Universidad Autónoma de Yucatán

Programación Estructurada - MEFI

Licenciatura en Ingeniería de Software

Emilio Gabriel Rejón Herrera

Entrega Final de Proyecto: “Consola de videojuegos”

Cauich Uc Uriel Eliseo

May Rodriguez Mario Angel

Pérez Vázquez Jamart Uriel

Tovar Márquez Luis Enrique

Viernes 25/Junio/2021

1. Antecedentes de la propuesta.

La programación es el proceso por el cual, mediante algoritmos basados en un lenguaje de programación, se realiza una cierta tarea. Este campo de la informática tiene diversas aplicaciones en la vida cotidiana. Para poder desarrollar un software se requiere una serie de pasos para poder obtener el resultado deseado, así como diversas herramientas que ayuden a la ejecución del mismo.

C es un lenguaje de programación de propósito general. Mediante una investigación previa a la elección del proyecto, pudimos encontrar la mejor opción a desarrollar. Nuestra primera idea fue hacer un juego sencillo. En la investigación pudimos encontrar muchos juegos desarrollados en otros proyectos. Así que decidimos enfocarnos en algo más completo y mejor desarrollado.

Nuestro proyecto se diferencia de otros al ser un programa que cuenta con más de un juego, cada juego fue elegido por el gusto y preferencia de cada uno de nosotros, lo cual le proporcionará un sello de peculiaridad basada en nuestro gusto. Además, el programa se visualiza con un menú amigable y con características de guardado de datos e información que, a diferencia de los programas de juegos sencillos antes mencionados, tendrá como resultado un estímulo de interés de uso para el usuario.

2. Descripción del producto software.

El proyecto con nombre: “Consola de videojuegos” (arcade) consiste en la implementación y recopilación de diversos videojuegos clásicos en un solo programa de software, mediante el cual se le brinde al usuario la oportunidad de escoger la opción que le parezca más agradable entre esta selección de juegos a su disposición.

El programa de software incluirá un menú de inicio con cinco opciones principales, el primer apartado será el de: “menú principal” en este apartado estarán los nombres de los juegos disponibles y el usuario podrá escoger entra esas opciones, el juego que desee jugar.

El segundo apartado será el de: “instrucciones”, el cual le permitirá al usuario conocer la forma correcta de interactuar con el programa y los juegos, de esta forma, el usuario podrá tener conocimiento previo del modo de uso de los juegos a su disposición.

La tercera opción dentro del menú principal de inicio será el de: “créditos”, en el cual se incluirán los nombres de los integrantes del equipo que participaron en la elaboración del programa de software, el cuarto apartado corresponderá a: “jugadores y puntajes”, en donde aparecerán los nombres y puntajes de los usuarios que hayan interactuado, concluido y registrado su nombre al finalizar con un juego.

Por último, tendremos la opción “salir” para que el usuario pueda salir del programa de software cuando éste así lo decida.

La recopilación de videojuegos incluidos en el proyecto “Consola de videojuegos” son:

- Ahorcado
- Juego de Gato.
- Snake.
- Buscaminas.

3. Objetivos generales y específicos del sistema.

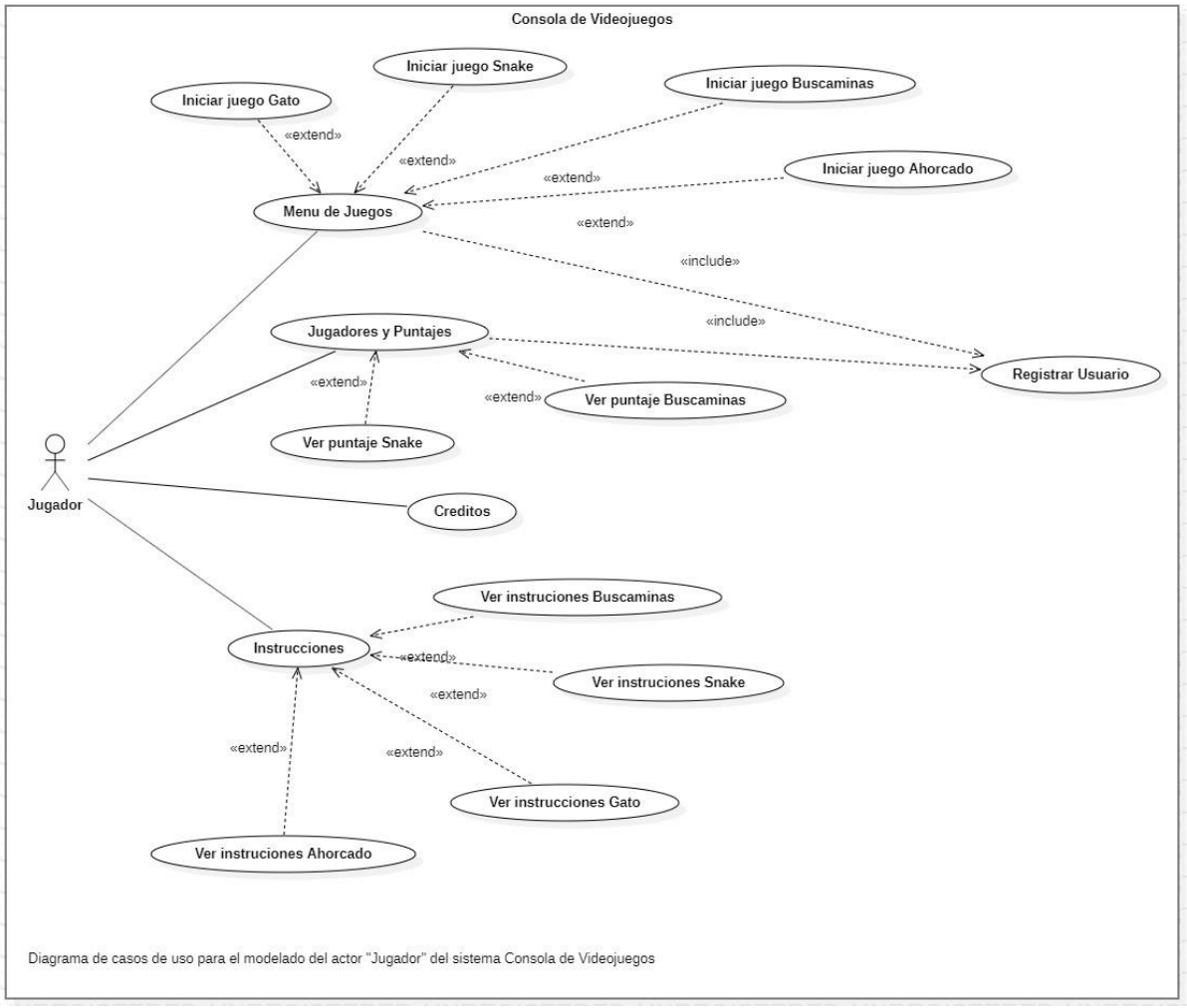
3.1. Objetivos generales

Divertir y entretener al usuario de una forma sana y en un ambiente amigable y de confort. Además, propiciar una experiencia cómoda y lo suficientemente entretenida para lograr que el usuario desee utilizar el programa más de una sola vez.

