***TPN°7: Programación Avanzada y Uso de Heap***

**A fin de asegurarse de usar y liberar correctamente la memoria reservada, agregar el siguiente flag al compilar:**

**-fsanitize=address**

**Se recomienda agregar el flag -g para obtener una salida más precisa sobre errores encontrados**

**Este flag no es soportado por MinGW**

# ***Ejercicio 1***

1. Generar el ejecutable para el siguiente programa, primero sin usar el flag **-fsanitize=address** y verificar que funcione.
2. Volver a generar el ejecutable pero con el flag. ¿Qué sucede al ejecutarlo? ¿A qué se deben los errores? ¿Es fácil viendo los errores detectar el origen de los mismos?
3. Vuelva a compilar agregando además el flag **-g** y ejecute nuevamente. Ahora debería poder ver el origen de algunos errores. Corregir el programa para que funcione sin errores.

| #include <stdlib.h>  #include <stdio.h>  int  main(void) {  char \* p = malloc(10);  for (int i=0; i <= 10; i++)  p[i] = i \* 10;  for(int i=0; i <= 10; i++)  printf("%d\n", p[i]);  return 0;  } |
| --- |

# ***Ejercicio 2***

Escribir un programa que convierta enteros de una base a otra. Los números entre y después de los símbolos '<' y '>' indican la base de entrada y de salida respectivamente. Dichos números serán decimales entre 2 y 10.

*Ejemplo: la cadena de entrada* ***< 5 > 2***  *indica que leerá números enteros en base 5 y los imprimirá en su equivalente binario. Luego, si se ingresa* ***41*** *se lo deberá pasar a* ***10101.***

***Ejercicio 3***

¿ Qué hace este programa ?



***Ejercicio 4***

¿ Qué hace este programa ?



# ***Ejercicio 5***

Sean Nombre y Curso dos arreglos que contienen el nombre y el curso( ‘G’ o ‘H’) de cada uno de los alumnos de una materia. Escribir una función que reciba dichos arreglos como parámetros de entrada y retorne en parámetr2os de salida otros dos arreglos CursoG y CursoH con el nombre de los alumnos separados por curso. El string vacío en el nombre indica fin del arreglo.

# ***Ejercicio 6***

Escribir un programa para desarrollar el juego Mastermind descrito por las siguientes reglas:

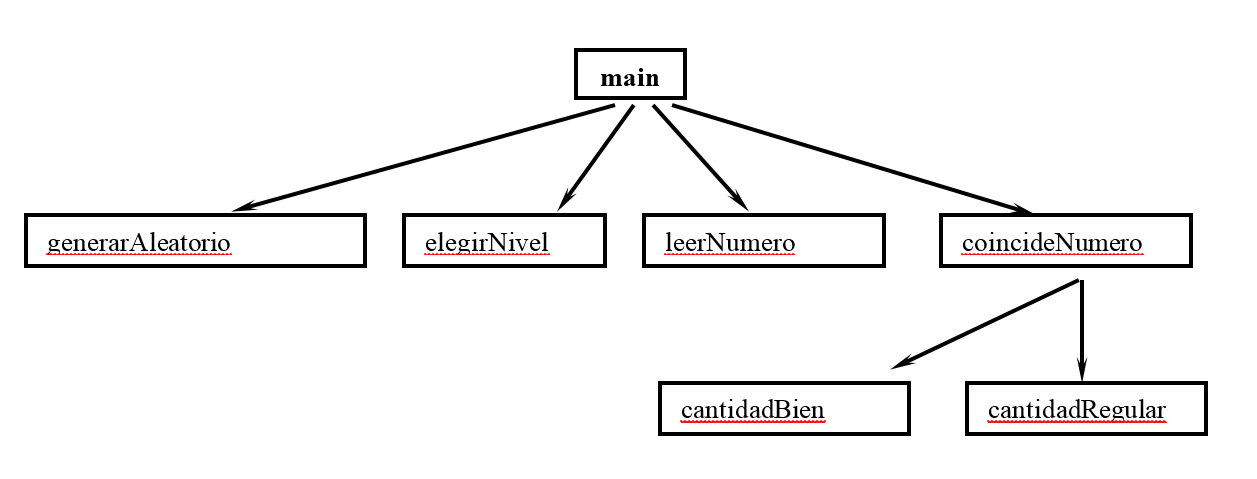
* La computadora construye un conjunto aleatorio de X dígitos distintos (para X usar una constante simbólica). Sólo se usan los dígitos del 1 al 9
* Se le pide ingresar al usuario el nivel en que quiere jugar. Según esto se trabajará con una tolerancia de hasta N pasos, dentro de los cuales si no logra ganar el juego se termina automáticamente.
* El jugador entra un conjunto de X dígitos distintos por teclado.
* Si el número propuesto por el jugador coincide con el que generó la máquina o la cantidad de chances excedió el número tope N , el juego finaliza; sino el programa debe informarle cuántos dígitos están BIEN (son correctos y están en su lugar) y cuántos son REGULARES (son correctos pero están fuera de lugar) y se vuelve a ejecutar el paso anterior. Así se va guiando al jugador hasta que adivine el número.

