

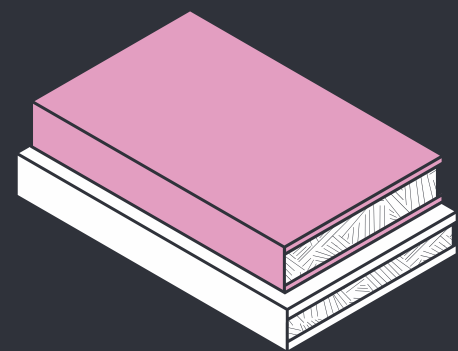
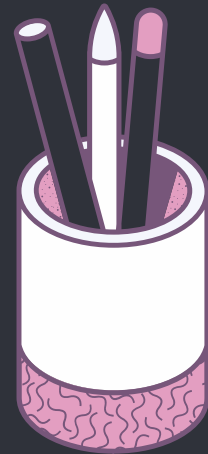
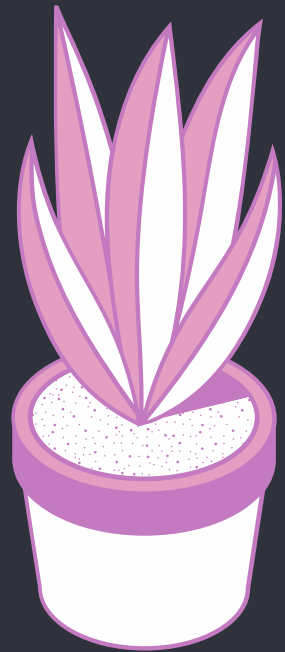


FUNDAMENTOS DE PROGRAMACIÓN

PROYECTO FINAL

-HABITUS-

Control de hábitos



García Olvera José Ignacio
López Morales Fernando Samuel
González Falcón Luis Adrián
Arias Jiménez Alejandro
Ortega Ortega Genaro Raziel

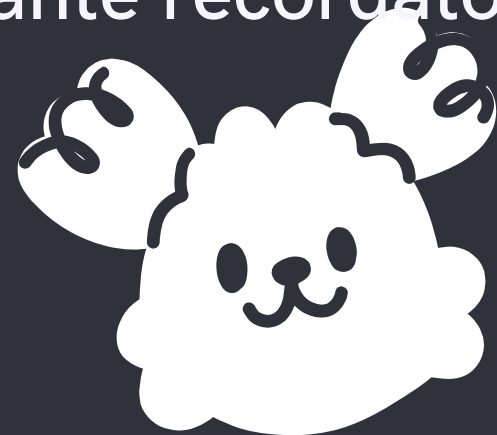


CONTEXTO

El control de hábitos es esencial para el éxito y bienestar diario, ya que los hábitos impactan en nuestra salud y rendimiento.



Las rutinas diarias y el apoyo, mediante recordatorios o personas cercanas, son cruciales.



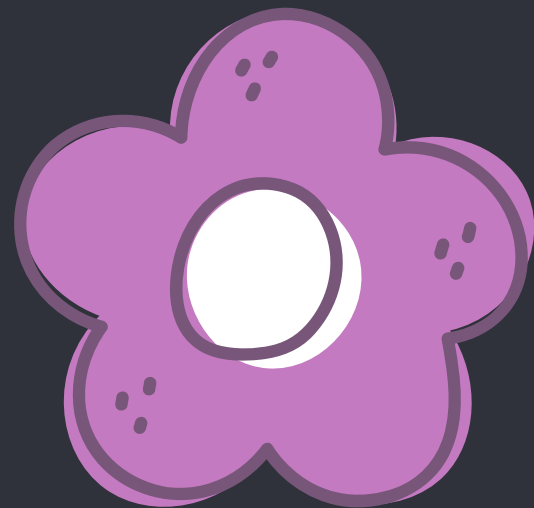
Celebrar pequeños logros impulsa el auto-mejoramiento, creando un camino hacia el bienestar constante.



