|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería de Software | **GUÍA**  No. 01 | **TIEMPO ESTIMADO:**  1h y 20 min. |
| **ASIGNATURA:**  Estructura de Datos  NRC: 2967 | **FECHA DE ELABORACION:** 26/01/2020  **SEMESTRE**: septiembre 2019 – febrero 2020 | |
| **TÍTULO:**  Arimaa (Backtraking) | **DOCENTE:** Ing. Fernando Solís | |

**OBJETIVO**

Implementar mediante el backtraking un juego derivado del ajedrez llamada Arimaa que consiste en tener 4 casillas trampa y es prohibido caer ahí.

**INSTRUCCIONES**

1. Utilice como material principal, aquel indicado en clase por el docente.
2. Utilice información consultada en Internet y conocimiento adquirido en clase.
3. Respetar los movimientos del caballo
4. No caer en el ismo sitio más de una vez
5. Encontrar la mejor solución

**ACTIVIDADES**

1. **Ubicación de recursos**
2. Formar grupos de máximo 2 personas por computador
3. Instalar la herramienta C++
4. Investigar temas por cuenta propia
5. **Planteamiento del problema**

**Ejercicio No 1**

Desarrollar un programa que servirá para demostrar el uso del algoritmo del Backtraking (Recursividad) en donde un cabello del tablero de ajedrez tiene que recorrerlo sin caer más de una vez, no obstante, la implementación se desarrollará en un juego llamada Arimaa (derivado del ajedrez) que tiene cuatro casillas trampa y no puede caer en ellas más de una vez.

**Funciones C++ a utilizar:**

* **kbhit() -** Revisa si una tecla pulsada está disponible
* **getch() -** Lee un solo carácter directamente desde el teclado, sin mostrar tal carácter en pantalla.
* **rand() -** Se utiliza para obtener un [número aleatorio](https://es.wikipedia.org/wiki/N%C3%BAmero_aleatorio).
* **system() -** Sirve para ejecutar subprocesos o comandos del sistema operativo.
* **pthread\_mutex\_lock() -** Permite solicitar acceso al mutex, el hilo se bloquea hasata su obtención
* **pthread\_mutex\_unlock()** - Permute liberar un mutex.
* **abort() –** Si no se encontró el pthread sale de la ejecución

**Marco Teórico**

**Arimaa**

Es un [juego de mesa](https://es.wikipedia.org/wiki/Juego_de_mesa) de [estrategia por turnos](https://es.wikipedia.org/wiki/Juego_de_estrategia_por_turnos) para dos jugadores que puede ser jugado con el mismo equipo, tablero y piezas que el [ajedrez](https://es.wikipedia.org/wiki/Ajedrez). Arimaa ha probado hasta ahora, ser más difícil para jugadores con [inteligencia artificial](https://es.wikipedia.org/wiki/Inteligencia_artificial) ([computadoras](https://es.wikipedia.org/wiki/Computadora)) que el ajedrez.

Arimaa fue inventado por [Omar Syed](https://es.wikipedia.org/w/index.php?title=Omar_Syed&action=edit&redlink=1), un ingeniero en computación con especialización en [inteligencia artificial](https://es.wikipedia.org/wiki/Inteligencia_artificial). Syed se inspiró en la derrota de [Garry Kasparov](https://es.wikipedia.org/wiki/Garry_Kasparov) en las manos del famoso [computador para ajedrez](https://es.wikipedia.org/wiki/Ajedrez#Ordenadores_y_ajedrez) [IBM](https://es.wikipedia.org/wiki/IBM) [Deep Blue](https://es.wikipedia.org/wiki/Deep_Blue_(computadora)), para diseñar un juego nuevo que pudiera ser jugado con el mismo conjunto de piezas de ajedrez, que fuera difícil que un computador lo jugara bien, pero que tuviera reglas tan fáciles y simples como para que su hijo de cuatro años, Aamir, lo entendiera. El origen del nombre "Arimaa", viene de "Aamir" escrito al revés más una "a" inicial. En el 2002, Syed publicó las reglas para Arimaa y anunció un premio de 10.000 [USD](https://es.wikipedia.org/wiki/USD), disponible anualmente hasta el año 2020, para el desarrollador, grupo o Corporación que creara el primer programa de computadora (ejecutándose en [hardware](https://es.wikipedia.org/wiki/Hardware) barato y de común obtención) que fuese capaz de vencer a un jugador de alto nivel en un partido de seis juegos o más.[1](https://es.wikipedia.org/wiki/Arimaa#cite_note-1)​ Syed también obtuvo una patente sobre las reglas de Arimaa, y logró conseguir que "Arimaa" fuera una marca registrada bajo su nombre.

