Experiencia N°2

Métodos de Programación

1-2022

UNIVERSIDAD DE SANTIAGO DE CHILE

CONTENIDO

- Introducción
- Solicitando información al usuario
- Operadores & y *
- Bifurcaciones
- Iteraciones
- Actividad 1
- Funciones
- Paso por valor
- Paso por referencia
- Actividad 2

UNIVERSIDAD DE SANTIAGO DE CHILE

Introducción

- En la sección anterior se utilizó la función `printf` de la biblioteca stdio.h.
- `stdio.h` posee otras funciones que pueden ser de interés, una página que facilita información sobre estas es:

http://c.conclase.net/

UNIVERSIDAD DE SANTIAGO DE CHILE

Solicitando información al usuario

scanf es una función que permite capturar información a través de la entrada estándar (consola).

`scanf(Arg1,Arg2);`

Arg1: carácter de formato tipo de dato.

Arg2: dirección de memoria de la variable.

```
int a;
printf("Ingrese un numero: ");
scanf("%d",&a);
printf("%d", a);
```

Primer argumento:

Necesita la especificación del tipo de dato a capturar a través de una carácter de formato (%d, %f, %c, etc.).

Segundo argumento:

Necesita la dirección de memoria de la variable en donde se desea guardar el valor capturado.

UNIVERSIDAD DE SANTIAGO DE CHILE

Operadores & y *

Existen dos operadores unarios que nos permitirán trabajar las direcciones de memoria en C:

Referenciación (&): Anteponerlo a una variable permite obtener la dirección de memoria de ésta.

```
int a = 5;
```

&a → 0061FF28

Desreferenciación (*): Anteponerlo a una dirección de memoria permite obtener el valor guardado en ésta.

Declaración de una dirección de memoria para un entero . "puntero a entero"

```
int * p;
p = &a;
```

p → 0061FF28

*p → 5

UNIVERSIDAD DE SANTIAGO DE CHILE

Operadores & y *

scanf es una función que permite capturar información a través de la entrada estándar (consola).

`scanf(Arg1,Arg2);`

Arg1: carácter de formato tipo de dato.

Arg2: dirección de memoria de la variable.

```
int a;
printf("Ingrese un numero: ");
scanf("%d",&a);
```

En la dirección de memoria de la variable a se guarda el valor capturado.

O sea, a tiene el valor capturado.

A diferencia del input de Python, scanf no permite mostrar un mensaje al usuario, sino que se debe enviar a parte.

UNIVERSIDAD DE SANTIAGO DE CHILE

Bifurcaciones

Recordemos como se realizan las sentencias if, else if y else

```
1  if(<condicion1>){
2      <Instrucciones en caso condicion1>
3  }else if(<condicion2>){
4      <Instrucciones en caso verdadero condicion2>
5  }else{
6      <Instrucciones en caso falso ambas condiciones>
7  }
```

Sentencia	Se activa si	
	Valor booleano if	Valor booleano else if
Instrucciones en caso condicion1	Verdadero	Verdadero o falso
Instrucciones en caso verdadero condicion2	Falso	Verdadero
Instrucciones en caso falso ambas condiciones	Falso	Falso

UNIVERSIDAD DE SANTIAGO DE CHILE

Bifurcaciones

Recordemos como se realizan las sentencias if, else if y else

```
1  if(<condicion1>){
2      <Instrucciones en caso condicion1>
3  }else if(<condicion2>){
4      <Instrucciones en caso verdadero condicion2>
5  }else{
6      <Instrucciones en caso falso ambas condiciones>
7  }
```

Sentencia	Se activa si
Instrucciones en caso condicion1	Valor boolean... leano else if
Instrucciones en caso verdadero condicion2	Verdader... o o falso
Instrucciones en caso falso ambas condiciones	Falso ... dero / ...so

Si el bloque de sentencia del if solo tiene una sentencia, las llaves pueden ser omitidas.