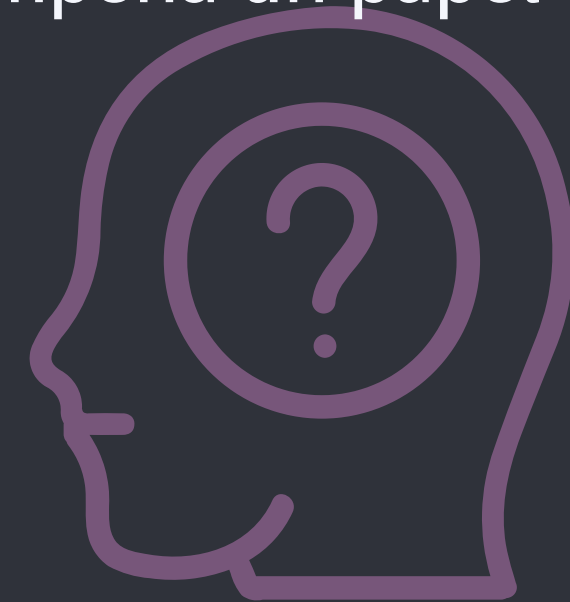
TEORIA DE WILLIAM JAMES

James creía que gran parte de la vida cotidiana está gobernada por hábitos. Sostenía que la voluntad desempeña un papel crucial en la creación y modificación de hábitos.

Explicó que al prestar atención a una tarea específica y al repetirla, se puede convertir en un hábito arraigado.



James describió el ciclo del hábito, "bucle de hábito", donde un hábito se forma a través de la repetición de este ciclo.



ANALISIS DEL PROBLEMA

Establecer un control consciente que nos permita moldear positivamente nuestra vida, identificando hábitos que contribuyen a metas y eliminando limitantes.

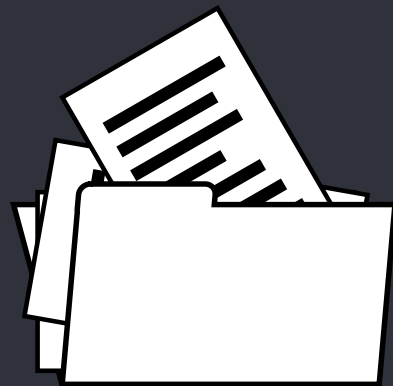
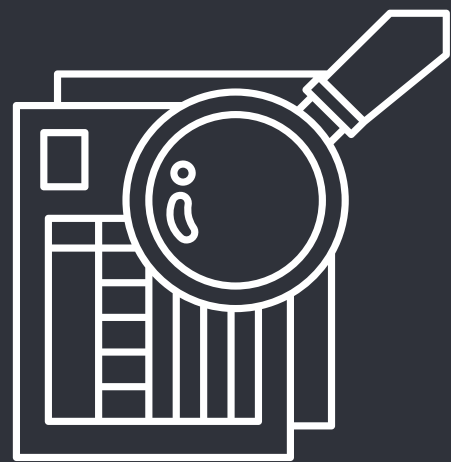
No hay una solución única, debemos adaptar estrategias a nuestra personalidad.





OBJETIVO

Desarrollar un programa en lenguaje C que funcione como una herramienta efectiva para el control de hábitos, brindando a los usuarios la capacidad de monitorizar, analizar y mejorar sus comportamientos diarios.



- Interfaz Intuitiva: Crear una interfaz de usuario amigable
- Registro de Hábitos: Implementar una función para registrar hábitos específico
- Análisis de Datos: Desarrollar algoritmos que analicen los datos ingresados, proporcionando a los usuarios información útil sobre patrones, tendencias y áreas de mejora en sus hábitos
- Alertas y Recordatorios: Incorporar un sistema de alertas y recordatorios personalizables
- Establecimiento de Metas: Permitir a los usuarios establecer metas específicas relacionadas con sus hábitos

PLANEACION DE UN PROYECTO

Manual de instalación de la biblioteca

Allegro y Clion

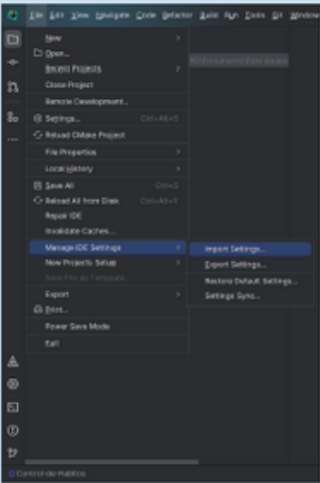
Elaborado por @TheUniqueFersa

Para la elaboración del proyecto de Fundamentos de Programación
Facultad de Ingeniería UNAM 2024-1

