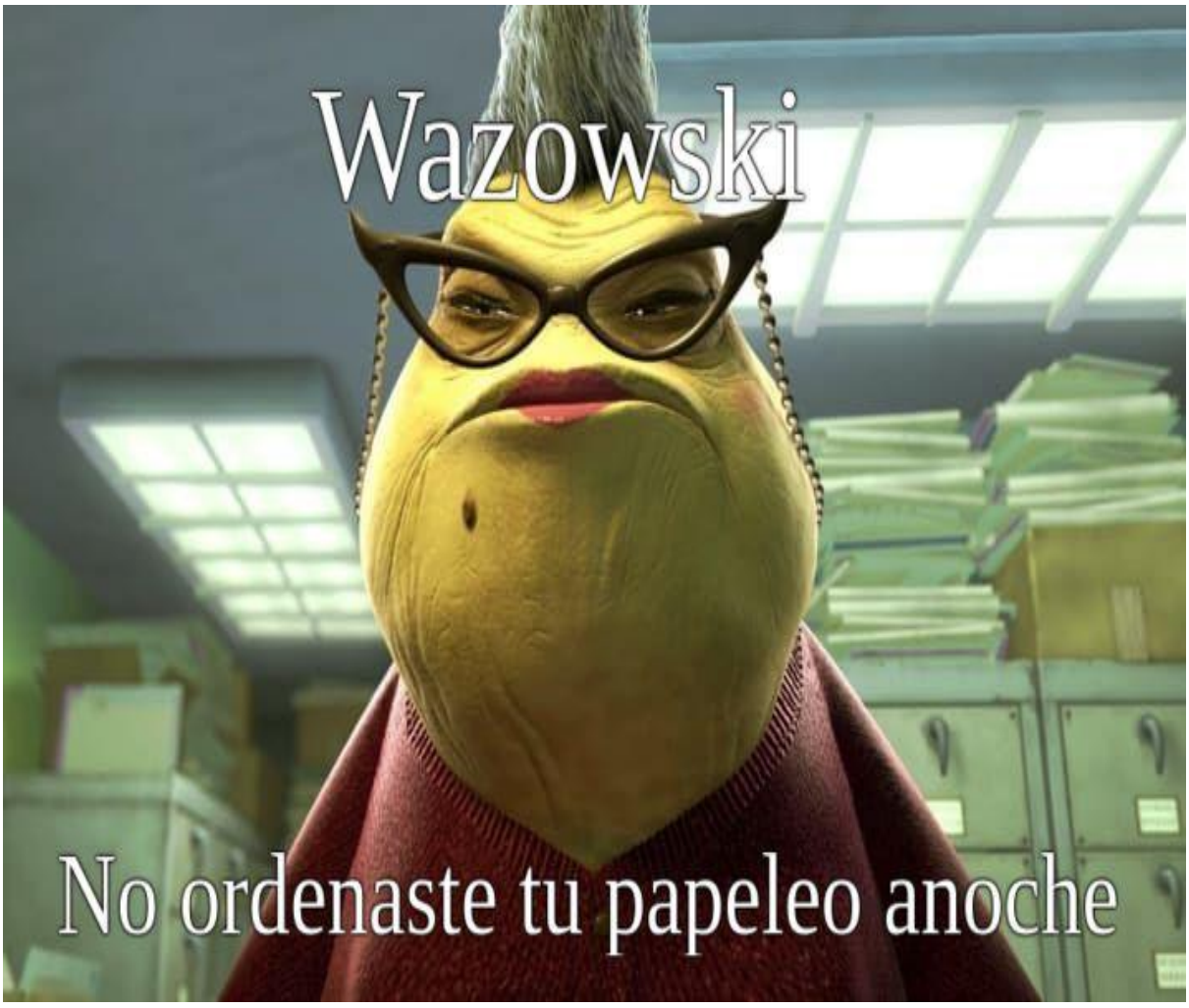
Algoritmos y Programación I

Curso Camejo

.UBAfiuba
FACULTAD DE INGENIERÍA

Trabajo Práctico N° 2

No ordenaste tu papeleo anoche



Fecha Presentación	28/04/2021
Fecha Entrega	12/05/2021

1. Introducción

Luego de todos los problemas que tuviste para llegar a FIUBA, finalmente pudiste rendir el temible exámen. Ahora llegó el momento de firmar la libreta, pero te diste cuenta que el día que está el profesor vos no podes ir. Te acordas que un amigo tuyo también tiene que ir, y le pedis si te hace el favor de llevar tu libreta a firmar.

