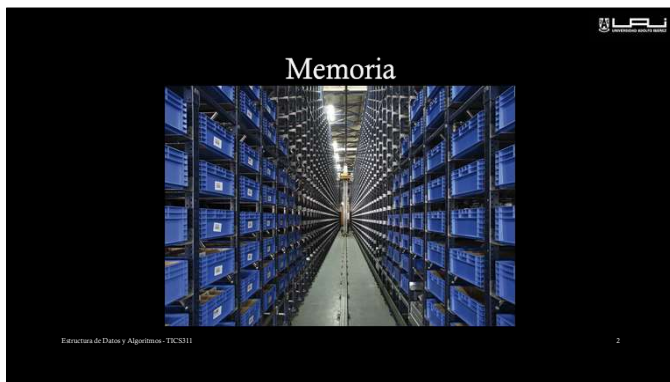
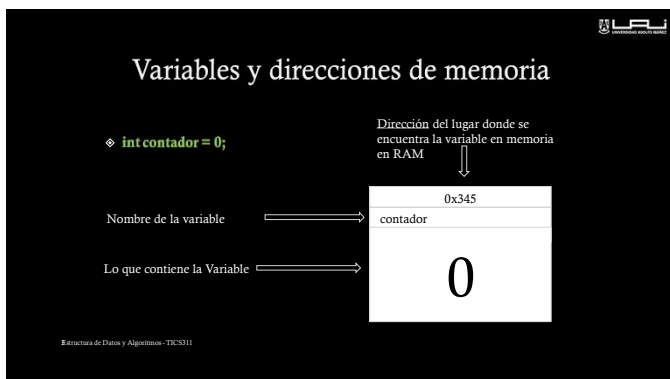


1



2



3

Direcciones de memoria

Las direcciones de memoria se escriben, por convención, usando **representación hexadecimal** (16 dígitos).

$$16^3 \ 16^2 \ 16^1 \ 16^0$$

$$0x \ 0 \ 1 \ A \ F$$

↓

$$0 + 256 + 160 + 15 = 431$$

Dec.	Hexadec.	Dec.	Hexadec.
0	0x0	16	0x10
1	0x1	17	0x11
2	0x2	18	0x12
3	0x3	19	0x13
4	0x4	20	0x14
5	0x5	21	0x15
6	0x6	22	0x16
7	0x7	23	0x17
8	0x8	24	0x18
9	0x9	25	0x19
10	0xA	26	0x1A
11	0xB	27	0x1B
12	0xC	28	0x1C
13	0xD	29	0x1D
14	0xE	30	0x1E
15	0xF	31	0x1F

Estructura de Datos y Algoritmos - TIC3011

4

Punteros

◆ ¿Qué es un puntero?

◆ Es una variable que almacena una dirección de memoria

```
int main()
{
  int x=1;
  int *y;
```

x: Es una variable que **almacena un valor entero** (en este caso)

y: Es un puntero (variable que **almacena una dirección de memoria**) a un entero (en este caso).

¿Puntero a un entero? 🤖


Quiere decir que en la dirección de memoria almacenada por el puntero hay un entero

Estructura de Datos y Algoritmos - TIC3011

5


Variables vs. Punteros

Variable entera x
int x = 1;



0x54F1
(Ejemplo de la dirección de memoria donde "vive" la variable x)

Puntero a entero y
int *v;



(Ejemplo de la dirección de memoria donde "vive" el puntero y)

Aquí se va a almacenar la dirección de memoria de una variable entera

¿Puedo asignar yo (como programador) una dirección de memoria al puntero y?


NO

Estructura de Datos y Algoritmos - TIC3011

6


Variables vs. Punteros

Variable entera x
`int x = 1;`



(Ejemplo de la dirección de memoria donde "vive" la variable x)

Puntero a entero y
`int *y;`



(Ejemplo de la dirección de memoria donde "vive" el puntero y)

Aquí se va a almacenar la dirección de memoria de una variable entera

¿Cómo se asigna una dirección de memoria al puntero y?

a) Pides al sistema que asigne una (más tarde en este curso)

b) Le asignas la dirección de una variable que ya existe.

Estructura de Datos y Algoritmos - TICS311

7

Asignando la dirección de memoria de una variable existente

◆ ¿Qué es un puntero?

◆ Es una variable que almacena una dirección de memoria

```
int main()
{
    int x=1,
    int *y;

    y = &x;
}
```

`&x`: dirección de memoria de la variable x

El operador unitario `&` entrega la dirección de memoria de una variable.

