.UBAfiuba
FACULTAD DE INGENIERÍA

ALGORITMOS Y PROGRAMACIÓN I - CURSO CAMEJO

Trabajo Práctico Nº 2

Over-chambuchito

Fecha presentación	20/04/2023
Fecha entrega	18/05/2023

Primer cuatrimestre 2023

1. Introducción

Reuben no se esperaba para nada tener la cantidad de pedidos que le están haciendo y se le está complicando bastante, se ve que vende los chambuchitos demasiado baratos... Para poder completar los pedidos que le faltan tuvo que ir a otra sucursal para buscar stock y poder hacer los platos para sus clientes. Además, le pidió ayuda a su amigo Stitch, pero este le dijo que iba a ayudarlo solo a alcanzarle los vegetales y Reuben debería hacer el resto del trabajo.

Parecía que estaba todo encaminado para empezar a trabajar, pero Stitch y Reuben se encontraron con otro problema: tienen la lista de pedidos pero no saben leer, así que decidieron que van a cocinar los platos en base al dinero recaudado.

Luego de decidir esto y dividir las tareas, los experimentos se pusieron manos a la obra.

2. Objetivo

El presente trabajo práctico tiene como objetivo que el alumno:

- Diseñe y desarrolle las funcionalidades de una biblioteca con un contrato preestablecido.
- Se familiarice con y utilice correctamente los tipos de datos estructurados.
- Desarrolle una interfaz gráfica amigable y entendible para el usuario.

Por supuesto, se requiere que el trabajo cumpla con las buenas prácticas de programación profesadas por la cátedra.

Se considerarán críticos la modularización, reutilización y claridad del código.

3. Enunciado

Como desarrolladores de este juego, debemos ayudar a Reuben a armar todos los pedidos posibles con el dinero disponible.

El juego consiste en una cocina dividida en dos cuadrantes horizontales, donde en la mitad superior estará Stitch que se encargara de cortar ingredientes y de lavar los platos, y del lado inferior estará Reuben cocinando y emplatando. Para pasar los ingredientes de un lado a otro, en la franja horizontal divisoria, habrá una mesa para apoyar.

La cocina tendrá distintas estaciones con las herramientas a utilizar para llevar a cabo el pedido.

Se iniciará el juego siendo Stitch, y el pedido se considerará completo una vez que todos los ingredientes se posicionen en la puerta de salida, cortados y cocinados.

Stitch deberá recolectar uno por uno los ingredientes de su cuadrante, cortarlos cuando sea necesario y luego dejarlos en la mesa para que Reuben los recolecte, los cocine y los entregue.

Para inicializar, primero deberan ubicarse las paredes, luego los obstaculos, las herramientas, los ingredientes y por último los personajes y la puerta de salida.

3.1. Obstáculos

Los obstáculos deben posicionarse aleatoriamente al inicializar el juego. Cabe destacar que no pueden posicionarse distintos obstáculos en la misma posición que otro objeto, pared, o personaje.

3.1.1. Agujeros

En el piso habrá 20 agujeros (10 en cada cuadrante) que provocaran que el personaje se caiga, lo que hará que pierda el juego.

3.1.2. Fuego

Cada 15 movimientos, aparecerá un fuego en un lugar aleatorio en uno de los dos cuadrantes disponibles. El personaje que este en el mismo cuadrante que el fuego, no podrá realizar otra acción hasta que agarre el matafuego y lo active en una distancia manhattan menor o igual 2 del fuego. El contador dejará de contar, y empezará nuevamente en 0 una vez que se apague. No apareceran otros fuegos si ya esta activo uno. Una vez eliminado, se deberá eliminar del vector obstáculos.

3.2. Herramientas

En cada cuadrante, habrá herramientas que permitaran cocinar a los personajes.

Las herramientas deben posicionarse de forma aleatoria al inicio del juego, y no deben estar en la misma posición que otro objeto.

3.2.1. Matafuegos

Al mismo tiempo que aparezca el fuego, deberá aparecer en el mismo cuadrante un matafuegos. Para agarrarlo, se deberá estar en la misma posición del mismo y éste deberá ser eliminado del terreno. Para poder activarlo, se deberá apretar la letra **M**. Una vez usado, se deberá eliminar del vector herramientas.

3.2.2. Cuadrante Stitch

Habrá 2 cuchillos ubicados en el cuadrante de Stitch, que para usarlos, será necesario tener en la mano el alimento que se quiere cortar y apretar la letra **C**.

3.2.3. Cuadrante Reuben

Habrá dos hornos para cocinar los ingredientes que lo requieran. Para usarlos, se deberá estar a una distancia manhattan de 1 y se usará la letra **H** También, habra una puerta de salida donde habrá que armar el plato para entregar.

