Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Bachillerato en Ingeniería en Computación

IC-1803 Taller de Programación

Profesor: Ing. Eduardo A. Canessa Montero, M.Sc.

Informe Técnico Ejecutivo

Estudiantes:

Ericka Céspedes Moya 2017239557

Javier Chiong 2016254702

II Semestre, 2017

Introducción

El siguiente proyecto tiene como objetivo encontrar la solución a cualquier laberinto, el laberinto será representado por medio de una matriz, tomando los valores de “I” como el punto de inicio, “F” como la salida del laberinto, “True” como los lugares en los que hay una pared, y “False” cuando no hay una pared y hay camino. La interfaz gráfica cuenta con botones para que el usuario pueda elegir entre cargar un archivo con una matriz o para que se genere una matriz al azar para resolver. Además, para ambas opciones, se puede elegir entre “Modo automático,” que resuelve automáticamente la matriz y “Modo usuario,” que permite que el usuario navegue dentro del laberinto hasta que encuentre la solución al laberinto; tanto para resolver el laberinto como para generar uno, se utilizaron los conceptos de matrices.

Además, a la hora de cargar un archivo, se utilizó un algoritmo para que a la ahora de presionar las teclas del teclado, se pudiera ingresar el nombre del archivo y usar la matriz en ese archivo; mientras que, para generar una matriz, se usa el mismo algoritmo, pero pide los parámetros de cuántas filas y columnas se desea que tenga el laberinto, y de esta manera generar la matriz para el laberinto acorde a estos parámetros. Luego de ingresada la matriz, se abre una ventana nueva tomando en consideración la cantidad de filas y columnas que tiene la matriz, y se forma el laberinto, sea con la solución automática o con el modo del usuario para que él mismo encuentre la solución al laberinto. Incluso en cualquier momento, a la hora de ingresar al modo automático o al modo usuario, se puede regresar a la interfaz inicial para reiniciar el ciclo del juego, y volver a elegir si se carga o se genera una matriz, o si se desea usar el modo automático o el modo usuario.

Resultados y análisis

En este proyecto se usó un algoritmo de backtracking para encontrar la solución automática al laberinto y se utilizó uno iterativo que asignaba al azar True o False a cada fila de la matriz para crear el laberinto. Se pudo haber usado recursividad en vez de iteración para la creación del laberinto. También se pudo haber implementado backtracking tomando en cuenta las casillas por las que ya se ha pasado para generar un laberinto.

Según Erickson (2014), un **algoritmo de backtracking** intenta construir una solución a un problema computacional incrementalmente. Cuando el algoritmo necesita decidir entre alternativas múltiples, el siguiente componente de la solución simplemente intenta todas las posibles opciones recursivamente. Como la mayoría de los algoritmos recursivos, la ejecución del algoritmo de backtracking puede ser ilustrado usando un árbol recursivo. La raíz del árbol recursivo corresponde a la invocación original del algoritmo; los lados del árbol corresponden a llamadas recursivas.

> "El backtracking es un algoritmo que utiliza una estrategia de búsqueda en profundidad y busca la mejor combinación de las variables para solucionar el problema. Durante la búsqueda, si se encuentra una alternativa incorrecta, la búsqueda retrocede hasta el paso anterior y toma la siguiente alternativa. Cuando se han terminado las posibilidades, se vuelve a la elección anterior y se toma la siguiente opción. Si no hay más alternativas la búsqueda falla." (Calvo y Corisco, 2006)

Algoritmo de Solución

La implementación que se ha llevado a cabo busca en primera instancia las coordenadas del punto inicial "I," y la toma como la casilla de origen para la función que encuentra la solución. En este proyecto, se implementa el algoritmo de forma recursiva, a cada llamada se le asigna una dirección posible a la cual puede desplazarse. La función que busca la solución al laberinto, evalúa los posibles caminos a tomar mediante llamadas recursivas partiendo de la casilla origen. En caso de que se llegue a un callejón sin salida, se devuelve y vuelve a tomar una elección. Continúa de esta manera hasta

encontrar el punto final. De este modo el algoritmo irá recorriendo todo el espacio del laberinto hasta encontrar una salida.