3.2. Objetivos específicos:

1. Desarrollar un programa de software que implemente de manera amigable con el usuario lo visto en clases.
2. Generar un sistema de puntuaciones que incentive al usuario a seguir haciendo uso de la “consola de videojuegos” de manera constante a lo largo del tiempo.
3. Crear una consola de videojuegos que pueda ser utilizada y disfrutada por personas de manera individual o en grupo, buscando que la competitividad entre ellos estimule lazos de amistad, a través de juegos aptos para todos los rangos de edad, libres de violencia y contenido explícito.

4. Diagrama de Casos de Uso



4.1. Descripción de tipos de usuarios

- **Jugador casual:** Estos usuarios se enfocan solamente en jugar de manera casual o por diversión los diferentes minijuegos que ofrece la aplicación “consola de videojuegos”, su interés está en ocupar su tiempo de ocio pasando un buen rato jugando algún minijuego o superando sus propias puntuaciones.

5. Interfaces principales

- **Interfaz Principal:** Cuando se ejecuta el sistema se muestra el título del juego y se invita al usuario al presionar una tecla para continuar.
- **Menú de Juego:** Al seleccionar la opción se despliega la lista de juegos disponibles en el sistema, que son: Gato, Buscaminas, Snake, Ahorcado y Regresar.

- **Instrucciones:** Para aumentar la facilidad de uso del sistema se integrará una sección con las instrucciones de cada juego presentado en el programa.
- **Créditos:** Muestra en pantalla los datos creadores del software. En éste, gracias a una función, con la pulsación de cualquier tecla se regresa a la interfaz principal.
- **Jugadores y Puntajes:** Con esta opción el usuario podrá visualizar los puntajes más altos registrados en las partidas de cada juego propuesto en el sistema.
- **Salir:** Opción con la que usuario terminará la ejecución del sistema.

6. Mapeo de requerimientos

6.1. Requerimientos Funcionales

Identificador RF-001			
Título	Generar Usuario		
Descripción	El programa ofrecerá la opción de generar un usuario al inicio del juego, para de esta forma identificar al usuario que está usando el sistema en ese momento, y así tener la posibilidad de guardar su puntuación.		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Esencial		
Fuente	Desarrollador		

Identificador RF-002			
Título	Proporcionar instrucciones		
Descripción	El menú de inicio tendrá un apartado de instrucciones que le permitirán al usuario una experiencia más sencilla al momento de utilizar el programa, ya que es posible que no conozcan a la perfección como funciona algún juego.		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Deseable		
Fuente	Desarrollador		

Identificador RF-003			
Título	Créditos		
Descripción	El sistema debe tener una opción para mostrarle al usuario aquellos miembros del equipo de desarrollo que crearon el software.		
Prioridad	Baja	**Estabilidad**	Sí
Claridad	Media	**Verificabilidad**	Sí

Necesidad	Opcional
Fuente	Desarrollador

Identificador RF-004			
Título	Puntajes		
Descripción	El sistema debe almacenar las puntuaciones obtenidas por el usuario para así poder mostrar la clasificación de mejores puntajes en los diferentes juegos.		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Esencial		
Fuente	Desarrollador		

Identificador RF-005			
Título	Selección de juego		
Descripción	El sistema debe proporcionarle al usuario una lista de los diferentes juegos que se ofrecen, para que así el usuario pueda escoger el juego deseado.		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Esencial		
Fuente	Desarrollador		

Identificador RF-006			
Título	Progreso de Partida		
Descripción	El usuario podrá conocer el progreso obtenido al finalizar la partida, así obtiene retroalimentación instantánea acerca de que tan bien lo hizo.		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Deseable		
Fuente	Desarrollador		

Identificador RF-007			
Título	Juegos disponibles		
Descripción	El sistema contará con 4 videojuegos disponibles para ser jugados: Gato, Ahorcado, Snake y Buscaminas.		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Esencial		
Fuente	Desarrollador		

6.2. Requerimientos No Funcionales

Identificador RNF-001			
Título	Sistema operativo		
Descripción	El sistema sólo funcionará en equipos de cómputo que cuenten con el sistema operativo Windows 7 o versiones más recientes.		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Esencial		
Fuente	Desarrollador		

Identificador RNF-002			
Título	Tiempo de respuesta		
Descripción	El sistema no debe tardas más de 8 segundos en cargar cualquier funcionalidad, ya sea la carga de uno de los juegos, los créditos o las mayores puntuaciones.		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Esencial		
Fuente	Desarrollador		

Identificador RNF-003			
Título	Almacenamiento		
Descripción	El sistema no debe ocupar más de 15 MB de espacio de almacenamiento.		
Prioridad	Media	**Estabilidad**	Sí
Claridad	Media	**Verificabilidad**	Sí
Necesidad	Deseable		
Fuente	Desarrollador		

Identificador RNF-004			
Título	Usabilidad		
Descripción	El uso del sistema será de muy fácil aprendizaje, no requerirá más de un par de minutos para comprender el funcionamiento del mismo, pues deben ser simples comandos de teclado.		
Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Esencial		
Fuente	Desarrollador		

Identificador RNF-005	
Título	Fiabilidad
Descripción	El sistema debe funcionar correctamente por lo menos el 95% de las veces que sea utilizado.

Prioridad	Alta	**Estabilidad**	Sí
Claridad	Alta	**Verificabilidad**	Sí
Necesidad	Esencial		
Fuente	Desarrollador		

7. Especiación de Casos de Uso:

1. **Inicio de una partida.** Al usuario se le presenta en pantalla las diferentes opciones del menú y selecciona la opción de “Menú de juegos”, y se le presentan en pantalla cuatro opciones de juego: Gato, Buscaminas, Snake y Ahorcado, de entre los cuales selecciona uno.
    - **1.1.** En caso de que ingrese una opción de juego inválida, se le volverá a pedir que ingrese una opción válida.
    - **1.2.** Al finalizar la partida, se le pedirá al jugador que ingrese un nombre de usuario para registrar su puntuación.
2. **Consulta de puntajes.** Al usuario se le presenta en pantalla las diferentes opciones del menú y selecciona la opción de “Jugadores y Puntajes”.
    - **2.1.** En caso de querer consultar las puntuaciones más altas en el juego Buscaminas, selecciona la opción correspondiente.
    - **2.2.** En caso de querer consultar las puntuaciones más altas en el juego Snake, selecciona la opción correspondiente.
    - **2.3.** De igual forma, en caso de ingresar una opción inválida, se le solicitará que ingrese de nuevo una opción válida.
3. **Consulta de instrucciones.** Al usuario se le presenta en pantalla las diferentes opciones del menú y selecciona la opción de “Instrucciones”, y se le presentan en pantalla cuatro opciones de juego: Gato, Buscaminas, Snake y Ahorcado, de entre los cuales selecciona uno y se le presentan sus respectivas instrucciones para jugar ducho juego.
    - **3.1.** En caso de que ingrese una opción de juego inválida, se le volverá a pedir que ingrese una opción válida.
4. **Consulta de créditos.** Al usuario se le presenta en pantalla las diferentes opciones del menú y selecciona la opción de “Créditos”, con lo cual al usuario se le mostrará en pantalla los créditos de los creadores del sistema.

