```
/*
* Juego de la Serpiente v1
*
* Pablo_Villa 08/11/2023
*
* version en la que por cada manzana
* comida se incrementa en 1 el tamaño
*/
#include <iostream>
#include "terminal.h"
#include <cstdlib>
#include <ctime>

using namespace std;

const char TECLA_SIGUIENTE = ' ';
const char TECLA_FIN = 'F';
const char SERPIENTE = '@';
const char CUERPO_SERPIENTE = 'o';
const char ARRIBA = 'W';
const char ABAJO = 'S';
const char IZQUIERDA = 'A';
const char DERECHA = 'D';
const char MANZANA = 'M';
const char SIMBOLO_VERTICAL = '|';
const char SIMBOLO_INTERMEDIO = ' ';
const char SIMBOLO_HORIZONTAl = '-';
const char SIMBOLO_EXTERIOR = '+';
const int PREMIO = 100;
const int LONG_SERPIENTE = 100;
const int BASE = 80;
const int ALTURA = 22;
const int RETARDO = 50;
const int SERPIENTE_X_INICIAL = 10;
const int LIMITE_SUPERIOR = 1;
const int LIMITE_INFERIOR = 20 ;
const int LIMITE_IZQUIERDA = 2;
const int LIMITE_DERECHA = 78;
const int SERPIENTE_Y_INICIAL = 15;
const int MOVIMIENTO_X_DERECHA = 1;
const int MOVIMIENTO_Y_DESCENDENTE = 1;
const int MOVIMIENTO_X_IZQUIERDA = -1;
const int MOVIMIENTO_Y_ASCENDENTE = -1;
const int MARGEN_INI_MANZANA = 5;
const int MARGEN_MARCADOR = 5;
const int MAX_MANZANAS = 10;
const string TITULO = "Juego de la serpiente ";
const string VERSION = "5.0";
const string TECLA_CONTINUAR = "ESPACIO";

struct posicion {
    int x = 0;
    int y = 0;
};

struct inc_unitario_posicion {
    int x = 0;
    int y = 0;
};

void iniciar_pantalla_inicial();
void inicializar_juego(char tecla,posicion serpiente[],posicion& manzana,
                       inc_unitario_posicion& inc_unitario_posicion, int longitud_serpiente);
void pantalla_incial();
void dibujar_linea(const char c_exterior, const char c_interior, const int largo);
void dibujar_rectangulo(const int base, const int altura);
```

```
 67  void inicializar_serpiente(posicion serpiente[], inc_unitario_posicion& inc_unitario_posicion,int
longitud_serpiente);
 68  bool juego_terminado(char tecla,posicion serpiente[],int longitud_serpiente);
 69  void obtener_direccion_serpiente(const char tecla,inc_unitario_posicion& inc_unitario_posicion);
 70  void pintar_serpiente(const posicion serpiente[], int longitud_serpiente);
 71  void borrar_serpiente( const posicion serpiente[], int longitud_serpiente);
 72  bool serpiente_tocada(const posicion serpiente[],int longitud_serpiente);
 73  void inicializar_manzana(posicion& manzana);
 74  void pintar_manzana(const posicion& manzana);
 75  void borrar_manzana(const posicion& manzana);
 76  bool manzana_tocada(const posicion& manzana, const posicion serpiente[]);
 77  void mover_serpiente(posicion serpiente[], inc_unitario_posicion inc_unitario_posicion);
 78  void actualizar_marcador(int& puntos);
 79  void actualizar_longitud(int& longitud_serpiente);
 80
 81  int main() {
 82      int puntos = 0;
 83      bool hay_manzana = false;
 84      char tecla = '\0';
 85      int longitud_serpiente = 5;
 86      posicion serpiente[LONG_SERPIENTE];
 87      posicion manzana = {0, 0};
 88      inc_unitario_posicion inc_unitario_posicion = {0, 0};
 89
 90      inicializar_juego(tecla, serpiente, manzana, inc_unitario_posicion,longitud_serpiente);
 91      while ( ! juego_terminado(tecla, serpiente,longitud_serpiente)) {
 92
 93          if( ! hay_manzana){
 94              inicializar_manzana(manzana);
 95              pintar_manzana(manzana);
 96              hay_manzana = true;
 97          }
 98          if( manzana_tocada(manzana, serpiente)) {
 99              hay_manzana = false;
100              actualizar_longitud(longitud_serpiente);
101              poner_cursor(1,28);
102              cout << longitud_serpiente;
103              actualizar_marcador(puntos);
104          }
105
