ANALISIS PROGRAM RESPONSI 1
MATA KULIAH KONSEP PEMROGRAMAN
PROGRAM STUDI S-1 INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SEBELAS MARET
2020

Nama: Michael Raditya Krisnadhi
NIM: M0520047
Kelas: B

1. Awal Permainan

```
!! WELCOME TO ANIMAL KAISER !!
Choose your fighter:
 [1] Electric Ray (H: 75, S: 10)
 [2] Cheetah (H: 100, S: 7)
 [3] Hippo Barry (H: 150, S: 5)
Your choice?
```

Pada awal permainan, program masuk ke dalam fungsi main() di mana awalnya melakukan deklarasi untuk variabel-variabel yang akan digunakan, lalu melakukan seeding untuk random number generator menggunakan perintah srand(time(NULL)) (harus mengimpor header stdlib.h dan time.h agar bisa bekerja).

Kemudian program menampilkan pilihan spesies yang dapat digunakan oleh pemain untuk melawan musuh. Pada program tersebut terdapat tiga spesies yang mana terkandung dalam array yang tersimpan dalam variabel global ‘g_human_species’ di mana tipe data elemen dari array tersebut adalah sebuah structure untuk mendeskripsikan suatu spesies yang berisi informasi name, health, dan strength dari spesies yang bersangkutan. Setelah menampilkan pilihan spesies menggunakan for loop, maka program akan membaca input berupa angka dari pemain yang mana menunjukkan nomor spesies.

Setelah menerima input, program melakukan inisiasi pemain (init_human), musuh (init_computer), dan barang-barang yang terdapat pada toko (init_packages) menggunakan function call dan pass-by-reference. Program melakukan inisiasi pemain dengan input berupa pointer menuju variabel ‘human’ yang memiliki tipe data ‘human_t’ dan pointer menuju spesies yang dipilih di mana dapat diakses dalam ‘g_human_species[spec_id – 1]’ di mana spec_id adalah input yang diterima sebelumnya (operasi - 1 digunakan untuk menyesuaikan indeks pada array yang dimulai dari 0). Kemudian program melakukan inisiasi musuh dengan input berupa pointer menuju variabel ‘computer’ yang memiliki tipe data ‘computer_t’ dan pointer menuju spesies level pertama (atau ke-0) untuk komputer yang disimpan dalam array ‘g_computer_spesies’ pada global variable.

Berikut adalah potongan kode untuk fungsi ‘init_human’.

```
151 void init_human(human_t *human, const species_t *spec) {
152     human->spec = spec;
153     human->health = spec->health;
154     human->strength = spec->strength;
155     human->credits = 0;
156 }
```

Inisiasi tersebut menggunakan bantuan pointer untuk mengubah nilai yang terdapat pada variabel yang dikandung dalam pointer ‘human’ menggunakan operasi → untuk mengakses masing-masing member pada struct. Kemudian ‘spec’ digunakan sebagai masukan berupa informasi spesies yang

akan digunakan oleh pemain. Meskipun dijamin tidak ada operasi menulis (read-only), pemrogram tetap menggunakan pointer dan diberi ‘const’ karena dengan tujuan untuk menghemat memori sehingga tidak harus pass-by-reference sehingga menciptakan copy-an yang baru dari informasi spesies yang bersangkutan. Structure ‘human_t’ mengandung member health, strength, dan credits yang mana member tersebut dapat berubah-ubah nilainya dikarenakan terkena damage ataupun membeli barang pada toko.

