



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Marco Antonio Martínez Quintana

Profesor:

Fundamentos de Programación

Asignatura:

3

Grupo:

11

No de Práctica(s):

Sánchez Hernández Marco Antonio

Integrante(s):

*No. de Equipo de
cómputo empleado:*

No aplica

48

No. de Lista o Brigada:

2021-1

Semestre:

10/enero/2021

Fecha de entrega:

Observaciones:

La práctica fue realizada en un ordenador con Sistema operativo Kubuntu 20.04 LTS y un gestor de ventanas TWM y el editor de texto NeoVim.

CALIFICACIÓN: _____

Arreglos unidimensionales y multidimensionales

Introducción

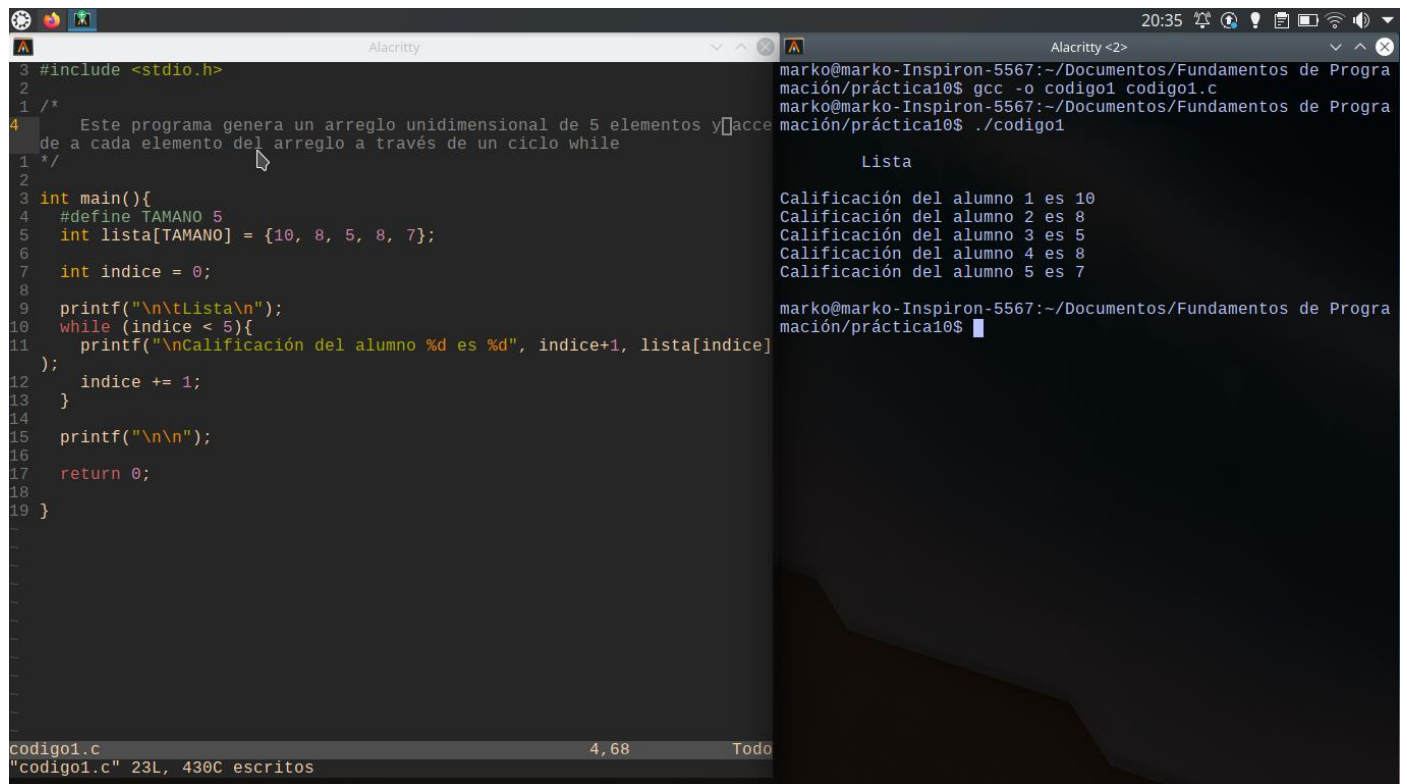
Los arreglos unidimensionales o multidimensionales son conjuntos de datos del mismo tipo que se disponen contiguamente. El tamaño de los arreglos será dado por el programador al momento de definirlos. Cada elemento del arreglo ocupa una posición, las cuales empiezan a contarse desde el número 0, es decir, la primera posición es 0. Se dice que un arreglo es unidimensional si solamente posee una sola dimensión, mientras que será multidimensional si posee más de una. Las dimensiones en lenguaje C serán representadas por [], en la cual se escribirá un número entero dentro, en una dimensión, se puede decir que el número entero representará el número de renglones, mientras que un arreglo de dos dimensiones, el segundo número entero introducido en el segundo corchete representa el número de columnas.