Estructura de Datos y Algoritmos - TICS311

8

Ejemplo 1



```
#include <stdio.h>

int main(void)
{
    int a=5;
    int *p;

    p = &a;

    printf("El valor almacenado en a es: %i\n", a);
    printf("La dirección de memoria de la variable a es: %p\n", &a);
    printf("La dirección de memoria almacenada en el puntero p es: %p\n", p);
    printf("La dirección de memoria del puntero p es: %p\n", &p);

    return 0;
}
```

Estructura de Datos y Algoritmos - TICS311

9

Punteros

◆ Podemos alterar el contenido de la variable apuntada por un puntero

```

int main()
{
    int a=1,
    int *p;
    p= &a;
    *p=3;
}

```

OJO: cuando se declara un puntero, el símbolo `*` no es un operador, sino un "recordatorio" de que la variable es un puntero

`*p`: operador con el que accedemos al **contenido** apuntado por la variable puntero.

Estructura de Datos y Algoritmos - TIC3011

10

Punteros

◆ Podemos alterar el contenido de la variable apuntada por un puntero

```

1. int main()
2. {
3.     int a=1,
4.     int *p;
5.
6.     p= &a;
7.     *p=3;
8. }

```

Línea 3. Se crea variable `a`, que almacena valor 1

Línea 4. Se crea puntero a entero `p`

Línea 6. Se almacena la dirección de memoria de la variable `a` en puntero `p` (`p` "queda apuntando" a variable `a`)

Línea 7. Se almacena número 3 en la dirección apuntada por puntero `p`

Estructura de Datos y Algoritmos - TIC3011

11

Punteros

◆ Podemos alterar el contenido de la variable apuntada por un puntero

```


1. int main()
2. {
3.     int a=1,
4.     int *p;
5.
6.     p= &a;
7.     *p=3;
8. }

```

Estructura de Datos y Algoritmos - TIC3011

12

RESUMEN




- ▶ ¿Qué es un puntero?
 - ♦ Es una variable que almacena una dirección de memoria
- ♦ Operadores unitarios de punteros
 - ♦ & : entrega la dirección de memoria de una variable
 - ♦ * : acceso al valor contenido en una dirección.


Estructura de Datos y Algoritmos - TIC3011


13

PUNTEROS Y FUNCIONES



PASO POR VALOR Y PASO POR REFERENCIA






Estructura de Datos y Algoritmos - TIC3011

14

RECORDATORIO FUNCIONES



¿Qué información se despliega por pantalla al ejecutar este programa?

```
#include <stdio.h>

void intercambia (int a, int b);

int main(void)
{
    int a=1;
    int b=3;

    printf("Valor de a: %i. Valor de b: %i\n", a,b);
    intercambia(a,b);
    printf("Valor de a: %i. Valor de b: %i\n", a,b);
}


void intercambia(int a,int b)
{
    int aux;

    aux=b;
    b=a;
    a=aux;
    printf("Dentro de función, a: %i, b: %i\n", a,b);
}
```

Estructura de Datos y Algoritmos - TIC3011

15

RECORDATORIO FUNCIONES



PASO POR VALOR
(LA FUNCIÓN RECIBE COPIAS)

```
#include <stdio.h>

void intercambia (int a, int b);

int main(void)
{
    int a=1;
    int b=3;

    printf("Valor de a: %i. Valor de b: %i\n", a,b);
    intercambia(a,b);
    printf("Valor de a: %i. Valor de b: %i\n", a,b);
}

void intercambia(int a,int b)
{
    int aux;

    aux=b;
    b=a;
    a=aux;
    printf("Dentro de función, a: %i, b: %i\n", a,b);
}
```


Valor de a: 1. Valor de b: 3
 Dentro de función, a: 3, b: 1
 Valor de a: 1. Valor de b: 3

Estructura de Datos y Algoritmos - TIC3011

16

PASO DE VALORES POR REFERENCIA (ORIGINAL)

Si a una función se le entregan **las direcciones de memoria** de las variables (en vez de sus valores), la función puede **acceder directamente al original** (el contenido de la variable).



Estructura de Datos y Algoritmos - TIC3011

17

RECORDATORIO FUNCIONES

¿Qué información se despliega por pantalla al ejecutar este programa?

```
#include <stdio.h>

void intercambia (int a, int b);

int main(void)
{
    int a=1;
    int b=3;

    printf("Valor de a: %i. Valor de b: %i\n", a,b);
    intercambia( a, b);
    printf("Valor de a: %i. Valor de b: %i\n", a,b);
}


void intercambia(int a,int b)
{
    int aux;

    aux= b;
    b= a;
    a=aux;
    printf("Dentro de función, a: %i, b: %i\n", a, b);
}
```