Instalar Allegro en Clion como plantilla de CMakeLists

Antes de intentar crear manualmente el archivo CMakeLists.txt debemos asegurarnos de que CLion reconozca como plantilla un formato específico para Allegro, para ello:

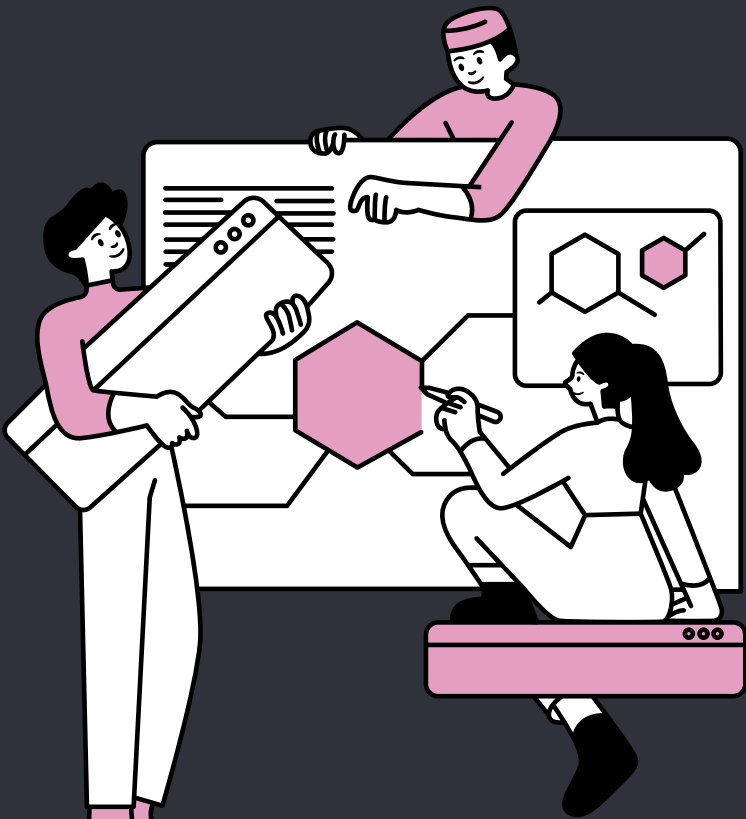
1. Abre Clion
2. Crea un proyecto cualquiera de prueba. (recuerda que el nombre no debe tener espacios).
3. Descarga el siguiente zip
https://drive.google.com/file/d/1X1nwHorhqIP1s9wDU2dCA_4GALeJaTx-/view?usp=sharing
4. Una vez descargado, en Clion vé a *Files>Manage IDE Settings>Import Settings*.



5. Selecciona el archivo zip que acabas de descargar “settings.zip” y da OK
6. Da clic nuevamente en OK y después da clic en “Import and Restart”, se reiniciará el IDE

DEFINICIÓN PROYECTO

ASPECTOS GENERALES





CLION-ALLEGRO(5)



```
CMakeLists.txt  tests.txt  main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <math.h>
5  #include <time.h>
6  #include <allegro5/allegro.h>
7  #include <allegro5/allegro_image.h>
8  #include <allegro5/allegro_primitives.h>
9  #include <allegro5/allegro_font.h>
10 #include <allegro5/allegro_ttf.h>
11 #include <allegro5/allegro_audio.h>
12 #include <allegro5/allegro_acodec.h>
13 //#include "../include/structs.c"
14 #include "../include/CRUD.c"
15 #include "../include/resources.c"
16 #include "../tests/depuracion.c"
17 /* ----> <---- */
18 /* ----> Prototipos <---- */
19 int inicializar_allegro(); //INICIALIZA TODO LO NECESARIO PARA QUE ALLEGRO FUNCIONE
20
```


PSEUDOCODIGO



```
Funcion modoHabitos()
//EVENTOS DE TECLADO PARA REGISTRAR
Si estado=0 Entonces
  //Desplazarse por el menú
SiNo
  Si estado=1 Entonces
    //Crear Habitos
SiNo
  Si estado=2 Entonces
    //EDITAR HABITO
SiNo
  Si estado=3
    //ELIMINAR HABITO
  Fin Si
Fin Si
Fin Si
Fin Si
FinFuncion
Funcion main_habitus()
  pedirUsuario();//pide el nombre del usuario
  cargarRegistros();
  Mientras !ventanaCerrada Hacer
    activarEventos();//Se inicializan los evento para cada Caso
    //(eventos de teclado, de ventana, etc)
  Segun momento Hacer
    -1:
      depuracion(); //Depuracion de registros
    0:
      pedirNombreUsuario();//Registrar el nombre de usuario
    1:
      modoHabitos(); // Gestor de habitos
    2:
      horarios();//GESTOR HORARIOS
    3:
      recoradtorios();//GESTOR RECORDATORIOS
    4:
      ajustes();
  De Otro Modo:
  Fin Segun
Fin Mientras
```

