Clase IX:
Punteros
Funciones y arreglos

Informática I
Centro Regional Universitario Córdoba
UNDEF

08 de junio de 2021

Mecanismo de paso de argumentos a funciones I

Paso por valor

El valor del argumento es **copiado** en el parámetro de la subrutina, por lo cual si se realizan cambios en el mismo dentro de la función, el valor original no es modificado.

```
1  #include<stdio.h>
2
3  /*PROTOTIPO*/
4  int suma_tres(int);
5
6  int main (void)
7  {
8     int n1=0;
9     printf("Ingrese un numero \n");
10    scanf("%d",&n1);
11    int rdo;
12
```

Mecanismo de paso de argumentos a funciones II

```
13      printf("N1 antes de llamar a suma_tres %d \n",n1);
14      rdo=suma_tres(n1);
15      printf("N1 despues de llamar a suma_tres %d \n",n1);
16      printf("Valor de resultado %d \n",rdo);
17
18  }
19
20  int suma_tres (int num)
21  {
22      num=(num+3);
23      return(num);
24  }
```

Mecanismo de paso de argumentos a funciones III

Paso referencia

Se copia la *dirección de memoria* del argumento como parámetro de la función. En este caso, al realizar cambios en parámetro formal este si se ve afectado.

```
1  #include<stdio.h>
2
3  /*PROTOTIPO*/
4  int suma_tres(int *);
5
6  int main (void)
7  {
8     int n1=0;
9     int resultado;
10    printf("Ingrese un numero \n");
11    scanf("%d",&n1);
12
13    printf("N1 antes de llamar a suma_tres %d \n",n1);
```

Mecanismo de paso de argumentos a funciones IV

```
14      resultado=suma_tres(&n1);
15      printf("N1 despues de llamar a suma_tres %d \n",n1);
16      printf("Valor de resultado %d \n",resultado);
17
18  }
19
20  int suma_tres (int *n1)
21  {
22      int resultado=0;
23      *n1=*n1+3;
24      resultado=(*n1+3);
25      return(resultado);
26  }
```

¿Qué significan los símbolos * y &?

Punteros I

Definición

Los punteros son variables cuyos valores son direcciones de memoria. En general, esta dirección de memoria es la ubicación de otra variable.

La declaración de una variable puntero se realiza indicando el tipo de dato, seguido de un * y el nombre de la variable:

```
1  /*Puntero a entero*/
2  int *dato=NULL;
3  /*Puntero a float*/
4  float *dato=NULL;
5  /*Puntero a char*/
6  char *dato=NULL;
7  /*Puntero a puntero*/
8  int **dato=NULL;
```

Punteros II

Operador ampersand (&)

Se lo conoce como *operador de dirección*. Devuelve la dirección de memoria de un operando.

Operador *

Se lo conoce como operador *operador de indirección o desreferencia*, devuelve el valor del objeto al que apunta su operando