Estructura de Datos y Algoritmos - TIC3011

18

RECORDATORIO FUNCIONES



Valor de a: 1. Valor de b: 3
Dentro de función, a: 3, b: 1
Valor de a: 3. Valor de b: 1

```
#include <stdio.h>

void intercambia (int a, int b);

int main(void)
{
    int a=1;
    int b=3;

    printf("Valor de a: %i. Valor de b: %i\n", a,b);
    intercambia(&a, &b);
    printf("Valor de a: %i. Valor de b: %i\n", a,b);
}

void intercambia(int a,int b)
{
    int aux;

    aux= b;
    b= a;
    a=aux;
    printf("Dentro de función, a: %i, b:%i\n", a, b);
}
```

Estructura de Datos y Algoritmos - TIC3011

19

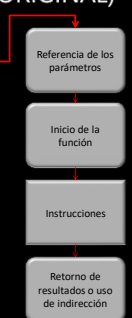
PASO DE VALORES POR REFERENCIA (ORIGINAL)

Llamado de la función

```
mi_funcion(&var1,...,&varN);
```

Ejemplo función

```
tipo1 *funcion(tipo1 *param1,..., tipoN *paramN)
{
    /* Instrucciones de
    mi función */
}
```



Estructura de Datos y Algoritmos - TIC3011

20

Paso de punteros por referencia

Dirección de memoria	Contenido
1	
...	
34	x
35	y
36	
37	
38	
49	
50	
51	
52	
53	

```
int main()
{
    int x,y;
```

Estructura de Datos y Algoritmos - TIC3011

21

Paso de punteros por referencia

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y=Getint();
}

```

Supongamos que usted escribe "28" y "12", y presiona enter

Estructura de Datos y Algoritmos - TICS311

22

Paso de punteros por referencia

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y=Getint();
}

```

Supongamos que usted escribe "28" y "12", y presiona enter

Estructura de Datos y Algoritmos - TICS311

23

Paso de punteros por referencia

```

int main()
{
    int x, y;

    intercambie(&x, &y);
}

```

¿Qué valor enviamos?
¿La dirección 34 o bien el valor 28?

Estructura de Datos y Algoritmos - TICS311

24

Paso de punteros por referencia

Variable	Dirección de memoria	Contenido
x	34	28
y	35	12
a	50	
b	51	

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y =Getint();

    intercambio(&x, &y);
}

void intercambio(int *a, int *b)
{
    // ¿Qué valor recibe? ¿Qué valor recibe?
}

```

Estructura de Datos y Algoritmos - TICS311

25

Paso de punteros por referencia

Variable	Dirección de memoria	Contenido
x	34	28
y	35	12
a	50	34
b	51	35

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y =Getint();

    intercambio(&x, &y);
}

void intercambio(int *a, int *b)
{
    // La función recibe una referencia.
    // Dado que *a es una variable puntero, recibe la dirección 34
}

```

Estructura de Datos y Algoritmos - TICS311

26

Paso de punteros por referencia

Variable	Dirección de memoria	Contenido
x	34	28
y	35	12
a	50	34
b	51	35
temp	52	

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y =Getint();

    intercambio(&x, &y);
}

void intercambio(int *a, int *b)
{
    int temp;
}

```

Estructura de Datos y Algoritmos - TICS311

27

Paso de punteros por referencia

	Dirección de memoria	Contenido
	1	
	...	
x	34	28
y	35	12
	36	
	37	
	38	
	49	
a	50	34
b	51	35
temp	52	
	53	

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y =Getint();

    intercambio(&x, &y);

}

void intercambio(int *a, int *b)
{
    int temp;
    temp= *a;  → Accedemos al valor contenido de *a
    *a = *b;
    *b = temp;
}

```

Estructura de Datos y Algoritmos - TICS311

28

Paso de punteros por referencia

	Dirección de memoria	Contenido
	1	
	...	
x	34	28
y	35	12
	36	
	37	
	38	
	49	
a	50	34
b	51	35
temp	52	28
	53	

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y =Getint();

    intercambio(&x, &y);

}

void intercambio(int *a, int *b)
{
    int temp;
    temp= *a;  → Accedemos al valor contenido de *a
    *a = *b;
    *b = temp;
}

```

Estructura de Datos y Algoritmos - TICS311

29

Paso de punteros por referencia

	Dirección de memoria	Contenido
	1	
	...	
x	34	28
y	35	12
	36	
	37	
	38	
	49	
a	50	34
b	51	35
temp	52	28
	53	

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y =Getint();

    intercambio(&x, &y);

}

void intercambio(int *a, int *b)
{
    int temp;
    temp= *a;  → Accedemos al valor contenido de *a
    *a = *b;
    *b = temp;
}

```

Estructura de Datos y Algoritmos - TICS311

30

Paso de punteros por referencia

Dirección de memoria	Contenido
1	
...	...
34	12
35	12
36	
37	
38	
49	
50	34
51	35
52	28
53	

