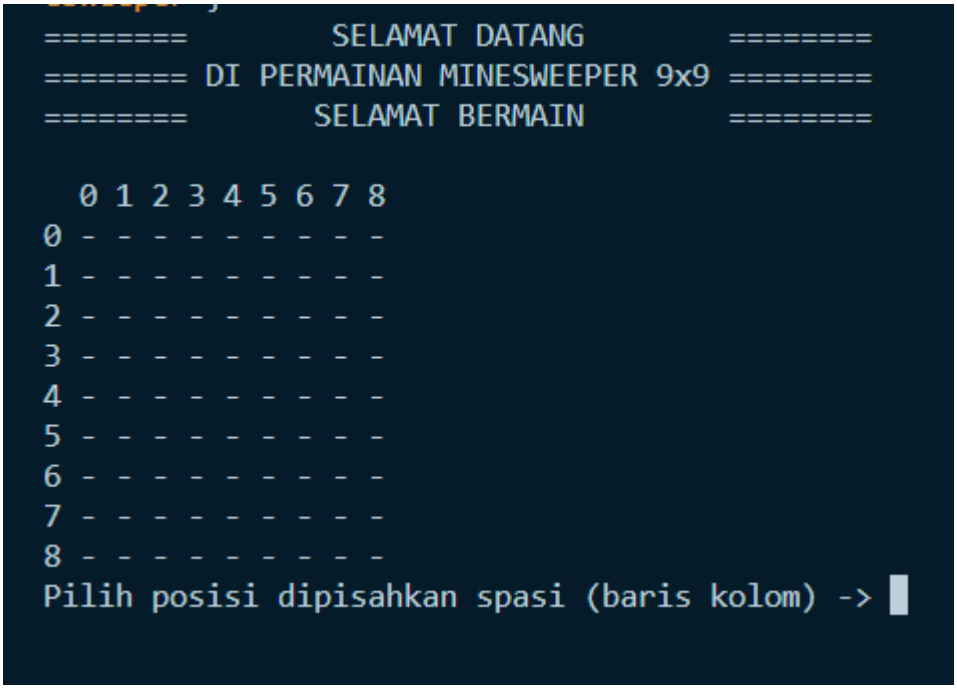
```
            if (isBom (baris-1, kolom+1, realBoard) == true)
                jmlBom++;
        }

    //----------- 6. North-West ------------

        if (isValid (baris-1, kolom-1) == true){
            if (isBom (baris-1, kolom-1, realBoard) == true)
                jmlBom++;
        }

    //----------- 7. South-East ------------

        if (isValid (baris+1, kolom+1) == true){
            if (isBom (baris+1, kolom+1, realBoard) == true)
                jmlBom++;
        }

    //----------- 8. South-West ------------

        if (isValid (baris+1, kolom-1) == true){
            if (isBom (baris+1, kolom-1, realBoard) == true)
                jmlBom++;
        }

    return jmlBom;
}
```

Fungsi hitungBom digunakan untuk menghitung jumlah bom yang terletak di sekitar posisi yang dipilih pada papan Minesweeper. Fungsi ini menerima beberapa argumen dan mengembalikan jumlah bom yang ditemukan di sekitar posisi tersebut. Berikut adalah penjelasan bagian-bagian dari fungsi hitungBom:

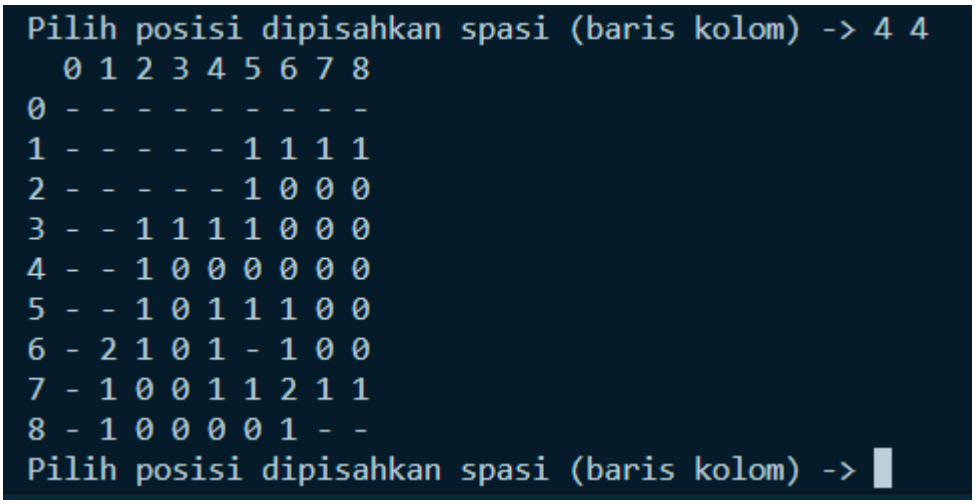
- int hitungBom(int baris, int kolom, int bom[MAXBOM][2], char realBoard[MAXSISI][MAXSISI]): Ini adalah deklarasi fungsi dengan tipe kembalian int yang menerima beberapa argumen sebagai berikut:

- baris dan kolom: Koordinat yang menunjukkan posisi yang akan diperiksa untuk menghitung bom di sekitarnya.
- bom[MAXBOM][2]: Array 2D yang berisi koordinat semua bom pada papan.
- realBoard[MAXSISI][MAXSISI]: Array 2D yang merepresentasikan papan permainan yang berisi informasi tentang lokasi bom.
- Fungsi ini menginisialisasi variabel jmlBom dengan nilai 0. Ini adalah variabel yang akan digunakan untuk menghitung jumlah bom di sekitar posisi yang dipilih.
- Fungsi menyediakan komentar yang menjelaskan posisi-posisi yang akan diperiksa di sekitar posisi yang dipilih. Ini membantu untuk memahami arah dan lokasi yang akan diperiksa.
- Untuk setiap dari 8 arah yang mungkin (Utara, Selatan, Timur, Barat, dan arah diagonal), fungsi melakukan pemeriksaan sebagai berikut:
- Pertama, fungsi memeriksa apakah posisi yang akan diperiksa adalah posisi yang valid pada papan dengan memanggil fungsi isValid. Jika posisi tersebut valid, maka pemeriksaan dilanjutkan.
- Selanjutnya, fungsi memeriksa apakah posisi tersebut mengandung bom atau tidak dengan memanggil fungsi isBom. Jika posisi tersebut mengandung bom, variabel jmlBom ditingkatkan sebesar 1.
- Fungsi mengembalikan nilai jmlBom yang merupakan jumlah bom yang ditemukan di sekitar posisi yang dipilih.

Analisis Program (saat dijalankan)



- Saat kode pertama dijalankan, kode tersebut menampilkan pesan selamat datang di permainan Minesweeper dengan ukuran papan 9x9. Pemain diperlihatkan papan 9x9 yang masing kosong.
- Setelah mencetak papan yang masih kosong, pemain dapat memilih posisi pada papan yang ingin di-*reveal* dengan menginputkan baris (spasi) kolom. Misalnya kita akan menginputkan baris 4 dan kolom 4



- Setelah kita menginputkan baris 4 dan kolom 4, secara kebetulan baris dan kolom tersebut tidak memiliki bom di sekitarnya. Oleh karena itu, count pada kode bernilai 0

dan kode memulai untuk menjalankan fungsi rekursifnya untuk membuka semua posisi baris dan kolom yang juga tidak memiliki bom di sekitarnya.

- Pemain dapat menginputkan baris dan kolom sampai permainan berakhir kalah atau berakhir menang.

```
  0 1 2 3 4 5 6 7 8
0 1 - - 1 0 0 1 - 1
1 2 3 3 2 1 1 1 1 1
2 1 - 1 1 - 1 0 0 0
3 2 2 1 1 1 1 0 0 0
4 - 2 1 0 0 0 0 0 0
5 2 - 1 0 1 1 1 0 0
6 2 2 1 0 1 - 1 0 0
7 - 1 0 0 1 1 2 1 1
8 - 1 0 0 0 0 1 - 1
Pilih posisi dipisahkan spasi (baris kolom) -> 8 0

KAMU MENANG!
```

