Trabajo Práctico 2 - Parte 4

Temas:

- **Utilizar variables auto y static**
- **Funciones que emplean paso de parámetros por referencia**

ES OBLIGATORIO UTILIZAR NOMBRES DE VARIABLES Y DE FUNCIONES SIGNIFICATIVOS

Problemas resueltos

USO DE STATIC

Mostrar por pantalla los N primeros números de la serie de Fibonacci. Para realizar este ejercicio debe considerar el uso de una función que no reciba parámetros y que puede o no retornar un valor.

```
#include <stdio.h>
//Declaración de funciones
int fibonacci();

int main(int argc, char *argv[]){
      int Cant_N,cont;
      printf("Ingrese un numero: ");
      scanf("%d",&Cant_N);
      for (cont=1; cont<=Cant_N;cont++)
         printf("\t%d", fibonacci());
      return 0;
}
//Definición de funciones
int fibonacci(){
      static int Num_a, Num_b=1, fib=0, ban=0;
      /*Las variables estáticas se inicializan la primera vez en 0*/
      if (ban==0){
         ban=1;
      }
      else
         {
         Num_a=Num_b;
         Num_b=fib;
         fib=Num_a+Num_b;
         }
      return(fib);
}
```

VARIABLES SIMPLES. USO DE PARÁMETROS POR REFERENCIA

Sugerencia: repasar el concepto de puntero con ejemplos explicados visto en el TP 1 Parte 2

Ingresar un número entero y un número real, luego presentar por pantalla la suma y el producto de dichos números.

```
#include <stdio.h>
//Declaración de funciones
float producto(int x, float y);
void suma(int x, float y, float *sum); //sum es un parámetro por referencia

int main(int argc, char *argv[]){
	int entero;
	float real,sum,producto;
	printf("Ingresar un nro. entero: ");
	scanf("%d",&entero);
	printf("Ingresar un nro. real: ");
	scanf("%f",&real);
	producto=producto(entero,real);
	suma(entero,real,&sum);
	//El valor de suma retorna modificado por ser parámetro por referencia
	printf("\nEl valor de %d más %.2f es = %.2f\n\n",entero,real,sum);
	printf("El valor de %d por %.2f es = %.2f\n\n",entero,real,producto);

 return 0;
}

//Definición de funciones
float producto(int x, float y){
	float p;
	p=x*y;
	return p;
}
void suma(int x, float y, float *sum){ //sum apunta a suma
	*sum=x+y; //sum en un puntero o apuntador a la variable suma
	//todos los cambios realizados utilizando *sum se efectúan directamente
en suma
}
```

REGISTROS (ESTRUCTURAS O STRUCT). USO DE PARÁMETROS POR REFERENCIA

Ingresar dos caracteres (dos letras por ejemplo), luego presentar por pantalla dichos caracteres, pero ordenados en forma ascendente, o un mensaje en caso de que sean iguales.

```
#include <stdio.h>
//Definición de las estructuras
typedef struct{
	char primera;
	char segunda;
} referencia;

//Declaración de funciones
```

```
//la función utiliza un parámetro tipo estructura llamado e por referencia y
retorna una variable simple
int ordenar(referencia *e);

int main(int argc, char *argv[]){
	referencia est;
	int iguales;
	printf("Ingresar un caracter: ");
	scanf("%c",&est.primera);//ingresa el primer caracter en la estructura
	fgetc(stdin);
	printf("Ingresar otro caracter: ");
	scanf("%c",&est.segunda);//se ingresa el segundo caracter
	printf("\nLos caracteres ingresados son: %c,
%c\n",est.primera,est.segunda);
	iguales=ordenar(&est);//el parámetro e un puntero, apunta a la
estructura est
	if (iguales==0)
		printf("\nLos caracteres son iguales\n");
	else
		printf("\nLos caracteres ordenados son: %c,
%c\n",est.primera,est.segunda);


	return 0;
}

//Definición de funciones
int ordenar(referencia *e){//e es un puntero a la estructura est
	char aux;
	int i=1;
	if(e->primera == e->segunda){
	/*Para acceder a un campo en una estructura por referencia, se utiliza
el operador
	flecha -> en lugar del operador punto
	en este caso se compara el valor del primer campo de est con el segundo
campo de est*/
		i=0;
	}
	else{
	  if(e->primera > e->segunda){ //se compara el primer caracter contenido
en la estructura est con el
	segundo caracter contenido en la misma estructura
		aux = e->primera;//aux copia el primer caracter de est
		e->primera = e->segunda;//el primer campo de est copia el valor
del segundo campo de est
		e->segunda = aux;//el segundo campo de est copia el valor de aux
		}
	}
	return i;
}
```

