Principios y Herramientas de Programación

Dra. Jessica Andrea Carballido

jac@cs.uns.edu.ar

Dpto. de Ciencias e Ingeniería de la Computación

UNIVERSIDAD NACIONAL DEL SUR

Características

- Es un lenguaje de **alto nivel** pero con facilidades de bajo nivel.
- En su desarrollo se buscó lograr:
  - Un compilador pequeño y eficiente.
  - Generar programas eficientes y robustos.
- Es un lenguaje **compilado**.
- Es un lenguaje con **tipado estático**.

Estructura de un programa


```
declaración de importaciones
definición de constantes
definición de tipos de datos
main()
{
   declaración de variables
   instrucciones ejecutables
}
```

"main" es el algoritmo PRINCIPAL, sería el globo más grande de arriba cuando dividimos en subproblemas.

- El nombre del programa es el nombre del archivo de texto que lo contiene.
- Este documento es comúnmente denominado como el "código fuente" del programa (*.c).

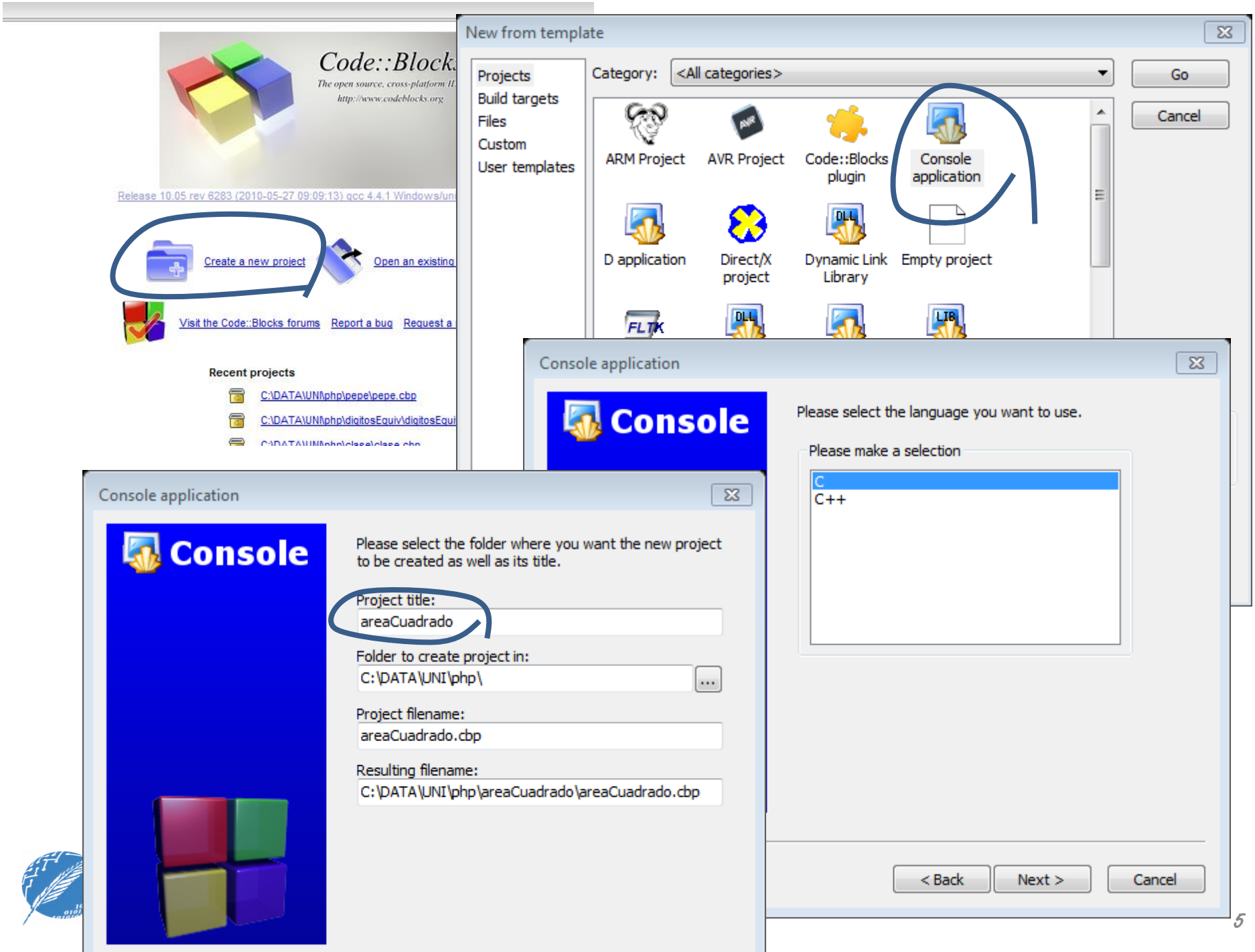
Programa *Hola, mundo!*

Lenguaje C

```
#include <stdio.h>
 void main( void )
 {
     printf( “Hola, mundo! \n” );
 }
```