- Gambar di atas adalah contoh kode saat pemain berhasil memenangkan permainan dengan mengungkap seluruh posisi yang tidak berisi bom. Setelah pemain menginputkan baris dan kolom 8 0, pemain memenangkan permainan dan kode pun berakhir.

```
Pilih posisi dipisahkan spasi (baris kolom) -> 5 5
  0 1 2 3 4 5 6 7 8
0 * - - - - - - - -
1 - - - - - - - - -
2 - - - - - * - - -
3 - - - 1 - - - - -
4 * - * - * * - - -
5 - - - - - * - - -
6 - - - * - - - - -
7 - - - - - - - - -
8 - - - - - - - * *

KAMU KALAH!
```

- Gambar di atas adalah contoh kode saat pemain tidak berhasil memenangkan permainan dengan memilih posisi yang berisi bom yaitu 5 5. Setelah pemain

menginputkan baris dan kolom 5 5, pemain kalah dalam permainan dan kode pun berakhir.

Tabel pengimplementasian materi :

No.	MATERI	POIN	PERAN DALAM PROGRAM

1	Tipe Data	5	`// INT`
			`Untuk mendeklarasikan variabel x, y, bom[MAXBOM][2], movesLeft, movePertama, random, count, main();`
			
			`// BOOL`
			`Untuk mendeklarasikan variabel gameOver, adaBom[][], isValid(int, int)`
			
			`// CHAR`
			`Untuk mendeklarasikan variabel array realBoard[][] dan playerBoard[][]`
			
			`// VOID`
			`Untuk mendeklarasikan fungsi pilihPosisi (int *x, int *y);`

2	I/O	5	`Untuk mencetak papan permainan di fungsi cetakPapan (char[])`
			
			`Untuk menginput posisi yang ingin dipilih di permainan pada fungsi pilihPosisi (int, int)`

3	Looping	5	`Untuk looping agar permainan terus berjalan selama belum game over` `while (gameOver == false)`

			Untuk iterasi menginput inisialisasi awal pada array realBoard[][] dan playerBoard[][] serta untuk mencetak array 2D tersebut
			`for (int i = 0; i < SISI_PAPAN; i++){`
			`for (int j = 0; j < SISI_PAPAN; j++)`

4	Conditional	5	Untuk mengecek apakah suatu posisi dalam array realBoard[][] berisi bom atau angka
			`if (realBoard[baris][kolom] == '*')`
			`return true;`
			`else return false;`
			
			Untuk digunakan pada fungsi hitungBom()
			`if (isValid == true){`
			`if (isBom == true)`
			`jmlBom++;}`
			
			Untuk digunakan pada fungsi rekursi Minesweeper()
			`if (isValid == true){`
			`if (isBom == false)`
			`Minesweeper();}`

5	Rekursi	20	Untuk memanggil kembali fungsi Minesweeper() jika di sekitar posisi yang dipilih tidak terdapat bom sama sekali
			
			`if (isValid == true){`
			`if (isBom == false)`
			`Minesweeper(playerBoard, realBoard, bom, baris-1, kolom, movesLeft);`

			}
			
			// Rekursi pada fungsi Minesweeper(), memanggil fungsi Minesweeper() di dalam fungsi Minesweeper()

6	Array	10	Untuk mendeklarasikan array of char realBoard[][] dan playerBoard[][]; Untuk mendeklarasikan array integer bom[MAXBOM][2];

7	Pointer	20	Untuk menerima alamat var x dan y di fungsi pilihPosisi()
			void pilihPosisi (int *x, int *y){
			...
			}
			pilihPosisi (&x, &y);
			
			Untuk menerima alamat var movesLeft di fungsi Minesweeper()
			bool Minesweeper (char [][], char [][], int [][], int, int, int *movesLeft)
			
			gameOver = Minesweeper (playerBoard, realBoard, bom, x, y, &movesLeft);

8	Formatted I/O	10	Untuk menerima input x dan y yang bertipe integer scanf("%d %d", x, y); Untuk mencetak variabel i yang bertipe integer printf ("%d ", i); Untuk mencetak array of char playerBoard printf ("%c ", playerBoard[i][j]);

Total : 80			

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