RESPONSI

PRAKTIKUM KONSEP PEMROGRAMAN

2023

IDENTITAS

Nama : Michael Johanes Johansyah

NIM : L0123081

Kelas : C

Judul Program : C_Responsi1KP_L0123081_Michael_01.c

Deskripsi Program : Game Simulasi Damage Senjata / Weapon Damage Simulator Game

Dokumentasi Program

Analisis Kode

Pada bagian awal kode, saya memasukkan kedua baris include ini untuk menyiapkan fungsi-fungsi yang akan digunakan kedepannya

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

Kemudian saya memasukkan variable yang akan digunakan kedepannya, lalu menggabungkannya kedalam struct agar lebih terstruktur dan mudah dimengerti.

```
struct Weapon {
    char name[50];
    int damage;
};

struct Enemy {
    int defense;
    int health;
};
```

Pada bagian ini, saya menyiapkan berbagai jenis pointer untuk digunakan ketika memasuki inti kode atau int main. Untuk penjelasan singkat dari masing-masing kode, sudah dipaparkan dalam bentuk comment didalam kode.

```
void clearScreen() {
```

```
    // Fungsi untuk membersihkan layar konsol, bekerja di berbagai sistem operasi
    system("clear || cls");
}

void initializeEnemy(struct Enemy *enemy) {
    clearScreen();
    printf("Weapon Damage Simulator Game \n");
    printf("-----------------------------------\n\n");
    printf("Choose an option to initialize enemy status:\n");
    printf("1. Enter custom values\n");
    printf("2. Use default values (100 health, 50 defense): \n");

    int option;
    scanf("%d", &option);

    if (option == 1) {
        // Meminta pengguna memasukkan nilai defense dan health untuk enemy
        printf("Enter initial enemy defense: ");
        scanf("%d", &enemy->defense);

        printf("Enter initial enemy health: ");
        scanf("%d", &enemy->health);
    } else if (option == 2) {
        // Menggunakan nilai default jika pengguna memilih opsi default
        enemy->defense = 50;
        enemy->health = 100;
    } else {
        // Menggunakan nilai default jika pengguna memasukkan opsi tidak valid
        printf("Invalid option. Using default values.\n");
        enemy->defense = 50;
        enemy->health = 100;
    }
}

void newEnemy(struct Enemy *enemy) {
    // Menginisialisasi enemy baru dengan opsi kustom atau nilai default
    printf("\nChoose an option for new enemy status:\n");
    printf("1. Enter custom values\n");
    printf("2. Use default values (100 health, 50 defense): \n");

    int option;
    scanf("%d", &option);

    if (option == 1) {
        // Meminta pengguna memasukkan nilai defense dan health untuk enemy baru
        printf("Enter new enemy defense: ");
        scanf("%d", &enemy->defense);
```

```
        printf("Enter new enemy health: ");
        scanf("%d", &enemy->health);
    } else if (option == 2) {
        // Menggunakan nilai default jika pengguna memilih opsi default
        enemy->defense = 50;
        enemy->health = 100;
    } else {
        // Menggunakan nilai default jika pengguna memasukkan opsi tidak valid
        printf("Invalid option. Using default values.\n");
        enemy->defense = 50;
        enemy->health = 100;
    }
}

void displayEnemyStatus(struct Enemy enemy) {
    // Menampilkan status defense dan health dari enemy
    printf("Enemy Status:\n");
    printf("Defense: %d\n", enemy.defense);
    printf("Health: %d\n", enemy.health);
    printf("----------------------------\n");
}

void displayWeapons(struct Weapon weapons[], int numWeapons) {
    // Menampilkan daftar senjata yang tersedia
    printf("Weapons List:\n");
    for (int i = 0; i < numWeapons; i++) {
        printf("%d. %s\n", i + 1, weapons[i].name);
    }
}

void displayWeaponDetails(struct Weapon weapon, struct Enemy enemy) {
    clearScreen();
    displayEnemyStatus(enemy);
    // Menampilkan detail senjata, termasuk nama dan damage
    printf("Weapon Details:\n");
    printf("Name: %s\n", weapon.name);
    printf("Damage: %d\n", weapon.damage);
}

// Fungsi rekursif untuk simulasi Headshot
int simulateHeadshotDamage(int damage) {
    // Peluang Headshot 50%, menggandakan damage dalam kasus Headshot
    if (rand() % 2 == 0) {
        printf("Headshot! Damage doubled! \n");
        return damage * 2;
    } else {
        return damage;
    }
```

```
}

void attackEnemy(struct Weapon weapon, struct Enemy *enemy) {
    // Mendapatkan damage yang mungkin termasuk Headshot
    int inflictedDamage = simulateHeadshotDamage(weapon.damage);

    // Mengurangi defense terlebih dahulu, kemudian health jika defense 0 atau negatif
    enemy->defense -= inflictedDamage;

    if (enemy->defense < 0) {
        // Mengurangkan defense negatif dari health jika defense kurang dari 0
        enemy->health += enemy->defense;
        enemy->defense = 0;

        // Jika baik defense dan health 0, reset ke nilai awal
        if (enemy->health <= 0) {
            printf("Enemy defeated!\n");
            newEnemy(enemy);
        }
    }

    // Memastikan health tidak kurang dari 0
    if (enemy->health < 0) {
        enemy->health = 0;
    }

    // Menampilkan status terkini dari enemy
    printf("Enemy defense: %d, health: %d\n", enemy->defense, enemy->health);
}

// Pilihan untuk mengupgrade senjata
void upgradeWeapon(struct Weapon *weapon) {
    // Meningkatkan damage senjata sebesar 10
    weapon->damage += 10;
    printf("Weapon upgraded! New damage: %d\n", weapon->damage);
}

// Pilihan untuk mereset damage senjata
void resetWeapon(struct Weapon *weapon, int originalDamages) {
    // Mereset damage senjata ke nilai asli
    weapon->damage = originalDamages;
    printf("Weapon reset! Damage restored to: %d\n", weapon->damage);
}
```