Berikut adalah potongan kode untuk fungsi ‘init_computer’

```
158 void init_computer(computer_t *computer, const species_t *spec) {
159     computer->spec = spec;
160     computer->health = spec->health;
161 }
```

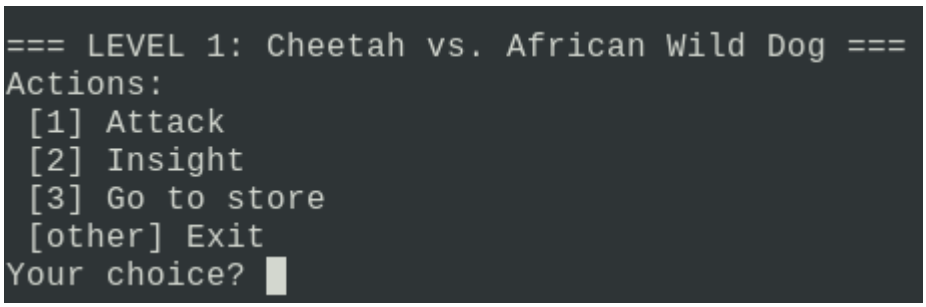
Pada dasarnya, mekanismenya sama dengan ‘init_human’ dengan perbedaan bahwa musuh hanya dapat mengalami perubahan health karena menerima serangan dari pemain.

Berikut adalah potongan kode untuk fungsi ‘init_packages’

```
163 void init_packages(void) {
164     int n = 0;
165 
166     g_packages[n].kind = pkgkind_bandage;
167     g_packages[n].name = "Fat";
168     g_packages[n].cost = 150;
169     g_packages[n].item.bandage_health = 25;
170     n++;
171 
172     g_packages[n].kind = pkgkind_bandage;
173     g_packages[n].name = "Mineral";
174     g_packages[n].cost = 250;
175     g_packages[n].item.bandage_health = 50;
176     n++;
177 
178     g_packages[n].kind = pkgkind_booster;
179     g_packages[n].name = "Carbohydrate";
180     g_packages[n].cost = 500;
181     g_packages[n].item.booster_strength = 1;
182     n++;
183 
184     g_packages[n].kind = pkgkind_booster;
185     g_packages[n].name = "Protein";
186     g_packages[n].cost = 800;
187     g_packages[n].item.booster_strength = 2;
188     n++;
189 }
```

Perintah-perintah dalam fungsi tersebut pada dasarnya mengisi seluruh array ‘g_packages’ dengan informasi mengenai masing-masing barang yang mana terdapat informasi kind, name, cost, dan item.bandage_strength atau item.booster_strength tergantung dengan tipe atau kind dari barang tersebut (kind bisa berupa pkgkind_bandage ataupun pkgkind_booster). Member ‘item’ tersimpan dalam union sehingga hanya nilai dari satu member saja yang dapat digunakan (misal kind berupa bandage, maka bandage_health saja yang bisa digunakan, begitu juga untuk booster).

2. Menu Utama



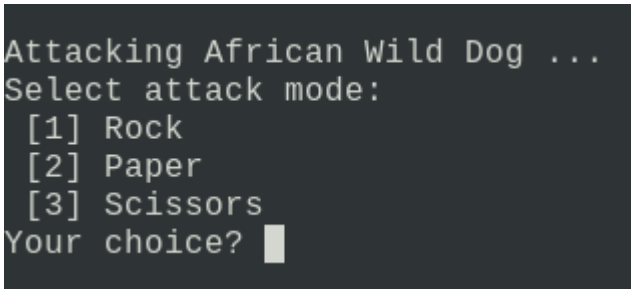
Setelah program melakukan inisiasi, maka program masuk ke dalam infinite loop yang bertujuan untuk menampilkan menu setiap saat dan berhenti saat diperlukan. Jika memilih 1, maka program akan navigasi ke pilihan menyerang. Jika memilih 2, maka program akan navigasi ke informasi permainan. Jika memilih 3, maka program akan navigasi ke menu toko.

