Práctico Nº 1

Tema: Lenguaje C y Punteros

Nota: los ejercicios deberán codificarse con **al menos 2 funciones**: la función *main* y una o más funciones definidas por el usuario. Los primeros 6 ejercicios son de repaso del lenguaje C, el resto deben ser realizados con punteros y sin usar variables globales. **Ejecutar TODOS LOS PROGRAMAS.**

Ej. 1 – Dado el siguiente código:

- Encontrar los **errores de sintaxis.**
- Luego **ejecutarlo.** Indicar que hace o intenta hacer
- **Corregir** todos los errores para que resuelva el problema indicado por Ud. en el punto anterior.
- **Re escribir** el código pero sin usar variables globales.

```
float P[27]
const int C=3;

void ingreso(float X[]){
   int i;
   for(i=0; ;i++){
      printf("\n Ingrese real ");
      scanf("%d", P[i]);
      getchar();
   }
}

void mostrar(float f) {
  printf("\n El valor real que falta es %c ",f);
}

determinar(int N){
 float f;
 int N,i=0;
 for (;i<C;i++)
       N == P[i];
       f = P[i]-N;
       mostrar (1-f,N);
 }
}

#include<stdio.h>
int main(){
      float f;
            INGRESO(P)
            determinar();
            return(0);
}
```

Ej. 2 - Codificar un programa que permita ingresar hasta 25 caracteres en un arreglo. Luego contar e imprimir cuantos caracteres mayúsculas, minúsculas, dígitos y caracteres especiales fueron ingresados. Además, dado un caracter ingresado por el usuario, realizar una función que cuente y retorne la cantidad de apariciones de dicho caracter. Finalmente, imprimir solo los dígitos ingresados en el arreglo, su correspondiente ASCII y el caracter ingresado por el usuario y la cantidad de apariciones.

Ej. 3 – Ej. 3 – <u>Operadores para manejo de bits</u>:

a) Realizar un programa que, dada una variable de tipo caracter inicializada con el valor 35, diga si el cuarto bit de dicha variable es 1 o 0. Nota: utilice operadores para manejo de bits que permita enmascarar el bit.

b) Mostrar qué imprime por pantalla el siguiente código:

```
#include <stdio.h>
int main ( )
{
char a = 23;
char b = 80;
char final;

      printf (''a = %d b =  %d'' , a , b ) ;
      resultado = a | b ;
      printf (''Resultado = %d '' , final ) ;
      return 0 ;
}
```

c) Mostrar qué imprime por pantalla el siguiente código y decir qué significa hacer corrimientos de bits hacia la derecha, en cuanto al número al cual se le aplica el corrimiento.

```
#include <stdio.h>

int main ( )
{
char a = 105;
char final;

      printf (''a = % d '' , a) ;
      final = a >> 1 ;
      printf (''  Resultado = %d'' , final);
      final = a >> 2;
      printf (''  Resultado = %d '' , final);
      return 0;
}
```

Ej. 4 – **Realizar** un programa que lea 5 números enteros positivos distintos, y mediante una función determine para cada número si es divisible por 6. La función solo puede tener un parámetro. El programa principal debe imprimir los números divisibles por 6. No usar variables globales.

Ej. 5 – **Escribir** un programa que lea números enteros hasta que se ingrese el cero. El segundo número ingresado se sumará al primero, luego el tercero se restará del resultado anterior, y así se deberá seguir alternando entre suma y resta hasta que se llegue al cero. Cuando se llegue a esta condición deberá imprimir el resultado en la función y el programa principal deberá imprimir la cantidad de operandos ingresados (sin incluir el cero). Ingresada la siguiente secuencia de números 22, 8, 9, -18, 15, -10, 0 ¿cuál es el resultado de su programa?

Ej. 6 – **Realizar** un programa que lea 3 números enteros distintos de cero en variables independientes (sin utilizar arreglo). Determine, luego, el mayor número par y el menor número impar e imprimirlos en la función *main*. Las funciones solo pueden tener dos parámetros. No puede usar variables globales.