8. Mapeo de los requerimientos (matriz de requerimientos)

Requerimientos Funcionales						
ID	**Título**	**Descripción**	**Prioridad**	**Objetivo 1**	**Objetivo 2**	**Objetivo 3**
RF-001	Generar usuario	El programa ofrecerá la opción de generar un usuario al inicio del juego, para de esta forma identificar al usuario que está usando el sistema en ese momento, y así tener la posibilidad de guardar su puntuación.	Alta		X	X
RF-002	Proporcionar instrucciones	El menú de inicio tendrá un apartado de instrucciones que le permitirán al usuario una experiencia más sencilla al momento de utilizar el programa, ya que es posible que no conozcan a la perfección como funciona algún juego.	Media			X
RF-003	Créditos	El sistema debe tener una opción para mostrarle al usuario aquellos miembros del equipo de desarrollo que crearon el software.	Baja			X
RF-004	Puntajes	El sistema debe almacenar las puntuaciones obtenidas por el usuario para así poder mostrar la clasificación de mejores puntajes en los diferentes juegos.	Alta		X	
RF-005	Selección de juego	El sistema debe proporcionarle al	Alta			X

		usuario una lista de los diferentes juegos que se ofrecen, para que así el usuario pueda escoger el juego deseado.				
RF-006	Progreso de partida	El usuario podrá conocer el progreso obtenido al finalizar la partida, así obtiene retroalimentación instantánea acerca de que tan bien lo hizo.	Media		X	
RF-007	Juegos disponibles	El sistema contará con 4 videojuegos disponibles para ser jugados: Gato, Ahorcado, Snake y Buscaminas.	Alta			X

Requerimientos No Funcionales						
ID	**Título**	**Descripción**	**Prioridad**	**Objetivo 1**	**Objetivo 2**	**Objetivo 3**
RNF-001	Sistema operativo	El sistema sólo funcionará en equipos de cómputo que cuenten con el sistema operativo Windows 7 o versiones más recientes.	Alta	X		
RNF-002	Tiempo de respuesta	El sistema no debe tardas más de 8 segundos en cargar cualquier funcionalidad, ya sea la carga de uno de los juegos, los créditos o las mayores puntuaciones.	Alta	X		
RNF-003	Almacena miento	El sistema no debe ocupar más de 15 MB de espacio de almacenamiento.	Media	X		
RNF-004	Fiabilidad	El uso del sistema será de muy fácil aprendizaje, no	Alta	X		

		requerirá más de un par de minutos para comprender el funcionamiento del mismo, pues deben ser simples comandos de teclado.				
RNF-005	Usabilidad	El sistema debe funcionar correctamente por lo menos el 95% de las veces que sea utilizado.	Alta	X		

8.1. Informe del avance general de la implementación de dichos requerimientos

- **RF-001. Generar Usuario:** Se implementó una opción similar a este requerimiento, al terminar la partida del juego, el sistema solicita el nombre o "Nickname" del usuario para guardar este dato al registro del juego correspondiente, Sólo se implementó para verificar que el guardado de registros funciona.
- **RF-002. Proporcionar instrucciones:** Este requerimiento ya se ha implementado en el menú del sistema, se creó una opción llamada "Instrucciones" que permite la visualización de cómo se juegan los videojuegos ofrecidos en el sistema.
- **RF-003. Créditos:** Este requerimiento ha sido implementado con éxito sin ningún inconveniente. Se encuentra disponible en el apartado de "Créditos" en el menú del sistema.
- **RF-004. Puntajes:** La implementación de este requerimiento ha sido parcial, ya que sólo funciona con un juego, pero con esta base, podremos implementar esta característica al resto del sistema.
- **RF-005. Selección de juego:** Este requerimiento se ha cumplido correctamente, en el menú principal del sistema se encuentra el apartado "Menú juegos" que permite visualizar los videojuegos disponibles en el sistema.
- **RF-006. Progreso de partida:** Por el momento, aún no se ha implementado este requerimiento en el sistema, pero se espera que sea cubierto para la entrega final.
- **RF-007. Juegos disponibles:** Este requerimiento ha sido cumplido parcialmente, ya que sólo se cuenta con un juego desarrollado por el momento. Se espera que para la entrega final ya se haya cubierto este punto por completo.

- **RNF-001. Sistema operativo:** El requerimiento se ha cumplido con éxito, el sistema funciona con el sistema operativo base especificado al igual que con versiones más recientes.
- **RNF-002. Tiempo de respuesta:** Este requerimiento se ha cumplido, la ejecución del sistema de software construido no tarda más de dos segundos incluso en una computadora con el 82% de memoria RAM utilizada.
- **RNF-003. Almacenamiento:** El requerimiento ha sido cubierto, por el momento, el tamaño del sistema es de 17 KB, y se espera que no supere de 40 KB de peso.
- **RNF-004. Usabilidad:** El requerimiento actual ha sido abarcado en el sistema hasta el momento desarrollado, se ofrece al usuario una interfaz sencilla e intuitiva para usar sin mayores problemas.
- **RNF-005. Fiabilidad:** Este requerimiento ha sido cubierto parcialmente, ya que, al estar en una etapa de desarrollo, es común descubrir errores tanto en los gráficos de la consola como en la lógica de programación, se espera que para la demostración final del sistema se logré cubrir este requerimiento en su totalidad.

9. Definición del estándar de codificación.

Para la elaboración de este sistema de software, se seguirán los estándares de codificación elaborados por *Robert C. Martin* en el libro *Clean Code: A Handbook of Agile Software Craftsmanship.*

Nombramiento de variables	• Las variables deben ser autodescriptivas por medio de su nombre. El nombre puede representar el ¿qué? o ¿cómo?, dependiendo de su uso. Ejemplo: numAsientosEstadio, inputRegistro, velocidad, checkTotal. • Se establecerá una longitud promedio del nombramiento entre 9 y 15 caracteres. • No habrá variables globales. • Las variables como *i, j, x, y*, tendrán un alcance corto, limitado a los ciclos. • Las variables de instancia deberán ser muy descriptivas. • Utilizar consistencia cuando se tengan variables con valores opuestos. Ejemplo: inicioConteo, finConteo.

