



Instituto Tecnológico de Costa Rica

Proyecto 4

Programación Imperativa lenguaje C

Jesus Araya Chaves 2021106062

Curso: Lenguajes de Programación

Número de grupo: 20

Profesor: Maria Auxiliadora Mora

Primer semestre del 2023

Descripción del sistema	3
Diseño de componentes	4
Tablero	4
Manejo de jugadores	4
Como se mantiene el estado del juego	4
Ejecución del código	5

Descripción del sistema

En esta sección se describirán brevemente las estructuras y funciones del programa, se describen de manera breve ya que en la documentación interna del código tienen una descripción más amplia:

1. ``struct Node``: Representa un único nodo en el tablero de juego. Cada nodo tiene un número de vértice único y es propiedad de un jugador (0 si está vacío, 1 para el Jugador 1 y 2 para el Jugador 2).
2. ``struct ListNode``: Representa un nodo en la lista de adyacencia. Cada nodo tiene un puntero a su Nodo correspondiente en el grafo, y un puntero al siguiente ListNode en la lista de adyacencia.
3. ``struct ListNodePath``: Representa un nodo en la lista de posibles caminos. Cada nodo contiene un valor (número de vértice) y un puntero al siguiente ListNodePath.
4. ``struct Graph``: Representa el tablero de juego, compuesto por todos los Nodos, sus listas de adyacencia y posibles caminos. Incluye propiedades para el tamaño de la cuadrícula y el número total de vértices (nodos).
5. ``createNode(int vertex, int player)``: Crea un nuevo Nodo con un número de vértice dado y lo asigna a un jugador.
6. ``createListNode(struct Node* node)``: Crea un nuevo ListNode que apunta a un Nodo dado.
7. ``addEdge(struct Graph* graph, int src, int dest)``: Añade un borde no dirigido entre dos nodos, es decir, añade cada nodo a la lista de adyacencia del otro.
8. ``initializeBoard(int gridSize)``: Inicializa el tablero de juego con un tamaño de cuadrícula dado. Genera dos cuadrículas conectadas por un único nodo. Cada celda de la cuadrícula es un vértice del grafo. Esta función también establece las listas de adyacencia para todos los nodos y define la colocación inicial de las piezas del jugador en el tablero.
9. ``printBoard``: Imprime el tablero de juego en la consola. El tablero se imprime en dos partes para representar dos cuadrículas para dos jugadores. Cada nodo se asigna a un color según el jugador al que pertenece. Un nodo de conexión especial se imprime entre las dos cuadrículas.
10. ``validPath``: Verifica si un nodo está en una ruta específica en el gráfico. Si el nodo está en el camino, devuelve verdadero, de lo contrario, devuelve falso.
11. ``possiblePlays``: Imprime todos los movimientos posibles para un jugador en el juego. Los movimientos posibles se basan en la posición actual de las piezas del jugador y las posiciones de las piezas del oponente.
12. ``movePiece``: Mueve una pieza de un jugador de una posición a otra en el tablero de juego. La función tiene en cuenta varios factores, como si la casilla de destino está vacía, si el destino está adyacente a la posición actual y si hay una pieza del oponente en el destino.
13. ``freeMemory``: Desasigna la memoria utilizada para el tablero y todas las estructuras asociadas.

Diseño de componentes

En esta sección se describirán algunos componentes utilizados para la creación del juego:

Tablero

El tablero del juego está representado por una estructura de datos de tipo grafo compuesta por nodos conectados por aristas. Cada nodo representa una celda en la cuadrícula del juego y puede pertenecer a uno de los jugadores o estar vacío. El grafo está compuesto de dos cuadrículas, cada una de tamaño `gridSize x gridSize`, conectadas por un solo nodo central. Las conexiones entre nodos, que representan las celdas directamente alcanzables entre sí, se manejan utilizando listas de adyacencia. Además, existe una lista de posibles rutas que representan posibles trayectorias para que una pieza se desplace. Todo el grafo, incluyendo los nodos, sus listas de adyacencia, y las posibles rutas, está encapsulado dentro de una estructura `Graph`.

Manejo de jugadores

Cada jugador tiene nueve piezas, y el objetivo es agotar las piezas del oponente, cada nodo en el grafo tiene un número que dice a qué jugador pertenece el nodo.

Al principio, el jugador que iniciará el juego se selecciona mediante una entrada del usuario.

En cada turno, se le pide al jugador actual que seleccione una pieza para mover e indique el destino de la pieza. Si el movimiento es válido, se realiza y el turno cambia al otro jugador. Si el movimiento es inválido, el jugador recibe un mensaje de error y debe intentar de nuevo.

El programa continuará cambiando turnos entre los jugadores hasta que uno de ellos se quede sin piezas.

Como se mantiene el estado del juego

El estado del juego se mantiene a través de una estructura de grafo, donde cada nodo representa una casilla del tablero y almacena información sobre la pieza en esa casilla. Cuando se realiza un movimiento, la función `movePiece()` cambia el estado del nodo de origen y destino. Si una pieza es movida, su nodo correspondiente se actualiza al jugador adecuado. Si una pieza es capturada, el nodo correspondiente se vacía, lo que indica que ya no hay ninguna pieza en esa casilla. Las actualizaciones del estado se reflejan visualmente en el tablero con la función `printBoard()`, la cual itera a través de cada nodo en el grafo, imprime un marcador de posición con un color específico que indica la pieza de un jugador. Los nodos vacíos se representan con el color blanco, las piezas del jugador 1 con rojo y las del jugador 2 con azul.