Ej. 7 - Dado el siguiente programa:

```
# include <stdio.h>
int i, *pi;

int main ( ){
          i = 20;
          pi = &i;
          *pi = 133;
          printf ("valor de i es:%d \n", i);
          printf ("valor apuntado por pi es: %d \n", *pi);
          return (0);}
```

¿De qué tipo es la variable **pi**?
¿Es sintácticamente correcta la asignación **pi = &i**?
¿Cuál es el contenido de la variable **i** luego de ejecutado el programa? Justificar

Ej. 8 - El siguiente es un trozo de código válido:

```
int x=8, y=245;
 int *p1;
 int *p2;
```

Suponiendo que la dirección de **x** es **Ox8000** y la de **y** es **Ox7500, graficar** para cada instrucción los efectos que produce la ejecución del siguiente trozo de programa ejecutándolo consecutivamente:

```
p1 = &x;
p2 = &x;
*p1 = x + 1;
*p1 = *p1 + 1;
p2 = p2 + 1;
p2 = y + 1;
p2 = *p1 + 1;
p2 = *p2 + 1;
```

Ej. 9 – Ejecutar el siguiente código asumiendo que la variable **"a"** se encuentra en la dirección 0x4000 y **"carac"** en la dirección 0x6000.

```
#include <stdio.h>
int main() {
    char *r, carac='A';
    int a=12;
    int *p;

  printf("Direccion de a=%p\n", &a);
    printf("a = %d \n", a);
    printf("Direccion de carac = %p\n", &carac);
    printf("carac = %c\n", carac);
    printf("Valor ASCII de carac = %d\n", carac);
    printf("Direccion de p = %p\n", &p);
    printf("Direccion de r = %p\n", &r);
    p = &a;
    r = &carac;
    *r = *r + '0' + a;
    *p = *r + '0' + a;
    printf("El puntero  p  apunta a la direccion: %p\n", p);
    printf("Y su contenido es: %d\n", *p);
    printf("El puntero  r  apunta a la direccion: %p\n", r);
    printf("Y el contenido de *r es: %d\n", *r);
    printf("Como caracter ASCII *r contiene: %c\n", *r);
    return (0); }
```

Ej. 10 - Dado el siguiente programa:

```
 # include <stdio.h>
int main ( )
{
 int temp;
 int *p1_arr, *p2_arr;
 *p1_arr = 45;
 *p2_arr = 78;
 printf ("valor apuntado por p1_arr es: %d \n", *p1_arr);
 printf ("El valor apuntado por p2_arr es: %d \n", *p2_arr);
 temp = *p1_arr;
 *p1_arr = *p2_arr;
 *p2_arr = temp;
 printf ("valor apuntado por p1_arr es: %d \n", *p1_arr);
 printf ("valor apuntado por p2_arr es: %d \n", *p2_arr);
 return (0);
}
```

¿**Es correcto**? Si su respuesta es negativa, modificar el programa de manera tal que solucione el error.
Ejecutar mostrando paso a paso los cambios producidos en la memoria.
¿Cuál es la **salida** del programa?

Ej. 11 – **Realizar** un programa que almacene en un arreglo de caracteres una frase de hasta 5 palabras en letras minúsculas, ingresar caracter por caracter. No se permite almacenar espacios en blanco.

a) Mediante una función "*Imprimir1*" mostrar por pantalla una de las palabras. (La palabra a imprimir 1^{era}, 2da,…o 5ta será seleccionada por el usuario).

b) Definir una **función** "Vocales" que permita contar la cantidad de vocales en una de las 5 palabras. (La palabra en la que se contara la cantidad de vocales, 1^{era}, 2da,…o 5ta será seleccionada por el usuario).

c) Imprimir en la función main la cantidad de vocales y la palabra seleccionada en la parte b.

Ejercicios Propuestos

Ej. 1.- Realizar las siguientes modificaciones en el programa del ej 11:

a) Definir una función *"Impri_todo"* que imprima todas las palabras ingresadas una debajo de la otra.

b) Definir una función *"Selec_palabra"* que permita seleccionar una palabra, la cantidad total de palabras es también un parámetro.

c) Agregar al programa el llamado a las funciones recién definidas.

Ej. 2 – **Realizar** un programa que lea y almacene 100 números enteros no mayores a 89. Posteriormente busque entre los números ingresados los números primos que se encuentren e imprimirlos.