106          pintar_serpiente(serpiente,longitud_serpiente);
107
108          retardar(RETARDO);
109
110          borrar_serpiente(serpiente,longitud_serpiente);
111
112          obtener_direccion_serpiente(tecla, inc_unitario_posicion);
113          mover_serpiente(serpiente, inc_unitario_posicion);
114
115          tecla = leer_tecla();
116      }
117      deshabilitar_modo_crudo_terminal();
118      borrar_terminal();
119  }
120
121  void iniciar_pantalla_inicial(){
122      retardar(RETARDO);
123      hacer_cursor_visible(false);
124      pantalla_incial();
125  }
126
127  void inicializar_juego(char tecla, posicion serpiente[], posicion& manzana,
128                         inc_unitario_posicion& inc_unitario_posicion,int longitud_serpiente){
129      srand(time(0));
130      setlocale(LC_ALL, "");
131      iniciar_pantalla_inicial();
```

```
132
133     while(leer_tecla()!= TECLA_SIGUIENTE){
134             retardar(RETARDO);
135     }
136     deshabilitar_modo_crudo_terminal();
137     borrar_terminal();
138
139     inicializar_serpiente(serpiente, inc_unitario_posicion,longitud_serpiente);
140
141     dibujar_rectangulo(BASE, ALTURA);
142
143     habilitar_modo_crudo_terminal();
144     hacer_cursor_visible(false);
145     tecla = leer_tecla();
146 }
147
148 void pantalla_incial(){
149
150     poner_cursor(1,1);
151     cout << " **************************** " << endl;
152     poner_cursor(1,2);
153     cout << " * "<< TITULO << VERSION << " * " << endl;
154     poner_cursor(1,3);
155     cout << " **************************** " << endl;
156     poner_cursor(1,6);
157     cout << "   ______" << endl;
158     poner_cursor(1,7);
159     cout << " _/      \\ " << endl;
160     poner_cursor(1,8);
161     cout << "  \\___    \\ " << endl;
162     poner_cursor(1,9);
163     cout << "      \\    \\_____" << endl;
164     poner_cursor(1,10);
165     cout << "       \\         \\ " << endl;
166     poner_cursor(1,11);
167     cout << "        \\____     \\______      _|_ " << endl;
168     poner_cursor(1,12);
169     cout << "             \\         0 \\_/  /   \\ " << endl;
170     poner_cursor(1,13);
171     cout << "              \\__________/ \\  \\___/" << endl;
172     poner_cursor(1,17);
173     cout << "Pulsa la tecla de " <<  TECLA_CONTINUAR << " para continuar" << endl;
174 }
175
176 void inicializar_serpiente(posicion serpiente[], inc_unitario_posicion& inc_unitario_posicion,int
longitud_serpiente) {
177     serpiente[0].x = SERPIENTE_X_INICIAL;
178     serpiente[0].y = SERPIENTE_Y_INICIAL;
179
180     inc_unitario_posicion.x = MOVIMIENTO_X_DERECHA;
181     inc_unitario_posicion.y = MOVIMIENTO_Y_ASCENDENTE;
182
183     for (int i = 1; i < longitud_serpiente; i++) {
184         serpiente[i].x = serpiente[i - 1].x + 1;
185         serpiente[i].y = serpiente[i - 1].y;
186     }
187 }
188
189 void dibujar_linea(const char c_exterior, const char c_interior, const int largo){
190     cout << c_exterior;
191     for(int i = 0; i < largo - 2; i++){
192         cout << c_interior;
193     }
194     cout << c_exterior << endl;
195 }
196
```

```
197 void dibujar_rectangulo(const int base, const int altura){
198     poner_cursor(2,1);
199     cout << "+------------------------- " << TITULO
200         << VERSION << " -------------------------+ " << endl;
201     for (int i = 2; i < altura -2; i++){
202         poner_cursor(2,i);
203         dibujar_linea(SIMBOLO_VERTICAL,SIMBOLO_INTERMEDIO,base);
204     }
205     poner_cursor(2,altura - 2);
206     dibujar_linea(SIMBOLO_EXTERIOR,SIMBOLO_HORIZONTAl,base);
207     poner_cursor(2, altura);
208     cout << ARRIBA << "-> Subir " << ABAJO << "-> Bajar " << IZQUIERDA
209          << "-> Izda " << DERECHA << "-> Dcha " << TECLA_FIN << "-> Fin" << endl;