Fin Funcion

```
Funcion inicializarVentanas()
// ... CÓDIGO DONDE SE INICIALIZA VENTANAS Y EVENTOS
main_habitus();
Escribir "Inicializar ventana";
Fin Funcion
```

```
Algoritmo control_de_Habitos
Si inicializaAllegro Entonces
  inicializarVentanas();//DESCRIPCION
SiNo
  Escribir "Ocurrió un error";
Fin Si
```

FinAlgoritmo

```
Algoritmo sin_titulo
  // Declaración de variables
  Entero acceso

  // Inicialización de Allegro y creación de recursos
  Si inicializar_allegro() Entonces
    disp <- al_crear_display(1200, 700)
    AFK <- al_crear_timer(30)
    al_establecer_titulo_ventana(disp, "Hábitus")
    cola_eventos <- al_crearCola_eventos()

    al_registrar_fuente_evento(cola_eventos,
al_obtener_fuente_evento temporizador(AFK))
    al_registrar_fuente_evento(cola_eventos,
al_obtener_fuente_evento_ventana(disp))
    al_registrar_fuente_evento(cola_eventos, al_obtener_fuente_evento_teclado())

    al_iniciar_temporizador(AFK)

    // Creación de estructuras y operaciones CRUD
    Si no verificar_existencia_archivo(rutaAPP) Entonces
      INSERT(rutaAPP, reseteoAPP, longitud(struct APP), 1)
      // fin_si debería ir aquí
    Fin Si

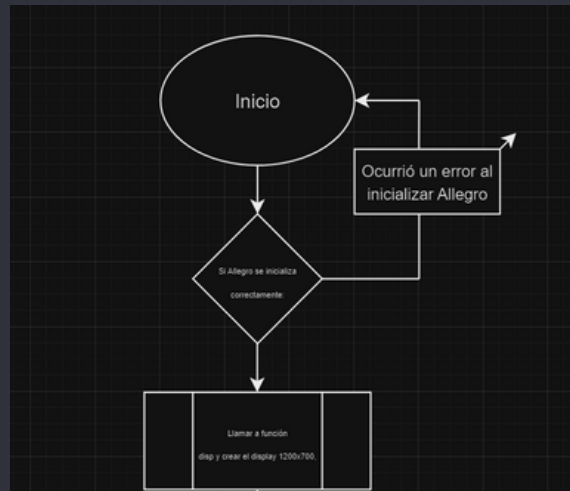
    SELECT(rutaAPP, app_recibe, longitud(struct APP), 1, 1)

    // Llamada a la función principal de la aplicación
    main_habitus(app_recibe.init, app_recibe.ID_last_user)

    // Liberación de recursos
    al_destruirCola_eventos(cola_eventos)
    al_destruir_display(disp)
    al_destruir_temporizador(AFK)
  Sino
    Imprimir("Ocurrio un error") // TODO: Registrar errores en un archivo de texto
  Fin Si
Fin Algoritmo
```

```
// Prototipos de funciones y procedimientos
Funcion Entero inicializar_allegro()
Procedimiento main_habitus(Entero, Entero)
Procedimiento actualizar_display()
Entero init_resources()
Procedimiento IUSD()
Procedimiento llamarINSERT()
Procedimiento llamarUPDATE()
Procedimiento llamarSELECT()
Procedimiento llamarDELETE()
Procedimiento creacionEstructuras()
```