Los apuntadores se definen como variables que poseen la dirección de almacenamiento de un dato contenido en una variable.

Objetivo:

Reconocer la importancia y utilidad de los arreglos, en la elaboración de programas que resuelvan problemas que requieren agrupar datos del mismo tipo, así como trabajar con arreglos tanto unidimensionales como multidimensionales.

Ejercicios



```
3 #include <stdio.h>
4
5 /* Este programa genera un arreglo unidimensional de 5 elementos y accede a cada elemento del arreglo a través de un ciclo while */
6
7 int main(){
8     #define TAMANO 5
9     int lista[TAMANO] = {10, 8, 5, 8, 7};
10     int indice = 0;
11     printf("\n\tLista\n");
12     while (indice < 5){
13         printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
14         indice += 1;
15     }
16     printf("\n\n");
17     return 0;
18 }
19 }
```

```
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ gcc -o codigo1 codigo1.c
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ ./codigo1

Lista
Calificación del alumno 1 es 10
Calificación del alumno 2 es 8
Calificación del alumno 3 es 5
Calificación del alumno 4 es 8
Calificación del alumno 5 es 7

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$
```

codigo1.c 4,68 Todo
"codigo1.c" 23L, 430C escritos

```
Alacritty
3 #include <stdio.h>
4 /*
5  Este programa genera un arreglo unidimensional de 5 elementos y accede a cada elemento del arreglo a través de un ciclo while.
6 */
7
8 int main(){
9
10 #define TAMANO 5
11 int lista[TAMANO] = {10, 8, 5, 8, 7};
12
13 int indice = 0;
14
15 printf("\n\tlista\n");
16 for (int indice = 0 ; indice < 5 ; indice++){
17     printf("\nCalaficación del alumno %d es %d", indice+1, lista[indice]);
18 }
19
20 printf("\n\n");
21
22 return 0;
23 }

codigo2.c 4,68 Todo
"codigo2.c" 23L, 441C escritos

Alacritty <2>
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ gcc -o codigo2 codigo2.c
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ ./codigo2

lista
Calaficación del alumno 1 es 10
Calaficación del alumno 2 es 8
Calaficación del alumno 3 es 5
Calaficación del alumno 4 es 8
Calaficación del alumno 5 es 7

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$
```

```
Alacritty
1 #include <stdio.h>
2 /*
3  Este programa crea un apuntador del tipo caracter.
4 */
5
6 int main (){
7
8 char *ap, c = 'a';
9 ap = &c;
10
11 printf("Caracter: %c\n",*ap);
12 printf("Código ASCII: %d\n",*ap);
13 printf("Dirección de memoria: %d\n",ap);
14
15 return 0;
16 }
17 }

codigo3.c 1,1 Todo

Alacritty <2>
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ gcc -o codigo3 codigo3.c
codigo3.c: In function 'main':
codigo3.c:14:35: warning: format '%d' expects argument of type 'int', but argument 2 has type 'char *' [-Wformat=]
14 |     printf("Dirección de memoria: %d\n",ap);
    |                                ^~      ~~
    |                                |      |
    |                                int   char *
    |                                %s
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ ./codigo3
Caracter: a
Código ASCII: 97
Dirección de memoria: 1107496047
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$
```

En el sistema Kubuntu, derivado de Debian, que es una distribución de Linux, al momento de compilar este código, muestra una advertencia referente a el tipo de dato que se maneja, sin embargo, es posible ejecutar el código con una correcta salida.

```
Alacritty
1 #include <stdio.h>
2 /*
3  Este programa accede a las localidades de memoria de distintas variables a través de un apuntador.
4 */
5
6 int main(){
7
8     int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
9     int *apEnt;
10    apEnt = &a;
11
12    printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n");
13    printf("apEnt = &a\n");
14
15    b = *apEnt;
16    printf("b = *apEnt \t-> b = %i\n", b);
17
18    b = *apEnt + 1;
19    printf("b = *apEnt + 1 \t-> b = %i\n", b);
20
21    *apEnt = 0;
22    printf("*apEnt = 0 \t-> a = %i\n", a);
23
24    apEnt = &c[0];
25    printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt);
26
27    return 0;
28
29 }
```