Setelah berbagai pointer sudah disiapkan, saatnya untuk memulai inti dari kode, dimulai dengan int main, saya menginisialisasi berbagai jenis senjata yang dapat digunakan, dengan masing-masing damage yang dimiliki.

```
int main() {
    const int numWeapons = 5;
    struct Weapon weapons[numWeapons];

    // Inisialisasi senjata-senjata yang tersedia
    weapons[0] = (struct Weapon){"Pistol", 20};
    weapons[1] = (struct Weapon){"Rifle", 30};
    weapons[2] = (struct Weapon){"Shotgun", 50};
    weapons[3] = (struct Weapon){"Sniper", 100};
    weapons[4] = (struct Weapon){"Machine Gun", 40};
```

Dibagian ini, saya mulai memasukkan variable enemy agar dapat menjalankan fungsi-fungsi dari pointer sebelumnya, seperti inisialisasi status enemy pada baris selanjutnya tersebut.

Kemudian saya menyiapkan variable choice sebagai variable untuk membaca pilihan dari pengguna nanti.

Selain itu saya juga langsung menyiapkan kode untuk menyimpan nilai asli dari damage masing-masing senjata agar ketika mengalami upgrade dapat kembali ke nilai damage semula ketika di reset.

```
    struct Enemy enemy;
    initializeEnemy(&enemy);

    int choice;

    // Simpan nilai asli dari damage senjata
    int originalDamages[numWeapons];
    for (int i = 0; i < numWeapons; ++i) {
        originalDamages[i] = weapons[i].damage;
    }
```

Pada bagian inilah saya mulai memasak, saya memulai dengan menampilkan status enemy dan juga berbagai jenis senjata yang dapat digunakan, kemudian saya meminta pengguna untuk memilih senjata mana yang akan dipakai

Nantinya setelah sudah memilih senjata yang mana yang akan dipakai, pengguna akan melihat tampilan detail dari senjata tersebut, dan juga 4 pilihan yang bisa dilakukan, yaitu menyerang enemy, mengupgrade senjata, mereset senjata dan kembali ke tampilan pilih senjata.

Untuk penjelasan singkat dari masing-masing kode sudah dipaparkan dalam bentuk comment didalam kode.

