

Programación I

Teoría I

<http://proguno.unsl.edu.ar>

Horarios

- Teorías:
 - Jueves 8:00 – 10:00; aula 58 (Bloque II)
- Prácticas:
- **Comisión 1: Licenciatura y Profesorado Plan 06/09**
 - Martes de 10 a 13 en sala 7, bloque 2
 - jueves de 10 a 13 en la sala 8, bloque 2
- **Comisión 2: Ingenierías y Profesorado Plan 02/16**
 - Martes de 10 a 12 en sala 8, bloque 2
 - viernes de 10 a 12 en la sala 8, bloque 2

Condiciones de Regularización y/o Aprobación de la Materia

- **Regularización:**

1. Mínimo de 70% de asistencia a clases.
2. Aprobar el 50% de los prácticas de laboratorio.
3. Aprobar la evaluación o sus respectivas recuperaciones con una nota mínima de seis.
4. Aprobar práctico máquina.

- **Promoción sin examen final:**

1. Mínimo de 80% de asistencia a clases.
2. Aprobar el 50% de los prácticas de laboratorio.
3. Aprobar la evaluación o sus respectivas recuperaciones con una nota mínima de siete.
4. Aprobar práctico máquina.
5. Aprobar el Coloquio

Fechas importantes 2017

- **Primer Laboratorio:** Martes 19/09, 10:00, sala 7 y 8
- **Segundo Laboratorio** Jueves 12/10, 8:00, aula 58
- **Recuperación Laboratorios:** Jueves 19/10, 8:00, aula 58
- **Parcial:** Martes 07/11, 10:00, sala 7 y 8
- **Primer recuperación Parcial:** Martes 14/11, 10:00, sala 8 y 7
- **Entrega Pco. Máquina:** Martes 21/11, 10:00, sala 8 y 7
- **2da Recuperación :** Viernes 24/11, hora y lugar a fijar.

Lenguajes de Programación

- Lenguajes de Máquina
- Lenguajes de Ensamblado (Assembly)
- Lenguajes de Alto Nivel

Lenguajes de Programación

- Lenguajes de Máquina
 - Lenguaje “natural” de la máquina
 - Definido por el diseño del hardware
 - Dependientes de la computadora: set de instrucciones propio.
 - Cadenas de 0s y 1s.



Lenguajes de Programación

- Lenguajes de Ensamblado (Assembly)
 - Facilitan la programación y depuración
 - Cuentan con instrucciones para abreviar cadenas de 0s y 1s.
 - Cuentan con un programa ensamblador.

```
*****  
* FUNCTION: INCH - Input character  
* INPUT: none  
* OUTPUT: char in acc A  
* DESTROYS: acc A  
* CALLS: none  
* DESCRIPTION: Gets 1 character from terminal
```

```
C010 B6 80 04 INCH LDA A ACIA GET STATUS  
C013 47 ASR A SHIFT RDRF FLAG INTO CARRY  
C014 24 FA BCC INCH RECIEVE NOT READY  
C016 B6 80 05 LDA A ACIA+1 GET CHAR  
C019 84 7F AND A #$7F MASK PARITY  
C01B 7E C0 79 JMP OUTCH ECHO & RTS
```

Lenguajes de Programación

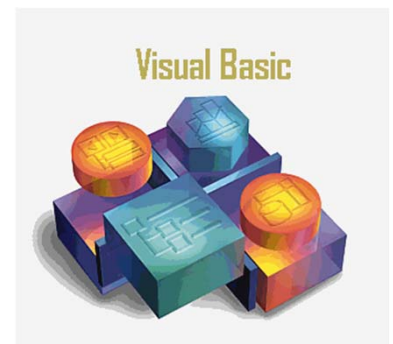
- Lenguajes de Alto Nivel
 - Facilitan aún más la programación y depuración
 - Sentencias para agrupar conjunto de instrucciones de máquina.
 - Programas traductores: compiladores e intérpretes – Sintaxis
 - Portabilidad - Estándares