	• Las variables booleanas deberán implicar valores booleanos. Ejemplo: done, success, error, found. • Las variables de tipo bandera deben identificarse por su uso (evitar nombrarse como *flag* o *bandera*).
Nombramiento de macros	• Deben ser nombradas de forma corta y clara. • Su tamaño máximo es de 20 caracteres. • Si tienen demasiada funcionalidad, deben transformarse en funciones. • No relacionar una macro con otras macros, sólo con constantes. • Si se tienen múltiples argumentos, separar con paréntesis.
Nombramiento de bibliotecas	No aplica.
Nombramiento de funciones	• Describir en el nombre todo lo que la función hace. • Evitar nombramiento con verbos sin sentido. • Nombrar las funciones tan largas como sea necesario. • Elegir el nombre de la función con base en la descripción del valor que retorna. • Utilizar consistencia con los opuestos.
Nombramiento de archivos	• Utilizar nombres descriptivos. • Nombres que den referencia a lo que el archivo contiene. • La longitud del nombre será tan larga como sea necesario.
Nombramiento de comentarios	• Escribir un comentario para explicar un intento o solución. • Para fines informativos. • Advertir sobre la modificación de una instancia o variable. • Aclarar una sección del código que pueda ser complicada de entender. • Para ampliar la información del código utilizado. • Evitar comentarios redundantes. • Evitar comentarios históricos (como escribir fechas de modificación). • No comentar código que no se utiliza en el sistema.
Formato del código	• Usar espacios en blanco para separar elementos de una sentencia no muy relacionados o mostrar precedencia de operadores.

	• Utilizar sangría para mostrar la jerarquía que tiene el código, clases con funciones, funciones con bloques, bloques con sentencias. Igual el alcance de las variables, constantes. Aplica también en sentencias cortas. • Separación vertical entre conceptos utilizando un espacio en blanco. • Para las funciones dependientes o relacionadas, colocarlas cerca una de la otra.

9.1. Cumplimiento del estándar de codificación

9.1.1. Bibliotecas a utilizar en el programa

Para poder hacer uso de las funciones del sistema, se requirió de bibliotecas del lenguaje de programación que permitieron la implementación de los comandos utilizados. Las bibliotecas usadas hasta el momento son las siguientes:

- **<stdio.h>:** Biblioteca estándar del lenguaje de programación C para hacer operaciones, estándar, de entrada y salida, así como la definición de tipos necesarias para dichas operaciones.
- **<time.h>:** Relacionado con formato de hora y fecha es un archivo de cabecera de la biblioteca estándar del lenguaje de programación C que contiene funciones para manipular y formatear la fecha y hora del sistema.
- **<iostream>:** Es la biblioteca estándar en C++ para poder tener acceso a los dispositivos estándar de entrada y/o salida.
- **<conio.h>:** Es un archivo de cabecera escrito en C usado mayormente por los compiladores de MS-DOS para proveer un sistema de entrada y salida por consola. Conio.h no pertenece a la biblioteca estándar de C.
- **<string.h>:** Biblioteca que contiene un conjunto de funciones para manipular cadenas: copiar, cambiar caracteres, comparar cadenas, etc.
- **<windows.h>**: Es un archivo de cabecera específico de Windows para la programación en lenguaje C/C++ que contiene las declaraciones de todas las funciones de la biblioteca Windows API, todas las macros utilizadas por los programadores de aplicaciones para Windows, y todas las estructuras de datos utilizadas en gran cantidad de funciones y subsistemas.

9.1.2. Nombramiento de variables

Se presenta una muestra de la aplicación del estándar de codificación determinado:

```
int buscaminas[fila][columna];
int nivel, bombas, repetir;
int decision;
```

9.1.3. Nombramiento de macros

Se presenta una muestra de la aplicación del estándar de codificación determinado:

```
#define fila 15
#define columna 15
```

9.1.4. Nombramiento de funciones

Se presenta una muestra de la aplicación del estándar de codificación determinado:

```
void generarTablero(int matriz[][15],int status);
int generarBombas(int x);
void asignarPuntaje(int puntosJugador);
```

9.1.5. Nombramiento de archivos

Se presenta una muestra de la aplicación del estándar de codificación determinado:

```
f = fopen("RegistroBuscaminas.txt","a+");
```

9.1.6. Nombramiento de comentarios

Se presenta una muestra de la aplicación del estándar de codificación determinado:

```
//Funcion principal
int main(int argc, char *argv[]) {
    //Establecemos el nombre a la ventana de la consola del sistema
    system("title Proyecto Final");{
        system("color 0a");
        //Llamamos a la funcion que muestra el titulo del sistema
        tituloPrincipal();
    }
    //Llamamos a la funcion que muestra el menu principal del sistema
```

```
    menuPrincipal();
    return 0;
}
```

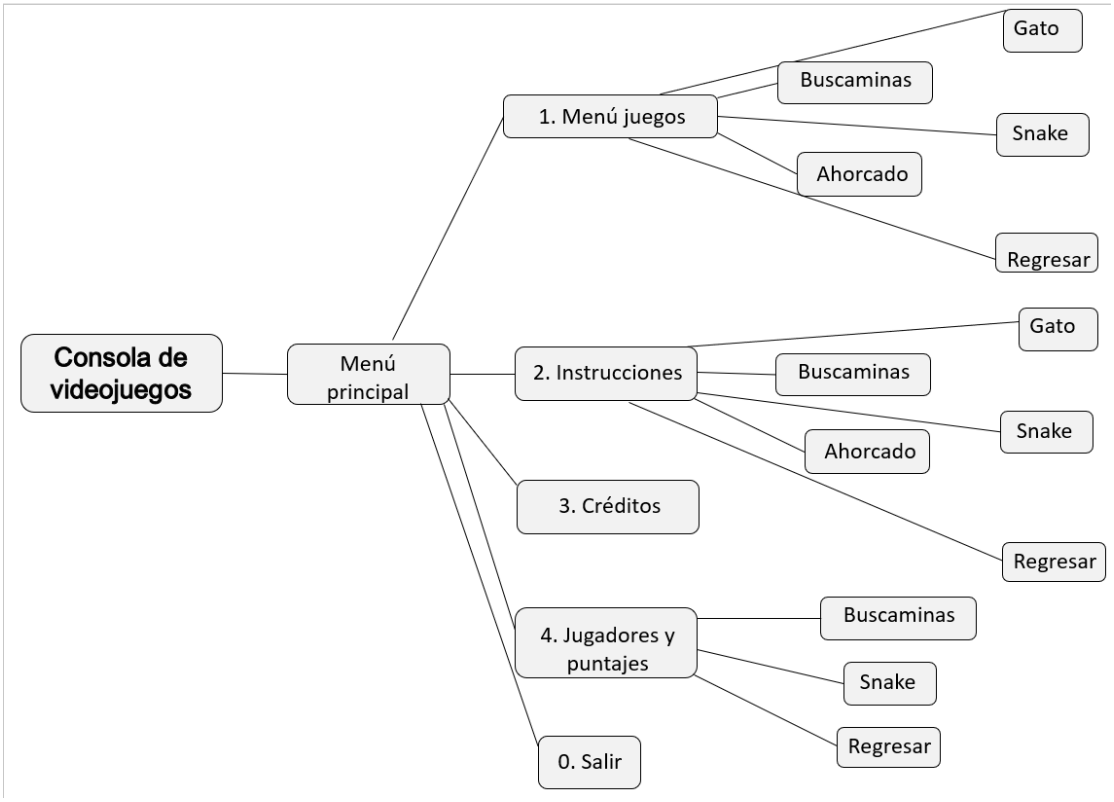
9.1.7. Formato del código

Se presenta una muestra de la aplicación del estándar de codificación determinado:

```
    //Funcion para llenar el tablero de Buscaminas
    void llenarMatriz(int matriz[][15]){
        for(int i = 0; i < fila; i++){
            for(int j = 0; j < columna; j++){
                matriz[i][j] = 2;
            }
        }
    }
```

10. Modularidad

- Organización del sistema en general.