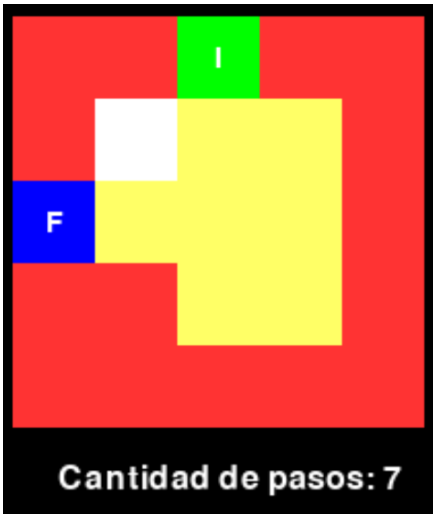


Figura 1.1. Solución al laberinto 5x5

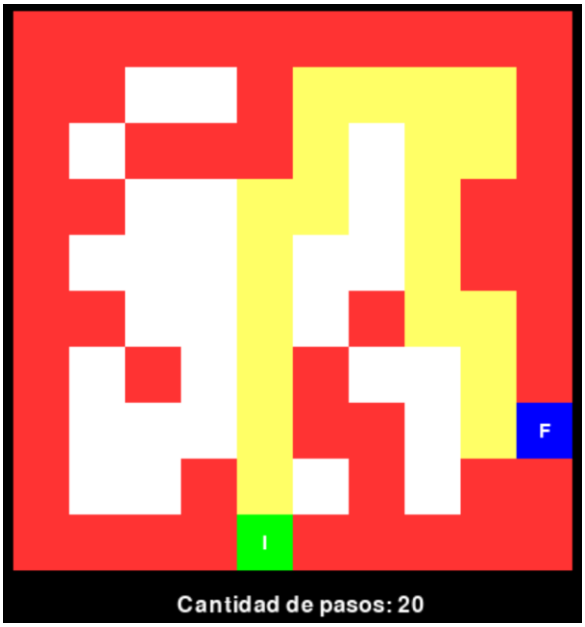


Figura 1.2. Solución al laberinto 10x10

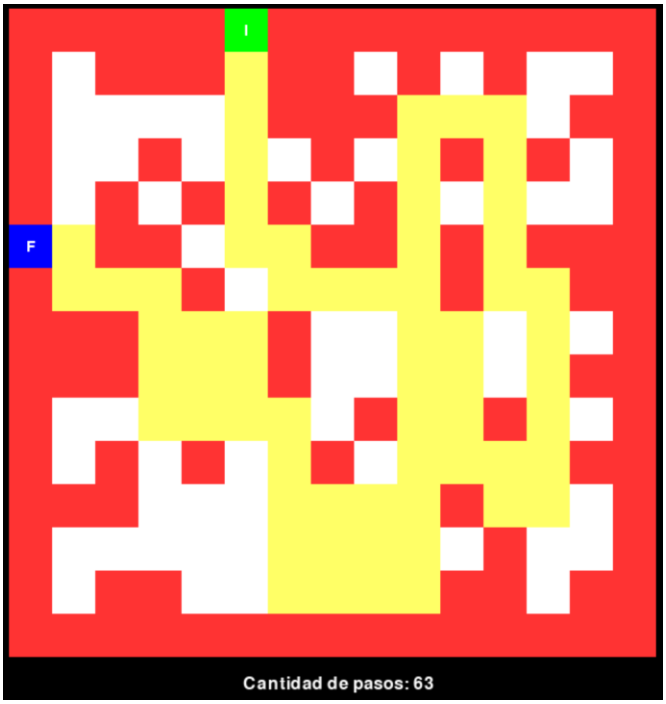


Figura 1.3. Solución al laberinto 15x15

Según Calvo y Corisco, una **solución óptima** a un laberinto es un camino que implica un número mínimo de pasos. En el caso de este proyecto, como se visualiza en las figuras 1.1, 1.2, y 1.3 las soluciones no son óptimas, no se considera la solución que requiera menos pasos, sólo considera que se encuentre la salida. En este proyecto, se podría efectuar el algoritmo que Calvo y Corisco emplearon compara los caminos obtenidos para elegir el que requiere el menor número de pasos, y de esta manera, esta implementación obtiene el camino más óptimo.

El algoritmo de solución de Calvo y Corisco obtiene los movimientos posibles desde cada casilla y se llama al método de forma recurrente para tomar la decisión de si se elige ese camino o no. Se comparan las distancias relativas hasta la meta para cada una de las posibilidades; se añade la casilla actual al camino de la mejor de las soluciones y se devuelve dicho camino al método llamante. De este modo el algoritmo irá recorriendo todo el espacio del laberinto hasta encontrar una salida, y una vez encontrada volverá hacia atrás para ver si es posible encontrar otro camino con un menor número de pasos.