3.3. Comidas

Se tendrá un vector de comidas que deberá ser inicializado antes de comenzar el juego. Cada comida tiene diferentes ingredientes que se deben inicializar en posiciones aleatorias del terreno y no deben estar en la misma posición que otro objeto. Es requisito que las comidas se inicialicen en éste orden: Ensalada, Pizza, Hamburguesa, Sandwich. En el terreno solo se deberán mostrar los ingredientes que se necesitan para la comida actual que se está armando. Para agarrar cada ingrediente, es necesario estar parado en la misma posición y apretar la letra **R**. Cuando esto pase, el ingrediente deberá eliminarse del terreno. Si se aprieta la **R** teniendo un ingrediente en mano, este deberá quedar en la posición actual del personaje.

3.4. Ensalada

La ensalada necesitará de 1 lechuga y 1 tomate. Ambos se inicializarán en el cuadrante de Stitch.

3.5. Pizza

La pizza necesitará una masa, 1 jamón y 1 queso. El jamón y el queso requieren cortarse por lo que se inicializarán del lado de Stitch, mientras que la masa deberá ser cocinada, por lo que se inicializará del lado de Reuben.

3.6. Hamburguesa

Los ingredientes de la hamburguesa serán pan, carne, 1 lechuga y 1 tomate. Las verduras y el pan deberán ir en el cuadrante de Stitch para poder cortarse, mientras que la carne estará en el cuadrante de Reuben para poder cocinarse.

3.7. Sandwich

El sandwich llevara 1 milanesa que deberá cocinarse, por lo que irá del lado de Reuben. Mientras que el pan, el tomate, la lechuga, el jamón y el queso, iran del lado de Stitch y precisarán cortarse.

Importante! Para poder pasarle los ingredientes a Rouben, se deberan cortar y dejarlos en la mesa del medio. Para esto, es necesario estar a una distancia manhattan de 1 y apretar la letra **T**.

Si un ingrediente ya se encuentra en la mesa, no podrá ubicarse otro.

3.8. Terreno

El terreno será representado con una matriz de 21x21, dividida en 2 cuadrantes con una mesa en el medio, de la siguiente forma:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
1	#																				#
2	#																				#
3	#					C	u	a	d	r	a	n	t	e							#
4	#																				#
5	#							S	t	i	t	c	h								#
6	#																				#
7	#																				#
8	#																				#
9	#																				#
10	#	#	#	#	#	#	#	#	#	#	_	#	#	#	#	#	#	#	#	#	#
11	#																				#
12	#																				#
13	#					C	u	a	d	r	a	n	t	e							#
14	#																				#
15	#							R	e	u	b	e	n								#
16	#																				#
17	#																				#
18	#																				#
19	#																				#
20	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#

3.9. Modo de juego

Los personaje se podrán mover en 4 direcciones:

- **Arriba**: W
- **Abajo**: S
- **Derecha**: D
- **Izquierda**: A

Para **cambiar de personaje** se usará la letra **X**.

Habrá un plato por turno, esto quiere decir que para pasar al siguiente plato, es necesario que el anterior este terminado, por lo que solo se deberá mostrar la comida que se esta armando en el momento. Para armar un plato, es necesario llevar cada ingrediente a la puerta de salida. Los ingredientes previamente tienen que ser procesados (cortados y/o cocinados). En caso contrario, no podrán dejarse en el plato. Para dejarlo en el plato, se requiere agregar el ingrediente al vector comida_lista. Una vez que esten todos los ingredientes, se podrá seguir con la siguiente comida.

Para ganar el juego, Stitch y Reuben deberán completar los platos dependiendo de cuanta plata tienen disponible:

- **Hasta $100 (incluido)**: Ensalada, pizza.

- **Hasta $150 (incluido)**: Ensalada, pizza, hamburguesa.
- **Más de $150**: Ensalada, pizza, hamburguesa, Sandwich

4. Especificaciones

4.1. Convenciones

Se deberá utilizar la siguiente convención para los obstáculos:

- **Fuego**: F.
- **Agujeros**: A.

Para las herramientas:

- **Matafuegos**: M.
- **Cuchillos**: C.
- **Horno**: H.
- **Puerta de salida**: P.

Para los ingredientes:

- **Lechuga**: L.
- **Tomate**: T.
- **Milanesa**: I.
- **Carne**: B.
- **Pan**: N.
- **Jamón**: J
- **Queso**: Q.
- **Masa**: O.

Para las comidas:

- **Ensalada**: E.
- **Pizza**: P.
- **Hamburguesa**: H.
- **Sandwich**: S.

Y para los personajes:

- **Stitch**: S.
- **Reuben**: R.