Una librería (*.h) provee un grupo de primitivas relacionadas.

Compilador Enlazador


Code::Block
The open source, cross-platform IL
http://www.codeblocks.org
Release 10.05 rev 6283 (2010-05-27 09:09:13) gcc 4.4.1 Windows/un
Create a new project
Open an existing
Visit the Code::Blocks forums
Report a bug
Request a
Recent projects
C:\DATA\UNI\php\pepe\pepe.cbp
C:\DATA\UNI\php\digitosEquiv\digitosEqui
New from template
Projects
Build targets
Files
Custom
User templates
Category:
<All categories>
Go
Cancel
ARM Project
AVR Project
Code::Blocks plugin
Console application
D application
Direct/X project
Dynamic Link Library
Empty project
Console application
Console
Please select the language you want to use.
Please make a selection
C
C++
< Back
Next >
Cancel
Console application
Console
Please select the folder where you want the new project to be created as well as its title.
Project title:
areaCuadrado
Folder to create project in:
C:\DATA\UNI\php\
Project filename:
areaCuadrado.cbp
Resulting filename:
C:\DATA\UNI\php\areaCuadrado\areaCuadrado.cbp

Declaración e inicialización de variables

Las variables representan locaciones de memoria (DATOS) y tienen asociado un nombre (fijo), un tipo (fijo) y un valor (cambia durante la ejecución).

tipo nombre;

```
int alfa;
int epsilon = 5;
int omega = 2, sigma, gamma = 12;
```

Algunos Tipos de Datos Simples

- void
- char (1 byte)
- int (2 bytes)
- float (4 bytes)
- double (8 bytes)
- bool (incluyendo la librería stdbool.h)

La cantidad de bytes requeridos para el almacenamiento de una variable de un tipo dado se determina con la función *sizeof.*

Literales

- Ejemplos de literales de cada tipo:

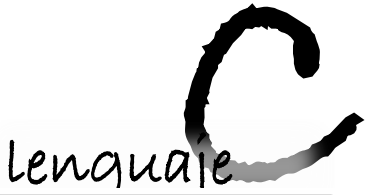
2173	int
28.69	float
81e-3	float // $81*10^{-3} = 0,081$
76.2	double

Operadores Aritméticos

Operador	Nombre	definicion
*	Multiplicación	Multiplica x por y
/	División	Divide x por y
%	Modulo	Resto de x dividido y
+	Suma	Suma x mas y
-	Substracción	Resta y de x
++	Incremento	++x x++
--	Decremento	--x x--
-	Negación	Multiplica x por -1

Al aplicar "/" a dos operandos de tipo entero, se realiza división entera.

Operadores Relacionales



Operador	Ejemplo	Definición
>	x > y	1 si x es mayor que y, en caso contrario es 0
>=	x >= y	1 si x es mayor o igual a y, en caso contrario es 0
<	x < y	1 si x es menor que y, en caso contrario es 0
<=	x <= y	1 si x es menor o igual a y, en caso contrario es 0
= =	x == y	1 si x es igual que y, en caso contrario es 0
!=	x != y	1 si x no es igual que y, en caso contrario es 0

Operadores Lógicos

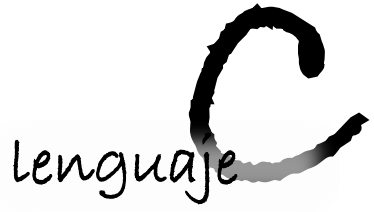
And

Or

!	!x	1 si x es 0, en caso contrario es 0
&&	x && y	0 si x o y es 0, en caso contrario 1
||	x || y	0 si x e y son 0, en caso contrario 1