Tu amigo, **Mike Wazowsi**, acepta el pedido, pero como buen interesado que es, lo hace a cambio de un obsequio.

Al llegar al Departamento de Computación, Mike se encuentra con **Roz**, quien, una vez más, le recrimina que **"no ordenó su papeleo anoche..."**. Y en venganza, repartió todos los papeles por distintas habitaciones para que se le haga más difícil a Mike llenarlos. Y, lamentablemente, Roz no le va a firmar la libreta hasta que él junte todo su papeleo.

Sin posibilidad de volver a evadir a Roz, llegó el momento de que Mike llene su papeleo, y nosotros vamos a acompañarlo en esta aventura.

2. Objetivo

El presente trabajo práctico tiene como objetivo evaluar a los alumnos en aspectos fundamentales de la programación. Estos aspectos son:

- Validación de datos ingresados por el usuario.
- Diseño y desarrollo de funcionalidades de una biblioteca con un contrato preestablecido.
- El correcto uso de estructuras de control.
- Tipos de dato simples y estructurados.
- Buenas prácticas de programación.
- Modularización.

3. Enunciado

Como desarrolladores de este juego, debemos ayudar a Mike a juntar todo su papeleo y llevárselo a Roz para que finalmente tu libreta sea firmada.

El juego consiste en pasar por una serie de raras habitaciones, para que Mike junte el papeleo en orden, que estará disperso en ellas.

Habrá 3 habitaciones donde buscar. Dichas habitaciones, estarán delimitadas por los bordes del terreno, y además habrá paredes, obstáculos y herramientas para ayudarte, o perjudicarte, en tu búsqueda.

Mike tendrá distinta cantidad de movimientos iniciales disponibles en cada nivel, y los movimientos que le sobren del nivel anterior, se sumarán a los iniciales del próximo nivel.

Pero cuidado! La habitación va a rotar con tus movimientos.

3.1. Orden de inicialización

El orden de inicialización de los elementos del juego deberá ser:

1. Paredes.
2. Objetos (obstáculos y herramientas).
3. Papeleos.
4. Personaje.

En esta primera etapa de desarrollo, deberás inicializar toda la estructura del juego para que quede lista para la próxima parte, y poder empezar con la búsqueda.

3.2. Obstáculos

Los obstáculos son elementos que estarán en cualquier posición del mapa e intentarán dificultar la tarea de Mike.

- **Fuegos**
- **Medias**

Todos los obstáculos empezarán en posiciones aleatorias, y en el caso de los fuegos próximos a una pared (adyacentes no diagonales). Cabe destacar que no pueden posicionarse distintos obstáculos en la misma posición que otro objeto, o personaje.

El orden de posicionamiento de los obstáculos al inicializar el juego es indiferente, siempre y cuando se respete lo enunciado en el párrafo anterior.

3.3. Herramientas

Las herramientas son elementos que ayudarán a Mike a recolectar todo su papeleo.

- **Extintores**
- **Martillos**
- **Botellas de gritos**
- **Interruptores 'ahuyenta Randall'**

Los extintores y martillos son herramientas que estarán presentes junto a Mike en su mochila, es decir que no tendrán una posición asociada. Para el caso de las demás herramientas deben posicionarse aleatoriamente al inicializar el juego. Cabe destacar que no pueden posicionarse distintas herramientas en la misma posición o en una posición donde ya existe un obstáculo, o en la posición de Mike. Los interruptores "ahuyenta Randall" comienzan inicialmente desactivados.

El orden de posicionamiento de las herramientas al inicializar el juego es indiferente, siempre y cuando se respete lo enunciado en el párrafo anterior.

3.4. Obsequio

Como se mencionó anteriormente, Mike solo nos ayudará a cambio de un obsequio, y éste dependerá del personaje que obtuvimos en el trabajo práctico 1.

- **Jasmín**: Obsequia un martillo extra en cada nivel, gracias al genio de la lámpara.
- **Rayo Mcqueen**: Obsequia 10 movimientos extras al comenzar el primer nivel, gracias a su extrema velocidad.
- **Stitch**: Obsequia una de sus pistolitas con una única munición, la cual quema un papeleo del último nivel.
- **Olaf**: Con su naturaleza helada, extingue dos fuegos del primer nivel, y uno del segundo nivel.

3.5. Niveles

Habrá 3 niveles a lo largo de todo el juego. En cada nivel habrá distinta cantidad de herramientas, obstáculos y movimientos.

