Programación I

Teoría I

http://proguno.unsl.edu.ar

Horarios

- Teorías:
  - Lunes 11:00 – 13:00; sala 3 (Rectorado)
- Prácticas:
- **Comisión 1: Licenciatura\Ingenierías**

  Lunes de 9 a 11 en sala 3

  Miércoles de 10 a 12 en sala 3

  jueves 11 a 13 en la sala 3

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 70% 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 2018

- **Primer Laboratorio**: Lunes 26/03, (funciones)
- **Segundo Laboratorio** jueves 12/04, (Arreglo-struct)
- **Tercer Laboratorio** 26/04,(recursión)
- **Cuarto Laboratorio** jueves 09/05, (memoria dinámica)
- **Recuperación Laboratorios**: lunes 21/05

- **Parcial**: Miércoles 11/06, 10:00 (recursión, memoria dinámica, TDA, Archivos)

- **Recuperación Parcial**: lunes 18/06,
- **2da Recuperación** : jueves 21/06, hora y lugar a fijar.

- **Entrega Pco. Máquina:** Miércoles 27/06, 10:00,

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

Java

PASCAL

C++

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

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

HASKELL

Made with Lisp

ProLog

php

Visual Basic

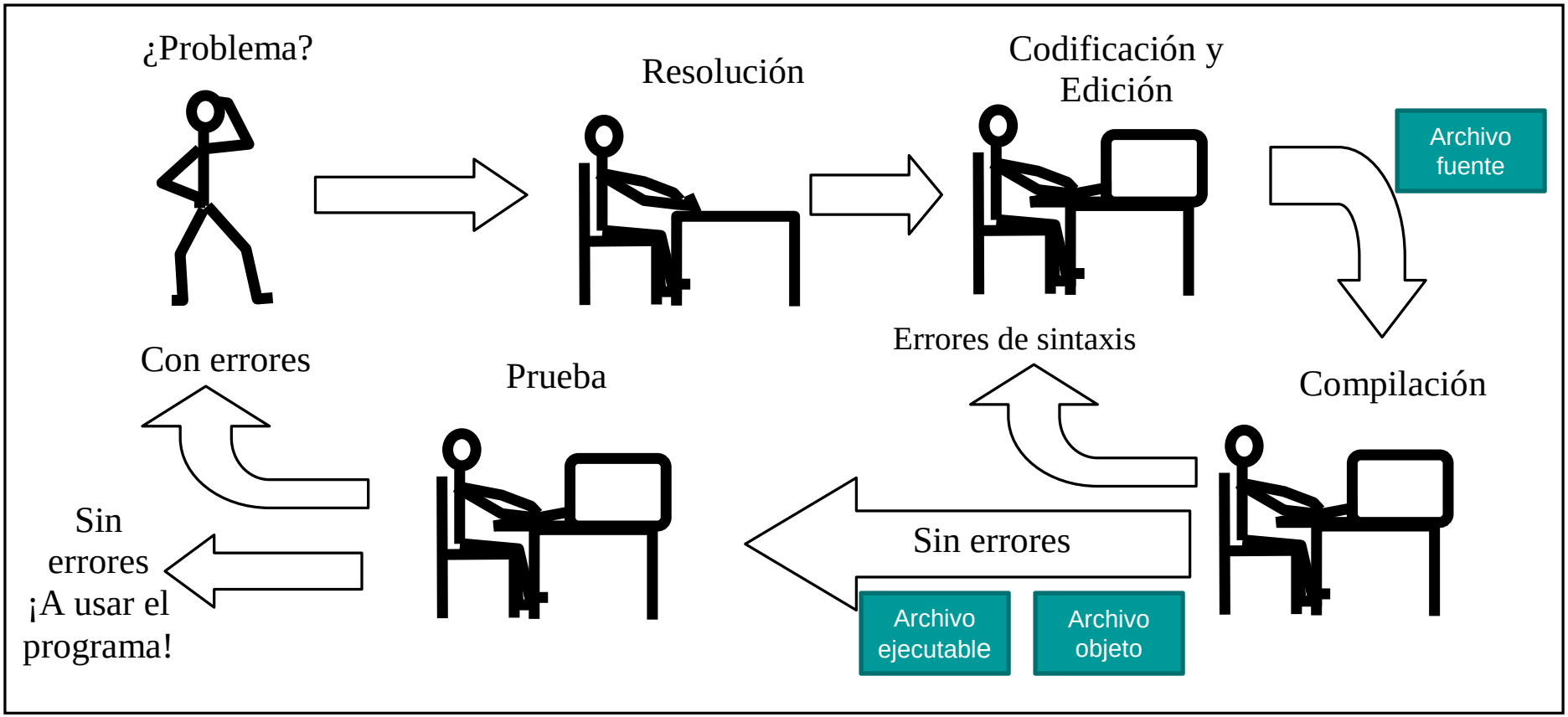
Paradigmas de Programación

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

  - Imperativo
  - Funcional
  - Lógico
  - Objetos

  } Declarativo

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

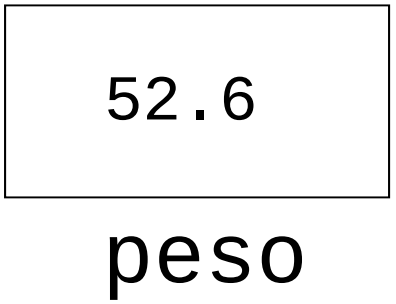
Datos y Tipos de Datos

- 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



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

<table>
<tr><td rowspan="6">Tipos de datos en C</td><td rowspan="3">Básicos
.
.</td><td rowspan="2">Aritméticos</td><td>Enteros</td></tr>
<tr><td>Caracteres</td></tr>
<tr><td colspan="2">Flotantes o reales</td></tr>
<tr><td colspan="2"><code>void</code></td></tr>
<tr><td rowspan="3">Estructurados o Compuestos</td><td colspan="2">Arreglos</td></tr>
<tr><td colspan="2">Estructuras o registros</td></tr>
<tr><td colspan="2">Uniones</td></tr>
<tr><td colspan="3">Punteros</td></tr>
</table>

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` ≡ 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	+	$x + 20$	`x + 20`
Resta	-	$a - b$	`a - b`
Multiplicación	*	xy	`x * y`
División	/	$x:y$	`x / y`
Módulo	%	$u \bmod k$	`u % k`

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
∧ and	`&&`	`i != 0 && j > 1`	`i` distinto de `0` y `j` mayor que `1`
∨ or	`||`	`c == 'a' || n == 0`	`c` igual al carácter 'a' o `n` igual a `0`
~ ¬ not	`!`	`!encontrado`	la variable `encontrado` no es verdadera (`encontrado` igual `0`)

Orden de evaluación:

`0 && _` → `0`

`1 || _` → `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 `int a = 3;`
`+=`	`a+=4`	`a = a + 4`	`7`
`-=`	`a-=4`	`a = a - 4`	`-1`
`*=`	`a*=4`	`a = a * 4`	`12`
`/=`	`a/=4`	`a = a / 4`	`0`

Operadores de incremento y decremento en C

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

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);
```