Idem usando *true* y *false* de la librería *stdbool*

Definición de Constantes



- Toda sentencia que empieza con el símbolo #
  es un directiva para el pre-procesador.
- Este preprocesador es un programita que por ejemplo hace el reemplazo literal de constantes antes de compilar.

```
# define Maximo 4
# define Tamano_Maximo 25
```

C no tiene predefinido el tipo de dato lógico,
por eso es común definir como constantes:

#define false 0
#define true 1

O se importa la librería *stdbool*

Estructuras de Control

Secuencia:

```
sentencia1;
sentencia2;
sentencia3;
....
sentencia-n;
```

Estructuras de Control

Condicional:

Si la expresión de la condición es **distinta de cero (true)**,
se ejecuta la sentencia-then.
En caso contrario, se ejecuta la sentencia-else.

Forma 1:

```
If (expresion)
   sentencia-then;
else
   sentencia-else;
```

Forma 2:

```
If (expresion)
{
  Sentencia1; sentencia2; …
}
else
  sentencia;
```

if y else son palabras reservadas

Estructuras de Control

Iteración:

```
int factorial( int n )
{
    int prod = 1;
    int i;
    for (i=1; i<=n; i++)
        prod *= i;
    return prod;
}
```

```
int factorial( int n )
{
    int prod = 1;
    int i=1;
    while (i<=n)
    {
        prod *= i;
        i++;
    }
    return prod;
}
```

```
int factorial( int n )
{
    int prod = 1;
    int i=1;
    do {
    prod *= i;
    i = i + 1;
    } while ( i<=n );
    return prod;
}
```

Acumuladores

Producto: prod=prod*i equivalente a prod*=i

Suma: suma=suma+i equivalente a suma+=i

Un programa simple

Lenguaje C

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

int main()
{

    int lado, area;

    printf("Ingrese el valor del lado: \n");
    scanf("%i", &lado);

    area=lado*lado;

    printf("El area de un cuadrado de lado %i es %i", lado, area);