codigo4.c 1,18 Todo

"codigo4.c" 30L, 591C escritos

```
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ gcc -o codigo4 codigo4.c
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ ./codigo4
a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}
apEnt = &a
b = *apEnt      -> b = 5
b = *apEnt + 1  -> b = 6
*apEnt = 0      -> a = 0
apEnt = &c[0]    -> apEnt = 5
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$
```

```
Alacritty
3 #include <stdio.h>
4 /*
5  Este programa trabaja con aritmética de apuntadores para acceder a los valores de un arreglo.
6 */
7
8 int main(){
9     int arr[] = {5, 4, 3, 2, 1};
10    int *apArr;
11    apArr = arr;
12
13    printf("int arr[] = {5, 4, 3, 2, 1}; \n");
14    printf("apArr = &arr[0]\n");
15
16    int x = *apArr;
17    printf("x = *apArr \t -> x = %d\n", x);
18
19    x = *(apArr+1);
20    printf("x = *(apArr+1) \t -> x = %d\n", x);
21
22    x = *(apArr+2);
23    printf("x = *(apArr+2) \t -> x = %d\n", x);
24
25    return 0;
26
27 }
```

codigo5.c 4,76-75 Todo

"codigo5.c" 26L, 482C escritos

```
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ gcc -o codigo5 codigo5.c
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ ./codigo5
int arr[] = {5, 4, 3, 2, 1};
apArr = &arr[0]
x = *apArr      -> x = 5
x = *(apArr+1)  -> x = 4
x = *(apArr+2)  -> x = 3
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$
```

```
#include <stdio.h>
/*
 * Este programa genera un arreglo unidimensional de 5 elementos y accede a cada elemento del arreglo a través de un apuntador utilizando un ciclo for.
 */
int main(){
#define TAMANO 5
int lista[TAMANO] = {10, 8, 5, 8, 7};
int *ap = lista;
printf("\n\tLista\n");
for (int indice = 0 ; indice < 5 ; indice++){
printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice));
};
printf("\n\n");
return 0;
}
```

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10\$ gcc -o codigo6 codigo6.c

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10\$./codigo6

Lista

Calificación del alumno 1 es 10
Calificación del alumno 2 es 8
Calificación del alumno 3 es 5
Calificación del alumno 4 es 8
Calificación del alumno 5 es 7

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10\$

En particular, este código dentro de la práctica fue realizado de 2 maneras diferentes a parte de esta, donde observamos que ocupan el mismo número de línea y básicamente la misma estructura, sin embargo, este código sería más útil en programas que manejen un mayor número de datos, por el uso de un apuntador.

```
#include <stdio.h>
/*
 * Este programa muestra el manejo de cadenas en lenguaje C.
 */
int main(){
char palabra[20];
int i=0;
printf("Ingrese una palabra: ");
scanf("%s", palabra);
printf("La palabra ingresada es: %s\n", palabra);
for (i = 0 ; i < 20 ; i++){
printf("%c\n", palabra[i]);
}
return 0;
}
```

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10\$ gcc -o codigo7 codigo7.c