Berikut adalah potongan kode yang digunakan untuk menerima masukan (perlu diperhatikan bahwa potongan kode di bawah terletak di dalam infinite loop).

```
129         printf("\n=== LEVEL %d: %s vs. %s ===\n", g_current_level, human.spec->name,
    computer.spec->name);
130         puts("Actions:");
131         puts(" [1] Attack");
132         puts(" [2] Insight");
133         puts(" [3] Go to store");
134         puts(" [other] Exit");
135         printf("Your choice? ");
136         scanf("%d", &action);
137
138         if (action == 1) {
139             attack(&human, &computer);
140         } else if (action == 2) {
141             insight(&human, &computer);
142         } else if (action == 3) {
143             go_to_store(&human);
144         } else {
145             break;
146         }
```

Program memanggil fungsi yang bersangkutan sesuai dengan pilihan pemain (action). Jika memilih 1, maka program akan memanggil fungsi ‘attack’ dengan input berupa pointer kepada variabel ‘human’ dan pointer kepada variabel ‘computer’. Jika memilih 2, maka program akan memanggil fungsi ‘insight’ dengan input berupa pointer menuju variabel ‘human’ dan pointer menuju variabel ‘computer’. Jika memilih 3, maka program akan memanggil fungsi ‘go_to_store’ dengan input berupa pointer menuju variabel ‘human’. Jika memilih angka selain 1, 2, dan 3, maka program akan berhenti karena bertemu dengan perintah ‘break’ yang mana digunakan untuk keluar dari perulangan while untuk infinite loop.

3. Pilihan Menyerang



Ketika pemain memilih opsi menyerang pada menu utama, maka program menyediakan pilihan antara batu, kertas, dan gunting yang masing-masing diberi nomor 1, 2, dan 3. Setelah pemain

menentukan pilihannya, giliran musuh yang menentukan pilihannya. Karena musuh dioperasikan oleh komputer, maka pilihan yang ditentukan oleh komputer adalah acak atau random menggunakan fungsi rand() yang didefinisikan pada header stdlib.h. Komputer memilih secara acak dalam rentang [0, 2] yang mana masing-masing menyatakan pilihan batu/gunting/kertas (0: batu, 1: kertas, 2: gunting).

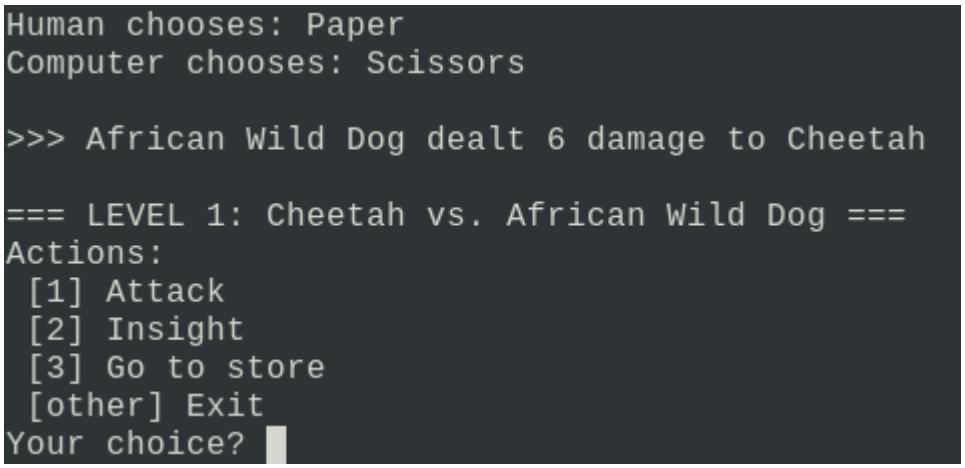
Berikut adalah potongan kode untuk fungsi ‘attack’

```
193 void attack(human_t *human, computer_t *computer) {
194     int human_mode;
195 
196     printf("\nAttacking %s ...\n", computer->spec->name);
197     puts("Select attack mode:");
198     puts(" [1] Rock");
199     puts(" [2] Paper");
200     puts(" [3] Scissors");
201     printf("Your choice? ");
202     scanf("%d", &human_mode);
203 
204     if ((1 <= human_mode) && (human_mode <= 3)) {
205         int damage = 0;
206         atkkind_t human_atk, computer_atk;
207 
208         switch (human_mode) {
209         case 1:
210             human_atk = atkkind_rock;
211             break;
212         case 2:
213             human_atk = atkkind_paper;
214             break;
215         case 3:
216             human_atk = atkkind_scissors;
217             break;
218         }
219 
220         computer_atk = (atkkind_t)random_range(atkkind_rock, atkkind_scissors);
221         printf("Human chooses: %s\n", g_atkkind_lookup[human_atk]);
222         printf("Computer chooses: %s\n", g_atkkind_lookup[computer_atk]);
223         if (wins(human_atk, computer_atk)) {
224             damage = randomize(human->strength, human->strength / 5);
225             computer->health = computer->health - damage;
226             printf("\n>>> %s dealt %d damage to %s\n", human->spec->name, damage,
    computer->spec->name);
227         } else if (wins(computer_atk, human_atk)) {
228             damage = randomize(computer->spec->strength, computer->spec->strength / 5);
229             human->health = human->health - damage;
230             printf("\n>>> %s dealt %d damage to %s\n", computer->spec->name, damage,
    human->spec->name);
231         } else {
232             puts("\n>>> No one attacked");
233         }
234 
235         return;
236     }
237     puts("Invalid input");
238     return attack(human, computer);
239 }
```