    return 0;
}
```

Lee los DATOS DE ENTRADA

Calcula (acciones)

Muestra los DATOS DE SALIDA

Las bibliotecas o librerías son contenedoras de primitivas.

Como mostrar datos de salida

Es necesario importar la libreria **stdio.h**
para usar la primitiva **printf**

Stdio (<stdio.**h**>)

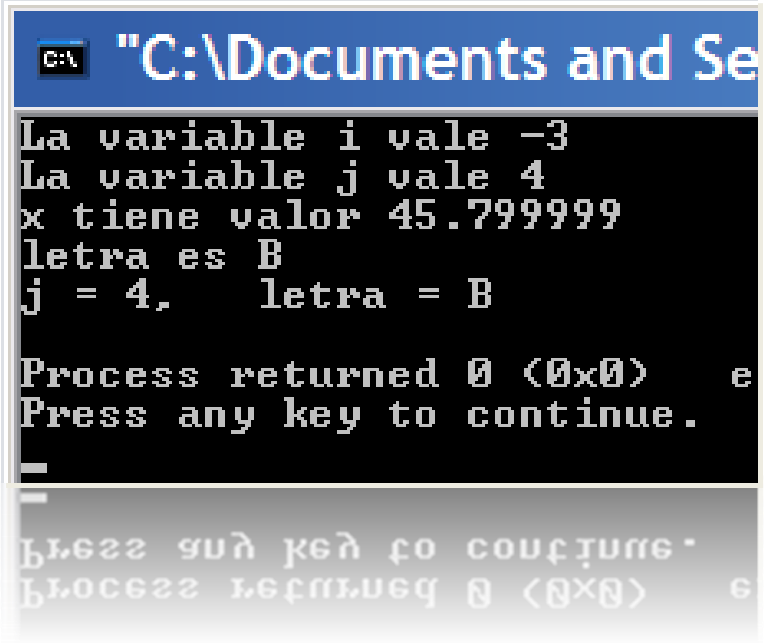
- **stdio.h**, que significa "**st**andar**d** **i**nput-**o**utput **h**eader" (cabecera estándar E/S), es el archivo de cabecera que contiene las definiciones de las macros, las constantes, las declaraciones de funciones de la biblioteca estándar del lenguaje de programación C para hacer operaciones, estándar, de entrada y salida, así como la definición de tipos necesarias para dichas operaciones.

Las funciones declaradas en stdio.h son sumamente populares.

- printf: usado para imprimir salidas de datos.
- scanf: utilizado para introducir entradas.
- puts: imprime una cadena de caracteres.
- getc: devuelve un carácter desde un fichero.
- ferror: comprueba el indicador de errores.

Ejemplos de salida

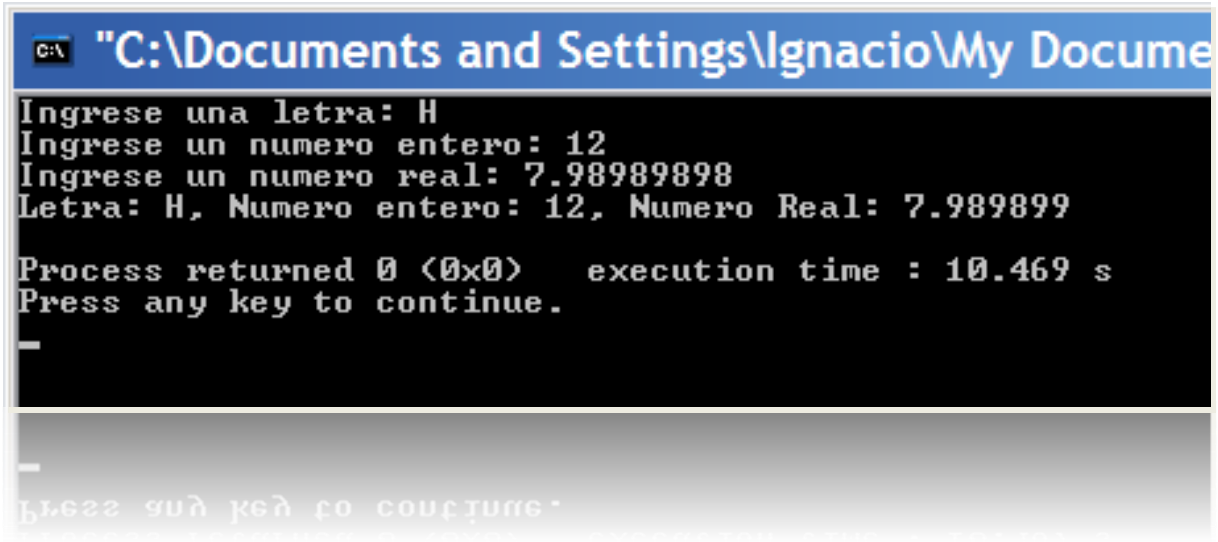
```
#include <stdio.h>
int main()
{
   char letra = 'B';
   int i = -3;
   int j = 4;
   float x = 45.799999;
   printf( "La variable i vale %i \n", i );
   printf( "La variable j vale %i \n", j );
   printf( "x tiene valor %f \n", x );
   printf( "letra es %c \n", letra );
   printf( "j = %i, \t letra = %c\n", j, letra );
   return 0;
}
```



Como leer datos de entrada

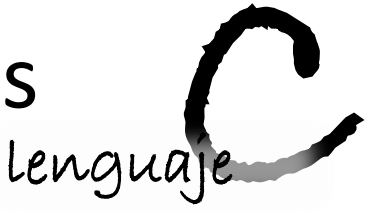
Es necesario importar la librería **stdio.h** para usar la primitiva **scanf**

Ejemplos de ingreso de datos




```
#include <stdio.h>

int main()
{
   char letra;
   int numero;
   float x;
   printf( "Ingrese una letra: " ); scanf( "%c", &letra );
   printf( "Ingrese un numero entero: " ); scanf("%i", &numero );
   printf( "Ingrese un numero real: " ); scanf( "%f", &x );
   printf( "Letra: %c, Numero entero: %i, Numero Real: %f  \n", letra, numero, x );
   return 0;
}
```

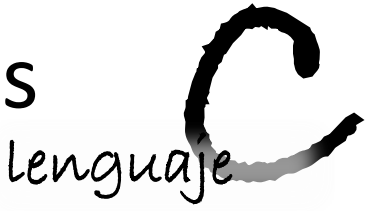
Descomposición de Problemas



- Un programa puede ser modularizado en C mediante el empleo de *funciones* (primitivas).
- La función principal que guía la ejecución de un programa es la llamada *main* (programa principal).
- La estructura de las funciones posee la siguiente sintaxis:

```
<tipo_dato_salida> <Id_funcion> (<Lista_de_parámetros>)
{
    …..
    return <expresión_de_tipo_”tipo_dato_salida”>;
}
```

Descomposición de Problemas



- La estructura de las primitivas posee la siguiente sintaxis:

```
<tipo_dato_salida> nombre (<Lista_de_parámetros>)
{
    …..
    return <expresión_de_tipo_”tipo_dato_salida”>;
}
```

Si la primitiva no se va a utilizar como una **función**, se devolverá *void* y se elimina el *return*

Descomposición de Problemas

Si la primitiva no se va a utilizar como una función, se devuelve *void* y se elimina el *return*

```
void mostrarCartel()
{
   printf("Bienvenidos!");
   printf("Autor: Jessica Carballido");
   printf("Ejercicio 2.a");
   printf("Suma de dígitos de un número.");
}

int main()
{
   mostrarCartel();
   …
   return(0);
}
```

También se va a retornar *void* cuando haya más de un dato de salida *(lo vemos más adelante)*

A función

Antes y fuera del main!!

```
ALGORITMO CantDigitos
DE: N (entero)
DS: Cantidad (entero)
DAux:  -
COMIENZO
  Cantidad ← 0
  MIENTRAS N > 0
      Cantidad ← Cantidad +1
      N ← N div 10
  FIN REPETIR
FIN ALGORITMO
```

```
int CantDigitos(int Numero)
{
   int Cantidad;

    Cantidad = 0;
    while (Numero > 0)
   {
      Cantidad = Cantidad+1;
      Numero = Numero/10;
   }
    return(Cantidad);
}
```

```
ALGORITMO CantDigitos
DE: N (entero)
DS: Cantidad (entero)
DAux:  -
COMIENZO
  Cantidad ← 0
  MIENTRAS N > 0
      Cantidad ← Cantidad + 1
      N ← N div 10
  FIN REPETIR
FIN ALGORITMO
```

Parámetros (DE)

```
int CantDigitos( int Numero)
{
   int Cantidad;

    Cantidad = 0;
    while (Numero > 0)
   {
      Cantidad = Cantidad+1;
      Numero = Numero/10;
   }
    return(Cantidad);
}
```

ALGORITMO CantDigitos

DE: N (entero)

DS: Cantidad (entero)

DAux: -

COMIENZO

  Cantidad ← 0

  MIENTRAS N > 0

    Cantidad ← Cantidad + 1

    N ← N div 10

  FIN REPETIR

FIN ALGORITMO


```
int CantDigitos( int Numero)
{
   int Cantidad;

    Cantidad = 0;
    while (Numero > 0)
   {
      Cantidad = Cantidad+1;
      Numero = Numero/10;
   }
    return(Cantidad);
}
```

ALGORITMO CantDigitos
DE: N (entero)
DS: Cantidad (entero)
DAux: -
COMIENZO
  Cantidad ← 0
  MIENTRAS N > 0
    Cantidad ← Cantidad + 1
    N ← N div 10
  FIN REPETIR
FIN ALGORITMO

Dato de salida

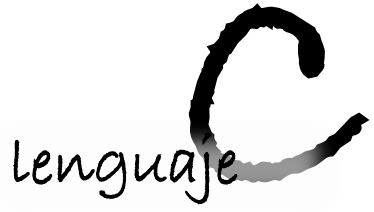
Lenguaje C

```
int CantDigitos( int Numero)
{
  int Cantidad;

   Cantidad = 0;
   while (Numero > 0)
   {
      Cantidad = Cantidad+1;
      Numero = Numero/10;
   }
   return(Cantidad);
}
```

```
#include <stdio.h>
        int CantDigitos( int Numero )
        {
           int Cantidad;
           Cantidad = 0;
           while (Numero > 0)
           {
                Cantidad = Cantidad+1;
                Numero = Numero/10;
           }
           return(Cantidad);
        }
 int main( )
{    int a, b;