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10\$./codigo7

Ingrese una palabra: prueba

La palabra ingresada es: prueba

p
r
u
e
b
a

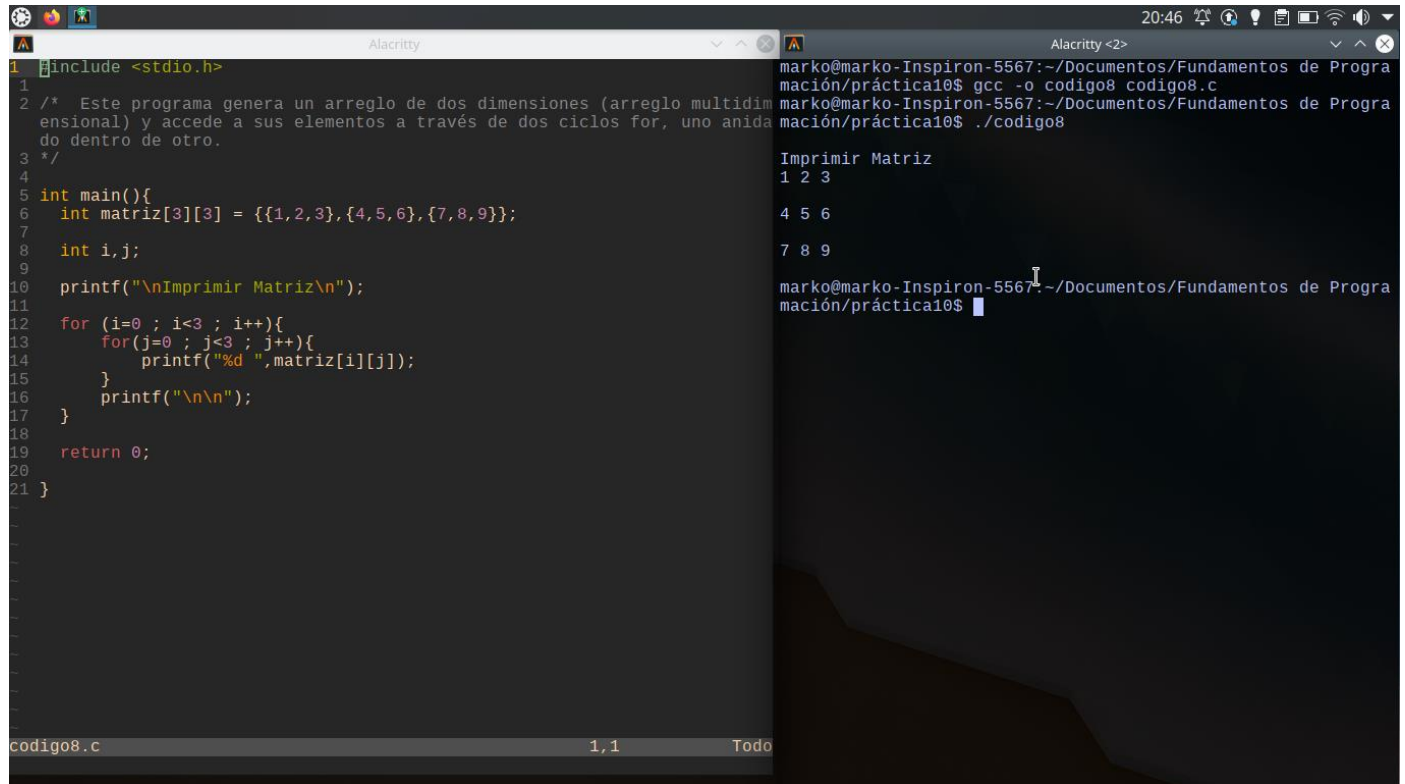
@
u
U

@
]

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10\$

La salida mostrada en esta captura de pantalla se debe a como está codificado el programa, char palabra [20] crea un arreglo que ocupará 20 espacios en memoria, la palabra ingresada tiene únicamente 5 caracteres, por ello, 15 espacios quedan libres, en el ciclo for, se describe que este se repetirá hasta que la variable i sea menor que 20, esto implica que aunque la palabra no

complete los 20 espacios, el ciclo se seguirá repitiendo, es por ellos que muestra en pantalla caracteres extra y espacios vacíos antes de volver a mostrar el texto de consola.



```
1 #include <stdio.h>
2 /* Este programa genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos a través de dos ciclos for, uno anidado dentro de otro.
3 */
4
5 int main(){
6     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
7
8     int i,j;
9
10    printf("\nImprimir Matriz\n");
11
12    for (i=0 ; i<3 ; i++){
13        for(j=0 ; j<3 ; j++){
14            printf("%d ",matriz[i][j]);
15        }
16        printf("\n\n");
17    }
18
19    return 0;
20 }
21 }
```