- Organización del código con base en entradas, procesamiento y salidas.

Entradas

1. Instrucciones

```
80  //Funcion que despliega las instrucciones de los juegos
81  void instrucciones(){
82      //Declaracion de Variables
83      int opcion;
84      system("color 0b");
85      system("cls");
86      bordesSistema();
87      gotoxy(5,1);printf("\t\t   *************** COMO JUGAR ***************  \n");
88      gotoxy(26,4);printf("1. GATO (Tic Tac Toe)");
89      gotoxy(26,6);printf("2. BUSCAMINAS");
90      gotoxy(26,8);printf("3. SNAKE");
91      gotoxy(26,10);printf("4. AHORCADO (Hangman)");
92      gotoxy(26,12);printf("0. Regresar al Menu Principal");
93      gotoxy(21,16);printf("Opcion: ");
94      gotoxy(30,16);scanf("%d",&opcion);
```

2. Menú principal

```
155
156 //Funcion que despliega el menu principal (interfaz principal)
157 void menuPrincipal(){
158     //Declaracion de variables
159     int opcion;
160
161     system("color 0a");
162     system("cls");
163     bordesSistema();
164     gotoxy(7,1);printf("\n\t\t   *************** ARCADE ***************  \n");
165     gotoxy(31,4);printf("1. Menu de Juegos");
166     gotoxy(31,6);printf("2. Instrucciones");
167     gotoxy(31,8);printf("3. Creditos");
168     gotoxy(31,10);printf("4. Jugadores y Puntajes");
169     gotoxy(31,12);printf("0. Salir");
170     gotoxy(26,16);printf("Opcion: ");
171     gotoxy(35,16);scanf("%d",&opcion);
172
```

3. Menú juegos

```
225 void menuJuegos(){
226     //Declaracion de variables
227     int opcion;
228
229     system("cls");
230     system("color 0f");
231     bordesSistema();
232     gotoxy(9,1);printf("\n\t\t  *************** JUEGOS ***************  \n");
233     gotoxy(31,4);printf("1. GATO (Tic Tac Toe)");
234     gotoxy(31,6);printf("2. BUSCAMINAS ");
235     gotoxy(31,8);printf("3. SNAKE");
236     gotoxy(31,10);printf("4. AHORCADO (Hangman)");
237     gotoxy(31,12);printf("0. Regresar");
238     gotoxy(26,16);printf("Opcion: ");
239     gotoxy(35,16);scanf("%d",&opcion);
240
```

4. Buscaminas

```
397  //Funcion que permite generar el juego de Buscaminas
398  void juegoBuscaminas(){
399      //Declaracion de variables
400      int buscaminas[fila][columna];
401      int nivel, bombas, repetir;
402      int decision;
403
404      do{
405          system("cls");
406          int i, j, contador = 0;
407          int estatus = -1;
408          //Se imprime y llena el tablero del buscaminas
409          llenarMatriz(buscaminas);
410          tituloBuscaminas();
411          //Se captura el nivel
412          do{
413              scanf("%d",&nivel);
414              fflush(stdin);
415              if(nivel == 0){
416                  return menuPrincipal();
417              }
418              if((nivel < 0) || (nivel > 4)){
419                  printf ("\n <<Caracter No Valido>>\n   Opcion: ");
420              }
421          }while((nivel < 0) || (nivel > 4));
```

5. Tiro

```
418              if((nivel < 0) || (nivel > 4)){
419                  printf ("\n <<Caracter No Valido>>\n   Opcion: ");
420              }
421          }while((nivel < 0) || (nivel > 4));
422
423          bombas = generarBombas(nivel);
424          srand(time(NULL));
425          colocarBombas(buscaminas, bombas);
426          system("cls");
427          printf("\n\t\t\t>>Intentalo mejor<<\t\t\t\n\n");
428          aumentarPuntosAciertos(contador);
429          generarTablero(buscaminas, estatus);
430
431          do{
432              puts("Ingrese su tiro: (x,y)");
433              do{
434                  printf("X=");
435                  scanf("%d", &i);
436                  if((i < 0 || i > 14)){
437                      printf("<<Tiro No Valido>> \n Ingrese su Tiro Nuevamente\n");
438                  }
439                  fflush(stdin);
440              }while(!(i >= 0 &&  i <= 14));
441              do{
442                  printf("Y=");
443                  scanf("%d", &j);
444                  if((j<0 || j>14)){
445                      printf("<<Tiro No Valido>> \n Ingrese su Tiro Nuevamente\n");
446                  }
```

6. ¿Jugar de nuevo?

```
473         }while(estatus != 1 && estatus != 0);
474         printf("¿Desea Jugar de Nuevo?: 1 - Si    0 - No: ");
475         fflush(stdin);//Limpia el Buffer de los Datos Almacenados
476 
477         do{
478             scanf("\t%d", &decision);
479             fflush(stdin);
480             if(decision == 1){
481                 repetir = 1;
482                 break;
483             }
484             if(decision == 0){
485                 repetir = 0;
486                 solicitarNombreBuscaminas();
487                 asignarPuntaje(contador * 25);
488                 guardarRegistroBuscaminas(Jugador);
489                 menuJuegos();
490             }
491             if((decision != 1) || (decision != 0)){
492                 puts("\n<<Caracter No Valido>>\n");
493             }
494         }while((decision != 1) || (decision != 0));
495         system("cls");
496     }while(repetir);
497 }
```

7. Puntajes

```
499 //Funcion que despliega los puntajes de los jugadores registrados
500 void puntajesJugadores(){
501     //Declaracion de variables
502     int opcion;
503 
504     system("color 0c");
505     system("cls");
506     bordesSistema();
507     gotoxy(1,2);printf("\t   *************** JUGADORES RECIENTES ***************  \n");
508     gotoxy(26,4);printf("1. BUSCAMINAS");
509     gotoxy(26,6);printf("2. SNAKE");
510     gotoxy(26,8);printf("0. Regresar al Menu Principal");
511     gotoxy(21,11);printf("Opcion: ");
512     gotoxy(30,11);scanf("%d",&opcion);
```