210 }
211
212 bool juego_terminado(char tecla, posicion serpiente[], int longitud_serpiente){
213     return(toupper(tecla) == TECLA_FIN ||
214            serpiente_tocada(serpiente,longitud_serpiente) ||
215            serpiente[0].x == LIMITE_IZQUIERDA ||
216            serpiente[0].x == LIMITE_DERECHA ||
217            serpiente[0].y == LIMITE_SUPERIOR ||
218            serpiente[0].y == LIMITE_INFERIOR);
219 }
220
221 void obtener_direccion_serpiente(const char tecla, inc_unitario_posicion& inc_unitario_posicion) {
222     switch (toupper(tecla)) {
223         case ARRIBA:
224             inc_unitario_posicion.x = 0;
225             inc_unitario_posicion.y = MOVIMIENTO_Y_ASCENDENTE;
226             break;
227
228         case ABAJO:
229             inc_unitario_posicion.x = 0;
230             inc_unitario_posicion.y = MOVIMIENTO_Y_DESCENDENTE;
231             break;
232
233         case IZQUIERDA:
234             inc_unitario_posicion.x = MOVIMIENTO_X_IZQUIERDA;
235             inc_unitario_posicion.y = 0;
236             break;
237
238         case DERECHA:
239             inc_unitario_posicion.x = MOVIMIENTO_X_DERECHA;
240             inc_unitario_posicion.y = 0;
241             break;
242   }
243 }
244
245 void mover_serpiente(posicion serpiente[], inc_unitario_posicion inc_unitario_posicion) {
246     posicion cabeza_anterior;
247     cabeza_anterior = serpiente[0];
248
249     serpiente[0].x = serpiente[0].x + inc_unitario_posicion.x;
250     serpiente[0].y = serpiente[0].y + inc_unitario_posicion.y;
251
252     for (int i = LONG_SERPIENTE - 1; i > 0; --i) {
253         serpiente[i] = serpiente[i - 1];
254     }
255     serpiente[1] = cabeza_anterior;
256 }
257
258
259 void pintar_serpiente(const posicion serpiente[],int longitud_serpiente) {
260     poner_cursor(serpiente[0].x, serpiente[0].y);
261     cout << SERPIENTE;
262
```

```
263     for (int i = 1; i < longitud_serpiente - 1; i++) {
264         poner_cursor(serpiente[i].x, serpiente[i].y);
265         cout << CUERPO_SERPIENTE;
266     }
267 }
268 
269 void borrar_serpiente( const posicion serpiente[], int longitud_serpiente) {
270     for (int i = 0; i < longitud_serpiente - 1; i++) {
271         poner_cursor(serpiente[i].x, serpiente[i].y);
272         cout << " ";
273     }
274 }
275 
276 bool serpiente_tocada(const posicion serpiente[],int longitud_serpiente) {
277     for (int i = 1; i < longitud_serpiente - 1; ++i) {
278         if (serpiente[0].x == serpiente[i].x &&
279             serpiente[0].y == serpiente[i].y) {
280             return true;
281         }
282     }
283     return false;
284 }
285 
286 void inicializar_manzana(posicion& manzana) {
287     manzana.x = LIMITE_IZQUIERDA + MARGEN_INI_MANZANA +
288                 rand() % (LIMITE_DERECHA - LIMITE_IZQUIERDA - MARGEN_INI_MANZANA );
289 
290     manzana.y = LIMITE_SUPERIOR + MARGEN_INI_MANZANA +
291                 rand() % (LIMITE_INFERIOR - LIMITE_SUPERIOR - MARGEN_INI_MANZANA);
292 }
293 
294 void pintar_manzana(const posicion& manzana){
295     poner_cursor(manzana.x, manzana.y);
296     cout << MANZANA;
297 }
298 
299 void borrar_manzana(const posicion& manzana){
300     poner_cursor(manzana.x, manzana.y);
301     cout << " ";
302 }
303 
304 bool manzana_tocada(const posicion& manzana, const posicion serpiente[]){
305     return(manzana.x == serpiente[0].x &&
306            manzana.y == serpiente[0].y);
307 }
308 
309 void actualizar_marcador(int& puntos){
310     puntos = puntos + PREMIO;
311     poner_cursor(LIMITE_IZQUIERDA,LIMITE_INFERIOR + MARGEN_MARCADOR);
312     cout << "PUNTOS: " << puntos;
313 }
314 void actualizar_longitud(int& longitud_serpiente){
315 
316  longitud_serpiente = longitud_serpiente + 1;
317 
318 }
```