3.5.1. Nivel 1

- Dimensión: 12x12
- Papeleos: 2
- Cantidad de movimientos iniciales: 40
- Cantidad obstáculos:
  - **Fuegos**: 10
  - **Medias**: 5
- Cantidad herramientas recolectables:

    - **Botellas de gritos**: 4
    - **Interruptores 'ahuyenta Randall'**: 1
- Cantidad herramientas personaje:
    - **Martillos**: 4
    - **Extintores**: 4

3.5.2. Nivel 2

- Dimensión: 17x17
- Papeleos: 3
- Cantidad de movimientos iniciales: 30
- Cantidad obstáculos:
    - **Fuegos**: 5
    - **Medias**: 4
- Cantidad herramientas:
    - **Botellas de gritos**: 3
    - **Interruptores 'ahuyenta Randall'**: 1
- Cantidad herramientas personaje:
    - **Martillos**: 5
    - **Extintores**: 2

3.5.3. Nivel 3

- Dimensión: 22x22
- Papeleos: 4
- Cantidad de movimientos iniciales: 20
- Cantidad obstáculos:
    - **Fuegos**: 3
    - **Medias**: 3
- Cantidad herramientas:
    - **Botellas de gritos**: 2
    - **Interruptores 'ahuyenta Randall'**: 0
- Cantidad herramientas personaje:
    - **Martillos**: 6
    - **Extintores**: 2

4. Especificaciones

Para poder lograr ayudar a Mike con su papeleo, se te pedirá implementar algunas funciones y procedimientos.

4.1. Funciones y procedimientos

```h
#ifndef __PAPELEO_H__
#define __PAPELEO_H__

#include "utiles.h"

#define MAX_OBSTACULOS 100
#define MAX_HERRAMIENTAS 100
#define MAX_PAPELEOS 10
#define MAX_NIVELES 3

typedef struct papeleo{
    coordenada_t posicion;
    int id_papeleo;
    bool recolectado;
} papeleo_t;

typedef struct objeto{
    coordenada_t posicion;
    char tipo;
} objeto_t;

typedef struct nivel{
    coordenada_t paredes[MAX_PAREDES];
    int tope_paredes;
    objeto_t obstaculos[MAX_OBSTACULOS];
    int tope_obstaculos;
    objeto_t herramientas[MAX_HERRAMIENTAS];
    int tope_herramientas;
    papeleo_t papeleos[MAX_PAPELEOS];
    int tope_papeleos;
    coordenada_t pos_inicial_jugador;
}nivel_t;

typedef struct jugador{
    coordenada_t posicion;
    int movimientos;
    int martillos;
    int extintores;
    bool ahuyenta_randall;
}jugador_t;

typedef struct juego{
    int nivel_actual;
    nivel_t niveles[MAX_NIVELES];
    jugador_t jugador;
    char personaje_tp1;
}juego_t;

/*
 * Procedimiento que recibe el juego e imprime toda su información por pantalla.
 */
void imprimir_terreno(juego_t juego);

/*
 * Inicializará un nivel, cargando toda la información inicial, las paredes,
 * los objetos, los papeleos y la posición inicial del jugador en dicho nivel.
 */
void inicializar_nivel(nivel_t* nivel, int numero_nivel, char amigo_pide_favor);

/*
 * Inicializará el juego, cargando toda la información inicial, los datos del jugador,
 * el personaje resultado del tp anterior, y los 3 niveles.
 */
void inicializar_juego(juego_t* juego, char amigo_pide_favor);

#endif /* __PAPELEO_H__ */
```

Observación: Queda a critero del alumno/a el hacer o no, más funciones y/o procedimientos para resolver los problemas presentados. No se permite agregar dichas firmas al .h.

4.2. Convenciones

Se deberá utilizar la siguiente convención para los elementos que tengan una coordenada asociada como obstáculos, algunas herramientas y para el personaje:

- **Fuegos**: F.
- **Medias**: M.
- **Botellas de gritos**: G.
- **Interruptores 'ahuyenta Randall'**: I.
- **Mike**: W.

5. Resultado esperado

Se espera que el trabajo creado cumpla con las buenas prácticas de programación y todas las funciones y procedimientos pedidos funcionen acorde a lo solicitado, respetando las pre y post condiciones propuestas.