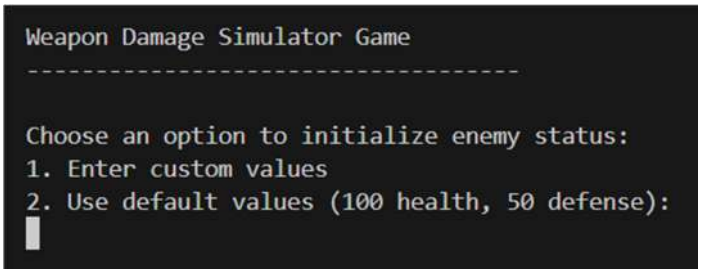
```
    do {
        clearScreen();
        displayEnemyStatus(enemy);
        displayWeapons(weapons, numWeapons);
```

```c
// Meminta pengguna memilih senjata untuk dilihat detailnya
printf("Enter the number of the weapon to view details (0 to exit): ");
scanf("%d", &choice);

if (choice > 0 && choice <= numWeapons) {
    int option;

    do {
        // Menampilkan detail senjata dan opsi-opsinya
        displayWeaponDetails(weapons[choice - 1], enemy);

        printf("\nOptions:\n");
        printf("1. Attack\n");
        printf("2. Upgrade Weapon\n");
        printf("3. Reset Weapon\n");
        printf("0. Back to Weapons List\n");

        // Meminta pengguna memilih opsi
        scanf("%d", &option);

        switch (option) {
            case 1:
                // Menyerang enemy dengan senjata yang dipilih
                attackEnemy(weapons[choice - 1], &enemy);
                getchar(); // Mengonsumsi karakter newline dari input sebelumnya
                printf("\nPress Enter to continue...");
                getchar(); // Menunggu penekanan tombol Enter
                break;
            case 2:
                // Meningkatkan damage senjata yang dipilih
                upgradeWeapon(&weapons[choice - 1]);
                getchar(); // Mengonsumsi karakter newline dari input sebelumnya
                printf("\nPress Enter to continue...");
                getchar(); // Menunggu penekanan tombol Enter
                break;
            case 3:
                // Mereset damage senjata yang dipilih
                resetWeapon(&weapons[choice - 1], originalDamages[choice - 1]);
                getchar(); // Mengonsumsi karakter newline dari input sebelumnya
                printf("\nPress Enter to continue...");
                getchar(); // Menunggu penekanan tombol Enter
                break;
            case 0:
                break;
            default:
                printf("Invalid option\n");
                break;
```

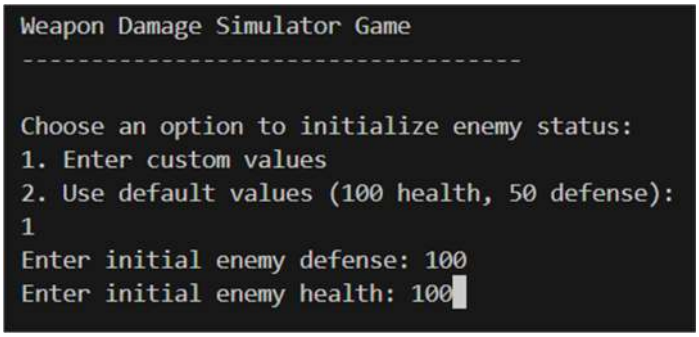
```
                }
            } while (option != 0);
        }
    } while (choice != 0);

    return 0;
}
```

Sekian hasil kode dari saya beserta penjelasannya, berikut ini analisis program dari hasil kode tersebut ketika dijalankan.

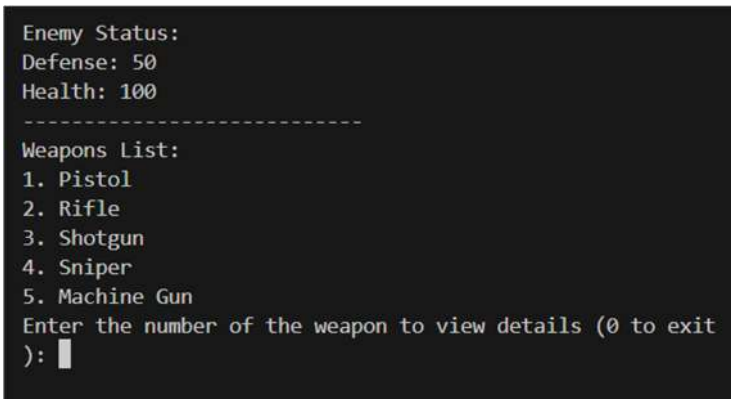
Analisis Program (saat dijalankan)



Berikut ini tampilan awal ketika kode mulai dijalankan, pengguna akan diminta untuk memilih cara untuk menampilkan status enemy, yaitu antara memasukkannya secara kustom maupun secara default, yang otomatis akan memasukkan nilai 100 health dan 50 defense ke enemy