Diagrama de Flujo

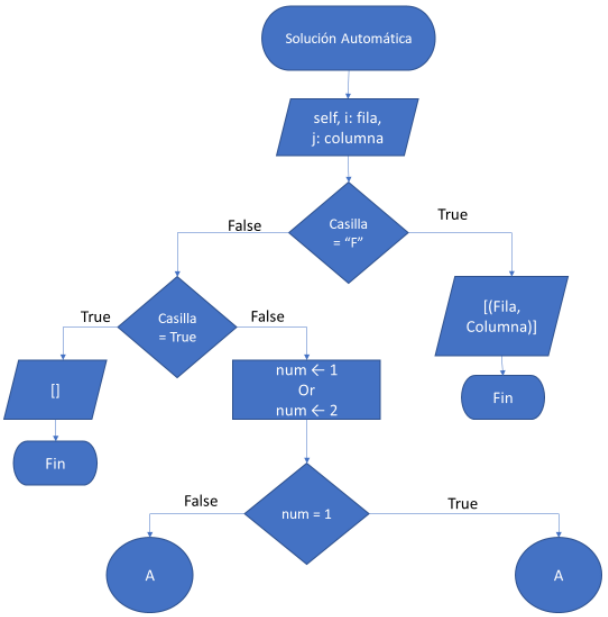


Figura 1.4. Diagrama de flujo de la solución automática, parte 1

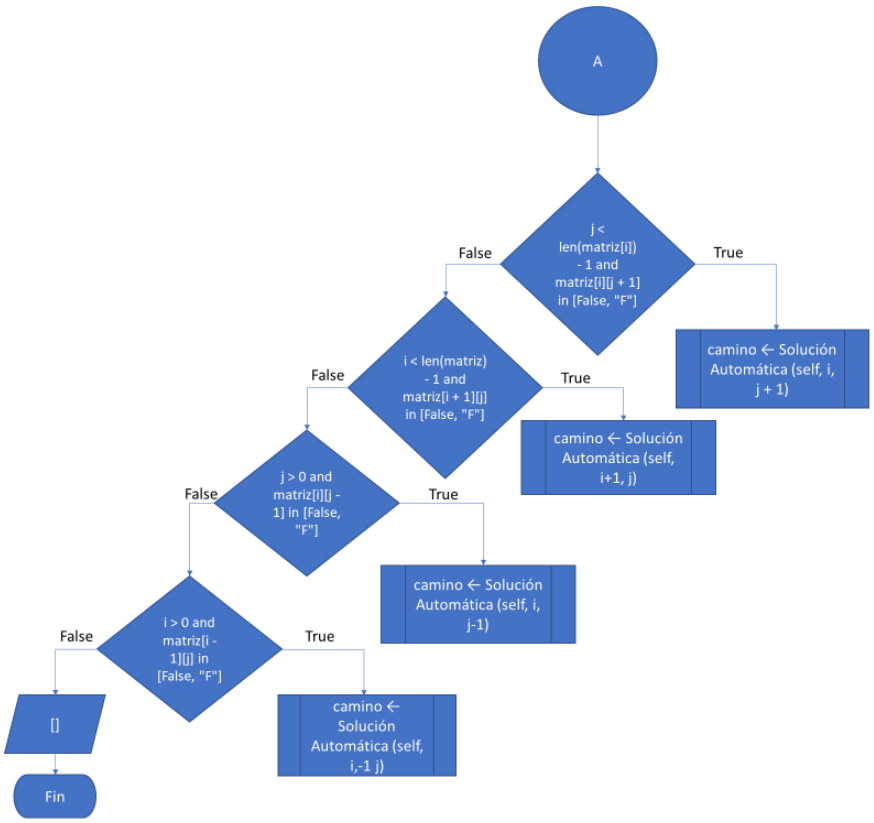


Figura 1.5. Diagrama de flujo de la solución automática, parte 2

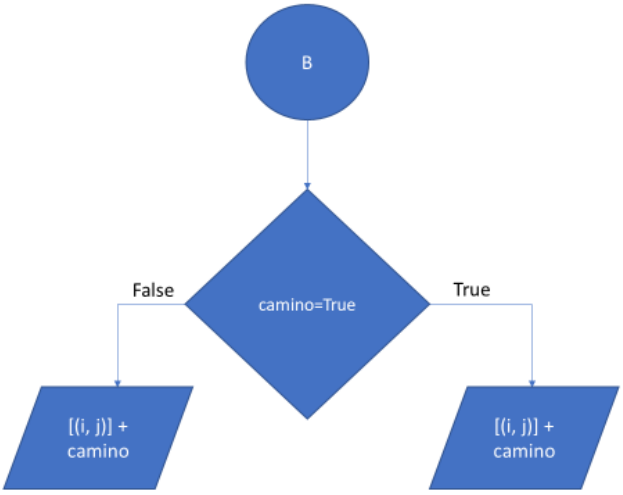


Figura 1.6. Diagrama de flujo de la solución automática, parte 3

Algoritmo de Creación

En este proyecto, el algoritmo para generar un laberinto crea recursivamente las filas del laberinto en la función auxiliar, mientras se van agregando a la función principal que forma la matriz, usando el método random.choice, se pudo hacer que eligiera al azar True o False para que pusiera una pared o un camino aleatoriamente. La función principal agrega los bordes de la matriz que deben ser True en toda la fila, y modifica la primera y última columna de la matriz cambiando las casillas que sean False por True. Luego se coloca la casilla que lleva la “I” y la que lleva la “F,” y seguidamente se verifica que la matriz tenga solución. De esta manera se crea el laberinto que es generado.

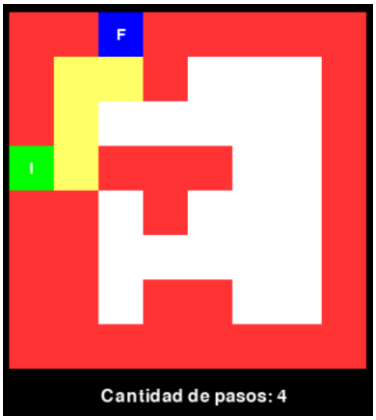


Figura 1.7. Creación del laberinto 8x8

Sin embargo, el algoritmo no es el mejor porque como se visualiza en la figura 1.7, en momentos la “I” y la “F” aparecen muy cerca una de la otra, no es muy complicado, o no aparecen callejones sin salida para aumentar la dificultad del laberinto.

Figura 1.8. Creación del laberinto 15x15

De la misma manera, aparecen paredes demás y cuadros a los cuales no se puede accesar, como se puede ver en la figura 1.8 en la esquina inferior izquierda y en la esquina superior derecha.