C++

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hola Facebook!\n";
    return 0;
}
```

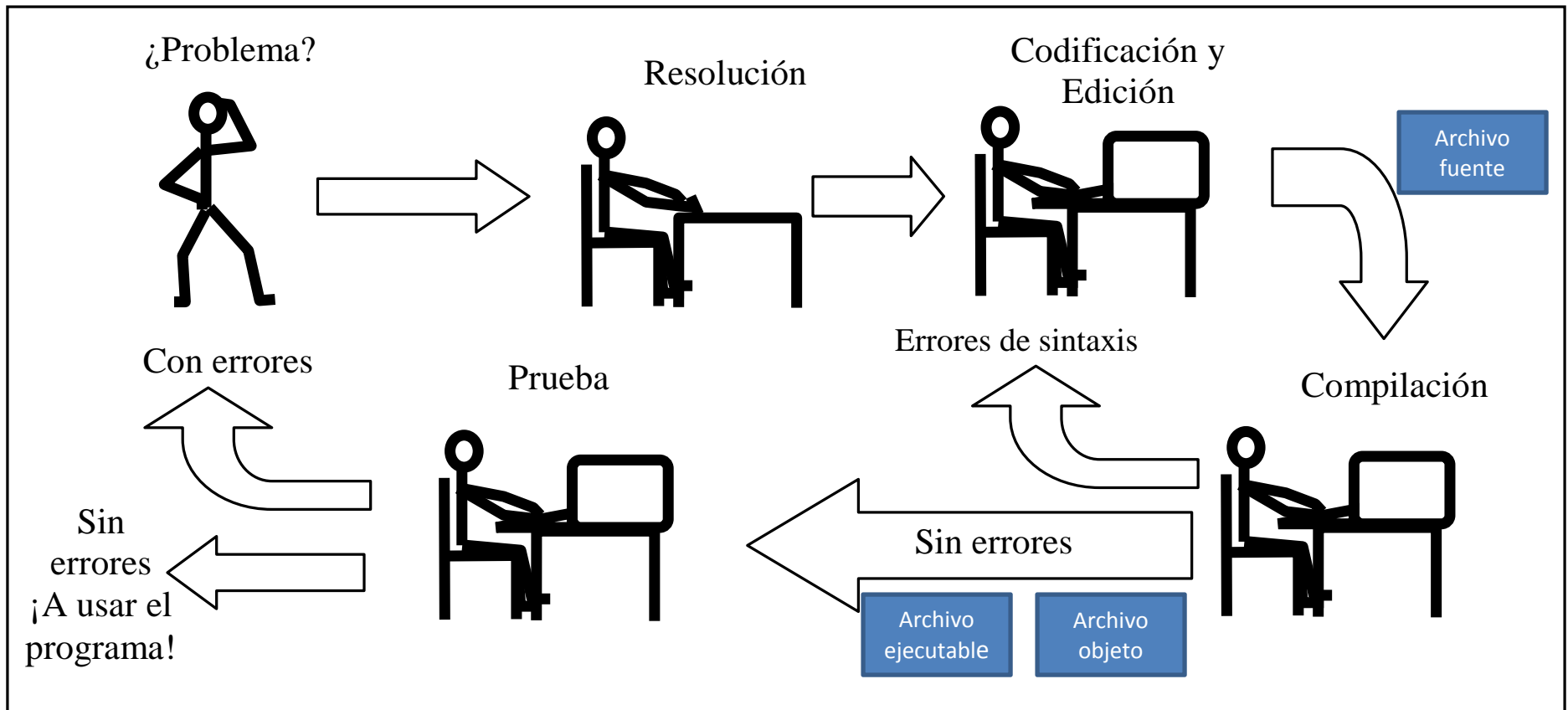


Paradigmas de Programación

- Conjunto de reglas, métodos, principios de programación, que comparten una filosofía común de programación.

- Imperativo
- Funcional } Declarativo
- Lógico
- Objetos

Etapas en la construcción de un programa



El Lenguaje de Programación C

- Orígenes
 - Dennis Ritchie, Laboratorios Bell, 1969-1972
 - B, BCPL
 - Usado para escribir S.O. Unix
- C Clásico (Kernighan & Ritchie)
- ANSI C



Estructura general de un programa C

- Todo programa C
 - Está formado por un conjunto de **funciones**
 - En particular, una función que no puede faltar es la función **main** (programa principal).
 - Llama a otras funciones:
 - Definidas por nosotros en el programa
 - Predefinidas, en bibliotecas.

Estructura general de un programa C