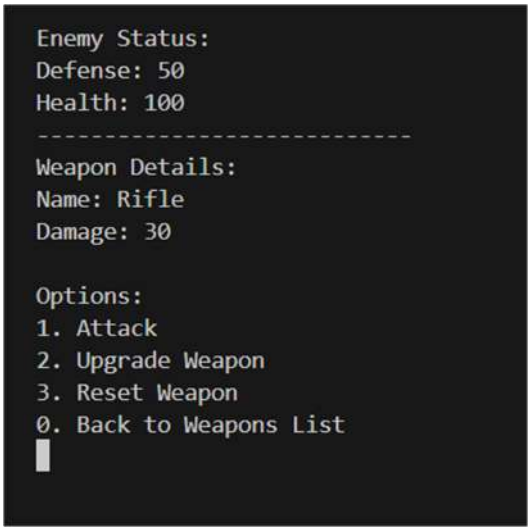


Begini tampilan jika kita memilih nomor 1, kita bisa memasukkan nilai health dan defense sesuai keinginan kita. Jika kita memilih nomor 2, tampilan akan berpindah menjadi seperti berikut :



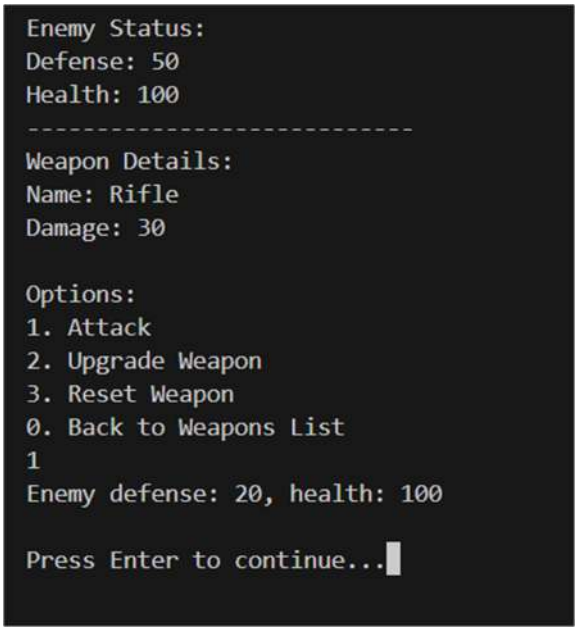
Pada gambar tampilan ini, diperlihatkan status enemy (jika tadi memilih pilihan 1, status enemy akan menyesuaikan dengan nilai yang kita inginkan, kali ini karena pilihan 2 default, nilai defensenya 50 dan health 100), kemudian diperlihatkan list senjata yang bisa dipilih mulai dari nomor 1-5, dan juga nomor 0 untuk keluar dan mengakhiri program.

Jika saya memilih salah satu senjata, pada skenario ini saya memilih nomor 2 yaitu “Rifle”, akan ditampilkan detail dari senjata Rifle tersebut beserta pilihan yang dapat kita lakukan sebagai berikut :



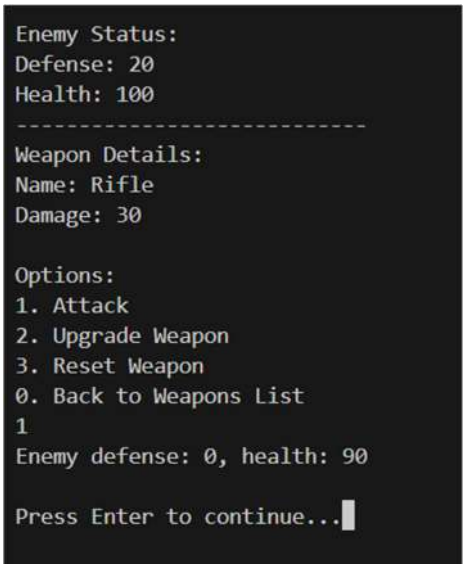
Pada gambar tersebut dapat dilihat bahwa kita masih bisa melihat status dari enemy, sehingga lebih mempermudah kita untuk memperhatikan damage yang disebabkan senjata kedepannya.
Disini terlihat ada 4 pilihan, yaitu menyerang, mengupgrade senjata dan mereset damage dari senjata yang sudah diupgrade, dan juga pilihan untuk kembali ke list senjata jika ingin mencoba senjata lain.

