Clase 2: Lunes 10 de Marzo

Ampliación de Teoría de punteros, arrays y asignación dinámica de memoria

Índice

1.- Puntero a puntero

- Es posible declarar un puntero que apunte a otro puntero
- La notación : **
  - Ej: int **p2=&p1;

Declaración	Dirección (aleatoria)	Valor almacenado
int n=1;	0x122f	1
int *p1=&n;	0x133f	0x122f
int **p2=&p1;	0x144f	0x133f

1.- Puntero a puntero

- Ejercicio 1

```
#include <stdio.h>
main(){
  int **p2,*p1,i;

  i=1;
  p1 = &i;
  p2 = &p1;
}
```

```
a.- printf (“%X”,&i);
b.- printf (“%d",i);
c.- printf (“%d",*i);
d.- printf(“%X",&p1);
e.- printf(“%X",p1);
f.-  printf(“%d",p1);
g.- printf(“%d",*p1);
h.- printf(“%X",&p2);
i.-  printf(“%X",p2);
j.-  printf(“%d",p2);
k.- printf(“%d”,*p2);
l.-  printf(“%d",**p2);
```

2.- Punteros y arrays

- Cuando declaramos un array, estamos generando un puntero con el mismo nombre que apunta a la dirección del primer elemento del array.
  - Ej:

```
int[3], *puntero;
puntero=array; // puntero apunta a la dirección del
               // primer elemento de array
```

"array[1] es equivalente a *(puntero+1)"

2.- Punteros y arrays

- Ejercicio 2
  - Si queremos declarar un vector de enteros y asignar valores, ¿qué asignación sería la correcta?

```
#include <stdio.h>
main() {
   int array;            //No. Estamos definiendo una variable ordinaria
   array[3]={1,2,3};     // No tiene tipo de dato (int)
   int array[3]={1,2,3}; // OK
   array={1,2,3};    // No tiene tipo ni dimensión
   array{1,2,3};         // No tiene tipo, ni dimensión, ni '='
   array[]={1,2,3};  // No tiene dimensión ni tipo
}
```

2.- Punteros y arrays

- Ejercicio 3: ¿qué printf son correctos?

```
#include <stdio.h>
main() {
   int array[3]={1,2,3};
   int *puntero;
   printf("Debería imprimir esto %d \n",array[0]); //OK
   printf("Debería imprimir esto %d \n",*array); //OK
   printf("Debería imprimir esto %d \n",array); //Mal
}
```

Ejercicio 4 (Voluntario)

- Ejercicio 4 : Si queremos apuntar con un puntero a la dirección del primer elemento del array llamado "array", ¿qué opción de todas las que aparecen en azul sería la correcta? ¿por qué?:

```
#include <stdio.h>
main() {
   int array[3]={1,2,3};
   int *puntero;
   puntero=array[];
   puntero=*(array);
   puntero=*array;
   puntero=array;
   &puntero=array;
}
```

Solución ejercicio 4

- puntero=array; // correcta
- puntero=array[]; // error. No se indica la posición en los corchetes
- puntero=*(array); // compila pero se cuelga el programa. Igualamos posiciones a valores y el array no trabajaría así
- puntero=*array; // compila pero se cuelga el programa. Igualamos posiciones a valores y el array no trabajaría así
- &puntero=array; // error. Intentamos modificar la propia dirección del puntero en lugar de donde debe apuntar.

3.- Arrays de punteros

- Los punteros también pueden estructurarse en arrays.

  Ejemplo:

```
int *puntero[3],a=1;
puntero[1]=&a; //asignamos la dirección de una
               //variable entera
*puntero[1];//para acccder al valor de a
```

4.- scanf()

- Cada argumento de scanf debe ser un puntero
- Para variables que no sean string:
  - *Enteros*

    int x;
    scanf (“%d”, &x);

  - *Char*

    char letra ;
    scanf (“%c”, &letra);

4.- scanf()

- Para arrays de variables que no sean string
  - *Arrays unidimensionales*

int entero[21];
scanf(“%d”,entero); // solo recoge el 1er elemento del vector

es equivalente a:

scanf(“%d”,&entero[0]);

4.- scanf ()

- *Ejercicio 6:* Si quiseramos acceder al elemento 0 del array y modificarlo, ¿cual de los métodos de scanf() es el correcto para realizar la operación?

```
#include <stdio.h>
main() {
    int entero[2]={10,20};
    printf("Antes: %d\n", entero[0]);
    scanf("%d",entero); /OK
    scanf("%d",&entero); //OK
    scanf("%d",entero[0]); // Mal. Hace falta el &
    scanf("%d",&entero[0]); // OK
    printf("Despues: %d", entero[0]);
}
```

4.- scanf ()

