RESPONSI 1

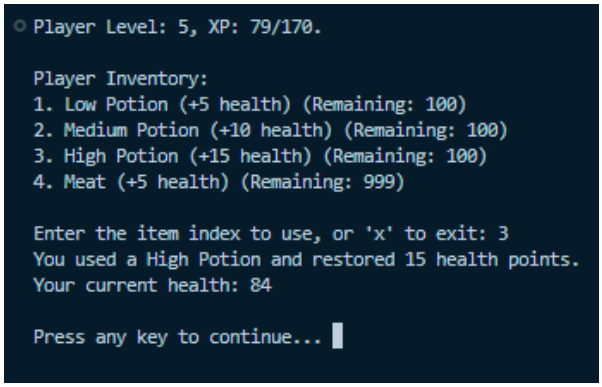
PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

2024

IDENTITAS		
Nama	:	**Mohammad Nazhiif Al-Ghoniy**
NIM	:	**L0123084**
Judul Program	:	**Mini Game RPG**
Deskripsi Program	:	**Program tentang game RPG tetapi versi simpelnya. Ada berburu monster untuk mendapatkan EXP, upgrade job class player**

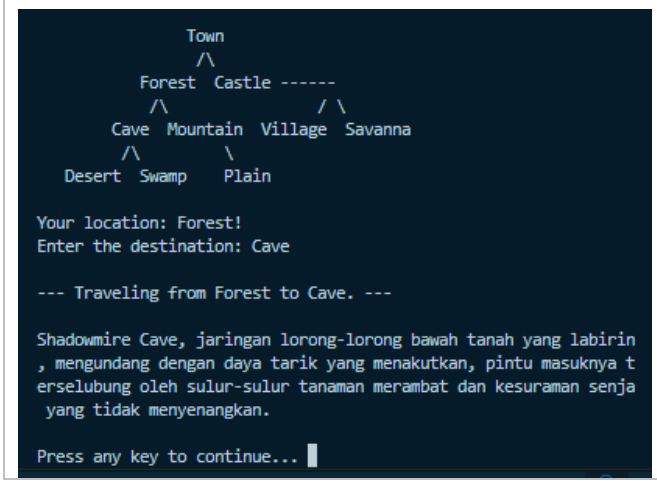
DOKUMENTASI PROGRAM

- **Fitur Inventory dan Level**



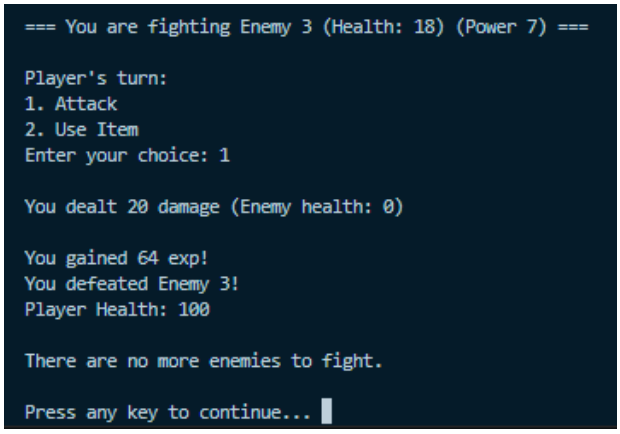
Pemain memiliki inventory yang berisi berbagai jenis item pemulihan health dengan nilai pemulihan yang berbeda. Pemain dapat menggunakan item dari inventaris untuk memulihkan health saat bertarung melawan musuh. Pemain juga dapat meningkatkan level dan mendapatkan EXP setelah bertarung melawan musuh.

- **Sistem Travel**



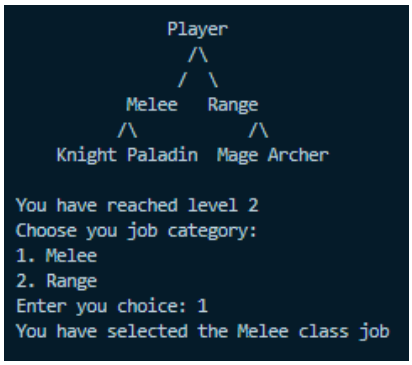
Terdapat peta dunia yang terdiri dari berbagai lokasi yang terhubung dalam struktur tree. Pemain dapat berpindah dari satu lokasi ke lokasi lain yang terhubung. Setiap kali pemain mengunjungi lokasi baru, lokasi tersebut akan dicatat dalam riwayat perjalanan. Program memberikan gambaran tentang kemajuan perjalanan pemain dengan menunjukkan lokasi yang telah dikunjungi dan sisa lokasi yang belum dikunjungi.

- **Sistem Pertarungan Musuh**



Setiap kali pemain mengunjungi lokasi baru, musuh acak akan dibangkitkan untuk dihadapi dalam pertarungan. Musuh memiliki tingkat kesulitan yang berbeda-beda tergantung pada lokasi tempat mereka dibangkitkan. Pemain dapat menyerang musuh atau menggunakan item untuk memulihkan kesehatan selama pertarungan. Pemain mendapatkan poin pengalaman setelah mengalahkan musuh, yang memungkinkan pemain untuk naik level.

- **Sistem Job Character**



Pemain dapat memilih kategori pekerjaan (Melee atau Range) pada level tertentu. Pada level yang lebih tinggi, pemain dapat memilih pekerjaan lanjutan berdasarkan kategori yang telah dipilih sebelumnya. Sistem ini menggunakan struktur binary tree untuk merepresentasikan hierarki pekerjaan. Pemain juga akan mendapatkan benefit tergantung dari job class yang dipilih.