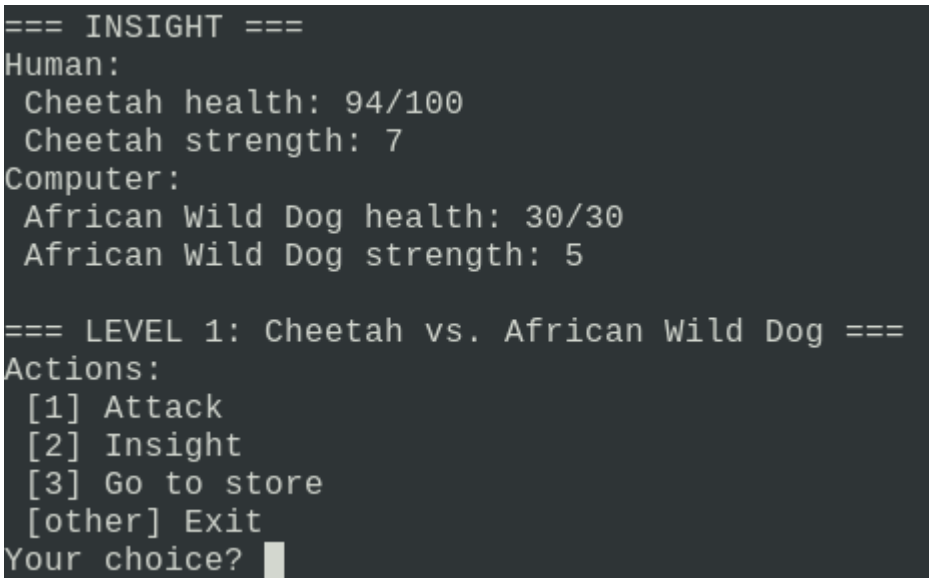
Input fungsi semuanya berupa pointer non-const yang berarti nilai pada variabel yang terkandung dalam pointer ‘human’ dan ‘computer’ berpotensi diubah oleh program. Kemudian komputer menentukan pilihan serangannya menggunakan fungsi ‘random_range’ yang hasilnya kemudian dikonversi ke tipe enum ‘atkkind_t’ di mana jika hasilnya 0, maka batu (atkkind_rock), jika 1 maka kertas (atkkind_paper), dan jika hasilnya 2 maka gunting (atkkind_scissors). Fungsi ‘random_range’ menerima input (a, b) di mana a merupakan batas bawah generasi bilangan acak dan b merupakan batas atas generasi bilangan acak, semuanya inklusif.