UNIVERSIDAD DE SANTIAGO DE CHILE

Bifurcaciones


```c
#include <stdio.h>

int main(){
    int a, b;
    printf("Ingrese un numero: ");
    scanf("%d",&a);
    printf("Ingrese otro numero: ");
    scanf("%d",&b);
    if(a<b){
        printf("El numero %d es menor que el numero %d\n", a, b);
    }else if(a>b){
        printf("El numero %d es mayor que el numero %d\n", a, b);
    }else{
        printf("Ambos numeros ingresados son iguales (%d)/n",a);
    }
}
```

UNIVERSIDAD DE SANTIAGO DE CHILE

Bifurcaciones

Estructura de la sentencia switch

```
switch(<variable>){
    case <valor 1>:
        <Sentencias del valor 1>
        break;
    case <valor 2>:
        <Sentencias del valor 2>
        break;
    ...
    case <valor n>:
        <Sentencias del valor n>
        break;
    default:
        <Sentencias por defecto>
}
```

Se evalúa una variable con switch, luego se revisan los valores esperados para esa variable con case. Si el valor coincide, el case se activa y se ejecutarán todas las lineas restantes del switch a menos que se encuentre un break o similares.
default se activa si no se activó ningún case.

UNIVERSIDAD DE SANTIAGO DE CHILE

Bifurcaciones

- Introducción
- Solicitando información al usuario
- Operadores & y *
- Bifurcaciones
- Iteraciones
- Actividad 1
- Funciones
- Paso por valor
- Paso por referencia
- Actividad 2

```c
#include <stdio.h>

int main(){
    int a;
    printf("Ingrese un numero: ");
    scanf("%d",&a);
    switch(a){
        case 1:
            printf("El numero es un 1.\n");
            break;
        case 3:
            printf("El numero es un 3.\n");
            break;
        case 5:
            printf("El numero es un 5.\n");
        case 2:
            printf("El numero es un 2.\n");
            break;
        case 4:
            printf("El numero es un 4.\n");
            break;
    }
}
```

UNIVERSIDAD DE SANTIAGO DE CHILE

Bifurcaciones

La asignación condicional o comparación ternaria ? :, permite condicionar el valor que se va asignar a una variable (de ahí su nombre). Veamos un ejemplo donde se le asigna a la variable valor un 0 si el número ingresado por el usuario es par o un 1 si el número es impar.

```
1  //"variable" ya definida
2  variable = <condicion>?<Sentencia verdadero>:<Sentencia falso>;
3
```

UNIVERSIDAD DE SANTIAGO DE CHILE

Bifurcaciones

Y si alguien se preguntaba ¿cómo quedaría el ejercicio anterior (el del 1, 2 u otro) con asignaciones condicionales? Pues, es algo más enredado, debido a que esta sentencia está ideada para otros propósitos, pero ahí les va…

```c
#include <stdio.h>

int main(){
    int op;
    printf("Ingrese un numero: ");
    scanf("%d",&op);
    op==1?printf("Ingreso un 1\n"):(op==2?printf("Ingreso un 2\n"):printf("Ingreso otro numero\n"));
}
```

Si, es una asignación condicional con una asignación condicional dentro. Les dije que no sería bonito.

UNIVERSIDAD DE SANTIAGO DE CHILE

Bifurcaciones

Y si alguien se preguntaba ¿cómo quedaría el ejercicio anterior (el del 1, 2 u otro) con asignaciones condicionales? Pues, es algo más enredado, debido a que esta sentencia está ideada para otros propósitos, pero ahí les va…

```c
#include <stdio.h>

int main(){
    int op;
    printf("Ingrese un numero: ");
    scanf("%d",&op);
    op==1?printf("Ingreso un 1\n"):(op==2?printf("Ingreso un 2\n"):printf("Ingreso otro numero\n"));
}
```

Pregunta op==1

Si op==1 entonces

Si op!=1 entonces

Si, es una asignación condicional con una asignación condicional dentro. Les dije que no sería bonito.