8. Nombre buscaminas

```
545 
546 //Funcion que solicita el nombre al usuario para el juego de Buscaminas
547 void solicitarNombreBuscaminas(){
548     fflush(stdin);
549     gotoxy(46,10);printf("Inserte Su Nickname:");
550     gotoxy(66,10);gets(Jugador.nombre);
551 }
```

Proceso

1. Seleccionar el juego

```c
//Funcion que despliega el menu de juegos
void menuJuegos(){
    //Declaracion de variables
    int opcion;

    system("cls");
    system("color 0f");
    bordesSistema();
    gotoxy(9,1);printf("\n\t\t  *************** JUEGOS ***************  \n");
    gotoxy(31,4);printf("1. GATO (Tic Tac Toe)");
    gotoxy(31,6);printf("2. BUSCAMINAS ");
    gotoxy(31,8);printf("3. SNAKE");
    gotoxy(31,10);printf("4. AHORCADO (Hangman)");
    gotoxy(31,12);printf("0. Regresar");
    gotoxy(26,16);printf("Opcion: ");
    gotoxy(35,16);scanf("%d",&opcion);

    do{
        fflush(stdin);
        system("cls");
        switch(opcion){
        case 1:
            //Juego de Gato
            system("cls");
            break;
        case 2:
            //Buscaminas
            system("cls");
            juegoBuscaminas();
            break;
            system("cls");
            juegoBuscaminas();
            break;
        case 3:
            //Juego de Snake
            system("cls");
            break;
        case 4:
            //Juego de ahorcado
            system("cls");
            break;
        case 0:
            //Regresa al menu principal
            system("cls");
            menuPrincipal();
            break;
        }
    }while((opcion >= 5) || (opcion < 0));
}
```

2. Colocar bombas

```
329
330 //Funcion para colocar las bombas previamente generadas en el tablero
331 void colocarBombas(int matriz[][15], int bomb){
332     int i, dominio, codominio;
333     for(i = 1; i <= bomb; i++){
334         dominio = 0 + rand() % 14;
335         codominio = 0 + rand() % 14;
336         matriz[dominio][codominio] = 3;
337     }
338 }
```

3. Puntos y tiempo

```
372 //Funcion para aumentar los puntos en caso de acertar
373 void aumentarPuntosAciertos(int cont){
374     int auxiliar;
375     auxiliar = cont * 25;
376     printf("\n\t\t\t\tPuntaje: %d \n\n", auxiliar);
377 }
378
379 //Funcion Para Llevar el Tiempo
380 void cronometroBuscaminas(int status){
381     int horas = 0, min = 0, sec = 0, x;
382     while(status != 1 && status != 0){
383         x = 1000;
384         sec++;
385         if(sec == 60){
386             sec = 0;
387             min++;
388         }if(min == 60){
389             min = 0;
390             horas++;
391         }
392         printf("Tiempo: %.2d:%.2d:%.2d", horas, min, sec);
393         Sleep(x);
394     }
395 }
396
```

4. Genera el buscaminas

```
397 //Funcion que permite generar el juego de Buscaminas
398 void juegoBuscaminas(){
399     //Declaracion de variables
400     int buscaminas[fila][columna];
401     int nivel, bombas, repetir;
402     int decision;
403 
404     do{
405         system("cls");
406         int i, j, contador = 0;
407         int estatus = -1;
408         //Se imprime y llena el tablero del buscaminas
409         llenarMatriz(buscaminas);
410         tituloBuscaminas();
411         //Se captura el nivel
412         do{
413             scanf("%d",&nivel);
414             fflush(stdin);
415             if(nivel == 0){
416                 return menuPrincipal();
417             }
418             if((nivel < 0) || (nivel > 4)){
419                 printf ("\n <<Caracter No Valido>>\n   Opcion: ");
420             }
421         }while((nivel < 0) || (nivel > 4));
422 
```

```
422 
423         bombas = generarBombas(nivel);
424         srand(time(NULL));
425         colocarBombas(buscaminas, bombas);
426         system("cls");
427         printf("\n\t\t\t>>Intentalo mejor<<\t\t\t\n\n");
428         aumentarPuntosAciertos(contador);
429         generarTablero(buscaminas, estatus);
430 
```

5. Puntaje

```
553 //Funcion que asigna los puntos al jugador
554 void asignarPuntaje(int puntosJugador){
555     fflush(stdin);
556     Jugador.puntos = puntosJugador;
557 }
558
559 //Funcion que guarda el registo de Buscaminas en un archivo txt
560 void guardarRegistroBuscaminas(player Jugador){
561     FILE *f;
562     f = fopen("RegistroBuscaminas.txt","a+");
563     if(f == NULL){
564         printf("No se pudo abrir el archivo.\n");
565     }else{
566         gotoxy(3,5);fprintf(f, "\n\t\t\t\t%s.....", Jugador.nombre);
567         gotoxy(17,5);fprintf (f,"%d", Jugador.puntos);
568     }
569     fclose(f);
570 }
571
572 //Funcion que despliega el registro de puntaje de Buscaminas en un archivo tx
573 void consultarRegistroBuscaminas(){
574     FILE *f;
575     char contenido[21];
576     f = fopen("RegistroBuscaminas.txt", "a+");
577     if(f == NULL)
578         printf("No se pudo abrir el archivo.\n");
579     else{
580         do{
```

```
577     if(f == NULL)
578         printf("No se pudo abrir el archivo.\n");
579     else{
580         do{
581             fgets(contenido,21, f);
582             puts(contenido);
583         }
584         while(!feof(f));
585     }
586 }
```