***Ejemplo:***

*Para* ***X = 5*** *y* ***N = 3****, suponiendo que el número escondido por la computadora fuera 98532, un diálogo posible sería:*

* *número introducido por el jugador:* 1**952**4
* *respuesta de la computadora:* 1 bien, 2 regular
* *número introducido por el jugado:* **9**1**52**7
* *respuesta de la computadora:* 2 bien, 1 regular
* *número introducido por el jugado:*  **9852**6
* *respuesta de la computadora:* 3 bien, 1 regular

Lo siento, el número es **98532**

****

| ***Nombre*** | ***Descripción*** | ***Parámetros*** | |
| --- | --- | --- | --- |
| **main** | Permite jugar al MasterMind | No tiene | |
| **GeneraAleatorio (\*)** | Arma un arreglo de X elementos no repetidos | **incognita** | Arreglo de enteros  De salida  V.F.: enteros positivos no repetidos |
| **elegirNivel** | Devuelve un número entero positivo entre uno y diez | No tiene | |
| **leerNumero** | Solicita al usuario ingresar un número de X dígitos no repetidos | **numero** | Arreglo de enteros  De salida  V.F.: enteros positivos no repetidos |
| **coincideNumero** | Dados dos arreglos de enteros, devuelve un valor booleano indicando si ambos arreglos son exactamente iguales o no. Imprime un cartel indicando cuántos elementos coinciden en valor y posición (BIEN) y cuántos sólo en valor (REGULAR). | **incognita** | Arreglo de enteros |
| De entrada |
| V.I.: enteros positivos no repetidos |
| **numero** | Arreglo de enteros  De entrada  V.I.: enteros positivos no repetidos |
| **cantidadBien** | Dados dos arreglos de enteros, devuelve cuántos de sus elementos coinciden en valor y posición | **incognita** | Arreglo de enteros  De entrada  V.I: enteros positivos no repetidos |
| **numero** | Arreglo de enteros  De entrada  V.I: enteros positivos no repetidos |
| **cantidadRegular** | Dados dos arreglos de enteros, devuelve cuántos de sus elementos coinciden en valor pero no en posición | **incognita** | Arreglo de enteros |
| De entrada |
| V.I.: enteros positivos no repetidos |
| **numero** | Arreglo de enteros  De entrada  V.I.: enteros positivos no repetidos |

(\*): en esta función es bastante común que el algoritmo para generar X dígitos aleatorios sin repetir sea el siguiente:

1. Crear un vector auxiliar de 9 chars en cero
2. Para i desde 0 hasta X-1 hacer
   1. Elegir un dígito al azar entre 1 y 9
   2. Si ya estaba marcado en el vector auxiliar volver al punto 2.a
   3. Marcarlo en el vector auxiliar
   4. Guardar el dígito en la posición i del vector incognita

El prob1lema con este algoritmo es que si X es cercano a 9 (por ejemplo 7), a medida que avanza i se incrementa la posibilidad de que el dígito ya haya salido, y algo similar pasará con el juego del bingo: las primeras bolillas se mostrarán rápidamente pero a medida que avanza el juego tardará más tiempo es mostrar las nuevas bolillas, ya que el algoritmo planteado arriba sería similar a: sacar una bolilla del bolillero, si no había salido entonces anunciarla, volver`` a meter la bolilla en el bolillero.