     printf( "Ingrese un numero entero: ");
     scanf( "%i", &a);
     printf( "Ingrese un numero entero: ");
     scanf( "%i", &b);
     If (CantDigitos(a) == CantDigitos(b))
                    printf("Tienen igual cantidad de digitos");
     else           printf("NO tienen igual cantidad de digitos");
     return(0);
}
```

Problema: Determinar si dos **números positivos** tienen igual cantidad de dígitos.

Ejercicio: Sucesiones



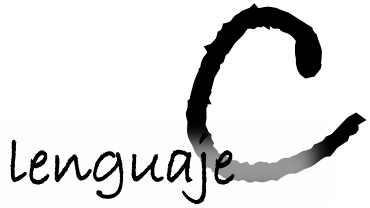
Escriba un programa en C que pida al usuario un valor natural N y muestre en pantalla el N-ésimo término de la siguiente sucesión infinita:

S = 2/1, 4/2, 8/6, 16/24, 32/120, 64/720…

Término general: $2^N/N!$

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

int pot(int base, int exp)
{
   int potencia=1, i;

   for (i=1; i<=exp; i++)
      potencia=potencia*base;
   return(potencia);
}

int fact(int numero)
{
   int factorial=1, i;

   for (i=1; i<=numero; i++)
      factorial=factorial*i;
   return(factorial);
}
```

Ejercicio: Sucesiones



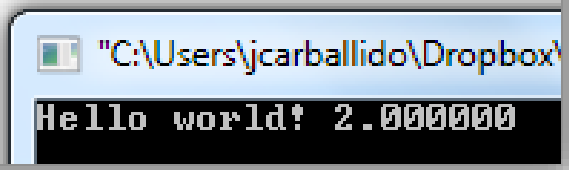

```c
float termino(int n)
{
    return ((float)pot(2,n)/fact(n));
}

int main()
{
    int N;

    printf("Ingrese el numero de termino que desea calcular: ");
    scanf("%i", &N);
    printf("El valor del termino %i en la sucesion es: %f", N, termino(N));

    return 0;
}
```

Conversión explícita de tipo

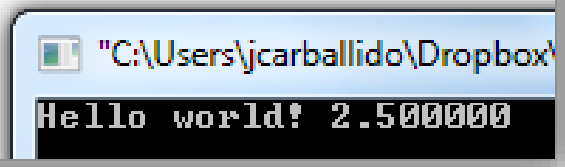
Cuidado con la división entera y las conversiones de tipos **implícitas**!

lenguaje C

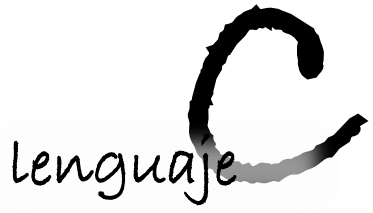
```c
int main()
{
    int a, b; float c;
    a=5; b=2;
    c=a/b;
    printf("Hello world! %f\n", c);
    return 0;
}
```




```c
int main()
{
    int a, b; float c;
    a=5; b=2;
    c=a/(float)b;
    printf("Hello world! %f\n", c);
    return 0;
}
```



Orígenes

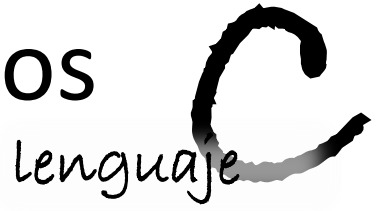


- El lenguaje **C** fue diseñado por Dennis Ritchie en el año 1972.
- Deriva de dos lenguajes anteriores llamados **BCPL** y **B**.
- Se usó para **implementar** buena parte del sistema operativo **UNIX**, y actualmente se lo sigue usando para codificar todo tipo de sistemas de software.

Compilación

Un **compilador** es un programa
que **traduce** un programa
escrito en un lenguaje de programación
a otro lenguaje de programación de más bajo nivel,
generando un programa equivalente
que la máquina será capaz de interpretar.

Interpretados vs. Compilados



- **Lenguajes interpretados (R):**
  - Las instrucciones se transforman en lenguaje máquina a medida que va siendo ejecutado el programa.
  - Los pequeños cambios pueden probarse rápidamente.
- **Lenguajes compilados (C):**
  - Los programas son traducidos en su totalidad a lenguaje máquina antes de ser ejecutados.
  - Para probar un pequeño cambio se debe compilar todo nuevamente.