Salidas

1. Impresión de las instrucciones de los juegos

```
80
81  //Funcion que despliega las instrucciones de los juegos
82  void instrucciones(){
83      //Declaracion de Variables
84      int opcion;
85      system("color 0b");
86      system("cls");
87      bordesSistema();
88      gotoxy(5,1);printf("\t\t   *************** COMO JUGAR ***************  \n");
89      gotoxy(26,4);printf("1. GATO (Tic Tac Toe)");
90      gotoxy(26,6);printf("2. BUSCAMINAS");
91      gotoxy(26,8);printf("3. SNAKE");
92      gotoxy(26,10);printf("4. AHORCADO (Hangman)");
93      gotoxy(26,12);printf("0. Regresar al Menu Principal");
94      gotoxy(21,16);printf("Opcion: ");
95      gotoxy(30,16);scanf("%d",&opcion);
96
97      do{
98          fflush(stdin);
99          system("cls");
100         switch(opcion){
101         case 1:
102             system("cls");
103             bordesSistema();
104             gotoxy(7,1);
105             printf("\t\t   *************** GATO ***************  \n");
106             printf("\n\t\t\t   Juego Para 2 Personas.\n\n\t El jugador 1 tendria la ficha
107             printf("\n\n\n\tPresione Cualquier Tecla Para Regresar");
108             getch();
109             menuPrincipal();
```

```
108             getch();
109             menuPrincipal();
110             break;
111         case 2:
112             system("cls");
113             bordesSistema();
114             gotoxy(7,1);
115             printf("\n\t\t   *************** BUSCAMINAS ***************\n");
116             printf("\n\t\t\t       Juego de 1 Persona.\n\n\tEl Jugador Decidira que Dificultad Jugar y Este\
117             printf("\n\n\n\tPresione Cualquier Tecla Para Regresar");
118             getch();
119             menuPrincipal();
120             break;
121         case 3:
122             system("cls");
123             bordesSistema();
124             gotoxy(7,1);printf("\n\t\t   *************** SNAKE ***************\n");
125             gotoxy(29,4);printf("Juego de 1 Persona.");
126             gotoxy(3,6);printf("El Juego Comienza Automaticamente Despues de Presionar Cualquier Tecla");
127             gotoxy(3,7);printf("Se Controla el Movimiento Con las Flechas del Teclado el Objetivo");
128             gotoxy(3,8);printf("Es Comer los Frutos, Hasta Obtener el Tamaï¿½o Maximo. Se Pierde");
129             gotoxy(3,9);printf("Tocando los Bordes o Si la Serpiente se Toca Asi Misma.");
130             gotoxy(3,10);printf("El Tiempo y el Puntaje se Muestra en la Parte Superior.");
131             gotoxy(3,15);printf("Presione Cualquier Tecla Para Regresar");
132             getch();
133             menuPrincipal();
134             break;
135         case 4:
136             system("cls");
```

```
            menuPrincipal();
            break;
        case 4:
            system("cls");
            bordesSistema();
            gotoxy(5,1);printf("\n\t\t   *************** AHORCADO ***************\n");
            gotoxy(27,4);printf("Juego de 2 o Mas Personas.");
            gotoxy(3,6);printf("El Juego Comienza Automaticamente Despues de Presionar Cualquier Tecla");
            gotoxy(3,7);printf("Cualquier Jugador Debera Ingresar la Palabra a Adivinar por el Jugador ");
            gotoxy(3,8);printf("Contrario. El Jugador Contario Debera Adivinar la Palabra Ingresando");
            gotoxy(3,9);printf("Letra por Letra. El Jugador Contara con 10 Vidas");
            gotoxy(3,15);printf("Presione Cualquier Tecla Para Regresar");
            getch();
            menuPrincipal();
            break;
        case 0:
            system("cls");
            menuPrincipal();
            break;
        }
    }while((opcion >= 5) || (opcion < 0));
}
```

2. Salida de los créditos

```
//Funcion que despliega los creditos de los autores del sistema
void creditos(){
    gotoxy(3,3);printf("\t\t\t    Proyecto Desarrollado Por:");
    gotoxy(3,4);printf("\t\tUriel Cauich, Mario May, Jamart Perez y Luis Tovar\n");
    gotoxy(6,6);printf("\t\t\t Ingenieria de Software - MEFI");
    gotoxy(5,9);printf("\t\t\t   Programacion Estructurada");
    gotoxy(5,12);printf("\t\t        Universidad Autonoma de Yucatan");
    gotoxy(4,15);printf("\t\t\t  Facultad de Matematicas");
    gotoxy(5,20); printf("Presione Cualquier Tecla Para Regresar");
    getch();
    menuPrincipal();
}
```

3. Puntajes

```
    do{
        fflush(stdin);
        system("cls");
        switch(opcion){
        case 1:
            bordesSistema();
            gotoxy(7,1);printf("\n\t\t   *************** BUSCAMINAS ***************\n");
            gotoxy(20,3);printf("\t     Jugador");
            gotoxy(39,3);printf("\t   Score");
            consultarRegistroBuscaminas();
            gotoxy(20,20);printf("Presione Cualquier Tecla Para Regresar");
            getch();
            menuPrincipal();
            break;
        case 2:
            system("cls");
            bordesSistema();
            gotoxy(7,1);printf("\n\t\t   *************** SNAKE ***************\n");
            gotoxy(20,3);printf("\t    Jugador");
            gotoxy(39,3);printf("\t Score");
            gotoxy(20,20);printf("Presione Cualquier Tecla Para Regresar");
            getch();
            menuPrincipal();
            break;
        case 0:
            system("cls");
            menuPrincipal();
            break;
        }
    }while(opcion>=3 || opcion<0);
```

Cohesión de las funciones

Invocación de las funciones

1. Función principal

```c
//Funcion principal
int main(int argc, char *argv[]) {
    //Establecemos el nombre a la ventana de la consola del sistema
    system("title Proyecto Final");{
        system("color 0a");
        //Llamamos a la funcion que muestra el titulo del sistema
        tituloPrincipal();
    }
    //Llamamos a la funcion que muestra el menu principal del sistema
    menuPrincipal();
    return 0;
}
```

2. Menú principal

```c
    do{
        fflush(stdin);
        system("cls");
        switch(opcion){
        case 1:
            //Ingresar al menu de juegos
            system("cls");
            menuJuegos();
            break;
        case 2:
            //Instrucciones de los juegos
            system("cls");
            instrucciones();
            break;
        case 3:
            //Creditos
            system("cls");
            system("color 0e");
            bordesSistema();
            creditos();
            break;
        case 4:
            //Puntajes
            system("cls");
            system("color 0a");
            puntajesJugadores();
```

```
194         case 4:
195             //Puntajes
196             system("cls");
197             system("color 0a");
198             puntajesJugadores();
199             break;
200         case 0:
201             //Finaliza la ejecucion del programa
202             system("cls");
203             printf("\n\n\n\n\t\t\t       ¡GRACIAS POR UTILIZAR!      ");
204             getch();
205             exit(1);
206             break;
207         }
208     }while((opcion >= 5) || (opcion < 0));
209 }
```

3. Menú de juegos

```
240 
241     do{
242         fflush(stdin);
243         system("cls");
244         switch(opcion){
245         case 1:
246             //Juego de Gato
247             system("cls");
248             break;
249         case 2:
250             //Buscaminas
251             system("cls");
252             juegoBuscaminas();
253             break;
254         case 3:
255             //Juego de Snake
256             system("cls");
257             break;
258         case 4:
259             //Juego de ahorcado
260             system("cls");
261             break;
262         case 0:
263             //Regresa al menu principal
264             system("cls");
265             menuPrincipal();
266             break;
267         }
268     }while((opcion >= 5) || (opcion < 0));
269 }
```

4. Buscaminas

```
476
477         do{
478             scanf("\t%d", &decision);
479             fflush(stdin);
480             if(decision == 1){
481                 repetir = 1;
482                 break;
483             }
484             if(decision == 0){
485                 repetir = 0;
486                 solicitarNombreBuscaminas();
487                 asignarPuntaje(contador * 25);
488                 guardarRegistroBuscaminas(Jugador);
489                 menuJuegos();
490             }
491             if((decision != 1) || (decision != 0)){
492                 puts("\n<<Caracter No Valido>>\n");
493             }
494         }while((decision != 1) || (decision != 0));
495         system("cls");
496     }while(repetir);
497 }
```