UNIVERSIDAD DE SANTIAGO DE CHILE

Bifurcaciones

- Introducción
- Solicitando información al usuario
- Operadores & y *
- Bifurcaciones
- Iteraciones
- Actividad 1
- Funciones
- Paso por valor
- Paso por referencia
- Actividad 2

Y si alguien se preguntaba ¿cómo quedaría el ejercicio anterior (el del 1, 2 u otro) con asignaciones condicionales? Pues, es algo más enredado, debido a que esta sentencia está ideada para otros propósitos, pero ahí les va…

```c
#include <stdio.h>

int main(){
    int op;
    printf("Ingrese un numero: ");
    scanf("%d",&op);
    op==1?printf("Ingreso un 1\n"):(op==2?printf("Ingreso un 2\n"):printf("Ingreso otro numero\n"));
}
```

Pregunta op==2

Si op ==2 entonces

Si op!=2 entonces

Si, es una asignación condicional con una asignación condicional dentro. Les dije que no sería bonito.

UNIVERSIDAD DE SANTIAGO DE CHILE

Iteraciones

La estructura de la sentencia while.

```
1  while(<condiciones>){
2      <Instrucciones a repetir>
3  }
```

Intentemos mostrar los números del 1 al 100 mediante una while.

Cabe destacar que while no necesita de un incrementador, sino que necesita algo que varíe el valor booleano resultado de la comparación. Pero es la forma más común de ver un while.

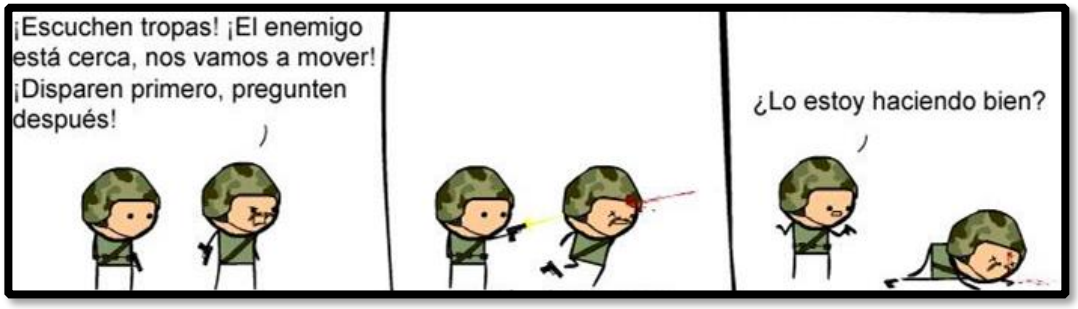
UNIVERSIDAD DE SANTIAGO DE CHILE

Iteraciones

- Con un do while se realizaría de la siguiente forma.

```
1  int i=1;
2  do{
3      printf("%d\n", i);
4      i++;
5  }while(i<100);
6
```

La única diferencia con while es que do while ejecuta las sentencias y luego pregunta, mientras que while pregunta antes de ejecutar.



UNIVERSIDAD DE SANTIAGO DE CHILE

Iteraciones

Un **for** no es más que el **while** típico (el que usa incrementador) escrito de forma ordenada.

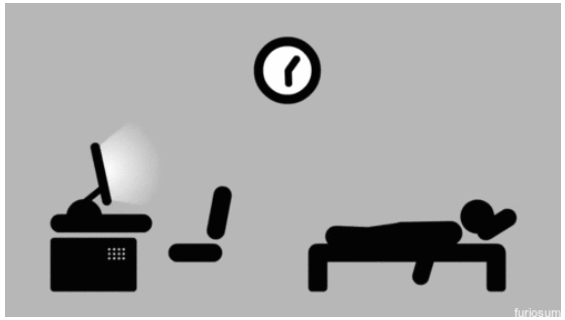
```
1  for (int i = 0; i < 100; ++i){
2      printf("%d\n", i);
3  }
4
```

El incremento del ciclo se realiza después de ejecutar las sentencias dentro del ciclo.

UNIVERSIDAD DE SANTIAGO DE CHILE

Actividad 1

Creemos un programa en donde el usuario decida si calcular el valor absoluto, valor al cuadrado o valor al cubo..