```
/* Mi primer programa C */  
  
#include <stdio.h>  
  
main()  
{  
    printf( "Hola mundo! \n" );  
}
```

Principales Secuencias de Escape en C

Secuencia de escape	Descripción
<code>\a</code>	Carácter de alarma (campana del sistema).
<code>\n</code>	New line (Nueva línea). Posiciona el cursor de la pantalla al comienzo de la próxima línea.
<code>\r</code>	Carriage return (Retrosceso de carro). Posiciona el cursor al comienzo de la línea corriente sin avanzar a la próxima.
<code>\t</code>	Tabulador horizontal. Mueve el cursor hasta la próxima marca de tabulación.
<code>\\</code>	Backslash. Usado para imprimir el carácter backslash.
<code>\"</code>	Comilla. Usado para imprimir la comilla.
<code>\'</code>	Apóstrofo. Usado para imprimir el apóstrofo.
<code>\?</code>	Signo de interrogación. Usado para imprimir el signo de interrogación.

Datos y Tipos de Datos

- Dato: representación en la computadora de un aspecto de la realidad.
 - Constantes simbólicas o literales
 - Variables
- Tipo de Dato: conjunto de valores que comparten las mismas características y operadores.

Constantes y Variables

- Constantes o literales
 - Ejemplos en C: 12, 'a', "hola", 23.5
- Variables
 - Nombre (Identificador)
 - En C, letras, números y el carácter de subrayado (_), no pueden comenzar con un número.
 - Cualquier longitud, reconoce hasta 31 caracteres.
 - Case sensitive.
 - No acepta palabras claves.
 - Tipo de dato
 - Valor

52.6

peso

Otro programa sencillo en C

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

```
main( )
```

```
{
```

```
    int entero1, entero2, suma;
```

```
    printf("Ingrese el primer entero\n");
```

```
    scanf("%d", &entero1);
```

```
    printf("Ingrese el segundo entero\n");
```

```
    scanf("%d", &entero2);
```

```
    suma = entero1 + entero2;
```

```
    printf("La suma es %d\n", suma);
```

```
}
```

Tipos de datos en C

Tipos de datos en C	Básicos :	Aritméticos	Enteros	
			Caracteres	
			Flotantes o reales	
		void		
	Estructurados o Compuestos	Arreglos		
		Estructuras o registros		
		Uniones		
Punteros				

Tipos de datos aritméticos en C

Enteros

- Tres tamaños
 - `short int`
 - `int` (tamaño del registro del procesador)
 - `long int`
- Valores con signo, por defecto
 - Modificador `unsigned`
- Se puede omitir `int` cuando va alguno de los modificadores (`short`, `long`, `unsigned`)

Tipos de datos aritméticos en C

Caracteres (`char`)

- 1 carácter (1 byte)

```
char x;
```

```
char w, y;
```

```
char z = 'A';
```

- un `char` \equiv un entero de 8 bits.

```
char z = 65;
```

```
z = z + 1;
```

```
printf("El código ASCII de %c es  
%d\n", z, z);
```

Tipos de datos aritméticos en C

Flotantes o Reales

- Tres tipos:

- `float` (precisión simple)
- `double` (precisión doble)
- `long double` (precisión extra)

Ejemplo

```
#include <stdio.h>
main()
{
    float radio;
    printf("Ingrese el radio del circulo en"
           " cm: ");
    scanf("%f", &radio); /* lee el radio*/
    printf("El circulo de radio %f tiene una"
           " superficie de %f cm2\n", radio,
           3.14159265358979323846 * radio *
           radio);
```

Uso de constante simbólica

Preprocesador de C