5. Puntajes

```
514     do{
515         fflush(stdin);
516         system("cls");
517         switch(opcion){
518         case 1:
519             bordesSistema();
520             gotoxy(7,1);printf("\n\t\t   *************** BUSCAMINAS ***************\n")
521             gotoxy(20,3);printf("\t    Jugador");
522             gotoxy(39,3);printf("\t   Score");
523             consultarRegistroBuscaminas();
524             gotoxy(20,20);printf("Presione Cualquier Tecla Para Regresar");
525             getch();
526             menuPrincipal();
527             break;
528         case 2:
529             system("cls");
530             bordesSistema();
531             gotoxy(7,1);printf("\n\t\t   *************** SNAKE ***************\n");
532             gotoxy(20,3);printf("\t    Jugador");
533             gotoxy(39,3);printf("\t Score");
534             gotoxy(20,20);printf("Presione Cualquier Tecla Para Regresar");
535             getch();
536             menuPrincipal();
537             break;
538         case 0:
539             system("cls");
540             menuPrincipal();
541             break;
542         }
543     }while(opcion>=3 || opcion<0);
544 }
```

11. Proceso de desarrollo

Como herramienta de codificación se utilizará el IDE ZinjaI.

La forma de organización partirá del mismo sistema de trabajo que hemos llevado como equipo desde el principio del semestre, por medio de mensajería en nuestro grupo de WhatsApp, y por video llamadas en nuestra sala de equipo de Microsoft Teams.

Nuestro esquema de monitoreo incluirá:

- Búsqueda constante de errores y problemas en la codificación planeada.
- Evaluación de avances.
- Análisis de los riesgos y factores externos en el proceso.
- Sesiones de guía con el profesor.
- Entrega de avances.
- Lista de cotejo por parte del profesor.

Nuestra bitácora contendrá un seguimiento por fecha de cada actividad en el proceso total del proyecto, así como las observaciones internas y las sugerencias y feedback del profesor.

La medición de trabajo se llevará a cabo con GitHub.

11.1. Trabajo de codificación acorde con la propuesta de herramientas, forma de organización, esquemas de monitoreo y bitácoras.

La herramienta de codificación que utilizamos desde un principio fue ZinjaI y posteriormente nos apoyamos de Visual Studio, específicamente de la herramienta Live Share.

La organización ha sido a través de mensajería en nuestro grupo de WhatsApp, de video llamadas en nuestra sala de equipo de Microsoft Teams y de un canal de chat de voz del programa Discord.

El trabajo resultante del esquema de monitoreo es el siguiente:

- Se corrigieron errores y se mejoraron procesos del código.
- Se encontró un factor externo no previsto en el uso de Git, la parte de la colaboración en el código se resolvió con el uso de la herramienta Live Share de Visual Studio. Para la parte de medición de trabajo se está resolviendo el problema con Git.
- Se modificaron los objetivos específicos de nuestro proyecto.

Bitácora actual:

Fecha	Actividad	Observaciones internas	Observaciones del profesor
04-Jun-21	Discusión como equipo para decidir el proyecto a realizar.	Presentaremos la propuesta de una consola de videojuegos.	N/A
07-Jun-21	Presentación de idea al profesor.	En el grupo privado de equipo en Teams, el profesor solicitó la idea de proyecto final.	Agregar un historial de puntuaciones de los juegos.
08-Jun-21	Investigación de antecedentes y preparación de la propuesta.	Hay varios juegos codificados en c, lo ideal sería codificar cada quién nuestro juego preferido y unirlos en con una interfaz amigable.	N/A
11-Jun-21	Presentación de la propuesta frente al grupo.	Todo bien.	Propuesta aceptada. Hay un error en nuestros objetivos específicos.
12-Jun-21	Inicio de codificación.	Avanzar con las codificaciones de los juegos de manera individual, usar la plantilla del Código de avance para agregar los juegos a GitHub	N/A
18-Jun-21	Presentación y entrega de avances.	Corregir el formato del reporte	Especificar secciones del mapeo de requerimientos
22-Jun-21	Adiciones de los juegos restantes	Verificar que el programa ejecuta correctamente y que todas las opciones funcionan.	N/A
24-Jun-21	Correcciones de formato y errores de código	Verificar que se siguen los estándares de codificación establecidos.	N/A
24-Jun-21	Elaboración del cartel científico	Agregar el link del repositorio de GitHub	N/A
25-Jun-21	Presentación del cartel científico y del código final.	Afinar detalles estéticos del código antes de realizar la entrega en Moodle	Sin observaciones

Evaluación basada en objetivos (medibles) de la participación individual y de grupo.

Integrante	Participaciones individuales	Evidencias	Aportación total para el equipo (%)

Pérez Vázquez Jamart Uriel	• Investigación de los antecedentes de la propuesta. • Elaboración de la presentación de propuesta de proyecto. • Codificación. • Descripción del proceso de desarrollo. • Elaboración de la presentación de avances de proyecto.	• Presentación de propuesta de proyecto. • Avance actual de código. • Presentación de avances de proyecto. • Correcciones de Bugs del Código Final. • Desarrollo del Cartel Científico. • Presentación del código final.	25%
Cauich Uc Uriel Eliseo	• Investigación de los antecedentes de la propuesta. • Estructuración de la descripción del proyecto. • Búsqueda de objetivos de proyecto. • Codificación. • Modularidad.	• Presentación de propuesta de proyecto. • Avance actual de código. • Presentación de avances de proyecto. • Correcciones de Bugs del Código Final. • Desarrollo del Cartel Científico. • Presentación del código final.	25%
May Rodríguez Mario Ángel	• Idea principal para proyecto. • Organización del sistema GitHub. • Codificación. • Definición estándar de codificación. • Revisión de formato de código.	• Repositorio en GitHub. • Avance actual de código. • Presentación de avances de proyecto. • Correcciones de Bugs del Código Final. • Desarrollo del Cartel Científico. • Presentación del código final.	25%
Tovar Márquez Luis Enrique	• Idea principal para proyecto. • Búsqueda de objetivos de proyecto. • Descripción de los tipos de usuario. • Codificación.	• Presentación de propuesta de proyecto. • Avance actual de código. • Presentación de avances de proyecto. • Correcciones de Bugs del Código Final.	25%

	• Definición de requisitos y mapeo. • Elaboración de los casos de uso.	• Desarrollo del Cartel Científico. • Presentación del código final.	

Participación de grupo	Evaluable		Puntuación obtenida
	SI	NO	
Búsqueda de propuesta.		X	
Presentación de idea.		X	
Propuesta de proyecto final.	X		17
Proceso de codificación.		X	
Avances de proyecto.	X		29

Enlace al Repositorio de GitHub

https://github.com/MarioMay/Proyecto_PE