PROBLEMA ANTERIOR RESUELTO CON ACCESO A LA ESTRUCTURA EMPLEANDO PUNTERO Y UTILIZANDO LOS OPERADORES DE INDIRECCIÓN (*) Y PUNTO (.) EN LUGAR DEL OPERADOR FLECHA (->)

```c
#include <stdio.h>
//Definición de las estructuras
typedef struct{
	char primera;
	char segunda;
	} referencia;

//Declaración de funciones
int ordenar(referencia *e);//la función utiliza un parámetro tipo estructura
llamado e por referencia y retorna una variable simple
int main(int argc, char *argv[]){
	referencia est;
	int iguales;
	printf("Ingresar un caracter: ");
	scanf("%c",&est.primera);//se ingresa el primer caracter en la
estructura
	fgetc(stdin);//Limpia el buffer del teclado
	printf("Ingresar otro caracter: ");
	scanf("%c",&est.segunda);//se ingresa el segundo caracter
	printf("\nLos caracteres ingresados son: %c,
%c\n",est.primera,est.segunda);
	iguales=ordenar(&est);//el parámetro e que es un puntero a estructura,
apunta a la estructura est
	if (iguales==0)
		printf("\nLos caracteres son iguales\n");
	else
		printf("\nLos caracteres ordenados son: %c,
%c\n",est.primera,est.segunda)
	return 0;
}
//Definición de funciones
int ordenar(referencia *e){//e es un puntero a la estructura est
	char aux;
	int i=1;
	if((*e).primera == (*e).segunda){
	/*Para acceder a un campo en una estructura por referencia, también se
puede utilizar el operador * y
	 el punto en este caso se compara el valor del primer campo de est con
el segundo campo de est*/
		i=0;
	}
	else{
	if((*e).primera > (*e).segunda){ //se compara el primer caracter
contenido en la estructura est con el
	 segundo caracter contenido en la misma estructura
			aux = (*e).primera;//aux copia el primer caracter de est
			(*e).primera = (*e).segunda;//el primer campo de est copia
el valor del segundo campo de est
			(*e).segunda = aux;//el segundo campo de est copia el valor
de aux
		}
	}
	return i;
}
```

⚠ Consideraciones para resolución de problemas propuestos

Los siguientes problemas, deben ser resueltos utilizando una o más funciones

En la función principal int main(int argc, char *argv[]) **SOLAMENTE deberá**:

a) Ingresar los datos de entrada solicitados en cada problema mediante funciones de control.

b) Invocar a la/s función/es para el procesamiento de los datos de entrada.

c) Presentar los resultados retornados por la/s función/es.

Recuerden que las funciones definidas por el usuario deben ser:

a) **declaradas** (declaración),

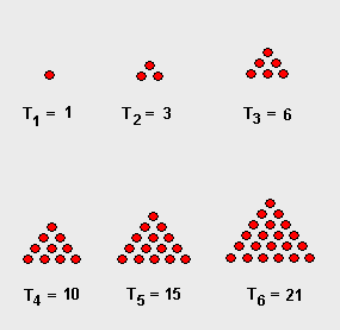
b) **implementadas** (definición) y

c) **usadas** (invocar o llamar).

Obs: En todos los siguientes ejercicios que se solicite el valor de N, realizar una función que solicite el ingreso de este valor y lo devuelva este número positivo.

Resolver utilizando una o más funciones que retornen valor, no utilicen parámetros y que empleen variables locales static (y auto si fuese necesario).

1. Presentar por pantalla, las N primeras letras del alfabeto en mayúsculas.
2. Un número triangular es un número que puede recomponerse en la forma de un triángulo equilátero (por convención, el primer número triangular es el 1). Es decir, los primeros números triangulares son 1, 3, 6, 10, etc. Elabore un programa que muestre en pantalla los N primeros números triangulares.