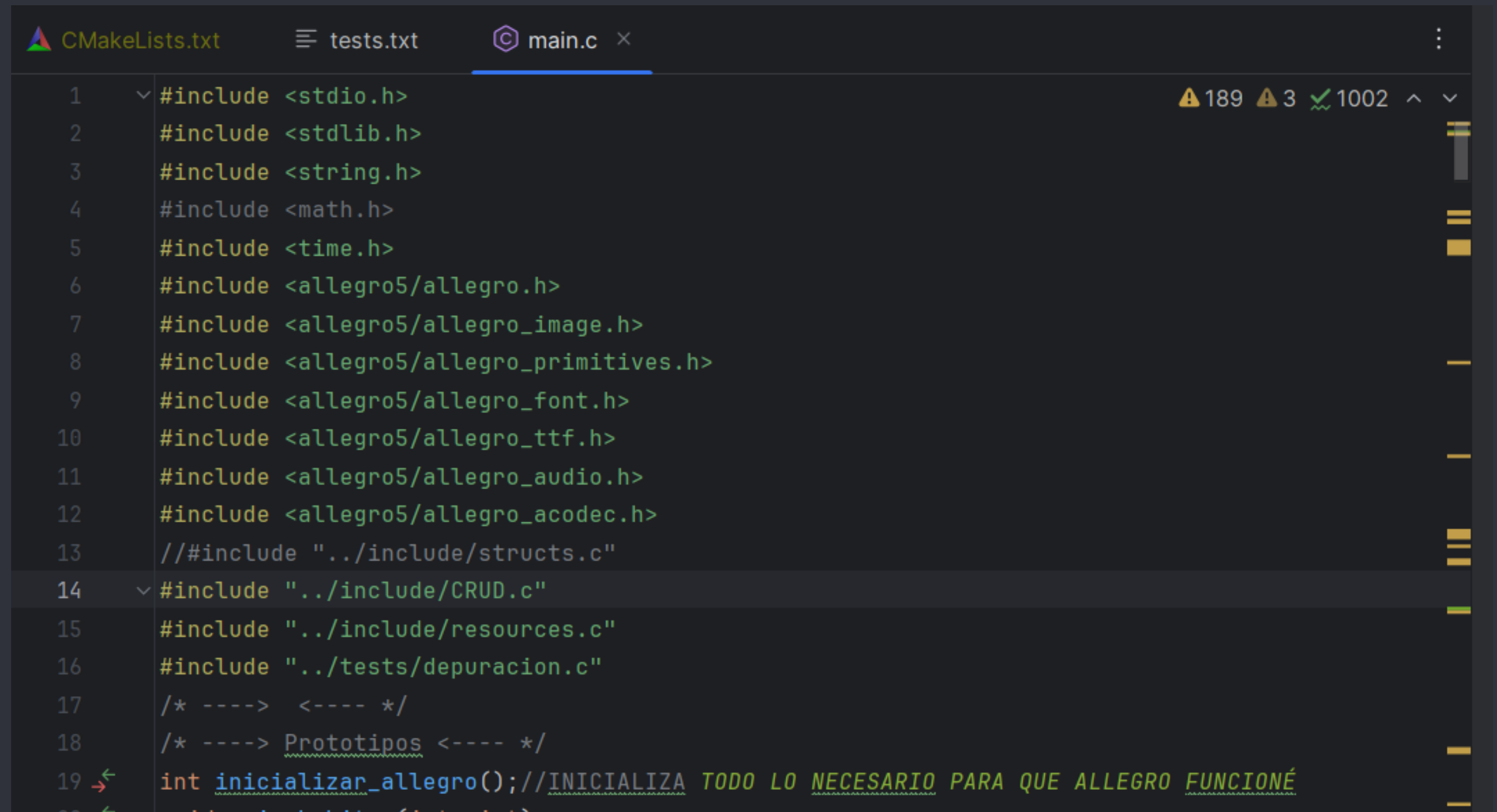
DIAGRAMA DE FLUJO



https://drive.google.com/file/d/1qK9I_KCsgkW0GWfPojT1I76yRdzcnJ_t/view?usp=sharing
Link del diagrama de flujo main

