Práctica 1: Arrays y punteros

Contenido:
Esta práctica está diseñada para que el estudiante comience a utilizar el concepto de puntero, íntimamente relacionado con el de array.

1) Descubriendo punteros: analizar los resultados del ejemplo.

```
#include <stdio.h>

int main(void)
{
        int i = 8, *pi=&i;
        long long l = 8, *pl=&l;
        float f = 102.8f, *pf=&f;
        double d=678.44, *pd=&d;
        int vec[100];
        vec[0] = 44;

        printf("variable int, tam.bytes: %d \tdir.&i: %p \tvalor: %d\n", sizeof(i), &i,  i);
        printf("puntero int, tam.bytes= %d \tdir.&pi: %p \tvalor: %p\n", sizeof(pi), &pi, pi);
        printf("variable long, tam.bytes: %d \tdir.&l: %p \tvalor: %ld\n",  sizeof(l), &l, l);
        printf("puntero long, tam.bytes: %d \tdir.&pl: %p \tvalor: %p\n",  sizeof(pl), &pl, pl);
        printf("variable float, tam.bytes: %d \tdir.&f: %p \tvalor: %.1f\n",  sizeof(f), &f, f);
        printf("puntero float, tam.bytes: %d \tdir.&pf: %p \tvalor: %p\n",  sizeof(pf), &pf, pf);
        printf("variable double, tam.bytes: %d \tdir.&d: %p \tvalor: %.2lf\n",  sizeof(d), &d, d);
        printf("puntero double, tam.bytes: %d \tdir.&pd: %p \tvalor: %p\n",  sizeof(pd), &pd, pd);
        printf("variable array, tam.bytes: %d \tdir.&vec[0]: %p \tvalor: %d\n", sizeof(vec[0]), &vec[0], vec[0]);
        printf("puntero array, tam.bytes: %d \tdir.&vec: %p \tvalor: %p\n", sizeof(vec), &vec, vec);
        return 0;
}
```

Verifique el tamaño de cada tipo de variable y del puntero asociado.

2) Escriba un programa que defina las siguientes variables:

```
int i=5, j[]={1,2,3,4,5,6,7,8,9,10};
char x = 'a', pal [] ="texto en c";
int *pi;
char *pc;
```

1. Mostrar la dirección de **"i"** y su valor.
2. Mostrar los mismos valores a través del puntero **"pi".**
3. Recorrer el vector **"j"** mostrando para cada elemento, su dirección y su valor.
4. Recorra el vector accediendo a través del puntero **"pi"** y usando álgebra de punteros.
5. Repita lo mismo con las variables **char**, el arreglo y el puntero.
6. Finalmente muestre la dirección donde se almacenan ambos punteros.

Genere una salida del tipo:

```
Por Variable: 'i'                Valor:   5   Dirección:   13FF5C
Por Puntero: 'pi'                Valor:   5   Dirección:   13FF5C
Por Variable: 'j[0]'             Valor:   1   Dirección:   13FF2C
Por Puntero: ' pi(=&j)+0'        Valor:   1   Dirección:   13FF2C
Por Variable: 'j[1]'             Valor:   2   Dirección:   13FF30
Por Puntero: 'pi(=&j)+1'         Valor:   2   Dirección:   13FF30
…
Por Variable: 'x'                Valor:   a   Dirección:   13FF23
Por Puntero: 'pc'                Valor:   a   Dirección:   13FF23
Por Variable: 'pal[0]'           Valor:   t   Dirección:   13FF0C
Por Puntero: 'pc(=&pal)+0'       Valor:   t   Dirección:   13FF0C
Por Variable: 'pal[1]'           Valor:   e   Dirección:   13FF0D
Por Puntero: ' pc(=&pal)+1'      Valor:   e   Dirección:   13FF0D
…
Dirección de *pi: 13FF00      De *pc: 13FEF4
```

3) Crear un programa que lea un número determinado (<100) de reales introducidos por teclado, los almacene en un vector para luego mostrarlos en orden inverso. Para recorrer el array deberá usar aritmética de punteros en lugar de índices del array.

4) Escribir una función que tome como argumento un entero positivo entre 1 y 7 y retorne un puntero a cadena con el nombre del día de la semana correspondiente al argumento. Probar dicha función.

5) Escribir una función void que tome como argumentos: la cantidad de kilos de determinado producto adquirido por un cliente y el precio por kilo del mismo (ambos números flotantes); la misma debe calcular el importe de la compra. El descuento efectuado a compras superiores a 100$ es del 10%, con lo cual la función deberá también calcular el nuevo monto, si es que corresponde el descuento. Usar argumentos pasados como punteros, donde corresponda.

6) Escribir las funciones que operan sobre cadenas de caracteres.