6. Compilación y Entrega

El trabajo práctico debe ser realizado en un archivo llamado papeleo.c, lo que sería la implementación de la biblioteca papeleo.h. Además, deberán crear una bilioteca propia con lo resuelto en el trabajo práctico 1, y deberá realizarse el cuestionario previo al juego. El trabajo debe ser compilado sin errores al correr desde la terminal el comando:

```
gcc *.c utiles.o -o juego -std=c99 -Wall -Wconversion -Werror -lm
```

utiles.o es un archivo compilado realizado por la cátedra, que pondrá a su disposición funciones que son necesarias para el armado del juego.

Lo que nos permite ***.c** es agarrar todos los archivos que tengan extensión .c en el directorio actual y los compile todos juntos. Esto permite que se puedan crear bibliotecas a criterio del alumno, aparte de la exigida por la cátedra.

Por último, debe ser entregado en la plataforma de corrección de trabajos prácticos **Chanutron2021** (patente pendiente), en la cual deberá tener la etiqueta **¡Exito!** significando que ha pasado las pruebas a las que la cátedra someterá al trabajo.

Para la entrega en **Chanutron2021** (patente pendiente), recuerde que deberá subir un archivo **zip** que contenga únicamente los archivos antes mencionados, sin carpetas internas ni otros archivos. De lo contrario, la entrega no será validada por la plataforma.

IMPORTANTE! Obtener la etiqueta **¡Exito!** en **Chanutron2021** (patente pendiente) no implica necesariamente haber aprobado el trabajo. El trabajo será corregido por un colaborador que verificará que se cumplan las buenas prácticas de programación.

7. Anexos

7.1. Cálculo de valor absoluto

Para obtener el valor absoluto de un valor se puede utilizar la función **abs**.

```c
#include <stdio.h>
#include <stdlib.h> //Para poder usar abs

int main(){
    int valor = -5;
    int valor_absoluto = abs(valor); //se obtiene como retorno un 5

    return 0;
}
```

7.2. Obtención de números aleatorios

Para obtener números aleatorios debe utilizarse la función **rand()**, la cual está disponible en la biblioteca stdlib.h.

Esta función devuelve números pseudo-aleatorios, esto quiere decir que, cuando uno ejecuta nuevamente el programa, los números, aunque aleatorios, son los mismos.

Para resolver este problema debe inicializarse una semilla, cuya función es determinar desde donde empezarán a calcularse los números aleatorios.

Los números arrojados por **rand()** son enteros sin signo, generalmente queremos que estén acotados a un rango (queremos números aleatorios entre tal y tal). Para esto, podemos obtener el resto de la división de **rand()** por el valor máximo del rango que necesitamos.

Aquí dejamos un breve ejemplo de como obtener números aleatorios entre 10 y 29 (inclusives).

```c
#include <stdio.h>
#include <stdlib.h> // Para usar rand
#include <time.h>   // Para obtener una semilla desde el reloj

int main(){
        srand ((unsigned)time(NULL)); // Genera la semilla aleatoria.
        int numero = rand() % 20 + 10; // La amplitud del rango es 20 y el valor mínimo es 10.
        printf("El valor aleatorio es: %i\n", numero);

        return 0;
}
```

7.3. Distancia Manhattan

Para obtener la distancia entre 2 puntos mediante este método, se debe conocer a priori las coordenadas de dichos puntos.

Luego, la distancia entre ellos es la suma de los valores absolutos de las diferencias de las coordenadas. Se ve claramente en los siguientes ejemplos:

- La distancia entre los puntos (0,0) y (1,1) es 2 ya que: $|0 - 1| + |0 - 1| = 1 + 1 = 2$
- La distancia entre los puntos (10,5) y (2,12) es 15 ya que: $|10 - 2| + |5 - 12| = 8 + 7 = 15$
- La distancia entre los puntos (7,8) y (9,8) es 2 ya que: $|7 - 9| + |8 - 8| = 2 + 0 = 2$

Observación: En este TP no están obligados a usar la distancia manhattan, sino que sólo se menciona como una posible herramienta.

7.4. Limpiar la pantalla durante la ejecución de un programa

Muchas veces nos gustaría que nuestro programa pueda verse siempre en la pantalla sin ver texto anterior.

Para ésto, podemos utilizar la llamada al sistema **clear**, de esta manera, limpiaremos todo lo que hay en nuestra terminal hasta el momento y podremos dibujar la información actualizada.

Y se utiliza de la siguiente manera:

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
        printf("Escribimos algo\n");
        printf("que debería\n");
        printf("desaparecer...\n");

        system("clear"); // Limpiamos la pantalla

        printf("Solo deberiamos ver esto...\n");
        return 0;
}
```