```
#include <stdio.h>
#define PI 3.14159265358979323846
main()
{
    float radio;
    printf("Ingrese el radio del circulo en cm: ");
    scanf("%f", &radio);
    printf("El circulo de radio %f tiene una
    superficie de %f cm2\n",radio, PI*radio*radio);
}
```

Operadores y expresiones aritméticas en C

Operador aritmético	Operador en C	Expresión aritmética	Expresión en C
Suma	+	$x + 20$	<code>x + 20</code>
Resta	-	$a - b$	<code>a - b</code>
Multiplicación	*	xy	<code>x * y</code>
División	/	$x:y$	<code>x / y</code>
Módulo	%	$u \text{ mod } k$	<code>u % k</code>

Operadores y Expresiones Relacionales en C

Operadores relacionales algebraicos	Operadores relacionales en C	Ejemplo de uso
=	==	x == y
≠	!=	x != y
≤	<=	x <= y
≥	>=	x >= y
>	>	x > y
<	<	x < y

Operadores y expresiones lógicas en C

- No hay tipo booleano o lógico
 - 0 representa falso
 - 1 representa verdadero
 - `1 == 1` \rightarrow 1 (verdadero)
 - `10 <= 5` \rightarrow 0 (falso)
 - `i = 10; if (i) printf("verdadero\n");`
 - `40 + (3 != 4)`

Operadores y Expresiones Lógicas en C

Operador lógico	Operador lógico en C	Ejemplo de uso	Significado
\wedge and	&&	<code>i != 0 && j > 1</code>	i distinto de 0 y j mayor que 1
\vee or		<code>c == 'a' n == 0</code>	c igual al carácter 'a' o n igual a 0
\sim \neg not	!	<code>!encontrado</code>	la variable encontrado no es verdadera (encontrado igual 0)

Orden de evaluación:

0 && _ \rightarrow 0

1 || _ \rightarrow 1

Operador de Asignación =

```
#define K -4
```

```
int i = 2;
```

```
i = K * 2;
```

```
i = i + 5;
```

- En C, la asignación es una operación que se vuelve una sentencia al seguirla por ;

```
r = (i = K + 1) + 4;
```

Operadores de asignación aritméticos

Operador de asignación aritmético	Ejemplo	Versión descomprimida	Resultado asignado a variable a asumiendo la declaración <code>int a = 3;</code>
<code>+=</code>	<code>a+=4</code>	<code>a = a + 4</code>	7
<code>--</code>	<code>a-=4</code>	<code>a = a - 4</code>	-1
<code>*=</code>	<code>a*=4</code>	<code>a = a * 4</code>	12
<code>/=</code>	<code>a/=4</code>	<code>a = a / 4</code>	0

Operadores de incremento y decremento en C

Operador	Explicación	Ejemplo de uso	Efecto
<code>a++</code>	Devuelve el valor de <code>a</code> y luego lo incrementa en 1.	<code>i = a++;</code>	Asigna el valor de <code>a</code> a la variable <code>i</code> y luego incrementa <code>a</code> en 1. Es equivalente a <code>i = a; a = a + 1;</code>
<code>++a</code>	Incrementa el valor de <code>a</code> en 1 y devuelve ese valor.	<code>i = ++a;</code>	Incrementa en 1 la variable <code>a</code> y ese valor lo asigna a la variable <code>i</code> . Es equivalente a <code>a = a + 1; i = a;</code>
<code>a--</code>	Devuelve el valor de <code>a</code> y luego lo decrementa en 1.	<code>printf("%d", a--);</code>	Imprime el valor de <code>a</code> y luego decrementa en 1 su valor. Es equivalente a <code>printf("%d", a); a = a - 1;</code>
<code>--a</code>	Decrementa el valor de <code>a</code> en 1 y devuelve ese valor.	<code>printf("%d", --a);</code>	Decrementa en 1 la variable <code>a</code> y luego muestra su valor por pantalla. Equivalente a <code>a = a - 1; printf("%d", a);</code>

Conversiones de Tipos

- Cambiar un tipo de dato por otro
- Implícitas o automáticas
 - En tiempo de compilación o de ejecución
 - En C: *promoción de tipos*.
- Explícitas
 - Por medio de una construcción sintáctica del lenguaje.
 - En C: *casting* → *operador cast*

Conversiones Implícitas de Tipos en C

Promoción de Tipos