Diagrama de flujo

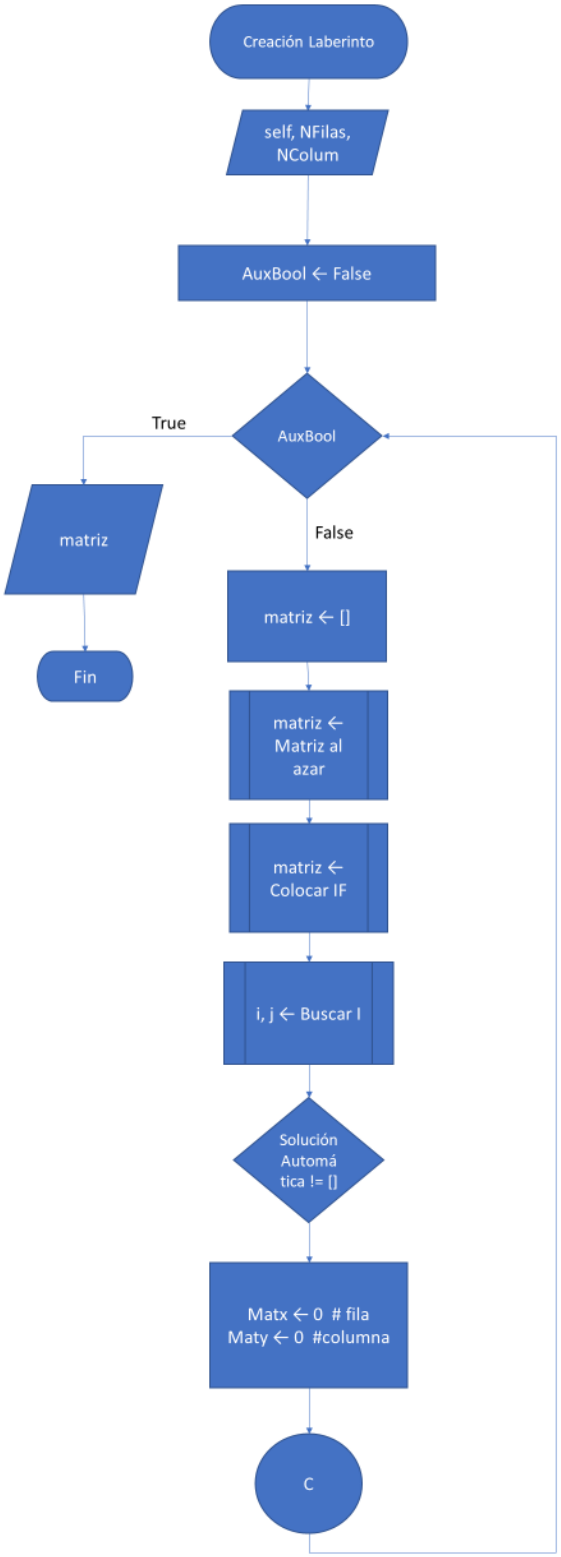


Figura 1.9. Diagrama de flujo de creación del laberinto, parte 1

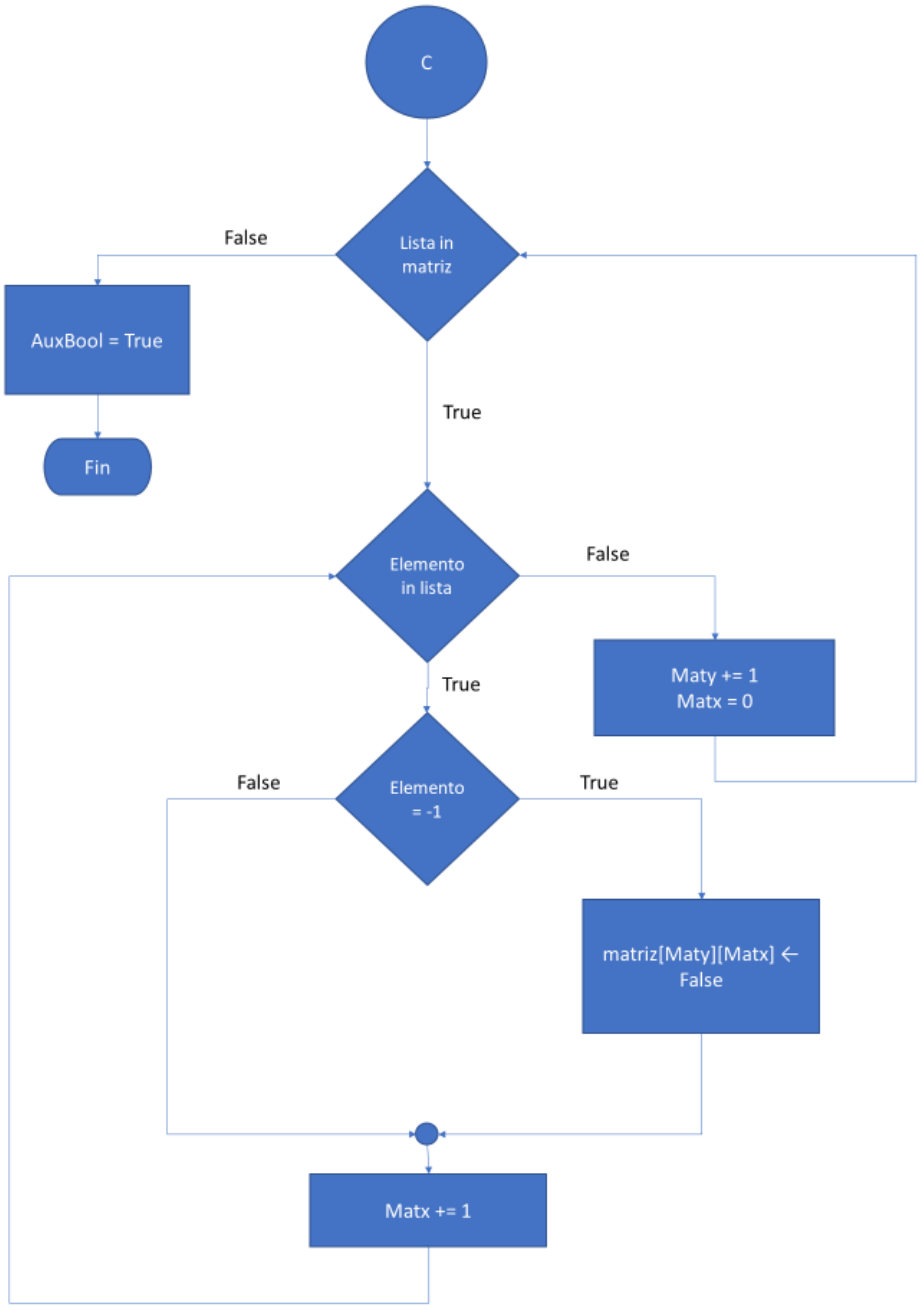


Figura 1.10. Diagrama de flujo de creación del laberinto, parte 2

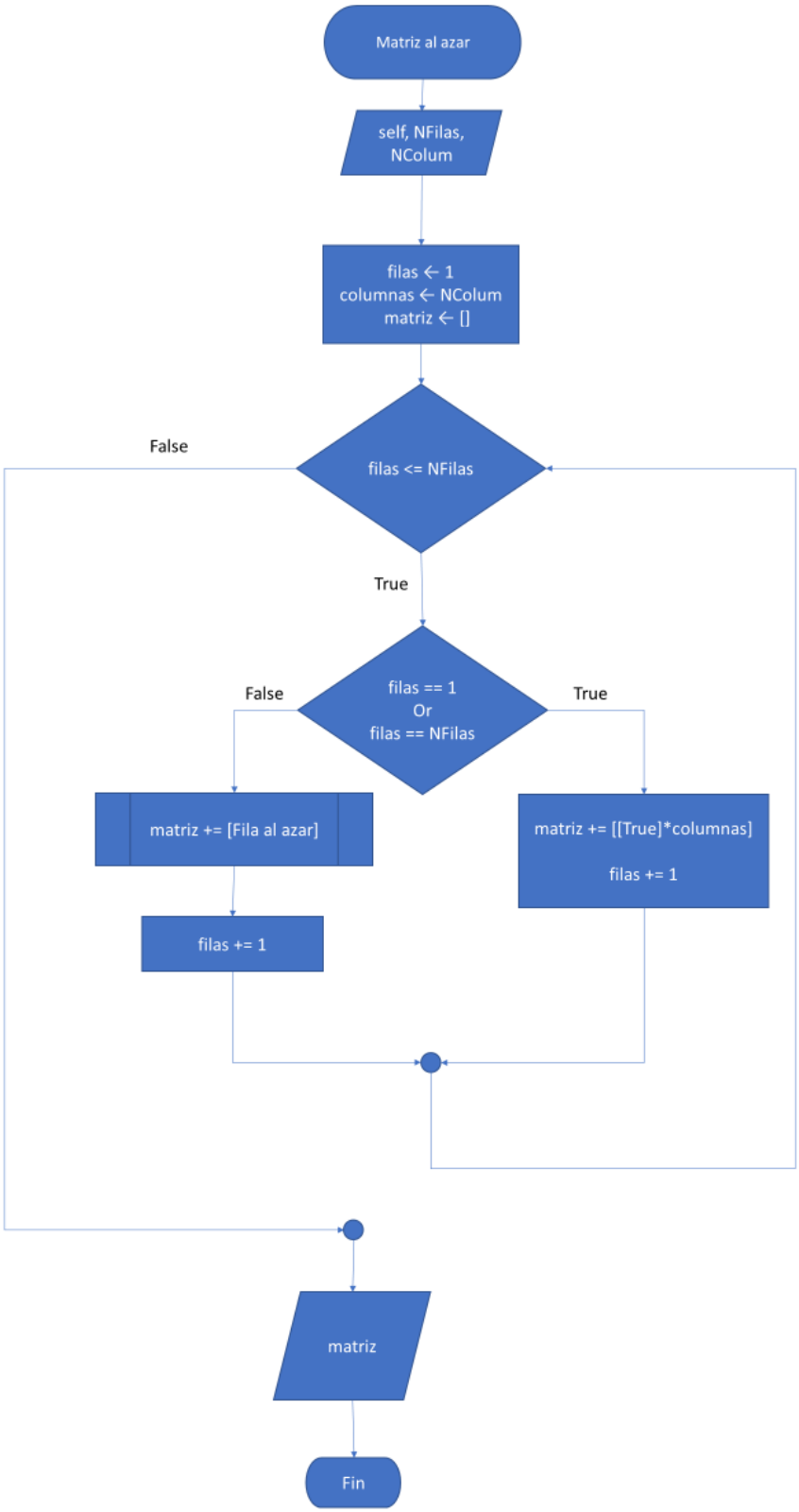


Figura 1.11. Diagrama de flujo de creación del laberinto, parte 3

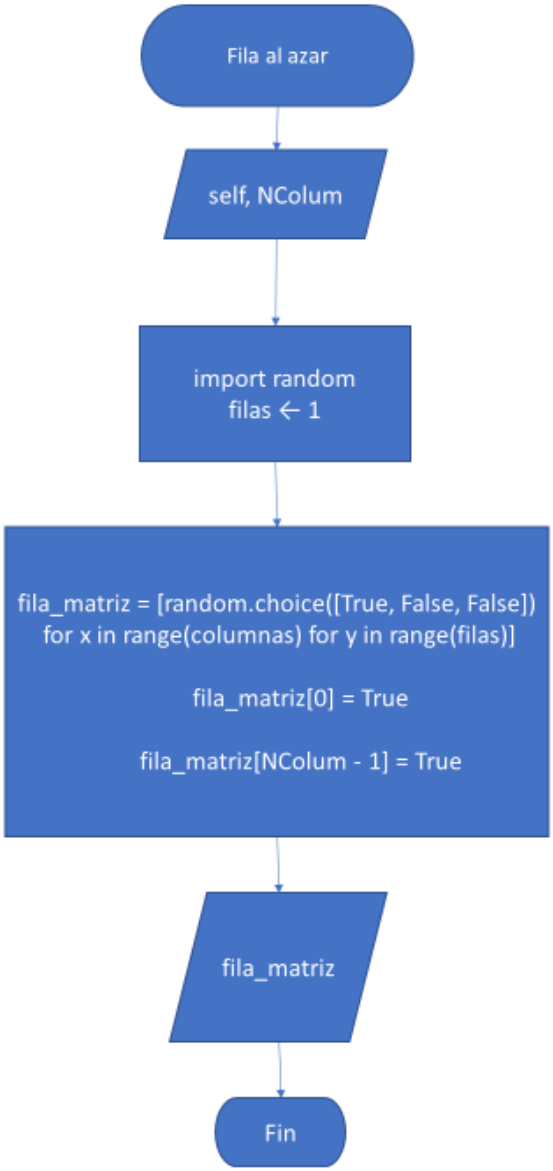


Figura 1.12. Diagrama de flujo de creación del laberinto, parte 4

Conclusiones

Se pudo realizar una interfaz con botones interactivos que le diera al usuario la opción de elegir entre generar o cargar un laberinto, además de que también se tienen los botones que permiten al usuario elegir entre el modo usuario o el modo automático. Luego de presionados los botones, se pudo generar otra ventana en el que se le pedía al usuario ingresar el nombre del archivo en el cual se encuentra la matriz en el caso de hacer click en el botón de cargar la matriz, o ingresar una tupla pidiendo el número de filas y columnas que se desean para generar el laberinto; usando el teclado para ingresar estos parámetros.

Seguidamente, se logró formar el laberinto de acuerdo con la matriz cargada o a los parámetros establecidos. Sin embargo, de vez en cuando se presentan laberintos que no son complicados de resolver, en el Manual del Usuario se hacen recomendaciones para evitar estos errores. También se pudieron crear las modalidades “Modo automático,” y “Modo usuario.” Para ambas se cuentan los pasos que se necesitaron para encontrar la solución, y se agregó un botón “Volver al Menú,” que no fue mencionado en los requisitos, pero regresa al usuario a la ventana principal para que seleccione nuevamente si se desea cargar o generar un laberinto, y las modalidades de juego. El “Modo automático” muestra cómo casilla por casilla se encuentra la solución obtenida, y se pudo implementar el método de backtracking en el algoritmo para devolverse y encontrar otro camino. El “Modo usuario” le permite al usuario moverse en el laberinto usando las flechas del teclado (↑,←,→,↓), y cada vez que se mueve, se le suma a la cantidad de pasos que requirió el usuario para encontrar la solución. Conjuntamente si el usuario ocupa ayuda para encontrar la solución, se implementó un botón “Ayuda” en el “Modo usuario” que muestra 2 casillas adyacentes por las cuales el usuario se puede mover.