Se debería encontrar un algoritmo que asegure que el siguiente dígito no sea uno de los que ya salieron, y lo mismo al sacar una bolilla en el bingo.

# ***Ejercicio 7***

Rehacer el problema del ejercicio anterior, pero considerando que la cantidad X de dígitos sea elegida por el usuario. ¿Cambia el árbol de módulos? ¿Cambia la tabla de interfaces? (edicion javi paja hacerlo) con malloc y realloc.

# ***Ejercicio 8***

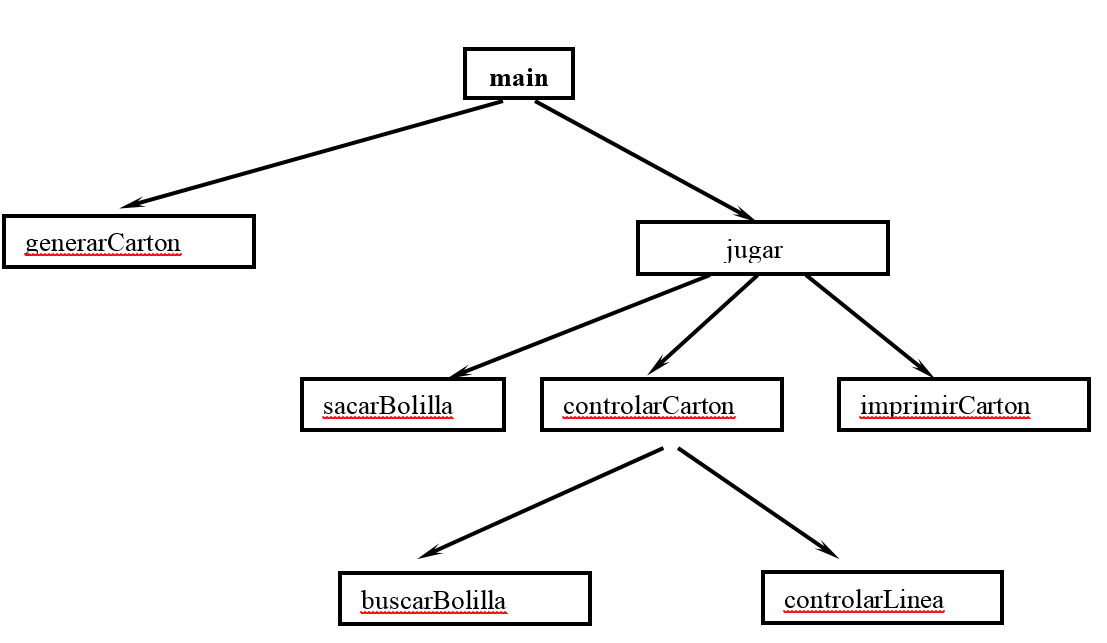
Escribir un programa que realice la simulación de un bingo para dos jugadores, el cual imprimirá paso a paso la bolilla extraída y los números faltantes (que aún no salieron)de cada cartón. Las bolillas están representadas por los números del 1 al 90 y cada jugador tiene un cartón con 15 números distintos. Los cartones son generados por el programa en forma aleatoria. No es necesario conservar los números que se van marcando en cada cartón (cuando hay coincidencia con la bolilla extraída)

El tipo de datos a utilizar para el cartón será :

**typedef int TipoLinea[5];**

**typedef TipoLinea TipoCarton [3];**

Se da a continuación un esquema del diseño del mismo.