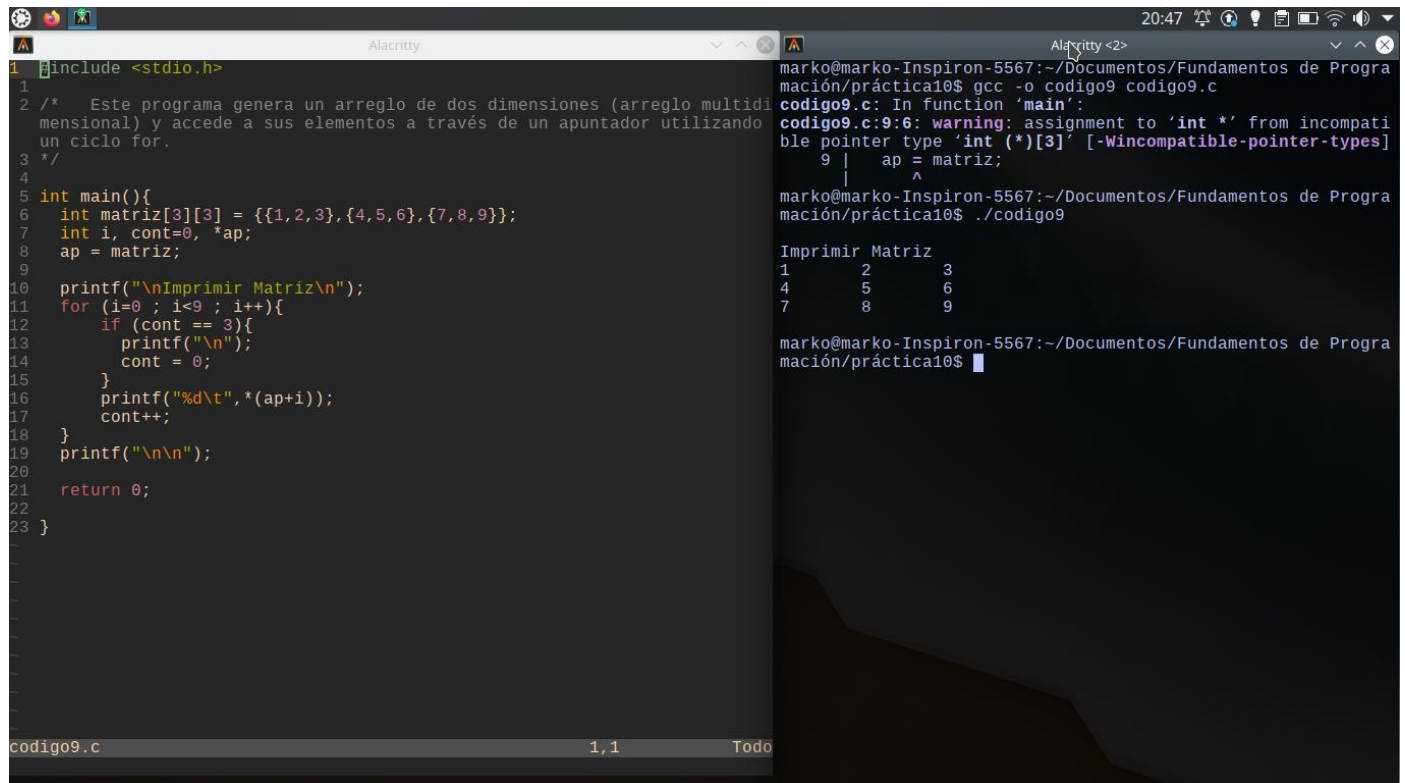
```
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ gcc -o codigo8 codigo8.c
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ ./codigo8

Imprimir Matriz
1 2 3

4 5 6

7 8 9

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$
```



```
1 #include <stdio.h>
2 /* Este programa genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos a través de un apuntador utilizando un ciclo for.
3 */
4
5 int main(){
6     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
7     int i, cont=0, *ap;
8     ap = matriz;
9
10    printf("\nImprimir Matriz\n");
11    for (i=0 ; i<9 ; i++){
12        if (cont == 3){
13            printf("\n");
14            cont = 0;
15        }
16        printf("%d\t",*(ap+i));
17        cont++;
18    }
19    printf("\n\n");
20
21    return 0;
22 }
23 }
```

```
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ gcc -o codigo9 codigo9.c
codigo9.c: In function 'main':
codigo9.c:9:6: warning: assignment to 'int *' from incompatible pointer type 'int (*)[3]' [-Wincompatible-pointer-types]
9 |     ap = matriz;
  |     ^
marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$ ./codigo9

Imprimir Matriz
1      2      3
4      5      6
7      8      9

marko@marko-Inspiron-5567:~/Documentos/Fundamentos de Programación/práctica10$
```

Conclusiones

El uso de los arreglos y apuntadores juega un papel importante al momento de manejar información que requiere ser almacenada de una forma especial, estas herramientas son de suma importancia en el diseño de bases de datos, ya que a partir de un solo dato se puede obtener todo lo contenido en una cierta parte del arreglo con mayor facilidad que si se hiciera sin ellos.

Referencias

Laboratorio de Computación Sala A y B. (2020). Facultad de Ingeniería, *Manual de prácticas de Fundamentos de Programación MADO-17 EP*. Recuperado de <http://lcp02.fi-b.unam.mx/poll/login/>