Recomendaciones

Crear una función para los botones

```
def button(msg, ancho, alto, tamAncho, tamLargo, funcion=None):  # Funcion para crear botones
    mouse = pygame.mouse.get_pos()  # Posicion del mouse
    click = pygame.mouse.get_pressed()  # Devuelve un vector para indicar si se presiono el mouse
    if (ancho + tamAncho > mouse[0] > ancho) and (alto + tamLargo > mouse[1] > alto):
        pygame.draw.rect(PANTALLA, GrisClaro,
                         (ancho + 1, alto + 1, tamAncho - 2, tamLargo - 2))  # Cambiar de color al boton
        if click[0] == 1:
            funcion()
    else:
        pygame.draw.rect(PANTALLA, Gris,
                         (ancho, alto, tamAncho, tamLargo))  # Definir color default
    # Lo de abajo son comandos para crear cuadros y meterles texto (Estetica del boton)
    smallText = pygame.font.Font("freesansbold.ttf", 20)
    textSurf, textRect = text_objects(msg, smallText)
    textRect.center = ((ancho + (tamAncho / 2)), (alto + (tamLargo / 2)))
    PANTALLA.blit(textSurf, textRect)


def DibujarBotones(Color):
    button("Cargar", 100, 250, ancho, alto,
                          OpcionesCarga)
    button("Generar", 550, 250, ancho, alto,
                          OpcionesGenerar)
    Emergente(Color)
```

Figura 2.1. Función de los botones

Debido a que el programa necesita de varios botones, como el botón “Generar,” “Cargar,” “Modo automático,” “Modo usuario,” “Ayuda” (Ayuda al usuario) y “Volver a Menú,” se recomienda la creación de una función para los botones, que pida las variables de “msg” (nombre del botón), los parámetros en los que se va a encontrar el botón para dibujar el cuadro y para definir en qué rango se debe hacer click para que funcione, y “función” que define lo que debe hacer el botón si se presiona.

Usar “event.unicode”

```
while True:
    barra_insertar_texto()
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
        if event.type == pygame.KEYDOWN:
            if pygame.key.name(event.key)=="return":
                return leer_archivo(nombre_matriz)
            elif pygame.key.name(event.key)=="backspace":
                nombre_matriz=nombre_matriz[0:len(nombre_matriz)-1]
            else:
                nombre_matriz+=event.unicode
            cargar_archivo(nombre_matriz)
```

Figura 2.2. Event.unicode

Usar “event.unicode” en vez de hacer un evento para cada una de las teclas del teclado. Para el “Modo usuario” se usó “event.key == pygame.K_X” (X=DOWN,UP,RIGHT,LEFT) para lograr que el usuario se pudiera movilizar en el laberinto. Sin embargo, a la hora de ingresar el nombre de la matriz a cargar o los parámetros para generar la matriz, se uso “pygame.key.name(event.key)==“return”,” “pygame.key.name(event.key)==“backspace”,” y else. Si se presiona return, se devuelve el nombre o los parámetros ingresados para que el programa genere el laberinto, si se presiona backspace se borra el texto ingresado, y si no ocurre ninguna de las anteriores, se usó “event.unicode” en vez de usar “event.key == pygame.K_X” y especificar cada una de las teclas del teclado.

Clasificar las diferentes ventanas utilizadas

```
# Introduccion del juego
class Interfaz:
    def Main():
        PANTALLA.fill(AzulOscuro)
        reloj.tick(30)
        while True:
            mouse = pygame.mouse.get_pos()  # Posicion del mouse
            color = PANTALLA.get_at(mouse)
            for event in pygame.event.get():
                if event.type == QUIT:
                    pygame.quit()
                    sys.exit()

                texto = "Laberinto"
                crear_texto(texto, 380, 100, 75)

            DibujarBotones(color)
            pygame.display.update()  # Actualiza
```

Figura 2.3. Class

Al usar “class,” se facilitó la movilización entre las diferentes ventanas del laberinto, la ventana principal en donde se seleccionan las preferencias del usuario, la segunda ventana en donde se ingresan sean el nombre de la matriz o el tamaño del laberinto, y la tercera ventana en la cual se forma el laberinto. En la tercera ventana, a la hora de formarse el laberinto, la ventana que aparece es del tamaño acorde con los parámetros de filas y columnas ingresados por el usuario anteriormente. De esta manera, “class” facilitó tanto la creación de la tercera ventana con las funciones mencionadas anteriormente y el botón “Volver a Menú,” en donde el usuario puede regresar a la ventana principal.

Literatura consultada

Corisco Nieto, A. y Calvo Atance, G. (2006). *Solución de un laberinto mediante la aplicación de un algoritmo de backtracking óptimo*. Recuperado de https://www.google.com/url?sa=t&source=web&rct=j&url=http://www.it.uc3m.es/jvillena/irc/practicas/06-07/09.pdf&ved=0ahUKEwiZ9aejjr7XAhXIQiYKHd-9BYoQFghlMA0&usg=AOvVaw37oEpfDMOXJcdxbD7hAsx0

Erickson, J. (2015). *Algorithms*. *Lecture 3: Bactracking*. Recuperado de http://jeffe.cs.illinois.edu/teaching/algorithms/notes/03-backtracking.pdf

Pygame. (s.f.). *pygame.font*. Recuperado de https://www.pygame.org/docs/ref/font.html

Pygame. (s.f.). *pygame.color*. Recuperado de https://www.pygame.org/docs/ref/color.html

Pygame. (s.f.). *Pygame Basics*. Recuperado de http://www.cogsci.rpi.edu/~destem/gamedev/pygame.pdf

Python. (s.f.). *Introduction to PyGame*. Recuperado de https://pythonprogramming.net/pygame-python-3-part-1-intro/

Python. (s.f.). *Displaying text to PyGame screen*. Recuperado de https://pythonprogramming.net/displaying-text-pygame-screen/