Setelah pemain dan musuh menentukan pilihan mereka masing-masing, maka tahap selanjutnya ialah menentukan pemenangnya dengan cara melakukan pemanggilan fungsi ‘wins’ yang memiliki

input (x, y) di mana x adalah jenis serangan penyerang dan y adalah jenis serangan yang diserang serta x atau y adalah salah satu di antara batu, gunting, dan kertas. Hasil dari pemanggilan fungsi tersebut bertipe boolean yang mana bernilai TRUE jika x menang dan bernilai FALSE jika x kalah. Pertama-tama, program mengecek apakah pemain menang terhadap musuh dengan melakukan substitusi x = human_atk dan y = computer_atk. Jika pemain menang, maka pemain melakukan serangan terhadap musuh dengan damage sebesar acak pada rentang [a, b] di mana a = max(s – s/5, 0) dan b = s + s/5, di mana s menyatakan strength atau kekuatan dari penyerang dalam integer. Langkah serupa juga diterapkan jika musuh menang terhadap pemain dengan cara yang berkebalikan (musuh melakukan serangan terhadap pemain).

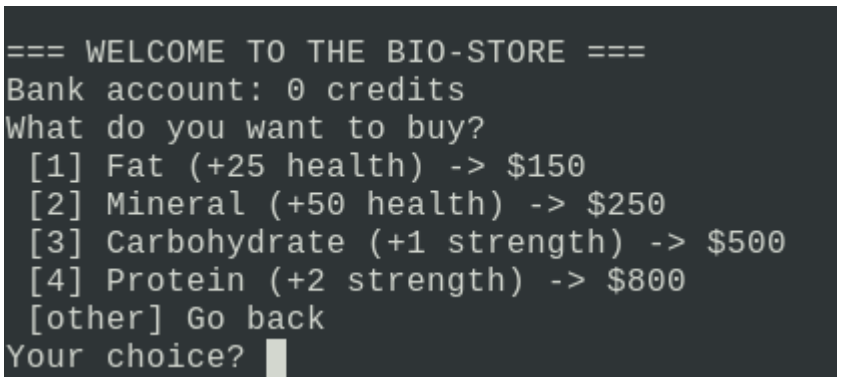


4. Informasi Pertandingan



Pada dasarnya informasi pertandingan hanya mencetak nilai-nilai dari struct member dalam variabel ‘human’ dan ‘computer’. Akan tetapi, pelewatan argumen ke fungsi ‘insight’ menggunakan pass-by-reference untuk menghemat memori sehingga tidak ada peng-copy-an seluruh isi dari variabel ‘human’ ataupun ‘computer’.

5. Menu Toko



Ketika pemain memilih toko pada menu utama, maka program menampilkan list barang yang ada di dalam variabel array ‘g_packages’ yang besarnya ditentukan oleh source code dan telah diisi oleh fungsi ‘init_packages’.

Berikut adalah potongan kode pada fungsi ‘go_to_store’

```
252
253 void go_to_store(human_t *human) {
254     int i, choice;
255
256     puts("\n=== WELCOME TO THE BIO-STORE ===");
257     printf("Bank account: %d credits\n", human->credits);
258     puts("What do you want to buy?");
259     for (i = 0; i < NUM_PACKAGES; i++) {
260         printf(" [%d] %s ", (i + 1), g_packages[i].name);
261         switch (g_packages[i].kind) {
262         case pkgkind_bandage:
263             printf("(+%d health) ", g_packages[i].item.bandage_health);
264             break;
265         case pkgkind_booster:
266             printf("(+%d strength) ", g_packages[i].item.booster_strength);
267             break;
268         }
269         printf("-> $%d\n", g_packages[i].cost);
270     }
271     puts(" [other] Go back");
272     printf("Your choice? ");
273     scanf("%d", &choice);
274
275     if ((1 <= choice) && (choice <= NUM_PACKAGES)) {
276         const package_t *pkg = &g_packages[choice - 1];
277
278         if (human->credits >= pkg->cost) {
279             switch (pkg->kind) {
280             case pkgkind_bandage:
281                 human->health = human->health + pkg->item.bandage_health;
282                 human->health = human->health > human->spec->health ? human->spec-
    >health : human->health;
283                 break;
284             case pkgkind_booster:
285                 human->strength = human->strength + pkg->item.booster_strength;
286                 break;
287             }
288             human->credits = human->credits - pkg->cost;
289             printf("\n>>> Successfully bought %s for $%d\n", pkg->name, pkg->cost);
290         } else {
291             puts("\n>>> Not enough credits :(");
292         }
293     } else {
294         puts("\n>>> Come back next time ;)");
295         return;
296     }
297     go_to_store(human);
298 }
299
```