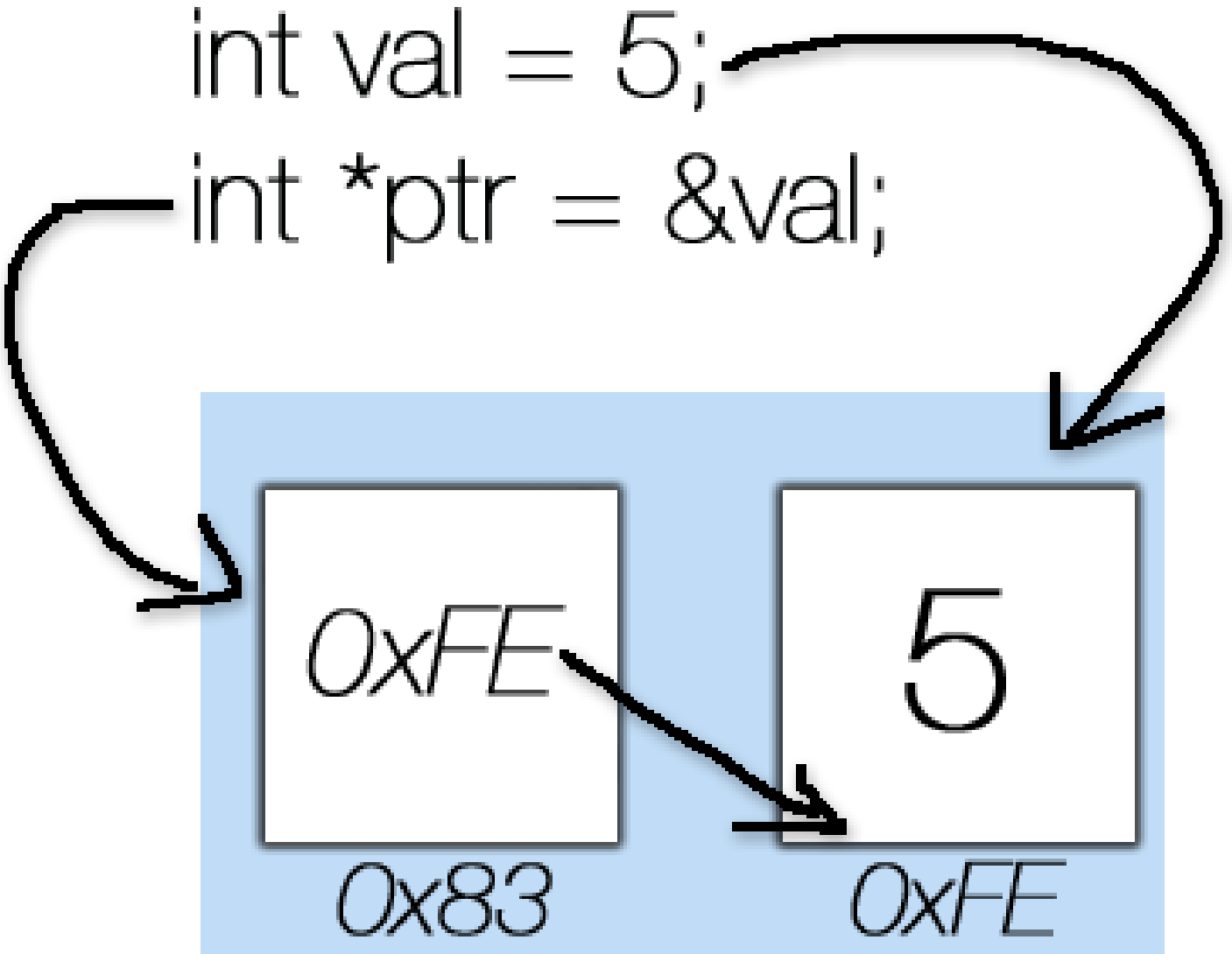
Ejemplo:

```
1  #include<stdio.h>
2  #include <stdlib.h>
3   int main(void)
4   {
5
6     int *ptr=NULL;
7     int   val=5;
```

Punteros III

```
 8
 9    ptr=&val;
10
11    printf("Valor de val %d\n",val);
12    printf("Posicion de memoria de val %X\n",&val);
13    printf("Valor de ptr %X\n",ptr);
14    printf("Posicion de memoria de ptr %X\n",&ptr);
15    printf("Val accedido por ptr %d\n",*ptr);
16
17  return(0);
18  }
```

Punteros IV



Volviendo a las funciones: paso por referencia I

```
1  #include<stdio.h>
2
3  /*PROTOTIPO*/
4  void swap(int *,int *);
5
6  int main (void)
7  {
8     int n1=0;
9     int n2=0;
10
11    printf("Ingrese un numero \n");
12    scanf("%d",&n1);
13    printf("Ingrese un numero \n");
14    scanf("%d",&n2);
15    printf("N1 antes de llamar a swap %d \n",n1);
16    printf("N2 antes de llamar a swap %d \n",n2);
17    swap(&n1,&n2);
18
```

Volviendo a las funciones: paso por referencia II

```
19      printf("N1 despues de llamar a swap %d \n",n1);
20      printf("N2 despues de llamar a swap %d \n",n2);
21      return(0);
22  }
23
24  void swap (int *n1, int *n2)
25  {
26      int auxiliar=0;
27      auxiliar=*n1;
28      *n1=*n2;
29      *n2=auxiliar;
30  }
```