USO DE LIBRERIAS

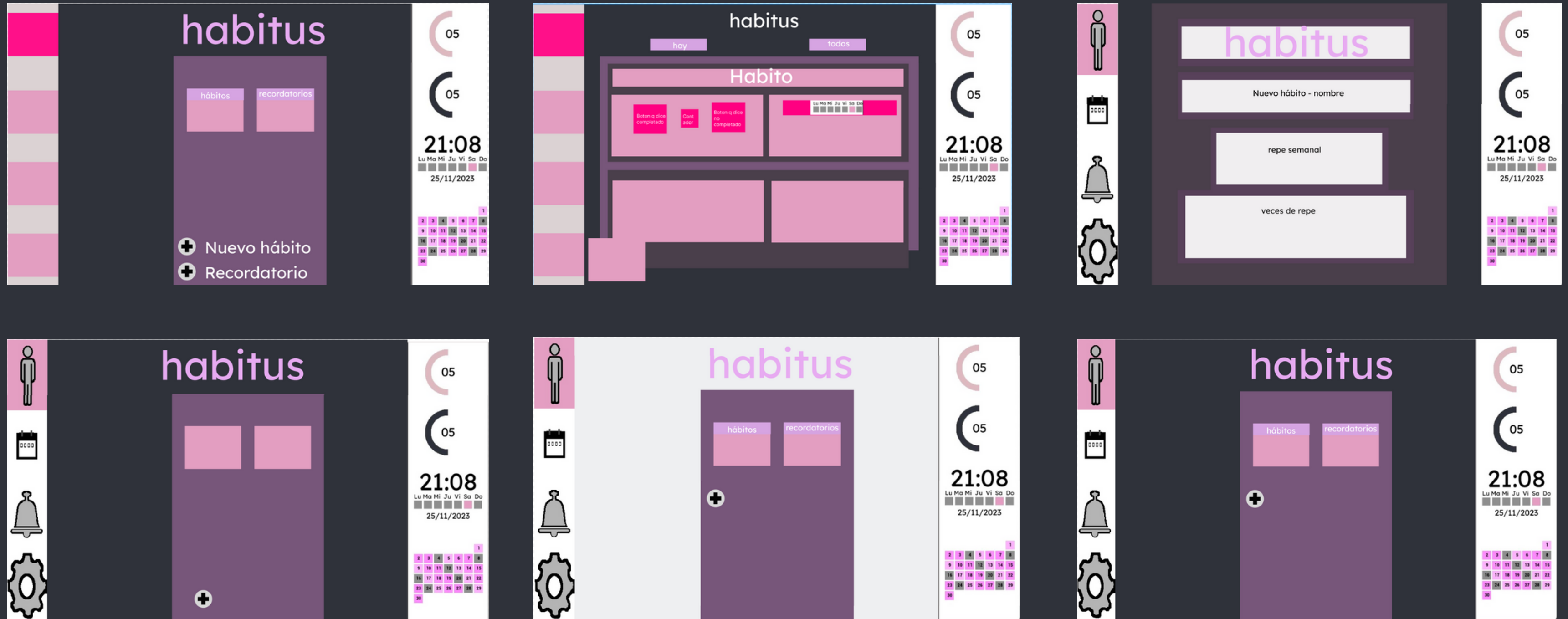
```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <ctype.h>
```