4.2. Biblioteca chambuchito.h

Se debe crear una biblioteca con el trabajo práctico 1, ésta biblioteca solo tendrá un procedimiento con la siguiente firma:

```
1 void calcular_precio_chambuchito(int* precio);
```

4.3. Funciones y procedimientos

Para poder cumplir con los pedidos, se pedirá implementar las siguientes funciones y procedimientos.

```c
#ifndef __COCINERITOS_H__
#define __COCINERITOS_H__

#include <stdbool.h>

#define MAX_INGREDIENTES 10
#define MAX_PAREDES 200
#define MAX_HERRAMIENTAS 50
#define MAX_OBSTACULOS 50
#define MAX_COMIDA 5

typedef struct coordenada{
        int fil;
        int col;
}coordenada_t;

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

typedef struct ingrediente{
        coordenada_t posicion;
        char tipo;
        bool esta_cortado;
        bool esta_cocinado;
}ingrediente_t;

typedef struct comida{
        ingrediente_t ingrediente[MAX_INGREDIENTES];
        int tope_ingredientes;
        char tipo;
}comida_t;

typedef struct personaje{
        coordenada_t posicion;
        char tipo;
        char objeto_en_mano;
}personaje_t;

typedef struct juego{
        personaje_t stitch;
        personaje_t reuben;
        char personaje_activo; //S / R
        char comida_actual;
        coordenada_t paredes[MAX_PAREDES];
        int tope_paredes;
        objeto_t herramientas[MAX_HERRAMIENTAS];
        int tope_herramientas;
        coordenada_t salida;
        coordenada_t mesa;
        objeto_t obstaculos[MAX_OBSTACULOS];
        int tope_obstaculos;
        comida_t comida[MAX_COMIDA];
        int tope_comida;
        ingrediente_t comida_lista[MAX_INGREDIENTES];
        int tope_comida_lista;
        int precio_total;
        int movimientos;
}juego_t;

/*
 * Inicializará el juego, cargando toda la información inicial de las comidas, personajes,
    herramientas y obstáculos.
 * El precio corresponde al precio obtenido en el TP1.
 */
void inicializar_juego(juego_t* juego, int precio);

/*
 * Moverá al personaje correspondiente y se realizarán todas las acciones necesarias.
 *
 */
void realizar_jugada(juego_t* juego, char movimiento);
```

```c

/*
 * Imprime el juego por pantalla
 */
void imprimir_terreno(juego_t juego);

/*
 * El juego se dará por ganado si los personajes cumplen con los platos pedidos. Si alguno de los personajes cae por
 * un agujero, se considerará perdido.
 * Devuelve:
 * --> 1 si es ganado
 * --> -1 si es perdido
 * --> 0 si se sigue jugando
 */
int estado_juego(juego_t juego);

#endif /* __COCINERITOS_H__ */
```

Observación: Queda a criterio del alumno/a hacer o no más funciones y/o procedimientos para resolver los problemas presentados. No se permite agregar funciones al .h presentado por la cátedra, como tampoco modificar las funciones dadas.

5. Resultado esperado

Se espera que se creen las funciones y procedimientos para que el juego se desarrolle con fluidez. Así mismo, se espera que se respeten las buenas prácticas de programación.

Muchas de las funcionalidades quedan a criterio del alumno, solo se pide que se respeten las estructuras y especificaciones brindadas.

El trabajo creado debe:

- Interactuar con el usuario.
- Mostrarle al usuario de forma clara el terreno y todos sus elementos.
- Mostrarle al jugador la información del juego a cada momento.
- Informarle al jugador correctamente cualquier dato que haya sido ingresado incorrectamente.
- Informarle al jugador si ganó o perdió.
- Cumplir con las buenas prácticas de programación.
- Mantener las estructuras propuestas actualizadas a cada momento.

6. Compilación y Entrega

El trabajo práctico debe ser realizado en un archivo llamado cocineritos.c, lo que sería la implementación de la biblioteca cocineritos.h. Además, deberán crear una biblioteca con lo resuelto en el trabajo práctico 1. El trabajo debe ser compilado sin errores al correr el siguiente comando:

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

Aclaración: El main tiene que estar desarrollado en un archivo llamado juego.c, el cual manejará todo el flujo del programa.

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

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

IMPORTANTE! Esto no implica necesariamente haber aprobado el trabajo ya que además será corregido por un colaborador que verificará que se cumplan las buenas prácticas de programación.

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

7. FAQS

Compartimos un documento con las preguntas más comúnes sobre el trabajo práctico:

8. Anexos

8.1. 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 mismo.

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 ésto, 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 30.

```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));
        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;
}
```

8.2. 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;
}
```

8.3. 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: $|7 - 9| + |8 - 8| = 2 + 0 = 2$