Ejemplos I

1. Diseñar y codificar un programa que recibiendo desde la función main los catetos de un triángulo rectángulo, imprima en la función el resultado de la hipotenusa.
   ▸ Ver en github
2. Modificar el programa anterior para que el resultado sea impreso en la función main.
   ▸ Ver en github
3. Modificar el programa anterior para que el valor de los catetos sea pasado a la función por referencia y la impresión sea realizada en main. ▸ Ver en github
4. Modificar el programa anterior para que la impresión siga siendo realizada en main, pero la función debe tener el siguiente prototipo: void calculoHipotenusa(int *, int*, float *). ▸ Ver en github

Ejemplos II

5. Diseñar y codificar un programa que implemente las siguientes funciones:
   - Impresión de datos personales del desarrollador: debe ejecutarse al inicio del programa. Esta función no recibe ni retorna datos.
   - Menú de opciones: debe permitirle al operador seleccionar entre las siguientes opciones:
     1. Sumar dos números
     2. Restar dos números
     3. Imprimir mayor
     4. Imprimir menor

     La opción seleccionada debe ser retornada a main.
   - Implementar las funciones del apartado anterior. Los dos números deben ser enviados desde main por valor. El resultado de la suma y resta debe ser impreso en la función. La impresión del mayor y el menor debe hacerse en main().

▸ Ver en github

Ejemplos III

6. Modificar el programa anterior para que cada uno de los números sea enviado por referencia.

   ▸ Ver en github

Funciones y arreglos unidimensionales I

A diferencia de las variables en las que podemos elegir pasarlas a una función por valor o referencia, los arreglos sólo se pasan a funciones **mediante referencia**.
Es decir que la función trabajará SIEMPRE con el arreglo original y no con una copia del mismo.

¿Cómo le indicamos a la función que va a recibir un arreglo?

Hay tres formas de señalar esto en el prototipo de la función:

- Como un array con tipo definido y sin dimensiones
- Como un array con tipo y dimensiones definidas
- Como un puntero

En C/C++ el nombre de un arreglo es un puntero al primer elemento del mismo.

Funciones y arreglos unidimensionales II

Ejemplo utilizando un array con tipo definido y sin dimensiones:

```
1  #define TAM 10
2  /*PROTOTIPOS*/
3  void cargar_vector(int []);
4  void imprimir_vector(int []);
5
6  int main (void)
7  {
8     int array_1d[TAM]={0};
9     /*LLAMADA A FUNCIONES*/
10    cargar_vector(array_1d);
11    imprimir_vector(array_1d);
12    return(0);
13 }
14
15
16
17
```

Funciones y arreglos unidimensionales III

```
18
19
20
21
22  void cargar_vector (int vector[])
23  {
24     int ii;
25     int dato;
26     for (ii=0;ii<TAM;ii++)
27     {
28        printf("Ingrese el elemento %d\n",ii);
29        scanf("%d",&dato);
30        vector[ii]=dato;
31     }
32  }
33
34
35
36
```

Funciones y arreglos unidimensionales IV

```
37
38
39
40  void imprimir_vector (int vector[])
41  {
42     int ii;
43     for (ii=0;ii<TAM;ii++)
44     {
45        printf(" %d",vector[ii]);
46     }
47  }
```

▸ Ver ejemplo completo en github

Funciones y arreglos unidimensionales V

Ejemplo utilizando un array con tipo y dimensiones definidas:

```
1  #include<stdio.h>
2  #define TAM 10
3  /*PROTOTIPO*/
4  void cargar_vector(int vector[TAM]);
5  void imprimir_vector(int vector[TAM]);
6
7
8
9  int main (void)
10 {
11    int array_1d[TAM]={0};
12    cargar_vector(array_1d);
13    imprimir_vector(array_1d);
14 }
15
16
17
```