– *Arrays multidimensionales*

```
float **matriz;
scanf("%f",&matriz[i][j]) // accedemos al elemento de la fila i,
                          //columna j
scanf("%f",*matriz); // accederíamos al primer elemento de
                     //la primera fila
scanf("%f",matriz); // error pq accedemos a la dirección de memoria
del puntero a puntero, es decir, si **matriz tiene asignado en
memoria la dirección fff2 accedemos a esa posición. Necesitamos
acceder a la dirección de la primera fila mediante *matriz
```

4.- scanf ()

- Para variables de cadenas de caracteres:

```
char palabra[21];
scanf("%s", palabra);
```

6.- Reserva dinámica de memoria en arrays mutidimensionales

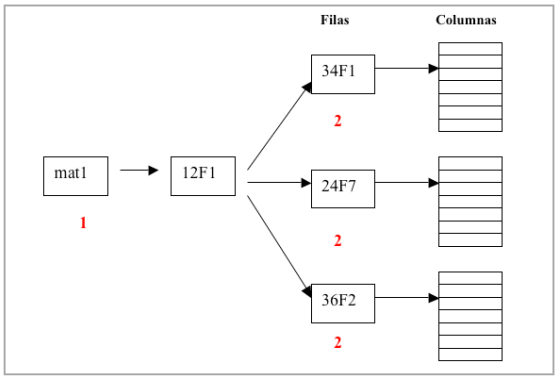
- *Reserva de memoria dinámica*

```
float **mat1;
mat1=(float **) malloc (filas*sizeof(float *)); // 1
for(i=0;i<filas;i++)  // 2
    mat1[i]=(float *) malloc(columnas*sizeof(float));
```

- *Liberación de memoria*

```
for(i=0;i<filas;i++)
    free(*(mat1+i));  // se libera cada uno de los vectores
free(mat1);          // se libera el puntero mat1
```

6.- Reserva dinámica de memoria en arrays mutidimensionales



Ejercicio 2. (Voluntario)

```
#include <stdio.h>
main(){
 int mat1[3][3];
 int i,j;

 mat1 = (int **)malloc(3*sizeof(int*));
 for(i=0; i < 3; i++)
    mat1[i] = (int *)malloc(3*sizeof(int));

 for(i=0;i<3;i++)
    for(j=0;j<3;j++)
    {
      printf("mat1[%d][%d]= ",i,j);
      scanf("%d",&mat1[i][j]);
      printf("\n%d\n",mat1[i][j]);
    }
 system("pause");
}
```

Cuando compilamos el siguiente ejercicio, el compilador nos proporciona el siguiente error:

"incompatible types in assignment"

¿Por qué?

Ejercicio 2 (Voluntario)

- Solución:

- En la línea 3 estamos realizando una reserva estática de memoria para los elementos de la matriz 3x3, y en las líneas 6,7 y 8 estamos realizando reserva de memoria dinámica.
- Debemos, por tanto, eliminar la reserva de memoria dinámica puesto que conocemos el número de elementos de la matriz, y como comentamos anteriormente, la reserva dinámica de memoria la debemos realizar cuando no conozcamos la dimensión o ésta sea variable.

Ejercicio 3 (Voluntario)

```
#include <stdio.h>
main(){
 int *mat1;
 int i,j;
mat1 = (int **)malloc(3*sizeof(int*));
 for(i=0; i < 3; i++)
     mat1[i] = (int *)malloc(3*sizeof(int));
 for(i=0;i<3;i++)
    for(j=0;j<3;j++)
    {
     printf("mat1[%d][%d]= ",i,j);
     scanf("%d",&mat1[i][j]);
        printf("\n%d\n",mat1[i][j]);
    }
 system("pause");
}
```

¿qué ocurrirá al compilar este programa?

Ejercicio 3 (Voluntario)

- Solución:

- Cuando compilemos el siguiente programa nos dará un error puesto que estamos declarando un puntero (array unidimensional) en el línea 3 y sin embargo estamos intentamos reservar memoria para un puntero de punteros.

7.- Paso de punteros como parámetros de funciones

	Caso	Declaración de la función	Llamada a la función	¿Qué se modifica?
1	Variable: Paso por valor	void CalculaNomina (float s)	CalculaNomina(sueldo);	nada
2	Variable: Paso por dirección	void CalculaNomina (float *s)	CalculaNomina(&sueldo);	Valor de sueldo
3	Puntero: paso por valor	Void CalculaNomina (float *s)	puntero=&sueldo; CalculaNomina(puntero);	Valor de sueldo
4	Puntero: paso por dirección	Void CalculaNomina (float *s)	Puntero=&sueldo; CalculaNomina(&puntero);	Valor de sueldo, dirección del puntero

9.- Paso de arrays multidimensionales como parámetros de una función

- Ejemplo:

```
Void LeerMatriz(float **m){
…
{
main(){
    …
    Float **mat1;
    LeerMatriz(mat1);
    }
```

Para que la función reciba el dato, deberemos llamarla con el nombre del array entre paréntesis y sin ningún *.