Jika kita memilih pilihan 1, yaitu menyerang enemy, berikut ini tampilan yang akan terjadi :



Jadi yang akan terjadi adalah diperlihatkan status enemy setelah diserang, yaitu pada gambar tersebut defense enemy berkurang 30 dan sekarang menjadi 20, serta health yang masih bernilai 100. Untuk melanjutkan aksi lainnya, saya perlu untuk menekan enter.

Berikut ini jika saya mencoba untuk menyerang enemy lagi :

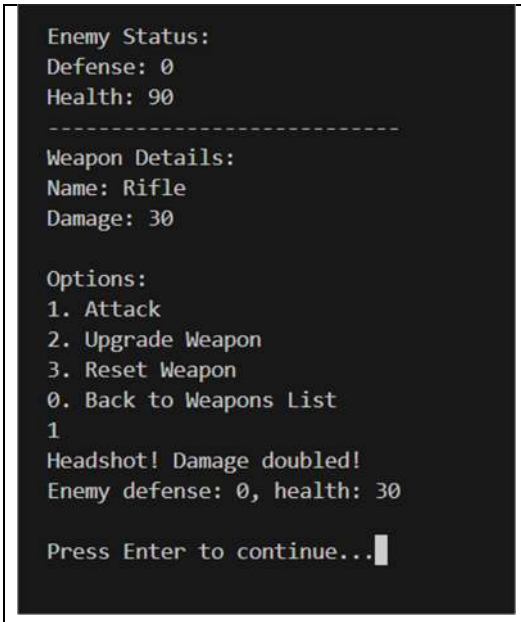


Dapat dilihat, ketika defense enemy mencapai nilai 0, sisa damage yang diterima akan mengurangi nilai health, hal tersebut sesuai dengan keinginan saya, karena ketika kita bermain game seperti contoh PUBG atau FORTNITE, mekanisme serupa akan terjadi.

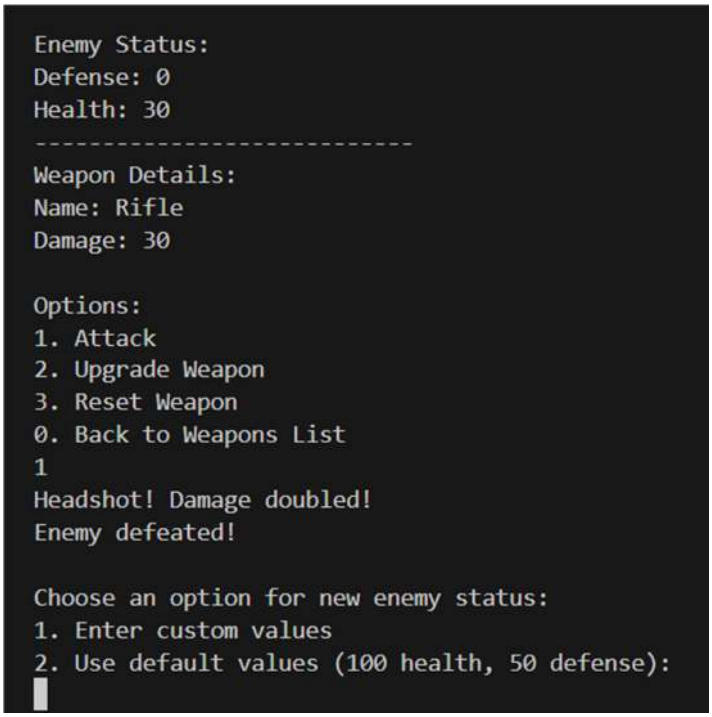
Selain itu, saya juga memanfaatkan fungsi rekursi untuk membuat fitur Headshot atau Critical Damage, dimana serangan memiliki kemungkinan untuk menggandakan damage yang diberikan. Dalam kode, yang terjadi adalah saya memanggil kembali variable damage agar dapat digandakan nilainya dan nilai tersebut yang akan diterima enemy. Seperti ini kodenya :

```
// Fungsi rekursif untuk simulasi Headshot
int simulateHeadshotDamage(int damage) {
    // Peluang Headshot 50%, menggandakan damage dalam kasus Headshot
    if (rand() % 2 == 0) {
        printf("Headshot! Damage doubled! \n");
        return damage * 2;
    } else {
        return damage;
    }
```