3. Presentar por pantalla los N primeros números primos. Para esto utilice una función que me devuelva el siguiente número primo, esta debe almacenar el anterior primo mostrado en una variable estática.

```
int siguientePrimo();
```

4. Realizar un programa que permita generar N códigos incrementales automáticos a partir de un cierto número base definido previamente. Por ejemplo, si se quieren generar 5 códigos incrementales a partir de un cierto número, como ser el 10000, se generarán los siguientes códigos: 10001, 10002, 10003, 10004, 10005.

   Para ello escriba una función que genere un código incremental. Utilice una variable estática para almacenar el último código generado y aumente el valor del código en 1 cada vez que se llame a la función. Esta función debe cumplir con la siguiente declaración:

```
int generadorNumero();
```

5. Se ingresan **N números enteros.** Determinar el mayor de los valores ingresados.
   Ejemplo: N = 7, Se ingresan: -12, 5, -8, 30, 0, -20, 15
   Mayor valor: 30.
   Controlar que el valor de N sea positivo. Usar una función que emplee variables static para encontrar el mayor. Verificar también el resultado si ingreso todos números negativos.

Resolver utilizando una o más funciones que utilicen solamente un parámetro por valor, que retorne un valor entero y que haga uso de variables static.

6. Ingresar N números enteros, luego presentar por pantalla la cantidad de series de números impares ingresados. Se considera serie si tiene 2 o más elementos seguidos.

   Ejemplo: 4, 9, 2, 5, 3, 2, 15, 3, 7, 9

```
          |-----|    |------------|
            1             2              2 series
```

7. Ingresar N números enteros no negativos, luego presentar por pantalla cuantas series de números estrictamente crecientes aparecen.

   Ejemplo: 5, 1, 9, 7, 6, 0, 1, 9, 3, 2, 4, 7, 9

```
          |-----|      |---------|    |------------|
            1              2              3       — → 3 series
```

8. Realizar un programa que permita ingresar N números enteros positivos y muestre la suma de todos ellos. Para esto, escriba una función que reciba un número entero como argumento y devuelva la suma acumulativa de todos los números ingresados. Utilice una variable estática para almacenar la suma acumulativa.

Codificar los siguientes problemas en lenguaje C, empleando funciones que retornen o no valor y que utilicen uno o más parámetros por referencia (y paso de parámetros por valor si fuese necesario)

Referencia ejercicios TP2-Parte 2 (Resolver utilizando funciones que NO retornen valor)

9. (Referencia ejercicio 2 del TP2-Parte 2) Implementar un programa en el cual: se define una constante numérica llamada "PI" con el valor 3.14; se solicite que se ingrese un número que será el radio de una circunferencia; se calcule el valor de la longitud de la circunferencia y se lo presente por pantalla. Recordar que la longitud de la circunferencia es 2 * PI * radio. Para este problema **diseñar una función que no retorne valor y que además de recibir los dos valores (PI y radio), reciba un parámetro por referencia donde se guarde el resultado del cálculo solicitado.**

Referencia ejercicios TP2-Parte 2 (Resolver utilizando funciones que retornen o no valor)

10. (Referencia ejercicio 2 del TP2-Parte 2).Definir un tipo struct que tenga dos campos, uno para contener la parte real y otro la parte imaginaria de un número complejo:

```
typedef struct{
      float real;
      float imaginario;
}complejo;
```

Emplear esta estructura para solicitar al usuario el ingreso de dos números complejos, realizar la suma de los mismos. Al finalizar los cálculos deberá presentar por pantalla los números complejos ingresados, la suma de dichos números. Para resolver deberá implementar funciones que empleen parámetros por referencia (y por valor si fuese necesario):

- Una función para solicitar al usuario el ingreso de un número complejo
- Una función para presentar por pantalla un número complejo dado

- Una función para calcular la suma de dos números complejos dados.

Ninguna de las funciones debe retornar valor.