Ejecución del código

Para la ejecución del código se puede hacer de dos maneras, la primera es utilizando un IDE como CLion (programa utilizado para el desarrollo de esta tarea) o se puede utilizar el compilador de C, ejecutando:

```
>gcc main.c  
>./a.exe
```

Para estos pasos, se utilizará el compilador de C, ya que muestra mejores resultados gráficamente ejecutarlo desde la terminal:

1. Compilar el programa con los comandos mencionados anteriormente.
2. Seleccionar el jugador que va a comenzar.

```
PS C:\Users\chuss\Documents\code\languages-project-4> gcc main.c  
PS C:\Users\chuss\Documents\code\languages-project-4> ./a.exe  
Who will start playing? (1 or 2): 1  
Enter the size of the board: |
```

3. Ahora podemos observar que se pide el tamaño del tablero, para efectos de la documentación se creará un tablero de tamaño 3, pero de igual manera, se adjuntan tableros de más tamaño.

```
Who will start playing? (1 or 2): 1  
Enter the size of the board: 10  
0 1 2 3 4 5 6 7 8 9 - 100 - 101 102 103 104 105 106 107 108 109 110  
10 11 12 13 14 15 16 17 18 19 - 100 - 111 112 113 114 115 116 117 118 119 120  
20 21 22 23 24 25 26 27 28 29 - 100 - 121 122 123 124 125 126 127 128 129 130  
30 31 32 33 34 35 36 37 38 39 - 100 - 131 132 133 134 135 136 137 138 139 140  
40 41 42 43 44 45 46 47 48 49 - 100 - 141 142 143 144 145 146 147 148 149 150  
50 51 52 53 54 55 56 57 58 59 - 100 - 151 152 153 154 155 156 157 158 159 160  
60 61 62 63 64 65 66 67 68 69 - 100 - 161 162 163 164 165 166 167 168 169 170  
70 71 72 73 74 75 76 77 78 79 - 100 - 171 172 173 174 175 176 177 178 179 180  
80 81 82 83 84 85 86 87 88 89 - 100 - 181 182 183 184 185 186 187 188 189 190  
90 91 92 93 94 95 96 97 98 99 - 100 - 191 192 193 194 195 196 197 198 199 200
```

Tablero de tamaño 10.

```
Who will start playing? (1 or 2): 1  
Enter the size of the board: 15  
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 - 225 - 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240  
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 - 225 - 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255  
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 - 225 - 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270  
45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 - 225 - 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285  
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 - 225 - 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300  
75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 - 225 - 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315  
90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 - 225 - 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330  
105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 - 225 - 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345  
120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 - 225 - 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360  
135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 - 225 - 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375  
150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 - 225 - 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390  
165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 - 225 - 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405  
180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 - 225 - 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420  
195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 - 225 - 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435  
210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 - 225 - 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450
```

Tablero de tamaño 15.

```
Who will start playing? (1 or 2): 1  
Enter the size of the board: 3  
0 1 2 - 9 - 10 11 12  
3 4 5 - 9 - 13 14 15  
6 7 8 - 9 - 16 17 18  
Possible plays for player 1:
```

Tablero de tamaño 3.

4. Cuando se selecciona el tamaño, se despliega el tablero. A partir de este momento el juego comienza, al momento de desplegar el tablero también se muestran unos datos de ayuda, como por ejemplo, una lista de posibles jugadas para el jugador y la cantidad de fichas que cada jugador tiene.

```
Who will start playing? (1 or 2): 1
Enter the size of the board: 3
0      1      2      - 9 -    10      11      12
3      4      5      - 9 -    13      14      15
6      7      8      - 9 -    16      17      18
Possible plays for player 1:
2 : 9   5 : 9   8 : 9

Pieces available for player 1 & player 2:
9 : 9

Player 1's turn.
Enter the index of the piece you want to move: |
```

5. Ahora, se pueden mover las fichas como el jugador desee.

```

Player 1's turn.
Enter the index of the piece you want to move: 2
Enter the index of where you want to move the piece to: 9
Move successful!
0      1      2      - 9 -    10      11      12
3      4      5      - 9 -    13      14      15
6      7      8      - 9 -    16      17      18
Possible plays for player 2:
16 : 2

Pieces available for player 1 & player 2:
9 : 9

Player 2's turn.
Enter the index of the piece you want to move: 16
Enter the index of where you want to move the piece to: 2
Move successful!
0      1      2      - 9 -    10      11      12
3      4      5      - 9 -    13      14      15
6      7      8      - 9 -    16      17      18
Possible plays for player 1:
1 : 9   5 : 9   5 : 9   8 : 9

Pieces available for player 1 & player 2:
8 : 9

Player 1's turn.
Enter the index of the piece you want to move: 5
Enter the index of where you want to move the piece to: 16
Invalid move, destination slot is not adjacent!
Possible plays for player 1:
1 : 9   5 : 9   5 : 9   8 : 9

Pieces available for player 1 & player 2:
8 : 9

Player 1's turn.
Enter the index of the piece you want to move: |

```

6. Se puede observar que el juego tiene validaciones para que las fichas no se puedan mover a posiciones inválidas. Muestra un mensaje de error y permite al jugador volver a intentar.
7. Cuando un usuario se queda sin fichas, muestra un mensaje diciendo cual jugador ganó.

Move successful!

0	1	2	- 9 -	10	11	12
3	4	5	- 9 -	13	14	15
6	7	8	- 9 -	16	17	18

Possible plays for player 1:

Pieces available for player 1 & player 2:

0 : 7

- Player 2 Wins! -

PS C:\Users\chuss\Documents\code\languages-project-4> |