Dan seperti inilah contoh tampilan yang muncul jika enemy terkena Headshot :

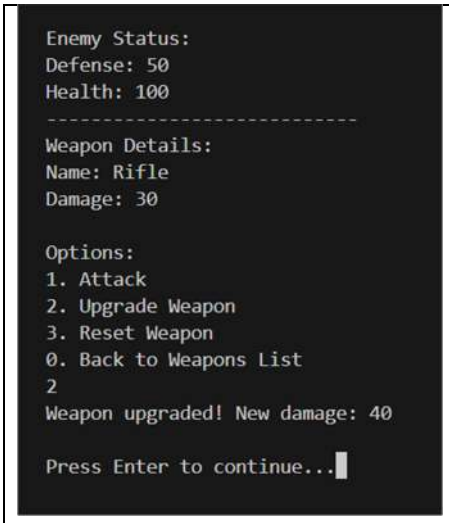


Dapat dilihat pada status awal enemy, nilai Healthnya adalah 90, setelah menerima Headshot, nilai Healthnya menjadi 30, hal ini dikarenakan damage Rifle yang awalnya 30, digandakan karena menjadi damage Headshot, yang nilai damagenya menjadi 60.



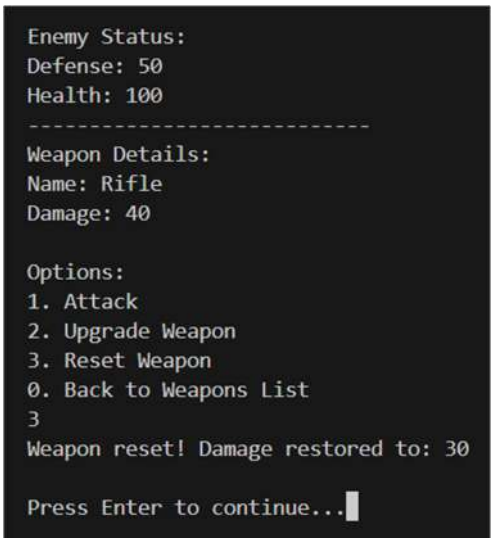
Jika kedua status nilai dari enemy yaitu Health dan Defense sudah menyentuh angka 0, maka pengguna akan diarahkan ke pilihan untuk memunculkan enemy yang baru, dengan nilai status yang sesuai keinginan atau nilai default, sama seperti pada awal kode ini dijalankan, dan begitu seterusnya.

Lanjut ke pilihan kedua, yaitu pilihan untuk mengupgrade weapon. Berikut ini tampilan yang akan muncul jika saya memilih pilihan upgrade.



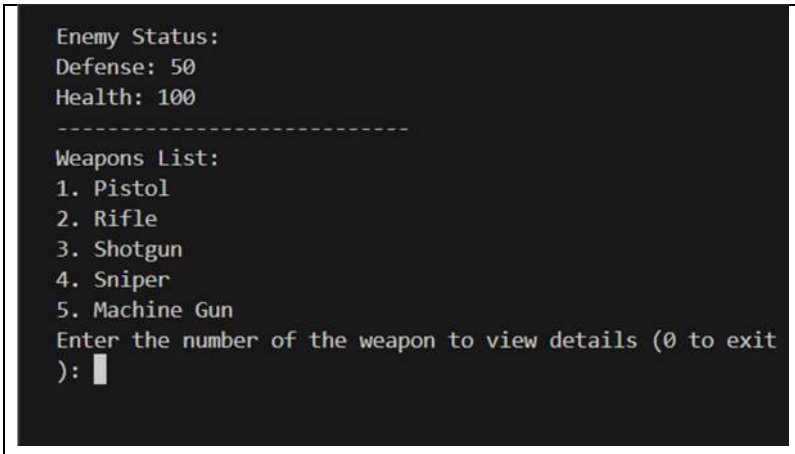
Disini damage weapon yang pada awalnya bernilai 30, dinaikkan menjadi 40. Pilihan ini dapat dipilih berkali-kali dan dapat ditumpuk atau di-stack (dari awalnya 30 bisa dinaikkan bahkan ke 300 damage).

Setelah itu, disinilah fungsi dari pilihan ketiga, yaitu untuk mereset damage senjata dan mengembalikan damagenya ke nilai semula. Berikut ini tampilan yang muncul ketika memilih pilihan ketiga :



Dapat dilihat damage dari Rifle yang tadi sudah diupgrade menjadi 40, setelah direset, damagenya kembali ke 30.