11. (Referencia ejercicio 3 del TP2-Parte 2) Ingresar un número entero positivo, luego presentar por pantalla un mensaje que indique si el número ingresado es o no es primo. Para resolver implementar:

- una función que realice el control de que el número sea positivo.
- una función que determine si un número es primo **(esta función no debe retornar un valor)**

12. (Referencia ejercicio 9 del TP2-Parte 2) Codificar un programa en el que se solicita el ingreso de un carácter alfabético, luego presentar por pantalla el carácter en mayúscula si el mismo fue ingresado en minúscula y viceversa. Para resolver este ejercicio deberá implementar:

a- una función para ingresar el carácter y controlar que sea alfabético.

b- una función para devolver el carácter convertido **(esta función no debe retornar un valor)**

c-una función para mostrar por pantalla el caracter convertido(esta función no es necesario que retorno valor)

13. (Referencia ejercicio 12 del TP2-Parte 2) Implementar un programa que permita determinar si un medicamento se encuentra o no vencido, conociendo la fecha de vencimiento y la fecha actual. Para resolver este ejercicio deberá:

- Usar del ejercicio (Referencia ejercicio 11 del TP2-Parte 2) las funciones para leer una fecha y controlar su validez.
- Implementar una función que determine si el medicamento está vencido y que presente por pantalla el mensaje correspondiente. **(esta función no debe retornar un valor)**

Almacenar la fecha en un dato tipo estructura con el siguiente formato:

```
typdef struct{
      int dia;
      int mes;
      int anio;
}fecha;
```

14. (Referencia ejercicio 2 del TP2-Parte 3)Ingresar N caracteres en un programa, identificar cuántos caracteres son letras minúsculas e indicar el porcentaje con respecto al total de caracteres.

Por ejemplo: se ingresa N= 10 y se ingresan los siguientes caracteres: e 5 T 1 C i ? / 1 >
Se debe mostrar que el porcentaje de minúsculas fue del 20%.

Para ello usar funciones para:

- controlar el valor de N,
- contar cuantos caracteres en minúsculas se ingresaron, con una función con la siguiente declaración:

  void contarMinusculas(char c, int *cantidadMinusculas);

- calcular el porcentaje con una función con esta declaración:

  void calcularPorcentaje(int cantidad, int total, float *porcentaje);

- presentar por pantalla el promedio, con una función con la siguiente declaración: **void mostrarPorcentaje(float porcentaje);**

15. Se reciben dos valores numéricos enteros, realizar con ellos la suma, resta, división y multiplicación y mostrar los resultados en pantalla.

Para resolver este problema debe:

- Crear un menú a través del cual se puedan seleccionar las operaciones a realizar
- Cada una de las operaciones deberá ser resuelta por una función y los resultados se mostrarán en la función principal main().

  Observación: Tener en cuenta que las funciones deben ser:

  menu(): Retorna Valor (devuelve la operación seleccionada) y no tiene parámetros.
  suma(): Retorna Valor (devuelve resultado de la suma) y usa parámetros por valor.
  resta(): No retorna valor, utiliza dos parámetros por valor y uno por referencia.
  cociente(): Retorna valor y utiliza parámetros por valor. ¡Cuidado con la división en cero!
  producto(): No retorna valor, utiliza dos parámetros por valor y uno por referencia.

16. Definir un tipo struct que contenga los siguientes atributos:

```
typedef struct{
      char automovil; //letra que representa al automovil
      float tiempo;
}participante;
```

Emplear esta estructura para resolver el siguiente problema: En una carrera, en donde compiten N automóviles, se registra una letra que identifica a cada automóvil y el tiempo que demora en llegar a la meta.

Utilizar funciones diferentes para:

- Verificar que N sea un número entero positivo.
- Ingresar un participante, controlando que el tiempo sea un real positivo. (esta función no debe retornar valor).
- Determinar cuál es él**automóvil ganador**, es decir el auto que llegó utilizando el menor tiempo posible y en qué posición se lo ingresó, con una función con la siguiente declaración:

  void determinarGanador(participante *participanteIngresado, participante *ganador, int *posicionGanador);

- Para mostrar por pantalla al automóvil ganador. Utilizar funciones de conversión de caracteres, presente la letra en mayúscula y el tiempo que este demoró en llegar a la meta. ***Revisar ctype.h***

17. De un grupo de N personas se debe registrar el peso, altura y edad. Realizar un programa que permita calcular el peso promedio, altura promedio y la edad promedio.