Dependiendo de la operación seleccionada, los mensajes de salida que se deben mostrar correspondientemente son los siguientes:

1. “El valor absoluto es: <Valor>”.
2. “El valor al cuadrado es: <Valor>”.
3. “El valor al cubo es: <Valor>”.

El programa debe terminar una vez que el usuario elige un opción y el mensaje es impreso. El código debe ser desarrollado con ayuda del profesor.

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones

- Las funciones nos permiten encapsular procesos y poder reutilizar el código.
- Ya que conocemos Python, veamos un ejemplo de una función que suma 2 números.

```
#Entrada: Dos numeros
#Salida: Un numero
def suma(a, b):
    resultado = a + b
    return resultado



valor1 = input("Ingrese el primer numero: ")

valor2 = input("Ingrese el segundo numero: ")

resultadoSuma = suma (valor1, valor2)

print "La suma de", valor1,"y",valor2,"es",resultadoSuma
```

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones


```
#Entrada: Dos numeros
#Salida: Un numero
def suma(a, b):
    resultado = a + b
    return resultado
```

```
valor1 = input("Ingrese el primer numero: ")

valor2 = input("Ingrese el segundo numero: ")

resultadoSuma = suma (valor1, valor2)

print "La suma de", valor1,"y",valor2,"es",resultadoSuma
```

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones


```
#Entrada: Dos numeros
#Salida: Un numero
def suma(a, b):
    resultado = a + b
    return resultado


valor1 = input("Ingrese el
valor2 = input("Ingrese el segundo numero: ")
resultadoSuma = suma (valor1, valor2)
print "La suma de", valor1,"y",valor2,"es",resultadoSuma
```

Parámetros formales: nombres genéricos para poder identificar los datos de entrada.

Parámetros actuales: valores o variables que contienen los datos reales que se deben procesar.

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones

- Analicemos la función.

Python

```python
#Entrada: Dos numeros
#Salida: Un numero
def suma(a, b):
    resultado = a + b
    return resultado
```

C

```c
//Entrada: Dos numeros
//Salida: Un numero
int suma(int a, int b){

    resultado = a + b;
    return resultado;

}
```

Pseudocódigo

```
Entrada: Dos numeros
Salida: Un numero
definir funcion suma(a,b)
        resultado =
a+b
        retornar
resultado
```

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones

- Encabezado.

```
int suma(int a, int b){
```

- Como se mencionó, C es un lenguaje fuertemente tipado. Por lo que a las variables se les debe indicar el tipo de dato.

- Esto no solo es relativo a las variables.

- Las funciones también deben especificar el tipo de retorno que tendrán, en este caso se indica que es un entero

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones

- Encabezado.

```
int suma(int a, int b){
```

Luego del tipo de retorno, se debe especificar el nombre de la función.

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones

- Encabezado.

```
int suma(int a, int b){
```

A continuación, se especifican los parámetros formales de la función, cada uno con su tipo de dato respectivo.

Estos se escriben entre paréntesis y separados por comas

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones

- Cuerpo de la función.

```
int suma(int a, int b){

    resultado = a + b;
    return resultado;

}
```

Las sentencias que componen la función se escriben entre llaves.

Como ya se debe haber notado, cada sentencia debe terminar con un punto y coma.

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones

- Retorno.

```
int suma(int a, int b){

    resultado = a + b;
    return resultado;

}
```

- El retorno o entrega de resultado se realiza a través de la sentencia **return.**

- Es importante destacar que la función debe retornar por obligación un valor del tipo de dato indicado en el encabezado.

- Esto significa que, si existen bifurcaciones, se debe asegurar que todos los caminos logren retornar el tipo antes indicado.

UNIVERSIDAD DE SANTIAGO DE CHILE

Funciones


```c

#include<stdio.h>

int suma(int a, int b){

    resultado = a + b;
    return resultado;

}

int main(){

    int valor1, valor2;
    int resultadoSuma;
    printf("Ingrese el primer numero: ");
    scanf("%d",&valor1);

    printf("Ingrese el segundo numero: ");
    scanf("%d",&valor1);

    resultadoSuma = suma(valor1,valor2);

    printf("La suma de %d y %d es %d",valor1,valor2,resultadoSuma);

    return 0;

}
```

Así quedaría el programa en C para sumar 2 números a través de una función.