Funciones y arreglos unidimensionales VI

```
18
19
20
21  void cargar_vector (int vector[TAM])
22  {
23     int ii;
24     int dato;
25     for (ii=0;ii<TAM;ii++)
26     {
27        printf("Ingrese el elemento %d\n",ii);
28        scanf("%d",&dato);
29        vector[ii]=dato;
30     }
31  }
32
33
34
35
36
```

Funciones y arreglos unidimensionales VII

```
37
38  void imprimir_vector (int vector[TAM])
39  {
40     int ii;
41     for (ii=0;ii<TAM;ii++)
42     {
43        printf(" %d",vector[ii]);
44     }
45  }
```

▸ Ver ejemplo completo en github

Funciones y arreglos unidimensionales VIII

Ejemplo utilizando un puntero :) :

```
1  #include<stdio.h>
2  #define TAM 10
3  /*PROTOTIPO*/
4  void cargar_vector(int *);
5  void imprimir_vector(int *);
6
7
8
9  int main (void)
10 {
11    int array_1d[TAM]={0};
12    cargar_vector(array_1d);
13    imprimir_vector(array_1d);
14 }
15
16
17
```

Funciones y arreglos unidimensionales IX

```
18
19
20
21  void cargar_vector (int *vector)
22  {
23     int ii;
24     int dato;
25     for (ii=0;ii<TAM;ii++)
26     {
27        printf("Ingrese el elemento %d\n",ii);
28        scanf("%d",&dato);
29        vector[ii]=dato;
30     }
31  }
32
33
34
35
36
```

Funciones y arreglos unidimensionales X

```
37
38  void imprimir_vector (int *vector)
39  {
40     int ii;
41     for (ii=0;ii<TAM;ii++)
42     {
43        printf(" %d",vector[ii]);
44     }
45  }
```

Funciones y arreglos unidimensionales XI

Escritura de una función genérica para trabajar con arreglos del mismo tipo, pero distinta longitud.

```
1  #include<stdio.h>
2  /*PROTOTIPO*/
3  void cargar_vector(int [], int);
4  void imprimir_vector(int [],int);
5
6
7
8
9
10
11
12
13
14
15
```

Funciones y arreglos unidimensionales XII

```
16  int main (void)
17  {
18     int array_1[10]={0};
19     int array_2[3]={0};
20     printf("Carga del primer arreglo\n");
21     cargar_vector(array_1,10);
22     printf("Carga del segundo arreglo\n");
23     cargar_vector(array_1,3);
24     printf("Impresion del primer arreglo\n");
25     imprimir_vector(array_1,10);
26     printf("Impresion del segundo arreglo\n");
27     imprimir_vector(array_1,3);
28     printf("Reutilizamos las funciones!!!\n");
29     return(0);
30
31  }
32
33
34
```

Funciones y arreglos unidimensionales XIII

```
35
36  void cargar_vector (int vector[], int tam)
37  {
38     int ii;
39     int dato;
40     for (ii=0;ii<tam;ii++)
41     {
42        printf("Ingrese el elemento %d\n",ii);
43        scanf("%d",&dato);
44        vector[ii]=dato;
45     }
46  }
47
48
49
50
51
52
53
```

Funciones y arreglos unidimensionales XIV

```
54
55  void imprimir_vector (int vector[], int tam)
56  {
57     int ii;
58     for (ii=0;ii<tam;ii++)
59     {
60        printf(" %d",vector[ii]);
61     }
62     printf("\n");
63  }
```

▸ Ver ejemplo completo en github

Ejemplos I

1. Diseñar y codificar un programa que implemente las siguientes funciones:
   - Saludo: imprimir los datos personales del desarrollador.
     Prototipo: void saludo(void);
   - Menú: permitir al operador seleccionar una de las siguientes opciones:
     - Cargar un vector de 10 elementos de tipo entero
     - Imprimir el vector
     - Imprimir el mayor elemento dentro del arreglo
     - Imprimir el menor elemento dentro del arreglo
     - Imprimir la media de todos los elementos del arreglo
     - Imprimir los elementos mayores a la media
     - Imprimir los elementos menores a la media

