ANALISTA UNIVERSITARIO EN SISTEMAS
TALLER DE PROGRAMACION II

PRÁCTICA 1

Ejercicio 1: Ejecutar y analizar el siguiente programa:

```
#include <stdio.h>
int main () {
  char ch;
  while ( (ch = getchar()) != '\n')
  printf(" %c ", ch);
  return 0;
}
```

¿Qué hace este código?

Ejercicio 2: Calcule mediante bucles for las siguientes sumatorias y productorias. Seleccione cuatro ítems y reescríbalos usando while.

$$\sum_{n=1}^{100} \frac{1}{n} \qquad \sum_{k=1}^{30} \frac{1}{k^2} \qquad \sum_{j=1}^{25} \frac{1}{j^j} \qquad \sum_{i=2}^{10} (i+1)i$$

$$\prod_{i=1}^{8} i \qquad \prod_{k=1}^{10} \frac{1}{k^2} \qquad \prod_{m=1}^{5} (m^2+3m) \qquad \prod_{n=0}^{10} (n+1)^2$$

Ejercicio 3: Una terna de números naturales (a, b, c) es una *terna pitagórica* si $a^2 + b^2 = c^2$.
Escribir un programa que imprima todas las ternas pitagóricas con $a \leq 20$ y $b \leq 30$.

Ejercicio 4: Mediante una única instrucción **for** y un **printf** genere la siguiente salida, emplee variables para la cadena de texto, un entero y un char:

```
b       5       T
bu      4       s
buc     3       R
bucl    2       q
bucle 1         P
```

Ejercicio 5: Año bisiesto, según el calendario gregoriano, es el año cuyo número es divisible por 4, excepto los seculares (divisibles por cien) de número no divisible por 400. (Ejemplo: 1700, 1800, 1900, son no bisiestos, 1996 y 2000 fueron bisiestos).
Construya un programa que determine si un año ingresado es bisiesto.
Referencia:
http://es.wikipedia.org/wiki/Anexo:Años_bisiestos_en_los_Siglos_XX,_XXI_y_XXII

Ejercicio 6: Realice un programa que convierta un número binario en su equivalente decimal.

Ejercicio 7:
Qué hace el siguiente programa?

```
#include <stdio.h>

int main() {
     int x, y;
     printf ("Ingrese dos enteros dentro del rango [1,20]:\n");
     scanf("%d %d", &x, &y);

     for (int i=1; i<=y; i++) {
          for (int j=1; j<=x; j++)
               printf ("@");
          printf ("\n");
     }

     return 0;
}
```

Ejercicio 8:
Escriba un programa que cumpla con los siguientes requisitos:

- El programa debe generar un número aleatorio entre 0 y 500, luego el usuario, mediante ingreso de números, deberá adivinar dicho valor.

- En cada intento el programa debe responder "el número es mayor" o "el número es menor", según corresponda.

- El usuario dispondrá como máximo de 15 intentos.

- En caso de éxito deberá presentarse un cartel diciendo:
  "Adivinó, felicitaciones!".
  En caso de fracasar deberá mostrar
  "El número era *número*".

Ejercicio 9:
Analice el siguiente código e introduzca comentarios en cada línea indicando el objeto de la misma.
Observar la 4ta línea a modo de ejemplo.
El algoritmo expuesto data del siglo III a.C. Y se lo denomina "Criba de Eratóstenes".

Corrija errores en caso de detectarlos.
Comente con sus palabras el objetivo del algoritmo.

```
#include (stdio.h)  // linea 1: …
#define N = 1000;   // linea 2: …

int main() {        // linea 3: …
  int i, j, a[N+1]; // linea 4: declaracion de dos enteros y
                       // un arreglo de … (completar) componentes
  for (a[1]=0, i=2; i <= N; i++)
     a[i] = 1;

  for (i=2; i <= N/2; i++)
     for (j==2; j <= N/i, j++)
        a[i*j] = 0;

  for (i=1; i <= N; i++)
     if(a[i]==true) printf('%d', I);

  printf("\n");
}
```

Ejercicio 10:
Crear una estructura punto (en el plano) y defina las siguientes funciones.:

proy(punto, e): devuelve el punto proyección sobre el eje e (e es un char que vale 'x' o 'y', cualquier
otro valor debe ser invalidado).

dist(punto1, punto2): devuelve la distancia entre dos puntos.

pos(punto): que devuelve '1', '2', '3' o '4' si el punto se encuentra estrictamente en un cuadrante, 'x' o 'y' si se encuentra en uno de los ejes u 'o' si es el origen de coordenadas.

Ejercicio 11:
Defina ahora una estructura circulo. Considerar la definición de un círculo por su centro y radio.
Crear las siguientes funciones:
area(circulo)
perimetro(circulo)
pertenece(circulo, punto): predicado que indica si el punto se encuentra dentro del círculo.

Ejercicio 12:
Dada las siguientes declaraciones

```
int vector[5] = {10, 20, 30, 40, 50};
int a = 3;
int *ptr = &a;
int *qtr = vector;
```

Determinar el tipo y valor de

a	&a	*a	ptr	&ptr	*ptr
qtr	&qtr	*qtr	vector	&vector	*vector
++qtr	+++qtr	+++vector	*&ptr	**&ptr	

Ejercicio 13:
Dados

```
int *ip1, ip2;
char ch, *cp;
```

¿Cuáles de las siguientes asignaciones son violaciones de tipo? Explique por que.

```
ip1 = “cadena de ejemplo”;    cp = 0;          ip1 = 0;
cp = &'a';                    ip1 = ip2;       cp = '\0';
ip1 = '\0';                   cp = &ch;        *ip = ip2;
```

Ejercicio 14:
¿Qué hacen las siguientes funciones?

```
void misterio1 ( char *s, char *t ) {
    while (*s != '\0' ) ++s;
    for( ; *s = *t; s++, t++)
            ;  /* instrucción vacía */
}
```

```
int misterio2 ( char *s ) {
    int x = 0;
    for (; *s != '\0' ; s++) ++x;

    return x;
}
```

```
int misterio3 ( char *s, char *t ) {
    for (; *s != '\0'  && *t != '\0' ; s++, t++)
        if (*s != *t) return 0;
    return 1;
}
```

Ejercicio 15:
Implemente una pila de tamaño dinámico con arreglos. El tamaño de la pila se debe doblar cuando la pila se llena y se debe reducir a la mitad, cuando menos de un cuarto de su tamaño actual está siendo utilizado.
Utilizar un tamaño inicial (y mínimo) de 4 elementos.
NOTA: La “cima” de la pila se relaciona con las posiciones “más altas” del arreglo.