UNIVERSIDAD DE SANTIAGO DE CHILE

Introducción a paso por valor y referencia

Referenciación (&): Anteponerlo a una variable permite obtener la dirección de memoria de ésta.

Desreferenciación (*): Anteponerlo a una dirección de memoria permite obtener el valor guardado en ésta.

No confundir con **<tipo de dato > ***
Ejemplo:

```
int * a;
char *c;
```

En estos casos **<tipo de dato> *** es el tipo de dato de la variable, declarando de esta manera a la variable **a** como una dirección de memoria para enteros y a la variable **c** como una dirección de memoria para guardar un carácter.

Cabe destacar que ni **a** ni **c** se pueden utilizar directamente, ya que no se les ha adjudicado memoria.
Esto se enseñará en clases posteriores.

UNIVERSIDAD DE SANTIAGO DE CHILE

Introducción a paso por valor y referencia

***a**: a debe ser una dirección de memoria y estamos accediendo al valor guardado en dicha dirección.

&b: b es una variable que puede contener cualquier tipo de valor y estamos obteniendo la dirección de memoria de donde esta guardado dicho valor

int *c: se esta declarando la variable c, la cual es una dirección de memoria que en algún momento almacenará un entero.

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor

- Introducción
- Solicitando información al usuario
- Operadores & y *
- Bifurcaciones
- Iteraciones
- Actividad 1
- Funciones
- Paso por valor
- Paso por referencia
- Actividad 2

```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del programa ********\n\n");

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

¿Qué muestra por pantalla el código en las 3 impresiones destacadas?

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del programa ********\n\n");

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Valor1: 0

Valor1: 0

Valor1: 1

UNIVERSIDAD DE SANTIAGO DE CHILE

- Introducción
- Solicitando información al usuario
- Operadores & y *
- Bifurcaciones
- Iteraciones
- Actividad 1
- Funciones
- Paso por valor
- Paso por referencia
- Actividad 2

Paso por valor

¿Por qué ocurre esto?

Veamos de forma simplificada que ocurre en la memoria al ejecutar las sentencias del código.

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	
0061FF21	
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	n = 0
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	n = 1
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	n = 1
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

Ya que el valor retornado por la función nunca se asignó a la variable *valor1*, esta conserva su valor de cero y se ha perdido el resultado de la función.

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	n = 0
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	n = 1
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	n = 1
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 1
0061FF21	
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor


```c
#include <stdio.h>

int funcion(int  n){
    n = n +1;
    return n;
}

int main(){
    printf("\n\n******** Inicio del prog

    int valor1 = 0;

    printf("< int valor1 = 0 >\n");
    printf("valor1: %d\n\n",valor1);

    printf("< funcion(valor1) >\n");
    funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("< valor1 = funcion(valor1) >\n");
    valor1 = funcion(valor1);
    printf("valor1: %d\n\n",valor1);

    printf("\n\n");
    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 1
0061FF21	
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

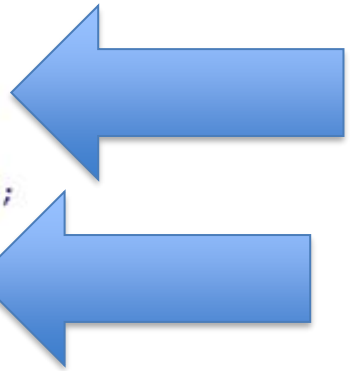
Paso por Referencia

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por referencia

¿Qué muestra por pantalla el código en las 2 impresiones destacadas?

```
1  #include <stdio.h>
2
3  void funcion(int * n){
4      *n = *n + 1;
5  }
6
7  int main(){
8      printf("\n\n******** Inicio del programa ********\n\n");
9      int valor1 = 0;
10
11
12     printf("< int valor1 = 0 >\n");
13     printf("valor2: %d\n\n",valor1);
14
15     printf("< funcion(&valor1) >\n");
16     funcion(&valor1);
17     printf("valor2: %d\n",valor1);
18
19
20     printf("\n\n******** Fin del programa ********\n\n");
21     return 0;
22 }
```



UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por referencia

¿Qué muestra por pantalla el código en las 2 impresiones destacadas?

```
1  #include <stdio.h>
2
3  void funcion(int * n){
4      *n = *n + 1;
5  }
6
7  int main(){
8      printf("\n\n******** Inicio del programa ********\n\n");
9      int valor1 = 0;
10
11
12     printf("< int valor1 = 0 >\n");
13     printf("valor2: %d\n\n",valor1);
14
15     printf("< funcion(&valor1) >\n");
16     funcion(&valor1);
17     printf("valor2: %d\n",valor1);
18
19
20     printf("\n\n******** Fin del programa ********\n\n");
21     return 0;
22 }
```

valor2: 0

valor2: 1

UNIVERSIDAD DE SANTIAGO DE CHILE

- Introducción
- Solicitando información al usuario
- Operadores & y *
- Bifurcaciones
- Iteraciones
- Actividad 1
- Funciones
- Paso por valor
- Paso por referencia
- Actividad 2

Paso por referencia

¿Por qué ocurre esto?

¿Por qué el llamado a función tenía un & y la función tenia * en su parámetro formal y también cuando lo utilizaba?

Veamos de forma simplificada que ocurre en la memoria al ejecutar las sentencias del código.

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por referencia


```c
#include <stdio.h>

void funcion(int * n){
    *n = *n + 1;
}

int main(){
    printf("\n\n******** Inicio del pro
    int valor1 = 0;


    printf("< int valor1 = 0 >\n");
    printf("valor2: %d\n\n",valor1);

    printf("< funcion(&valor1) >\n");
    funcion(&valor1);
    printf("valor2: %d\n",valor1);


    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por referencia


```c
#include <stdio.h>

void funcion(int * n){
    *n = *n + 1;
}

int main(){
    printf("\n\n******** Inicio del pr
    int valor1 = 0;


    printf("< int valor1 = 0 >\n");
    printf("valor2: %d\n\n",valor1);

    printf("< funcion(&valor1) >\n");
    funcion(&valor1);
    printf("valor2: %d\n",valor1);


    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por referencia


```
1  #include <stdio.h>
2
3  void funcion(int * n){
4      *n = *n + 1;
5  }
6
7  int main(){
8      printf("\n\n******** Inicio del
9      int valor1 = 0;
10
11
12     printf("< int valor1 = 0 >\n");
13     printf("valor2: %d\n\n",valor1)
14
15     printf("< funcion(&valor1) >\n");
16     funcion(&valor1);
17     printf("valor2: %d\n",valor1);
18
19
20     printf("\n\n******** Fin del programa ********\n\n");
21     return 0;
22 }
```

Dirección	Contenido
0061FF20	valor1 = 0
0061FF21	n = 0061FF20
0061FF22	
0061FF23	
0061FF24	
0061	
0061	

n* tiene la dirección de memoria de *valor1
o
n* “apunta” a *valor1

Paso por referencia


```
1  #include <stdio.h>
2
3  void funcion(int * n){
4      *n = *n + 1;
5  }
6
7  ir
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22 }
```

Dirección	Contenido
0061FF20	valor1 = 1
0061FF21	n = 0061FF20
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	
	4

***n indica que se quiere desreferenciar n, lo que significa que se quiere acceder al valor que está en la dirección de memoria n**
Lo que seria equivalente a querer acceder al valor de la variable *valor1*

Por lo tanto *n = *n + 1; indica que se quiere obtener el contenido de la dirección de memoria n, sumarle 1 y reemplazar el valor en la dirección de memoria n por este nuevo resultado. Modificando de esta manera el valor de la variable *valor1*.