Utilizar funciones para:

- Verificar que N sea un número entero positivo.
- Ingresar las personas y controlar que sus atributos sean números positivos
- Calcular el promedio del peso, la altura y la edad.

Presentar por pantalla los resultados. Utilizar parámetros por valor y por referencia y variables static si considera necesario.

Usar la siguiente estructura para almacenar la información de cada persona:

```
typedef struct {
	float peso;
	float altura;
	int edad;
} persona;
```

18. Se ponen a la venta entradas para un partido de fútbol internacional, cuyo precio depende de la ubicación o tribuna, así:

- Tribuna norte y sur cuesta 4000 pesos,
- Tribuna este cuesta 5000 pesos y
- Tribuna oeste cuesta 6500 pesos.

Se debe elaborar un programa que controle la venta de dichas entradas.

Se realizan N ventas, por cada venta se solicita cantidad de entradas a comprar y en qué tribuna.

Se pide resolver utilizando funciones: :

- Controlar el ingreso de un número N positivo.
- Elaborar un menú para que el usuario elija en qué tribuna quiere comprar sus entradas, y luego debe seleccionar la cantidad. Se recomienda usar la siguiente definición de función

  void ingresarVenta(tribunas *t);

- El porcentaje de personas que se ubicaron en la tribuna norte. El porcentaje se mostrará en main y deberá ser obtenido haciendo uso de una función que no retorne un valor.
- El monto total recaudado por la venta de todas las entradas

Se recomienda el uso de una estructura para ir almacenando la cantidad de entradas vendidas en cada tribuna

typedef struct {

int tribunaNorte;

int tribunaSur;

int tribunaEste;

int tribunaOeste;

} tribunas;

19. Ingresar dos caracteres; controlar que los mismos sean **letras del alfabeto**; en caso de que las letras ingresadas sean mayúsculas, convertirlas a minúsculas; luego presentar por pantalla dichas letras pero ordenadas en forma alfabética, o un mensaje en caso de que sean iguales (debe realizar una función para ordenar las letras y determinar si son iguales).

20. Un proveedor de tecnología, registra sus ventas diarias en un sistema diseñado para solicitar DNI del cliente, monto de la venta y el medio de pago de la misma ('t' para tarjeta de crédito o débito, 'e' para pago en efectivo y 'c' para cuenta corriente).
Almacenar la venta en un dato tipo estructura con el siguiente formato:

```
typedef struct{
    int dni;
    float monto;
```

```
        char medio_pago;
    }venta;
```

La carga de ventas diarias finaliza al ingresar el caracter F

Al finalizar cada día el encargado necesita saber:

- El total por cada medio de pago y el total vendido en el dia. El sistema debe mostrar en la función principal los distintos totales. Para esto realizar la siguiente función:
  float totalVenta(venta *v, float totalT,float *totalE, float *totalC);

- El importe de venta más alto y a que cliente perteneció.
  El sistema debe comparar cada vez que se ingresa una venta, si es la mayor venta ingresada, para poder mostrarla al final del ingreso. Realizar la siguiente función:
  void ventaMayor(venta ventaIngresada, venta *ventaMayor);
- El porcentaje de ingresos que representa cada condición, esto el sistema se lo muestra en la función principal con una función que no retorna valor y que tiene la siguiente declaración: **void porcentajes(float total, float individual, float *porcentaje);**

21. Realizar un programa que permita ingresar N números enteros positivos, luego determinar el valor promedio (o media aritmética) de los números ingresados que sean primos.

Para resolver este problema deberá ***implementar las siguientes funciones:***
a- **ingresarEntero():** permite ingresar un número entero y controlar si el mismo es positivo.

b- **esPrimo():** para determinar si un número es primo.

c- **promedioPrimos()**: calcula el promedio de los números primos **(esta función no debe retornar un valor)**

Se debe presentar por pantalla el promedio calculado, o en su defecto un mensaje si en la serie ingresada no hay números primos.

Aclaración: Ud debe definir si las funciones arribas mencionadas reciben o no parámetros, como así también si retornan o no valor