**Backtraking**

Es una estrategia para encontrar soluciones a problemas que satisfacen restricciones. El término "backtrack" fue acuñado por primera vez por el matemático estadounidense [D. H. Lehmer](https://es.wikipedia.org/wiki/Derrick_Henry_Lehmer) en la [década de 1950](https://es.wikipedia.org/wiki/D%C3%A9cada_de_1950).

Los problemas que deben satisfacer un determinado tipo de restricciones son problemas completos, donde el orden de los elementos de la solución no importa. Estos problemas consisten en un conjunto (o lista) de variables a la que a cada una se le debe asignar un valor sujeto a las restricciones del problema. La técnica va creando todas las posibles combinaciones de elementos para obtener una solución. Su principal virtud es que en la mayoría de las implementaciones se puede evitar combinaciones, estableciendo funciones de acotación (o poda) reduciendo el [tiempo](https://es.wikipedia.org/w/index.php?title=Eficiencia_de_los_algoritmos&action=edit&redlink=1) de ejecución.

Vuelta atrás está muy relacionado con la [búsqueda combinatoria](https://es.wikipedia.org/w/index.php?title=B%C3%BAsqueda_combinatoria&action=edit&redlink=1).

**Código**

A continuación, se presenta el código de la clase Arimaa

**Arimaa.h**

1. include <fstream>
2. #include <pthread.h>
4. using namespace std;
6. int contador=0;
7. pthread\_mutex\_t ptmutex1;
9. class Arimaa {
10. private:
11. ofstream archivo;
12. int tamanio;
13. int \*moverX;
14. int \*moverY;
15. public:
16. bool verificar(int,int,int,int\*\*);
17. int resolver(int,int,int,int,int\*,int\*,int\*\*);
18. void imprimir(int,int\*\*);
19. bool solucion();
20. void\* funcion1(void\*);
21. void\* funcion2(void\*);
22. void principal();
23. void menu();
24. };
25. Arimaa.cpp
26. #include <stdlib.h>
27. #include "Arimaa.h"
28. #include "VectorMatriz.cpp"
29. #include "Marquesina.cpp"
31. bool Arimaa::verificar(int x, int y, int tamanio, int \*\*matriz) {
32. **return** (x >= 0 && x < tamanio && y >= 0 && y < tamanio && \*(\*(matriz+x)+y) == -1);
33. }
34. int Arimaa::resolver(int x, int y, int numero, int tamanio, int \*moverX, int \*moverY, int \*\*matriz) {
35. int siguienteX, siguienteY;
36. **if**(numero == tamanio\*tamanio) {
37. **return** true;
38. }
39. **for**(int i=0; i<tamanio; i++) {
40. siguienteX = x + \*(moverX+i);
41. siguienteY = y + \*(moverY+i);
42. **if**(verificar(siguienteX, siguienteY, tamanio,matriz)) {
43. \*(\*(matriz+siguienteX)+siguienteY) = numero;
44. **if**(numero == 36 || numero == 27 || numero == 45 || numero == 20) {
45. \*(\*(matriz+siguienteX)+siguienteY) = 0;
46. }
47. **if**(resolver(siguienteX, siguienteY, numero+1, tamanio,moverX, moverY, matriz) == true) {
48. **return** true;
49. }**else** {
50. \*(\*(matriz+siguienteX)+siguienteY) = -1;
51. }
52. }
53. **if**(contador <= 10) {
54. imprimir(tamanio,matriz);
55. }
56. }
57. contador++;
58. **return** false;
59. }
61. void Arimaa::imprimir(int tamanio,int \*\*matriz) {
62. **for**(int i=0; i<tamanio; i++) {
63. **for**(int j=0; j<tamanio; j++) {
64. archivo << \*(\*(matriz+i)+j) << "     ";
65. }
66. archivo << endl;
67. }
68. archivo << endl;
69. }
71. bool Arimaa::solucion() {
72. int tamanio;
73. int \*\*matriz;
74. VectorMatriz generar;
75. Marquesina iniciar;
76. tamanio = iniciar.ingresarNumero();
77. matriz = generar.inicializar(tamanio);
78. matriz = generar.encerar(tamanio,matriz);
79. archivo.open("solucion.txt",ios::out);