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y =Getint();
    intercambio(&x, &y);
}

void intercambio( int *, int *)
{
    int temp;
    temp= *a;
    *a = *b;
    *b = temp;
}
  
```

Asignamos 12 contenido apuntado por *a

31

Paso de punteros por referencia

Dirección de memoria	Contenido
1	
...	...
34	12
35	28
36	
37	
38	
49	
50	34
51	35
52	28
53	

```

int main()
{
    int x, y;

    printf("ingrese valor 1, valor 2");
    x= Getint(); y =Getint();
    intercambio(&x, &y);
}

void intercambio( int *, int *)
{
    int temp;
    temp= *a;
    *a = *b;
    *b = temp;
}
  
```

copiamos el valor temp al contenido apuntado por *b

32

Paso de punteros por referencia

ANTES

Dirección de memoria	Contenido
1	
...	...
34	28
35	12
36	
37	
38	
49	
50	
51	
52	
53	

DESPUÉS

Dirección de memoria	Contenido
1	
...	...
34	12
35	28
36	
37	
38	
49	
50	34
51	35
52	28
53	

33

Función Scanf()

La función estándar de C para recuperar datos desde teclado es `scanf()`:

```
scanf("%d", &var)
```

Por ejemplo:

```
#include <stdio.h>

int main(void)
{
    int x;
    printf("Ingresa número\n");
    scanf("%d", &x);
    printf("Número ingresado: %d\n", x);
}
```

38

Función scanf()

En el caso de un arreglo:

```
#include <stdio.h>

int main()
{
    int i, vector[5];

    for(i=0;i<5;i++)
        printf(" Ingresa el valor %d del vector:",i);

    scanf("%d", &vector[i]);
}
```

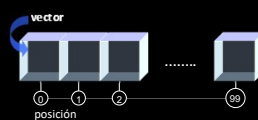
39

Paso por referencia de arreglos

```
#include <stdio.h>

int main()
{
    int vector[100];
}
```

Quiero hacer una función que reciba como argumento un arreglo para agregarle valores ¿Cómo lo hago?



40

Paso por referencia de arreglos

```
#include <stdio.h>
int main()
{
    int vector[100];
    mi_funcion(vector);
}
```

enviamos el vector como referencia.

El nombre del arreglo almacena la dirección de memoria del primer elemento del arreglo.

■ Estructura de Datos y Algoritmos - TICS311

41

Paso por referencia de arreglos

```
#include <stdio.h>
int main()
{
    int
}

int mi_funcion(int vector[])
{
    La función recibe la referencia.
}
```

■ Estructura de Datos y Algoritmos - TICS311

42

Paso por referencia de arreglos

```
#include <stdio.h>
int main()
{
}

int mi_funcion(int mi_vector[])
{
    int i;
    for (i=0; i<100; i++)
    {
        printf("Ingrese el valor %i del vector:", i);
        scanf("%d", &vector[i]);
    }
    return(0);
}
```

La función recibe la referencia.

■ Estructura de Datos y Algoritmos - TICS311

43

Ejemplo

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void llena_arreglo(int arreglo[]);
void imprime_arreglo(int arreglo[]);

int main(void)
{
    int A1[10], A2[10];
    srand(time(NULL));
    llena_arreglo(A1);
    llena_arreglo(A2);
    imprime_arreglo(A1);
    imprime_arreglo(A2);
}

void llena_arreglo(int arreglo[]){
    for(int i=0; i<10; i++)    arreglo[i]=rand()%10;}

void imprime_arreglo(int arreglo[]){
    for(int i=0; i<10; i++)    printf("%i\t", arreglo[i]);
    printf("\n");}
```

#Estructura de Datos y Algoritmos - TICS3011

44

```
#include <stdio.h>

void llena_arreglo(int arreglo[]);
void suma_2(int *x);
void imprime_arreglo(int arreglo[]);

int main(void)
{
    int A1[5], A2[5];
    llena_arreglo(A1);
    llena_arreglo(A2);
    suma_2(&A1[3]);
    imprime_arreglo(A1);
    imprime_arreglo(A2);
}

void llena_arreglo(int arreglo[]){
    for(int i=0; i<5; i++)    arreglo[i]=i;}

void suma_2(int *x)
{ *x=*x+2;      *x=*x+3;}

void imprime_arreglo(int arreglo[]){
    for(int i=0; i<5; i++)    printf("%i\t", arreglo[i]);
    printf("\n");}
```

Nota

```
También se puede escribir así:
void suma_2(int *x)
{
    x[0]=x[0]+2;
    x[1]=x[1]+3;}
```

#Estructura de Datos y Algoritmos - TICS3011

49

RESUMEN

Contenido	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]
Dirección	&A[0] &A[1] &A[2]...						
Dirección	A A+1 A+2 						

#Estructura de Datos y Algoritmos - TICS3011

50