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por referencia


```c
#include <stdio.h>

void funcion(int * n){
    *n = *n + 1;
}

int main(){
    int valor1 = 1;
    funcion(&valor1);
    printf("valor2: %d\n",valor1);

    printf("\n\n******** Fin del programa ********\n\n");
    return 0;
}
```

Dirección	Contenido
0061FF20	valor1 = 1
0061FF21	
0061FF22	
0061FF23	
0061FF24	
0061FF25	
0061FF26	

Es por esto que cuando se vuelve al main, independiente de que no se retornara ni se asignara el resultado de la función, el valor de la variable *valor1* cambió, ya que fue modificado directamente el la función

Paso por valor y referencia

Veamos un ejemplo de esto menos computacional.

Supongamos que tenemos dos alumnos

Allende

Resulta que Allende es un alumno responsable que toma muy buenos apuntes de sus clases

Lamentablemente Felipe faltó a una de las clases importantes del semestre

Como muchos hacen, Felipe le pidió el cuaderno prestado a Allende.

Felipe

Pero resulta que el profesor en la siguiente clase hace una actividad con nota a Felipe, en la que debe trabajar sobre los apuntes que haya tomado.

Paso por valor y referencia

Como sabemos, Felipe no tiene apuntes propios, sino que posee el cuaderno de Allende

Allende

En este punto, Felipe tiene 2 alternativas.

1. Correr a la fotocopiadora para copiar el cuaderno de Allende, para así trabajar sobre las fotocopias y no afectar los apuntes de Allende.
2. Ocupar el cuaderno de Allende para realizar la actividad directamente en él.

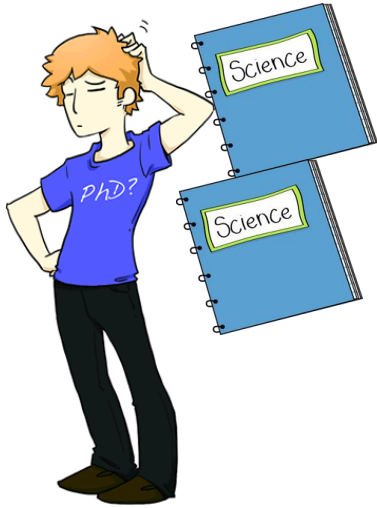
Felipe

Si vemos, en ambos casos Felipe tendrá la información de los apuntes.

Paso por valor y referencia

Allende

1. Correr a la fotocopiadora para copiar el cuaderno de Allende, para así trabajar sobre las fotocopias y no afectar los apuntes de Allende.
2. Ocupar el cuaderno de Allende para realizar la actividad directamente en él.



Felipe

En el primer caso, Felipe habrá ocupado un paso por valor, ya que no afecta el valor original (cuaderno de Allende).

Incluso, si Allende considera que los apuntes tomados por Felipe en la actividad son mejores que los originales, puede llegar a reemplazar si quiere su cuaderno por las fotocopias de Felipe con nuevos apuntes.

UNIVERSIDAD DE SANTIAGO DE CHILE

Paso por valor y referencia

Allende

1. Correr a la fotocopiadora para copiar el cuaderno de Allende, para así trabajar sobre las fotocopias y no afectar los apuntes de Allende.
2. Ocupar el cuaderno de Allende para realizar la actividad directamente en él.

Felipe

En el segundo caso, habrá ocupado un paso por referencia, ya que estará trabajando sobre la fuente original de la información.

Al momento de hacer el trabajo se habrán modificado la información original y a menos que se haya respaldado, no se podrá recuperar.

UNIVERSIDAD DE SANTIAGO DE CHILE

- Introducción
- Solicitando información al usuario
- Operadores & y *
- Bifurcaciones
- Iteraciones
- Actividad 1
- Funciones
- Paso por valor
- Paso por referencia
- Actividad 2

Consultas

UNIVERSIDAD DE SANTIAGO DE CHILE

Actividad 2

Usando como base el código de la actividad 1, encapsule el cálculo de cada una de las opciones.

La función del valor al cuadrado debe trabajar con paso por referencia.

La función del valor al cubo debe trabajar con paso por valor.

El programa debe seguir mostrando y realizando las opciones hasta que el usuario decida salir del programa.