| ***Nombre*** | ***Descripción*** | ***Parámetros*** | |
| --- | --- | --- | --- |
| **Main** | Simula un juego de bingo. Genera dos cartones, uno para cada jugador y va extrayendo bolillas hasta que alguno de los jugadores hace bingo. Imprime el o los ganadores. | No tiene | |
| **GenerarCarton** | Genera un cartón de bingo en forma aleatoria. | **carton** | ***TipoCarton***  De salida  V.F.: 15 enteros distintos entre 1 y 90 |
| **Jugar** | Va extrayendo bolillas hasta que alguno de los dos jugadores hace bingo. Con cada extracción se muestra el estado de los cartones e imprime el ganador de la primera línea.  Devuelve un número entero con un bit en uno por cada jugador que hizo bingo. | **bolillero** | Arreglo de enteros  De entrada / salida  V.I: vector de enteros con el bolillero inicial. |
| **jugador1, jugador2** | ***TipoCarton***  De entrada / salida  V.I: 15 números distintos entre 1 y 90, que representan el cartón de cada jugador. |
| **SacarBolilla** | Devuelve un número de bolilla aún no extraída y actualiza el bolillero y la cantidad de bolillas que aún quedan en el mismo. | **bolillero** | Arreglo de enteros  De entrada / salida  V.I: vector de enteros que representa las bolillas que aún no han sido extraídas.  V.F: ídem V.I. pero sin la bolilla extraída. |
| **cantBolillas** | Entero positivo  De entrada / salida  V.I: cantidad de bolillas aún no extraídas.  V.F: cantidad de bolillas que quedan en el bolillero. |
| **controlarCarton** | Dado un cartón y una bolilla extraída, verifica si pertenece al mismo, en cuyo caso lo marca y controla si se formó línea o bingo. Devuelve la cantidad de líneas completadas del cartón con esa bolilla. | **carton** | ***TipoCarton***  De entrada / salida  V.I: cartón con los números faltantes. |
| **bolilla** | Entero  De entrada  V.I: número de bolilla extraída. |
| **imprimirCarton** | Imprime el estado de un cartón. Se imprimen por cada línea los números que no han sido marcados aún. | **carton** | ***TipoCarton***  De entrada  V.I: cartón con los números faltantes. |
| **buscarBolilla** | Dado un cartón y la bolilla extraída la busca entre los números restantes y si la encuentra lo marca y devuelve 1 (verdadero), en caso contrario devuelve 0 (falso). | **carton** | ***TipoCarton***  De entrada / salida  V.I: cartón con los números faltantes. |
| **bolilla** | Entero  De entrada  V.I: número de bolilla extraída |
| **controlarLineas** | Recibe una línea del cartón y devuelve 1 si la misma está completa (tiene todos sus números ya marcados) y 0 en caso contrario. | **linea** | ***TipoLinea***  De entrada  V.I.: línea, con nros marcados |

# ***Ejercicio 9***

Rehacer el problema del ejercicio anterior para N jugadores, siendo N:

1. una constante simbólica.
2. Un número ingresado por el usuario al iniciar el juego.

Realice la menor cantidad de cambios posible en la interfase de las funciones.

# ***Ejercicio 10***

Escribir un programa interactivo para jugar al juego del ahorcado. Las palabras que almacena la máquina deben estar guardadas en un arreglo de strings constantes y cada vez que se elija jugar, utilizar aleatoriamente alguna de dichas palabras.

# Antes de escribir el programa, realizar el diseño y la modularización, documentando cada módulo interviniente, tal como se mostró en los ejercicios 8 y 9.

***Ejercicio 11***

Indicar -de ser posible- cual exs la salida estándar en cada uno de los siguientes fragmentos de código, suponiendo que previamente se definió la función ***aMayusculas*** de la siguiente forma:





a)

b)

c) 

d)

| void resize(char \*p, unsigned int bytes) {  p = realloc(p, bytes);  }  int  main(void) {  char \*p=NULL;  resize(p, 100);  strcpy(p, "hello world");  aMayusculas(p);  printf("%s\n", p);  return 0;  } |
| --- |

***Ejercicio 12***

Dada la declaración:

**char \*p[2][3] = {{"abc","defg","hi"},{"jklmn","pqrstuvw","xyz"}};**

a) ¿Cuántos bytes de memoria ocupa la declaración?