```
int i = 5;  
float f = 3.0;  
float y;  
  
f = i;  
y = f + i;
```


Conversiones Explícitas de Tipos en C

Operador cast

- Forma general: *(nombre-de-tipo) expresión*
- Ejemplos:

```
float resultado;
```

```
int i = 9;
```

```
int j = 5;
```

```
resultado = i / j;
```

```
resultado = (float)i / j;
```

```
resultado = i / (float)j;
```

```
resultado = (float)(i / j);
```

```
resultado = (float)i / (float) j;
```

Programación Estructurada

- Metodología de programación
- C. Böhm y G. Jacopini, 1966
- Edsger W. Dijkstra, 1968
- Estructuras de Control
 - Secuencia
 - Selección
 - Iteración

Secuencia

- Sigue el orden de lectura tradicional de los idiomas occidentales.
- La ejecución de las sentencias se hace en forma *secuencial*:
 - Una después de la otra, y no se ejecuta la segunda sentencia hasta que la primera haya terminado de ejecutarse, y así sucesivamente.

Secuencia en C

- `;` es un finalizador de sentencia:
 - `y = 4` y `j++` y `scanf(...)` son expresiones en C
 - `y = 4;` `j++;` `scanf(...);` son sentencias en C
- `{ }` para crear sentencias compuestas o bloques:
 - Agrupan declaraciones y sentencias.
 - Ejemplo: llaves en declaración de función `main`.

Selección

- Selección de rama vacía o simple

```
if (cond)  
    sentencia;
```

- Ejemplos:

```
if (a==1 && b)  
    a++;
```

```
if (a==1 && b) {  
    a++;  
    b+=5;  
}
```

Selección

- Selección de dos ramas

```
if (cond)
    sentencia;
else
    sentencia;
```

- Si hay if anidados ambiguos, asocia else al if mas interno:

```
if (a==1 && b)
if (b < 0)
    b = 0;
else
    a++;
```

```
if (a==1 && b){
    if (b < 0)
        b = 0;
}
else
    a++;
```

Selección

- Selección múltiple en C

```
switch (expresión) {  
    case constanteEntera1: sentencias;  
    case constanteEntera1: sentencias;  
    ...  
    case constanteEntera1: sentencias;  
    default: sentencias;  
}
```

- Ejemplo 1:

```
switch (i) {  
    case 1: printf("uno\n");  
    case 2: printf("dos\n");  
    case 3: printf("tres\n");  
}
```

Selección

- Ejemplo 2:

```
switch (i) {  
    case 1: printf("uno\n"); break;  
    case 2: printf("dos\n"); break;  
    case 3: printf("tres\n"); break;  
    default: printf("distinto de 1, 2 y 3\n");  
}
```

- Ejemplo 3:

```
switch (ch) {  
    case ` , ` :  
    case ` ; ` :  
    case ` . ` : printf("signo de puntuacion\n"); break;  
    default: printf("no es un signo de puntuacion\n");  
}
```


Iteración

- while
- do/while
- for

Iteración while

```
while (expresión)  
    sentencia;
```

Ejemplo:

```
int i = 0;  
while (i < 10) {  
    printf("i = %d", i);  
    i++;  
}
```

Iteración for

```
for ( expresión1; expresión2; expresión3 )  
    sentencia;
```

Ejemplos:

```
int suma = 0;  
for (i = 0; i <= n; i++)  
    suma = suma + i;
```

```
int suma;  
for (i=0, suma=0; i <= n; i++, suma+= i);
```

Iteración do / while

do

sentencia;

while (*expresión*)

Ejemplo:

```
do {  
    printf( "Ingrese un numero natural"  
           " mayor o igual a cero: " );  
    scanf( "%d", &n );  
} while ( n < 0 );
```