Terakhir, jika ingin mencoba senjata yang lain, kita bisa memilih pilihan ke 4 untuk kembali ke tampilan list senjata.



Dapat disadari, saya menggunakan fitur clearscreen, agar tampilan output tidak menumpuk dan lebih rapi.

Sekian hasil yang didapatkan ketika menjalankan kode saya, dengan membuat game simulasi ini, saya menjadi lebih memahami kegunaan masing-masing materi yang dipelajari sebelumnya, dan saya sekarang memiliki ide yang lebih luas mengenai game yang dapat dibuat dengan memanfaatkan materi-materi ini.

Tabel pengimplementasian materi :

No.	MATERI	POIN	PERAN DALAM PROGRAM
1	Tipe Data	5	Sebagai fondasi dalam kode ini, tipe data berperan dalam mendefinisikan variable yang saya perlukan didalam kode ini, seperti variable nama yang bertipe data char, variable damage yang bertipe data int, dan beberapa variable lainnya. Tanpa tipe data, kode ini tidak memiliki variable yang otomatis tidak memiliki nilai yang dapat dioperasikan.
2	I/O	5	I/O atau Input/Output dalam program saya contohnya adalah dibagian akhir, yaitu fungsi “getchar” yang memiliki peran yang simpel, yaitu mengonsumsi karakter newline dari input sebelumnya dan menunggu penekanan tombol enter untuk melanjutkan program.
3	Looping	5	Looping yang saya gunakan didalam program ini adalah Looping berjenis do-while yang berperan untuk terus menampilkan menu dan menerima input pengguna hingga pengguna memilih untuk keluar, yang pada program ini akan terjadi jika pengguna memilih angka 0.
4	Conditional	5	Conditional yang saya gunakan didalam program ini tentu saja if-else condition, dimana materi ini sangat berperan penting didalam program saya. Karena program saya banyak menyediakan pilihan, maka if-else condition ini selalu saya gunakan agar program dapat berjalan. Selain itu, untuk mempermudah dan lebih efisien dalam mengetik kode, saya juga menggunakan condition switch case.
5	Rekursi	20	Fungsi rekursi pada program ini berperan untuk memanggil kembali variable damage ketika serangan yang diberikan menjadi Headshot untuk digandakan. Pada program ini, fungsi rekursi yang digunakan sangatlah simpel, namun menghasilkan fitur yang unik dan memperluas pengetahuan serta ide saya untuk membuat game lain kedepannya.
6	Array	10	Pada program ini, Array juga memiliki peran yang penting, yaitu untuk menampilkan berbagai jenis senjata serta

			damage yang dimiliki masing-masing senjata tersebut. Array sangat memudahkan saya untuk memperlihatkan detail senjata, karena jika dimasukkan satu-persatu, akan sangat memakan waktu dan baris. Jadi, Array membuat program saya menjadi lebih efisien dan terlihat rapi.
7	Pointer	20	Saya menggunakan sangat banyak pointer pada program ini, seperti initializeEnemy, newEnemy, upgradeWeapon, resetWeapon, displayEnemyStatus, dan lain sebagainya. Alasannya agar program terlihat lebih rapi. Pointer juga berperan untuk menyimpan kode dari luar int main, yang ketika dibutuhkan, kita bisa langsung saja memanggilnya. Jika diandaikan seperti memasak nasi goreng, pointer adalah sosis yang sudah dipotong-potong, dan juga telur yang sudah dibumbui. Jadi saat memasak kita tidak perlu lagi memotong sosis ataupun membumbui telur, kita tinggal memasukkan sosis dan telur tersebut ke dalam wajan yang dapat kita anggap sebagai int main.
8	Formatted I/O	10	Formatted I/O merupakan versi I/O yang lebih leluasa dan dapat dikontrol dan diatur tata letaknya. Formatted I/O dalam program ini antara lain adalah printf dan scanf. Contoh Formatted I/O yaitu ketika pengguna diminta untuk memilih senjata apa yang mau digunakan, dan ketika dipilih, program akan membaca input dan memunculkan tampilan detail senjata yang diinginkan. Jadi I/O ini berperan untuk membuat program saya menjadi interaktif dan lebih fleksibel, karena dapat menerima input dari pengguna dan mencetak output yang diinginkan ke layar.
Total : 80			

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