The screenshot shows a code editor with three tabs: CMakeLists.txt, tests.txt, and main.c. The main.c tab is active, displaying a C program. The code includes several standard and custom headers, followed by a function declaration for `inicializar_allegro`. The editor interface includes a line number margin on the left, a status bar on the right showing 189 warnings, 3 errors, and 1002 successful checks, and a vertical scrollbar on the far right.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <math.h>
5  #include <time.h>
6  #include <allegro5/allegro.h>
7  #include <allegro5/allegro_image.h>
8  #include <allegro5/allegro_primitives.h>
9  #include <allegro5/allegro_font.h>
10 #include <allegro5/allegro_ttf.h>
11 #include <allegro5/allegro_audio.h>
12 #include <allegro5/allegro_acodec.h>
13 // #include "../include/structs.c"
14 #include "../include/CRUD.c"
15 #include "../include/resources.c"
16 #include "../tests/depuracion.c"
17 /* ----> <---- */
18 /* ----> Prototipos <---- */
19 int inicializar_allegro(); // INICIALIZA TODO LO NECESARIO PARA QUE ALLEGRO FUNCIONE
```

USO DE FIGMA



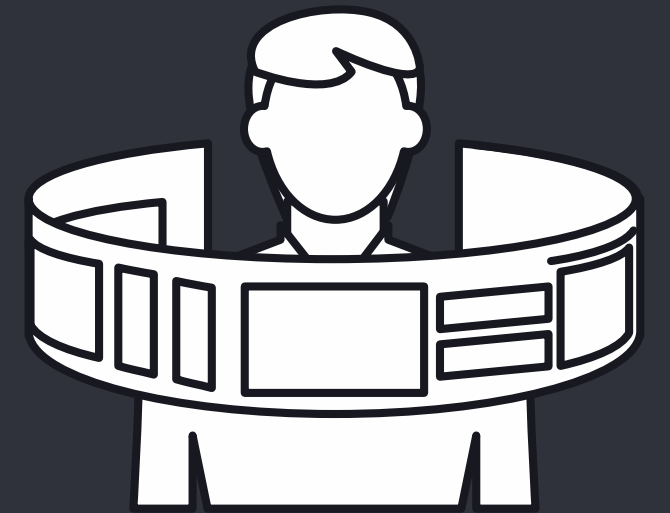
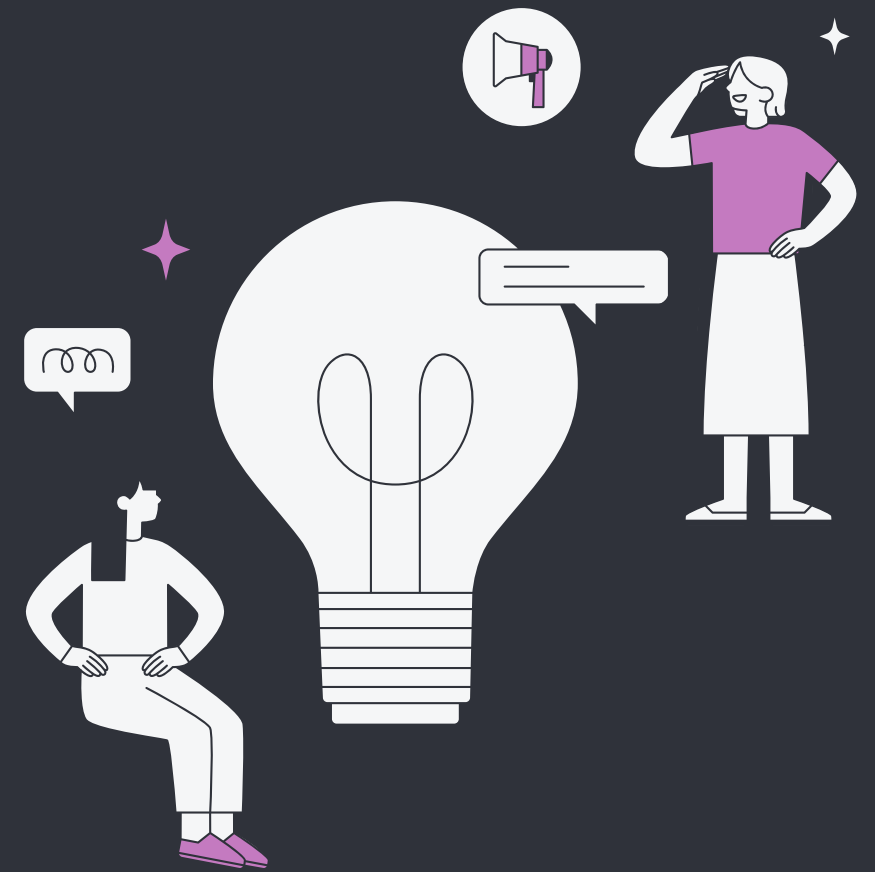
NO SIST ULT CON NO CON

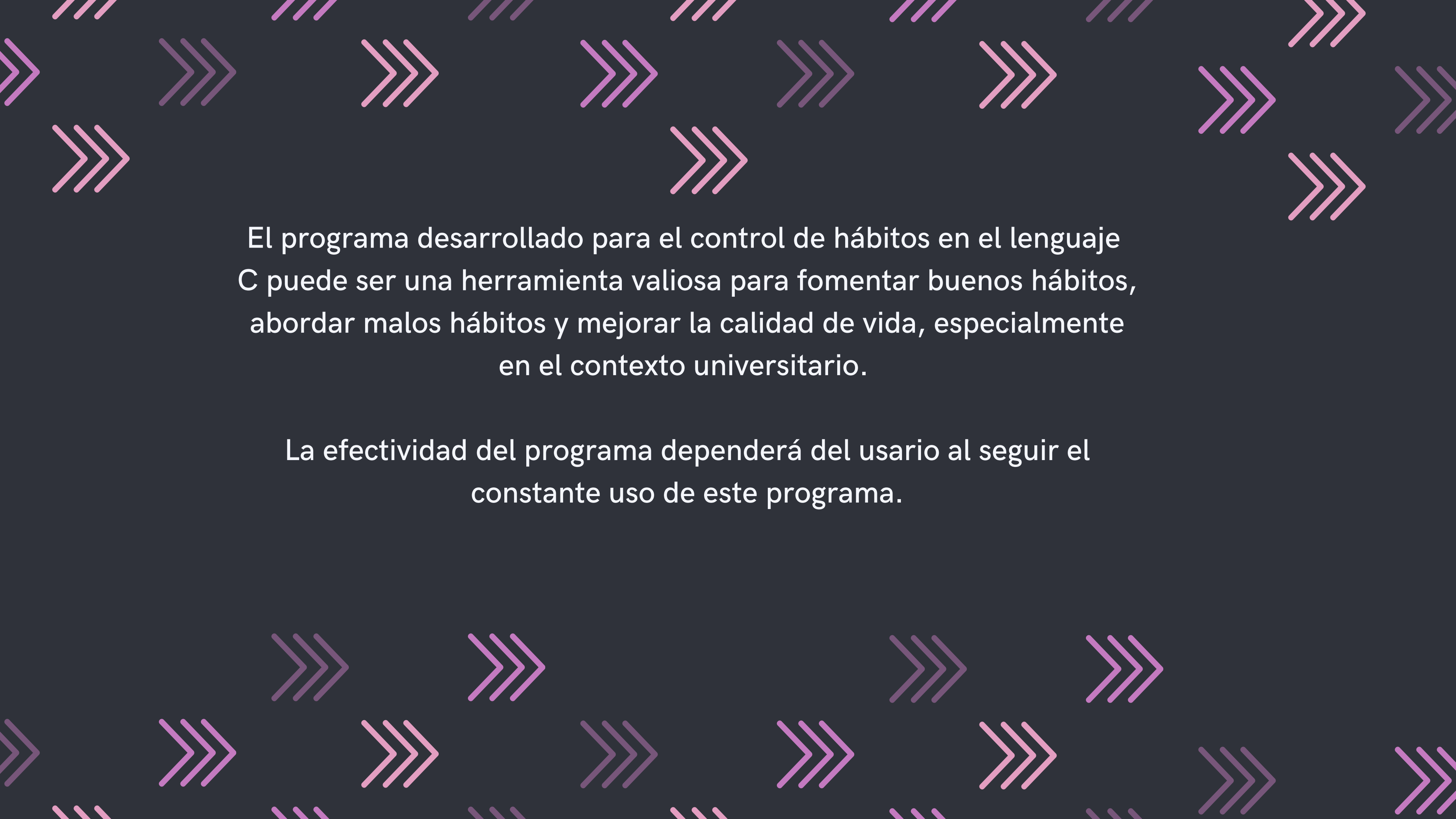
Los buenos hábitos son esenciales para una vida equilibrada y productiva.