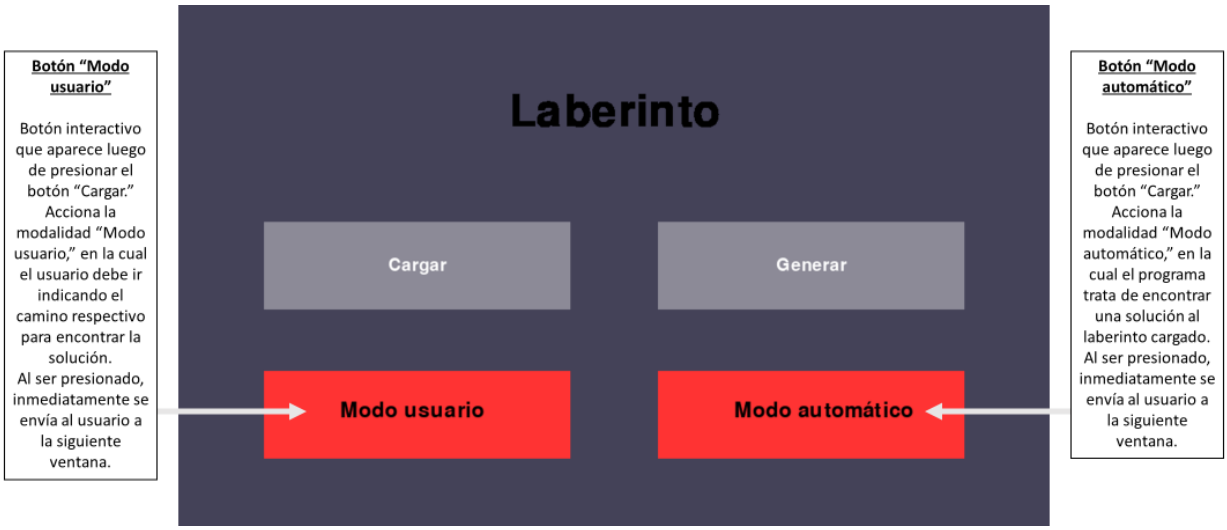
Python. (s.f.). *PyGame Buttons, part 1, drawing the rectangle*. Recuperado de https://pythonprogramming.net/pygame-buttons-part-1-button-rectangle/

Python. (s.f.). *PyGame Buttons, part 2, making the buttons interactive*. Recuperado de https://pythonprogramming.net/making-interactive-pygame-buttons/

Python. (s.f.). *PyGame Buttons, part 3, adding text to the button*. Recuperado de https://pythonprogramming.net/placing-text-pygame-buttons/

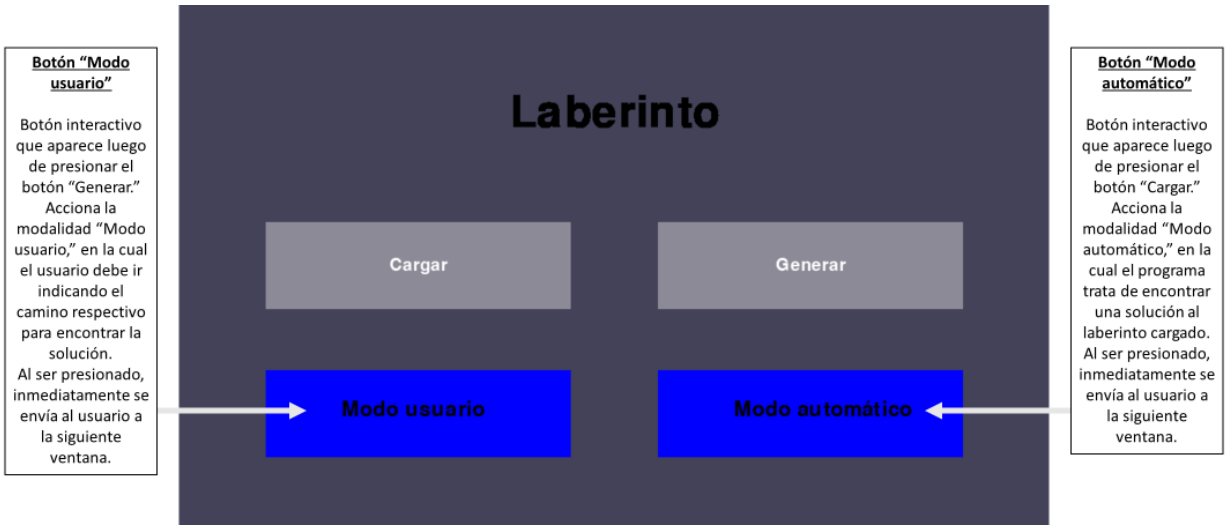
Python. (s.f.). *PyGame Buttons, part 4, creating a general PyGame button function*. Recuperado de https://pythonprogramming.net/pygame-button-function/