Setelah pemain menentukan pilihan barang, maka program mengecek apakah credits mencukupi atau tidak. Jika mencukupi maka barang tersebut bisa dibeli dan langsung dipakai sehingga efeknya dapat langsung diterapkan. Jika barang yang dibeli berupa bandage, maka poin health pada pemain akan bertambah sesuai dengan spesifikasi barang, akan tetapi tetap memerhatikan bahwa health maksimum adalah health awal dari spesies pemain tersebut sehingga ketika hasil jumlah melebihi health maksimum, maka nilainya langsung diatur menjadi health maksimumnya. Kemudian jika barang yang dibeli berupa booster, maka poin strength pada pemain akan bertambah sesuai dengan spesifikasi barang. Perlu diperhatikan bahwa fungsi menerima input berupa pointer pada variabel bertipe ‘human_t’ dan bersifat non-const sehingga nilai pada variabel yang terkandung dalam pointer tersebut dapat diubah atau dimanipulasi oleh fungsi tersebut. Namun jika credits tidak mencukupi, maka barang tidak bisa dibeli sehingga tidak ada efek apapun. Fungsi tersebut menerapkan rekursi supaya dapat menerapkan infinite loop ketika berada pada menu toko. Kondisi base (base case) terletak pada ketika pemain memasukkan angka yang tidak valid (bukan merupakan nomor urut barang) sehingga akan keluar dari rekursi.

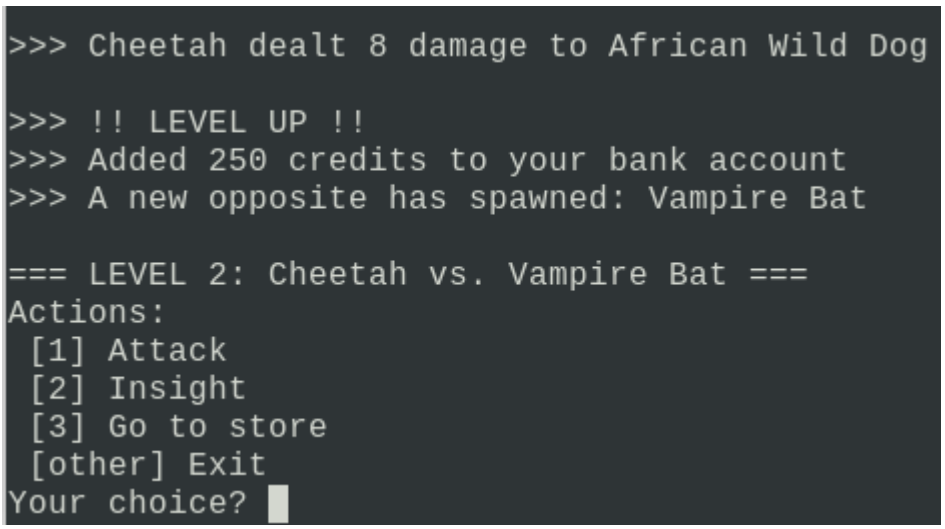
```
>>> Not enough credits :(

=== WELCOME TO THE BIO-STORE ===
Bank account: 0 credits
What do you want to buy?
 [1] Fat (+25 health) -> $150
 [2] Mineral (+50 health) -> $250
 [3] Carbohydrate (+1 strength) -> $500
 [4] Protein (+2 strength) -> $800
 [other] Go back
Your choice?
```

```
>>> Successfully bought Fat for $150

=== WELCOME TO THE BIO-STORE ===
Bank account: 100 credits
What do you want to buy?
 [1] Fat (+25 health) -> $150
 [2] Mineral (+50 health) -> $250
 [3] Carbohydrate (+1 strength) -> $500
 [4] Protein (+2 strength) -> $800
 [other] Go back
Your choice?
```

6. Event



Salah satu fitur yang menarik ialah ketika pemain berhasil membuat nyawa musuh menjadi kurang dari samadengan 0, maka level pemain akan bertambah sehingga lawan baru dengan spesies baru akan muncul (tentunya dengan konfigurasi nyawa dan kekuatan yang berbeda pula). Jika pemain naik dari level N menjadi level N+1, maka credits akan bertambah sebanyak N*250.