80. **for**(int i=0; i<tamanio; i++) {
81. **for**(int j=0; j<tamanio; j++) {
82. \*(\*(matriz+i)+j) = -1;
83. }
84. }
85. moverX = generar.inicializarVector(tamanio);
86. moverX = generar.encerarVector(tamanio,moverX);
87. moverY = generar.inicializarVector(tamanio);
88. moverY = generar.encerarVector(tamanio,moverY);
89. \*(moverX+0) = 2;
90. \*(moverX+1) = 1;
91. \*(moverX+2) = -1;
92. \*(moverX+3) = -2;
93. \*(moverX+4) = -2;
94. \*(moverX+5) = -1;
95. \*(moverX+6) = 1;
96. \*(moverX+7) = 2;
97. \*(moverY+0) = 1;
98. \*(moverY+1) = 2;
99. \*(moverY+2) = 2;
100. \*(moverY+3) = 1;
101. \*(moverY+4) = -1;
102. \*(moverY+5) = -2;
103. \*(moverY+6) = -2;
104. \*(moverY+7) = -1;
105. \*(\*(matriz+0)+0) = 0;
106. **if**(resolver(0,0,1,tamanio,moverX,moverY,matriz) == false) {
107. cout << "La solucion no existe";
108. **return** false;
109. }**else** {
110. imprimir(tamanio,matriz);
111. int esperar;
112. cout << endl << endl << endl << "Se creo el archivo";
113. muestraPDF();
114. }
115. archivo.close();
116. **return** true;
117. }
119. void Arimaa::menu() {
120. bool bandera=false;
121. char tecla;
122. **do** {
123. cout << endl << endl <<"1. Generar archivo" << endl;
124. cout << "2. Abrir QR" << endl;
125. cout << "3. Abrir codigo de barras" << endl;
126. cout << "4. Abrir ayuda" << endl;
127. cout << "5. Salir" << endl;
128. cout << "Elije una opcion: ";
129. cin >> tecla;
130. **switch**(tecla) {
131. **case** '1': {
132. system("cls");
133. Arimaa arimaa;
134. arimaa.solucion();
135. system("pause");
136. system("cls");
137. break;
138. }
139. **case** '2':
140. system("cls");
141. system("QR.png");
142. break;
144. **case** '3':
145. system("cls");
146. system("barra.jpeg");
147. break;
149. **case** '4':
150. system("cls");
151. system("Arimaa.chm");
152. break;
154. **case** '5':
155. exit(0);
156. break;
158. **default**:
159. system("cls");
160. cout << endl << endl << endl << endl << "Opcion no valida.**\a\n**";
161. break;
162. }
163. }**while**(bandera!=true);
164. }
166. void\* funcion1(void \*arg) {
167. Arimaa arimaa;
168. pthread\_mutex\_lock(&ptmutex1);
169. Sleep(50);
170. arimaa.menu();
171. pthread\_mutex\_unlock(&ptmutex1);
172. **return** NULL;
173. }
175. void\* funcion2(void \*arg) {
176. Marquesina marquesina;
177. pthread\_mutex\_lock(&ptmutex1);
178. marquesina.hacerMarquesina();
179. Sleep(5000);
180. pthread\_mutex\_unlock(&ptmutex1);
181. **return** NULL;
182. }
184. void principal() {
185. pthread\_t pthread1, pthread2;
186. **if** (pthread\_create(&pthread1, NULL, funcion1, NULL) ) {
187. cout << ("Error creando el hilo.");
188. abort();
189. }
190. **if** (pthread\_create(&pthread2, NULL, funcion2, NULL) ) {
191. printf("Error creando el hilo.");
192. abort();
193. }
194. **if** (pthread\_join(pthread1, NULL)) {
195. cout << ("Error creando el hilo.");
196. abort();
197. }
198. **if** (pthread\_join(pthread2, NULL)) {
199. cout << ("Error creando el hilo.");
200. abort();
201. }
202. }

**Conclusiones**

* Se desarrolló el algoritmo backtraking de forma correcta encontrando la solución al juego Arimaa.
* Se aprendió más sobre el funcionamiento del algoritmo y como debe ser implementado.
* Se fortaleció los conocimientos y aprendizajes de los algoritmos recursivos mediante el ejercicio de aplicación del Arimaa.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DOCENTE RESPONSABLE Integrantes del Grupo

Ing. Fernando Solis. MsC. Carlos Puco, Kevin Duy