b) Completar el siguiente cuadro:

| ***Expresión*** | ***Expresión Equivalente*** | ***Valor*** |
| --- | --- | --- |
| **\*\*\*p** | **p[0][0][0]** | **'a'** |
| **\*\*p[1]** |  |  |
| **\*\*(p[1]+2)** |  |  |
| **\*(\*(p+1)+1)[7]** |  |  |
| **(\*(\*(p+1)+1))[7]** |  |  |
| **\*(p[1][2]+2)** |  |  |

***Ejercicio 13***

Encontrar el error en cada uno de los segmentos de programa. De ser posible, explicar cómo corregirlo:

a)

| **int** \*numero;  printf ("%d\n", \*numero); |
| --- |

b)

| **float** \*realPtr;  **long** \*integerPtr;  integerPtr = realPtr; |
| --- |

c)

int \*x, y;

x = y ;

d)

| **char** s[] = "esta es una cadena de caracteres";  **for** ( ; \*s != '\0'; s++)  printf("%c ", \*s); |
| --- |

e)

| **short** \*numPtr, result, auxiliar=0;  **void** \*genericoPtr;  numPtr = &auxiliar;  genericoPtr = numPtr;  result = \* genericoPtr + 7; |
| --- |

f)

| **char \*s;  printf("%s\n", s);** |
| --- |

g)

| **char** \*s, \*aux;  s = malloc(20);  aux = realloc(s, 100);  free(s);  free(aux); |
| --- |
|  |

***Ejercicio 14***

Se tiene una función con la siguiente declaración:

**static int x[8] = { 10, 20, 30, 40, 50, 60, 70, 80 };**

1. ¿Qué se referencia con **x** ?
2. ¿Qué se referencia con **(x+2**)?
3. ¿Qué se referencia con **\*x** ?
4. ¿Qué se referencia con **\*x+2** ?
5. ¿Qué se referencia con **\*(x+2)** ?
6. Suponiendo que x[0] esté almacenado en la dirección de memoria $10F0, en qué dirección de memoria se encuentra almacenado **x[4]** ?

***Ejercicio 15***

Se tiene una función con la siguiente declaración:

**static float tabla[2][3] = { { 1.1, 1.2, 1.3 }, { 2.1, 2.2, 2.3 } };**

Graficar el almacenamiento en memoria del arreglo **tabla** (a partir de la dirección $1000) y decir qué tipo de dato es y qué se referencia con:

1. tabla
2. tabla+1
3. \*(tabla + 1)
4. \*(tabla + 1) + 1
5. \*tabla + 1
6. \*(\*(tabla + 1) + 1)
7. \*(\*tabla + 1)
8. \*(\*(tabla + 1))
9. \*(\*tabla + 1) + 1
10. tabla[1]
11. tabla[0] + 1
12. \*(tabla[1]) + 1
13. \*tabla[1]
14. tabla[0][1]

# ***Ejercicio 16***

Se tiene un programa con la siguiente declaración:

**static char \*color[6] = { “rojo”, “verde”, “azul”, “blanco”, “negro”, “amarillo” };**

1. Graficar el almacenamiento en memoria del arreglo **color** (a partir de $2004).
2. ¿Qué se referencia con **color**?
3. ¿Qué se referencia con **(color+2**)?

El resultado de (color+2) sería un nuevo puntero que apunta al tercer elemento del arreglo color, es decir, el elemento en la posición 2.(es importante remarcar que apunta).

1. ¿Qué se referencia con **\*color**?

Cuando se utiliza la expresión \*color, se está accediendo al primer elemento del arreglo color. El operador de desreferenciación \* se utiliza para acceder al valor al que apunta un puntero.

En este caso, \*color sería equivalente a color[0], es decir, está accediendo al primer puntero en el arreglo color. El valor al que apunta este puntero es la cadena de caracteres "rojo".

en este caso a diferencia del otro accede al valor.

1. ¿Qué se referencia con **\*(color + 2)**?
2. ¿En qué se diferencian **color[5]**  y  **\*(color + 5)**?

En resumen, tanto color[5] como \*(color + 5) hacen referencia al mismo sexto elemento del arreglo color, que es la cadena de caracteres "amarillo". La única diferencia es la notación utilizada para acceder a ese elemento: una utiliza corchetes y el otro utiliza aritmética de punteros.

1. ¿Qué se referencia con **\*\*color**?