     Prototipo: int menu(void);
   - Cargar un vector de 10 elementos de tipo entero.
     Prototipo: void cargar (int []);
   - Imprimir el vector
     Prototipo: void imprimir (int []);

Ejemplos II

- Imprimir el mayor elemento dentro del arreglo.
  Prototipo: void imprimeMayor (int []);
- Impimir el menor elemento dentro del arreglo
  Prototipo: void imprimeMenor (int []);
- Imprimir en main la media de todos los elementos del arreglo.
  Prototipo: float calculaMedia (int []);
- Imprimir los elementos mayores a la media, donde la media es recibida como parámetro desde la función main()
  Prototipo: void imprimeMayoresMedia (int [],float);
- Imprimir los elementos menores a la media, donde la media se obtiene llamando a la función calculaMedia desde imprimeMenoresMedia.
  Prototipo: void imprimeMenoresMedia (int []);

▸ Ver en github

Funciones y arreglos bidimensionales I

Para el caso de los arreglos bidimensionales, es necesario indicar en el prototipo de la función la cantidad de columnas que tiene arreglo. Este parámetro es utilizado por el compilador para poder determinar la posición de memoria de cada elemento y así poder operar con ellos.

```
1  #include<stdio.h>
2  #define FIL 2
3  #define COL 3
4
5  /*PROTOTIPOS*/
6  void cargar_matriz(int [][COL]);
7  void imprimir_matriz(int [][COL]);
8
9
10
11
```

Funciones y arreglos bidimensionales II

```
12
13
14
15
16  int main (void)
17  {
18      int array_2d[FIL][COL];
19      /*LLAMADA A FUNCIONES*/
20      cargar_matriz(array_2d);
21      imprimir_matriz(array_2d);
22      return(0);
23  }
24
25
26
27
28
29
30
```

Funciones y arreglos bidimensionales III

```
31  void cargar_matriz (int matriz[][COL])
32  {
33     int ii;
34     int jj;
35     int dato;
36     for (ii=0;ii<FIL;ii++)
37     {
38        for(jj=0;jj<COL;jj++)
39        {
40        printf("Ingrese el elemento %d %d\n",ii,jj);
41        scanf("%d",&dato);
42        matriz[ii][jj]=dato;
43        }
44
45     }
46  }
47
48
49
```

Funciones y arreglos bidimensionales IV

```
50
51  void imprimir_matriz (int matriz[][COL])
52  {
53     int ii;
54     int jj;
55     int dato;
56     for (ii=0;ii<FIL;ii++)
57     {
58        for(jj=0;jj<COL;jj++)
59        {
60        printf("%d \t",matriz[ii][jj]);
61        }
62        printf("\n");
63     }
64  }
```

▸ Ver en github

Ejemplos I

1. Diseñar y codificar un programa declare en main un arreglo de 7 filas por 9 columnas y luego implemente las siguientes funciones:
   - Cargar arreglo
   - Imprimir arreglo
   - Imprimir todas las columnas de una fila especifica. El valor de la fila es recibido por valor desde main.
   - Imprimir todas las filas de una columna especifica. El valor de la fila es recibido por referencia desde main..
   - Menú con las opciones citadas anteriormente

▸ Ver en github

Ejemplos II

2. Una estación meteorológica toma muestras de temperatura y humedad una vez por hora durante 30 días Dicha información se almacena en dos arreglos bidimensionales declarados de la siguiente manera:

```
int temperatura [30][24];
int humedad [30][24];
```

Diseñar y codificar un programa en C que implemente las siguientes funciones:

- Cargar arreglos: debe cargar valores aleatorios en los arreglos. La temperatura debe variar entre $-10^o C$ y $50^o C$ y la humedad entre 0 % y 100 %
- Imprimir las 24 muestras de temperatura y humedad de un día. El número de día se recibe por valor desde la función main
- Imprimir la mayor temperatura de un día. El número de día se recibe por referencia desde la función main

Ejemplos III

- Imprimir la menor temperatura de un día. El número de día se recibe por referencia desde la función main
- Imprimir el día y la hora de la temperatura máxima y mínima
- Imprimir el promedio de temperaturas y humedades de cada uno de los días

▸ Ver en github