Sistema de tipos

Tipado estático:

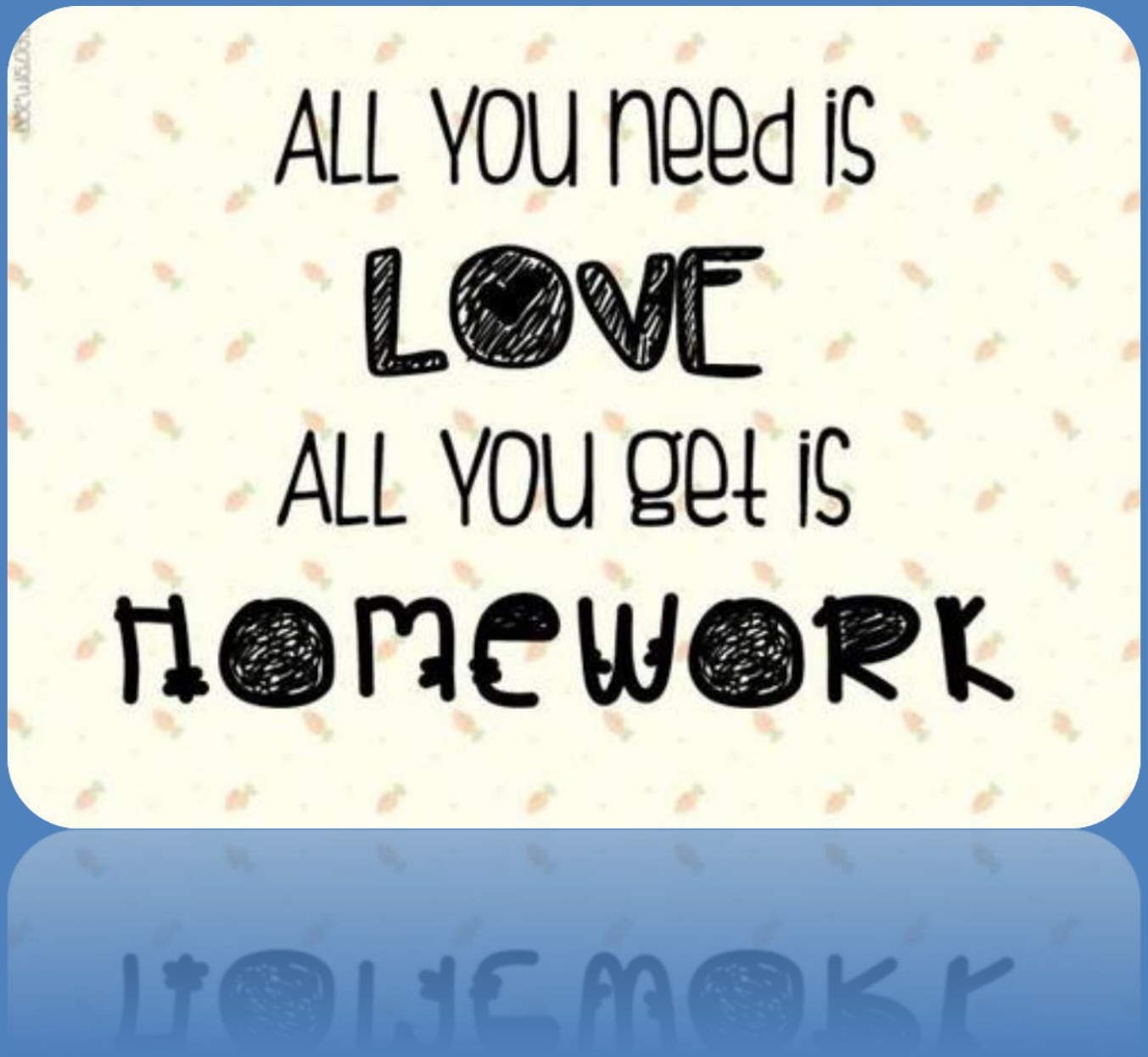
El chequeo de tipos (dominios) de las variables (datos) se realiza en compilación (C, C++, Java).

Tipado dinámico:

El chequeo de tipos se realiza en ejecución (R, Perl, Python, Lisp).

Comparado con el tipado dinámico, el estático permite que los errores de programación sean detectados antes, y que la ejecución del programa sea más eficiente.

El dinámico provee más flexibilidad.


ALL YOU need is
LOVE
ALL YOU get is
HOMEWORK

Ejercicio

Números Hermanos

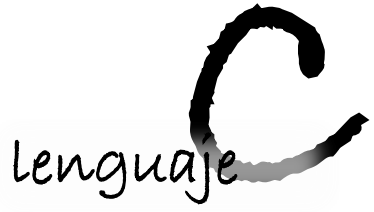
Dos números NATURALES se dicen “hermanos” cuando las sumas de sus dígitos son iguales.