Berikut adalah potongan kode program yang digunakan untuk mengecek apakah level pemain harus bertambah.

```
342 bool check_level_up(const computer_t *computer) {
343     if (computer->health <= 0) {
344         puts("\n>>> !! LEVEL UP !!");
345         g_current_level++;
346         return true;
347     }
348     return false;
349 }
```

Fungsi tersebut menerima input berupa pointer konstan kepada variabel bertipe ‘computer_t’ (dengan kata lain, nilai variabel yang terdapat pada ‘computer’ tidak akan mengalami perubahan). Lalu fungsi tersebut mengecek apakah nyawa musuh kurang dari atau samadengan 0. Jika iya, maka level (g_current_level) dinaikkan dengan cara melakukan operasi increment dan fungsi me-return nilai TRUE. Jika kondisi tidak memenuhi (nyawa musuh belum habis), maka fungsi tersebut me-return nilai FALSE.

Berikut adalah potongan kode yang terdapat di dalam infinite loop pada fungsi main yang berguna untuk mengecek apakah pemain sudah menaikkan levelnya.

```
116         if (check_level_up(&computer)) {
117             int prize = (g_current_level - 1) * 250;
118             if (g_current_level > NUM_SPECIES_COMPUTER) {
119                 puts("\n>>> !! CONGRATULATIONS !!");
120                 puts(">>> You have passed through all levels");
121                 puts(">>> Have a good day ^_^");
122                 break;
123             }
124             human.credits += prize;
125             printf(">>> Added %d credits to your bank account\n", prize);
126 
127             init_computer(&computer, &g_computer_species[g_current_level - 1]);
128             printf(">>> A new opposite has spawned: %s\n", computer.spec->name);
129         }
```

Kode tersebut diletakkan sebelum program menampilkan menu utama sehingga dilakukan pengecekan terlebih dahulu jika sudah level up. Kemudian jika level pemain sudah melebihi batas akhir atau jumlah musuh, maka permainan sudah selesai dan dimenangkan oleh pemain tersebut.

```
>>> Hippo Barry dealt 5331 damage to Narwhal

>>> !! LEVEL UP !!

>>> !! CONGRATULATIONS !!
>>> You have passed through all levels
>>> Have a good day ^_^
```

Namun, ketika nyawa pemain sudah habis terlebih dahulu, maka permainan akan selesai begitu saja. Berikut adalah potongan kode yang digunakan untuk mengecek apakah pemain gagal dalam permainan ini.

```
331
332 bool check_gameover(const human_t *human) {
333     if (human->health <= 0) {
334         puts("\n>>> !! GAME OVER !!");
335         printf(">>> Current level: %d\n", g_current_level);
336         puts(">>> Better luck next time ;)");
337         return true;
338     }
339     return false;
340 }
341
```

Fungsi tersebut mengecek apakah nyawa pemain kurang dari sama dengan 0 (dalam arti, nyawa sudah habis). Pemanggilan fungsi tersebut sejenis dengan pemanggilan fungsi ‘check_level_up’ yang mana terletak pada infinite loop tepatnya sebelum program menampilkan menu utama. Kemudian melakukan pengecekan apakah check_level_up(P) bernilai TRUE (jika iya, maka eksekusi perintah ‘break’ yang mana akan mengakhiri infinite loop sehingga program dapat keluar) di mana P adalah pointer kepada variabel pemain.

```
>>> African Wild Dog dealt 5 damage to Hippo Barry

>>> !! GAME OVER !!
>>> Current level: 1
>>> Better luck next time ;)
```