Este programa de control de hábitos puede ayudar a fomentar la creación y mantenimiento de hábitos positivos que contribuyan a nuestro bienestar general.

En cada uno esta el Identificar y abordar malos hábitos comunes entre los universitarios, como la procrastinación, la falta de ejercicio, el sueño insuficiente, y la mala alimentación, puede ser crucial para el éxito académico y la salud mental.

El programa podría servir como una herramienta para fomentar la disciplina al proporcionar recordatorios, establecer metas y ofrecer incentivos para cumplir con los hábitos deseados.





El programa desarrollado para el control de hábitos en el lenguaje C puede ser una herramienta valiosa para fomentar buenos hábitos, abordar malos hábitos y mejorar la calidad de vida, especialmente en el contexto universitario.

La efectividad del programa dependerá del usuario al seguir el constante uso de este programa.

GRACIAS



REFERENCIAS

Psicoadapta. (2018, 18 de diciembre). QUÉ ES EL HÁBITO. Blog de Psicoadapta.

Recuperado de <https://www.psicoadapta.es/blog/que-es-el-habito/>

García Díaz, R. (Fecha de publicación). El falso mito de los 21 días para crear un hábito. El primer diario digital de Castilla y León. Recuperado de <https://www.tribunasalamanca.com/blogs/para-profesionales/posts/el-falso-mito-de-los-21-dias-para-crear-un-habito>