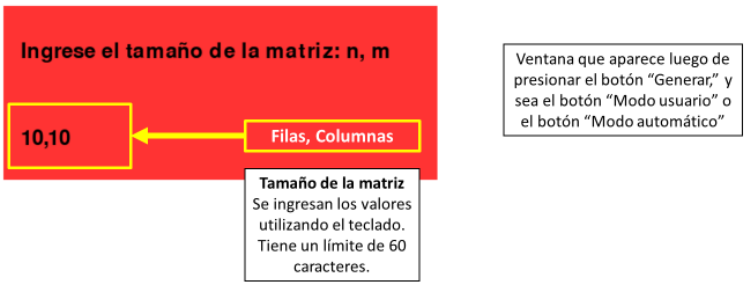
Anexo – Manual de usuario

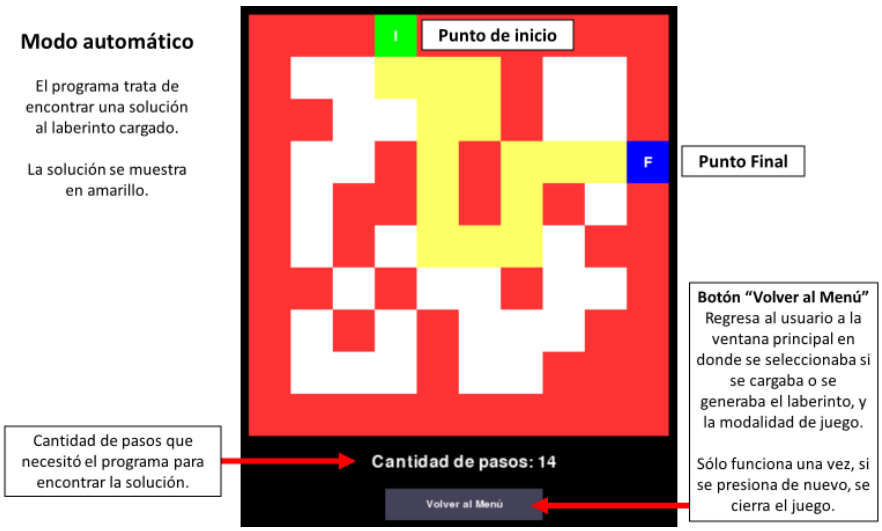


Ventana que aparece luego de presionar el botón "Cargar," y sea el botón "Modo usuario" o el botón "Modo automático"

Nombre del archivo
Se ingresa el nombre del archivo utilizando el teclado.
Tiene un límite de 60 caracteres.
Si el archivo no se encuentra, devuelve al usuario a la ventana principal.


Botón "Modo usuario"
Botón interactivo que aparece luego de presionar el botón "Generar." Acciona la modalidad "Modo usuario," en la cual el usuario debe ir indicando el camino respectivo para encontrar la solución.
Al ser presionado, inmediatamente se envía al usuario a la siguiente ventana.
Laberinto
Cargar
Generar
Modo usuario
Modo automático
Botón "Modo automático"
Botón interactivo que aparece luego de presionar el botón "Cargar." Acciona la modalidad "Modo automático," en la cual el programa trata de encontrar una solución al laberinto cargado.
Al ser presionado, inmediatamente se envía al usuario a la siguiente ventana.


Ingrese el tamaño de la matriz: n, m
10,10
Filas, Columnas
Tamaño de la matriz
Se ingresan los valores utilizando el teclado. Tiene un límite de 60 caracteres.
Ventana que aparece luego de presionar el botón "Generar," y sea el botón "Modo usuario" o el botón "Modo automático"

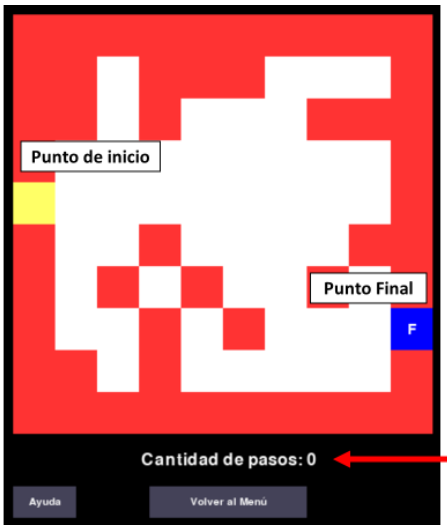

Modo automático
El programa trata de encontrar una solución al laberinto cargado.
La solución se muestra en amarillo.
I
Punto de inicio
F
Punto Final
Cantidad de pasos que necesitó el programa para encontrar la solución.
Cantidad de pasos: 14
Volver al Menú
Botón "Volver al Menú"
Regresa al usuario a la ventana principal en donde se seleccionaba si se cargaba o se generaba el laberinto, y la modalidad de juego.
Sólo funciona una vez, si se presiona de nuevo, se cierra el juego.

Modo usuario

El usuario debe ir indicando el camino respectivo para encontrar la solución.

El cuadro amarillo lo controla el usuario con las flechas del teclado.

Con las flechas del teclado, se mueve al usuario a la casilla siguiente:
↑: mueve el cuadro hacia arriba
←: mueve el cuadro hacia la derecha
→: mueve el cuadro hacia la izquierda
↓: mueve el cuadro hacia abajo



Cada vez que el usuario mueve al cuadro amarillo, va aumentando la cantidad de pasos que necesitó el usuario para encontrar la solución.

Modo usuario

Ayuda al usuario

Al presionar "Ayuda al usuario," aparecen dos cuadros celestes adyacentes al cuadro amarillo que controla el usuario, que ayudan al usuario a encontrar la solución.

Botón "Ayuda al usuario"

Botón "Volver al Menú" En cualquier momento se puede presionar para regresar a la ventana principal.

Modo usuario

Si el usuario encuentra la solución, al posicionarse sobre el **punto final**, se despliega un mensaje **"Has Ganado."**

Botón "Volver al Menú" En cualquier momento se puede presionar para regresar a la ventana principal.

Precauciones

Matriz demasiado pequeña o demasiado grande

A la hora de ingresar los parámetros del tamaño de la matriz, se recomienda de 7x7 a 15x15. Es recomendable que se ingresen parámetros para que el laberinto crezca a lo ancho y no lo alto, de esta manera, se recomienda un laberinto con parámetros menores a 15 en las filas y menores a 35 en las columnas.

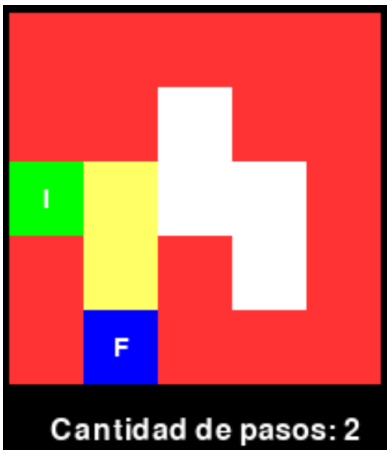


Figura 3.1. Laberinto 5x5

En la Figura 3.1, se muestra que si la matriz es demasiado pequeña entonces el laberinto generado no es muy complicado de resolver, mientras que si la matriz es demasiado grande (18x18) no cabe en la pantalla.

Botón “Volver al Menú”

No presionar el botón “Volver al Menú” más de 2 veces. De lo contrario, se cerrará el programa. El sólo funciona dos veces.

Nombre de matriz incorrecto

No ingresar la matriz incorrecta más de una vez o el programa se cerrará.