En este caso, color es un arreglo de punteros a caracteres, por lo que \*color representa el primer puntero en el arreglo, que apunta a la cadena de caracteres "rojo". Luego, \*\*color se desreferencia nuevamente para acceder al valor al que apunta el puntero "rojo". En este caso, el valor sería el primer carácter de la cadena "rojo", es decir, el carácter 'r'.

En resumen, \*\*color referencia al primer carácter de la primera cadena de caracteres en el arreglo color. En este caso, el valor referenciado sería el carácter 'r' de la cadena "rojo".

1. ¿Qué se referencia con **\*color[0]**?

En este caso, color[0] accede al primer puntero en el arreglo color, que apunta a la cadena de caracteres "rojo". Luego, se aplica el operador de desreferenciación \* para acceder al valor al que apunta ese primer puntero, es decir, a la cadena de caracteres "rojo".

Entonces, \*color[0] referencia a la cadena de caracteres "rojo".

1. ¿Qué se referencia con **\*\*color + 2**?

En este caso, color es un arreglo de punteros a caracteres, y \*color representa el primer puntero en el arreglo, que apunta a la cadena de caracteres "rojo". Luego, \*\*color se desreferencia nuevamente para acceder al valor al que apunta el puntero "rojo", que es el primer carácter de la cadena, es decir, el carácter 'r'.

Posteriormente, se realiza una suma con el valor 2. Esto se aplica al carácter 'r', que es un valor entero según su representación ASCII. Al sumar 2 al valor ASCII del carácter 'r', se obtiene un nuevo valor. Por ejemplo, si consideramos la tabla ASCII, el carácter 'r' tiene un valor ASCII de 114, y al sumarle 2, se obtendría el valor 116, que corresponde al carácter 't'.

Por lo tanto, \*\*color + 2 referencia al carácter 't', que se obtiene al sumar 2 al valor ASCII del primer carácter de la primera cadena en el arreglo color.

***Ejercicio 17***

Resolver el ejercicio **5.3** del Libro de Kernighan & Ritchie.



***Ejercicio 18***

Indicar y corregir los errores en cada una de las siguientes funciones.

a)



b)



***Ejercicio 19***

Escribir la función **copiaPrefijo** que dado un string ***s*** y un número entero ***n*** retorne en su nombre un nuevo string con los primeros ***n*** caracteres del string ***s***. Si la longitud de ***s*** es menor a ***n*** debe retornar una copia de ***s***. En caso de que no se pueda copiar debe retornar NULL. ¿Qué cuidado debe tener el usuario de esta función?

***Ejercicio 20***

Escribir la función **aprobados** que reciba dos vectores, uno con los nombres y apellidos de los alumnos de un curso y un vector de enteros donde se almacene el promedio de cada alumno en el curso. La función debe retornar en su nombre un nuevo vector con los alumnos que hayan aprobado el curso, esto es, que tengan un promedio mayor o igual a 4.

Como indicador de final del arreglo de alumnos se tiene el string vacío.

El código de la función debe ser tal que funcione tanto para vectores de punteros a char como para "matriz" de caracteres.

typedef \_\_\_\_\_\_ TAlumnos;

char \*\* aprobados(TAlumnos alumnos, const int notas[] );

Ejemplos de invocación:

typedef char TAlumnos[][15];

TAlumnos alumnos = {"Ana", "Juan Pablo", "Mariana", "Fernando", "Carolina", "" };

int notas[] = {8, 3, 10, 7, 5};

char \*\* alumAprob;

alumnAprob = aprobados( alumnos, notas );

......

typedef char \* TAlumnos[];

int notas[] = {8, 3, 10, 7, 5};

char \* alumnos [] = {"Ana", "Juan Pablo", "Mariana", "Fernando", "Carolina", "" };

char \*\* alumAprob;

alumnAprob = aprobados( alumnos, notas );

......

¿Podría hacerse una función que también retorne uno de los tipos?

TAlumnos aprobados(TAlumnos alumnos, const int notas[] );

***Ejercicio 21***

Escribir una función que permita liberar la memoria reservada por la función del ejercicio anterior.