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

    typedef enum { MAYUSCULAS, MINUSCULAS } may_min;

    int strLargo(const char *origen); //Cantidad de caracteres
    int strVacio(const char *origen); //retorna 1 si tiene al menos un caracter, 0 en otro caso
    void strCopia(char *destino, const char *origen); // Copiador
    /*prototipo modificado para permitir argumentos de tipo string literales, en casi todos los
    compiladores un literal string es considerado una constante, o sea la función no podría
    modificarlos pero, en algunos compiladores tales como GCC es posible modificarlos (según
    K&R el comportamiento es indefinido)*/
    char* reverse(char *string);//retorna una cadena que es string invertida
```

```
void strIzq(char *destino, const char *origen); // Saca blancos Izq.
void strDer(char *destino, const char *origen); // Saca blancos Der.
void strAmbos(char *destino, const char *origen); // Saca blancos Izq. y Der.
void strMayMin(char *destino, const char *origen, may_min m); // Convierte May. Min.

int main(){
	char *text1 ="   Sera Cierto ??   ";
	int largo=strLargo(text1)+1;
	char *result = (char *)malloc (largo);
	char* reves;
	if(result == NULL)
		return -1;//sino pudo reservar memoria para result
	printf("La cadena: ");
	puts(text1);
	printf("Se encuentra: %s\n",(strVacio(text1) ? "No vacia" : "Vacia"));
	printf("Largo : %d\n", strLargo(text1));
	strCopia(result,text1);
	printf("Copia : [%s]\n", result);
	strIzq(result,text1);
	printf("Sin blancos a la Izq:");
	puts(result);
	strDer(result,text1);
	printf("Der : [%s]\n", result);
	strAmbos(result,text1);
	printf("Ambos: [%s], sin blancos al principio ni al final.\n", result);
	strMayMin(result,text1, MAYUSCULAS);
		printf("Mayusculas : [%s]\n", result);
	strMayMin(result,text1, MINUSCULAS);
	printf("Minusculas : [%s]\n", result);
	reves=reverse(text1);
	printf("La cadena: %s invertida queda: %s\n",text1, reves);
	return 0;
}
```

Salida:

```
La cadena:    Sera Cierto ??
Se encuentra: No vacia
Largo : 20
Copia : [   Sera Cierto ??   ]
Sin blancos a la Izq:Sera Cierto ??
Der : [   Sera Cierto ??]
Ambos: [Sera Cierto ??], sin blancos al principio ni al final.
Mayusculas : [   SERA CIERTO ??   ]
Minusculas : [   sera cierto ??   ]
La cadena: Sera Cierto ?? invertida queda ?? otreiC areS
Presione una tecla para continuar . . .
```

7) Escribir una función que reciba como argumento un entero y retorne una cadena de caracteres en su representación decimal. Ídem para: representación octal, hexadecimal y binaria (genere una función por cada una de estas opciones).

8) Se necesita contar las letras de un texto ingresado. El texto puede tener varias oraciones. Crear una rutina que almacene las letras y la cantidad de veces que aparecen. Generar un informe con el detalle, en caso de no sean alfabéticos o números, mostrar su valor hexadecimal. Debe crear un vector de punteros que almacenen las frases y luego recorrerlo con un puntero de doble in-dirección.

9) Escriba una función que reciba como argumento un entero positivo (n) y que genere en forma dinámica una matriz identidad de dimensión n.

10) Escribir un programa para calcular el determinante de una matriz:

- Solicite la dimensión por teclado.
- Solicite memoria para el almacenamiento.
- Ingrese por teclado los coeficientes con valores.
- Desarrolle una función que realice el cálculo.

```
int determinante_Matriz(int tam, int **matriz);
```

- Mostrar por pantalla los rangos de la memoria asignada para la matriz.
- Verificar los resultados.

Nota: Recordar que el determinante de una matriz es $\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \Rightarrow |A| = a_{11} \times a_{22} - a_{12} \times a_{21}$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \times a_{22} \times a_{33} + a_{12} \times a_{23} \times a_{31} + a_{13} \times a_{21} \times a_{32} - a_{31} \times a_{22} \times a_{13} - a_{32} \times a_{23} \times a_{11} - a_{33} \times a_{21} \times a_{12}$$

11) Producto escalar de Matriz: Genera una nueva matriz al que se le multiplica cada elemento por un valor (escalar):

- Solicite el valor escalar y la dimensión de la matriz por teclado.
- Solicite memoria para el almacenamiento.
- Llene la matriz con valores aleatorios.
- Desarrolle una función que realice el cálculo cálculo y devuelva un puntero a la nueva matriz.

```
int ** producto_Escalar_Matriz(int fil, int col, int esc, int **matriz);
```

12) a) Escribir 3 funciones que, recibiendo una cadena de caracteres como argumento, permitan determinar si la cadena es válida como dirección IP, como dirección de correo electrónico, y como número de tarjeta de crédito.
b) Escribir una función denominada `validate_string()` que recibiendo una cadena de caracteres y una función de validación (pasada por puntero), determine si la cadena es válida conforme al criterio de validación indicado, retornando en consecuencia `true` o `false` por su nombre.

13) En un archivo CSV se recibe un listado con los clientes que han comprado algún producto a una empresa. En particular los campos 4 y 5 indican el número de sucursal y el código de cliente, respectivamente. El código de cliente puede ser:

- una dirección IP.
- una dirección de correo electrónico.
- un número de tarjeta de crédito.

a) Escribir una función que reciba sobre un argumento de tipo char * el código de cliente, y aunque sea rudimentariamente, los clasifique de acuerdo a las tres posibles categorías:

```
codigo_cliente_t clasificar_campo (char *);
```

b) Escribir una función que, recibiendo el código de cliente como string y su tipo, permita formatearlo convenientemente para su impresión.

c) Escribir una función que, recibiendo el código de cliente, lo clasifique internamente y retorne por el nombre un puntero a la función de impresión correspondiente. Utilizar un arreglo global de punteros a funciones.

Referencias:

Algunos ejercicios se extrajeron de:
http://materias.fi.uba.ar/7502E/material.html