- **Instruksi Pemakaian:**

1. Jalankan program.
2. Pilih opsi menu yang diinginkan dengan memasukkan angka yang sesuai.
3. Untuk berpindah lokasi, pilih opsi "Travel" dan masukkan nama lokasi tujuan.
4. Untuk melihat inventaris dan menggunakan item, pilih opsi "Show Level and Inventory".
5. Untuk bertarung melawan musuh, pilih opsi "Fight Enemy".
6. Untuk melihat riwayat perjalanan, pilih opsi "Travel History".
7. Untuk melihat kemajuan perjalanan, pilih opsi "Travel Progress".
8. Untuk memilih pekerjaan atau kelas karakter, pilih opsi "Choose Job Class".
9. Untuk keluar dari program, pilih opsi "Quit".

Tabel penggunaan struktur data

No.	STRUKTUR DATA	POIN	PERAN
1.	List	5	`std::list<Item> playerInventory;` • Digunakan untuk menyimpan item-item yang dimiliki oleh pemain dalam bentuk daftar. • Memungkinkan pemain untuk menambahkan, menghapus, atau mengakses item secara efisien.
2.	Queue	5	`std::queue<Enemy> enemyQueue;` • Digunakan untuk menyimpan riwayat perjalanan pemain dalam bentuk tumpukan (stack). • Setiap kali pemain berpindah lokasi, lokasi sebelumnya akan ditambahkan ke tumpukan. • Memungkinkan pemain untuk melihat riwayat perjalanan secara berurutan dari lokasi terakhir ke lokasi awal.
3.	Stack	5	`std::stack<std::string> travelHistory;` • Digunakan untuk menyimpan daftar musuh yang harus dihadapi pemain dalam bentuk antrian (queue). • Setiap kali pemain mengunjungi lokasi baru, musuh acak akan dibangkitkan dan ditambahkan ke antrian.

			• Pemain akan menghadapi musuh satu per satu sesuai urutan dalam antrian.
4.	Set	5	`std::set<std::string> visitedLoc;` • Digunakan untuk menyimpan kumpulan lokasi yang telah dikunjungi oleh pemain. • Kumpulan lokasi tersebut nantinya akan digunakan untuk menghitung progress perjalanan pemain mengelilingi dunia. • Memungkinkan program untuk melacak lokasi yang telah dikunjungi dan menghindari duplikasi dalam perhitungan kemajuan perjalanan.
5.	Map	5	`std::unordered_map<std::string, std::string> locDesc;` • Digunakan untuk menyimpan deskripsi setiap lokasi dalam bentuk pemetaan (map) antara nama lokasi dan deskripsinya. • Memungkinkan program untuk mengakses dan menampilkan deskripsi lokasi secara efisien.
6.	Tree	15	`struct Node { std::string location; Node *parent; // parent pointer std::vector<Node *> children; Node(const std::string &loc, Node *par = nullptr) : location(loc), parent(par) {} };` • Digunakan untuk merepresentasikan hierarki lokasi dalam peta dunia dan hierarki pekerjaan yang dapat dipilih pemain. • Struktur Node memiliki pointer parent yang menunjuk ke lokasi parent node, serta vektor children yang menyimpan lokasi child node. • Struktur JobNode mirip dengan Node, tetapi digunakan untuk menyimpan hierarki pekerjaan dalam bentuk binary tree.

			• Memungkinkan program untuk melacak hubungan antar lokasi dengan mudah.
7.	Binary Tree	10	`struct JobNode { std::string jobName; JobNode *left; JobNode *right; JobNode(const std::string &name) : jobName(name), left(nullptr), right(nullptr) {} };` • Digunakan untuk merepresentasikan hierarki pekerjaan yang dapat dipilih oleh pemain. • Setiap node dalam pohon biner mewakili sebuah job, dan memiliki pointer left dan right yang menunjuk ke job child node yang lebih spesifik. • Memungkinkan program untuk mengakses dan memilih pekerjaan secara hierarkis berdasarkan kategori dan tingkat pekerjaan.
TOTAL POIN : 50			

Analisis kompleksitas

No.	Nama part	Posisi part	Kompleksitas
1.	`void pressAny()`	main.cpp:15-25	O(1)
2.	`void job()`	main.cpp:79-89	O(1)
3.	`JobNode *createJobTree()`	main.cpp:93-115	O(1)
4.	`void selectJob`	main.cpp:119-149	O(1)
5.	`void chooseMeleeOrRangeJob()`	main.cpp:151-177	O(1)
6.	`void chooseAdvancedJob()`	main.cpp:179-219	O(1)
7.	`void chooseJob()`	main.cpp:221-274	O(1)
8.	`void history()`	main.cpp:295-316	O(N)
9.	`void createWorldMap()`	main.cpp:338-376	O(1)
10.	`void travelTo()`	main.cpp:384-438	O(1)
11.	`void useItem()`	main.cpp:440-471	O(N)
12.	`void showInventory()`	main.cpp:473-507	O(N)
13.	`void playerAttack()`	main.cpp:509-535	O(1)

14.	`void playerTurn()`	main.cpp:537-560	O(1)
15.	`void playerDefend()`	main.cpp:562-573	O(1)
16.	`void generateEnemies()`	main.cpp:575-603	O(N)
17.	`void fightEnemy()`	main.cpp:605-640	O(N)
18.	`int main()`	main.cpp:642-709	O(N)
19.	`void travelProgress()`	main.cpp:318-333	O(1)