Ejemplos:

- 3245 y 77 son hermanos.
- 88 y 556 son hermanos.
- 898 y 12 no son hermanos.

* Escribir un algoritmo para calcular la suma de los dígitos de un número.
* Escribir un programa en C en el que se defina la función SumaDigitos y utilizándola como primitiva, se reciban dos números y se decida si son hermanos. Mostrar un cartel con el resultado.

Ejemplo de Programa



Números Equivalentes en Dígitos

Dos números son equivalentes en dígitos si están formados exactamente por los mismos dígitos, sin importar el orden.

Ejemplos:

- 3245 es equivalente en dígitos 2453
- 898 es equivalente en dígitos 889
- 898 NO es equivalente en dígitos 9889
- 7673 NO es equivalente en dígitos 3676
- 7673 NO es equivalente en dígitos 367

```
ALGORITMO ContarVeces
DATOS DE ENTRADA: Número, Dígito
DATOS DE SALIDA: Cantidad
DATOS AUXILIARES:
COMIENZO
  Cantidad ← 0
  MIENTRAS (Número > 0)
     SI  (Número mod 10) = Dígito
              ENTONCES  Cantidad ← Cantidad+1
     FIN SI
     Número ← Número div 10
   FIN MIENTRAS
FIN ALGORITMO
```

ALGORITMO Números Equivalentes en Dígitos
DATOS DE ENTRADA: Num1, Num2 {naturales}
DATOS DE SALIDA: SonEqui {lógico}
DATOS AUXILIARES: Dígito, Cant1, Cant2 {natural}
COMIENZO
  Dígito ← 0
  SonEqui ← Verdadero
  MIENTRAS (Dígito≤9) Y (SonEqui)
    Cant1 ← ContarVeces(Num1,Dígito)
    Cant2 ← ContarVeces(Num2,Dígito)
    SI (Cant1 ≠ Cant2)
        ENTONCES SonEqui ← Falso
    FIN SI
    Dígito ← Dígito + 1
FIN ALGORITMO

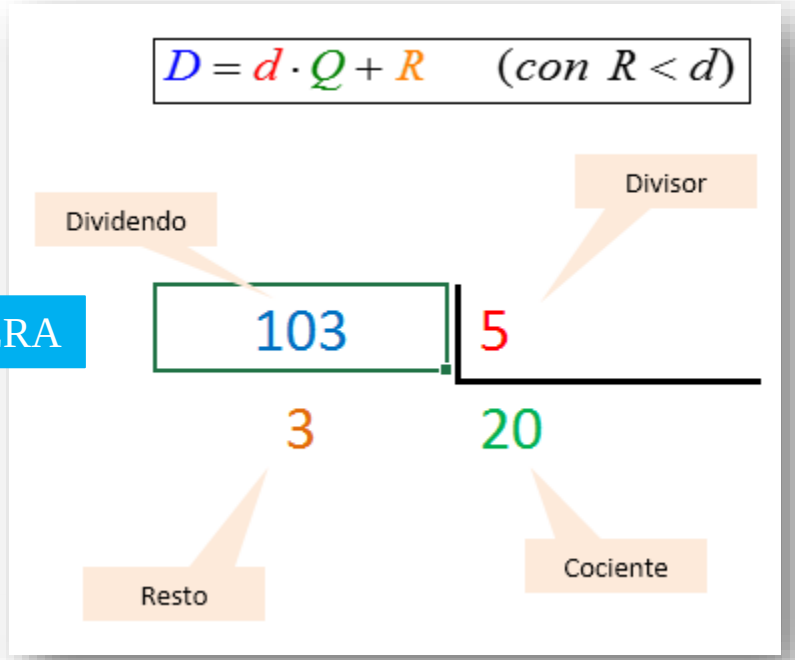
Código en C


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

int ContarVeces( int Numero, int Digito )
{
   int Cantidad;
   Cantidad = 0;
   while (Numero > 0)
   {
      if ((Numero%10) == Digito)
            Cantidad = Cantidad+1;
      Numero = Numero/10;
   }
    return(Cantidad);
}
```

Datos de entrada

Dato de salida

DIVISIÓN ENTERA



Dra. Jessica Andrea Carballido
CONICET - DCIC (UNS)

```
bool NumerosEquivalentesEnDigitos(int Num1, int Num2)
{
   bool SonEquivalentes;
   int Dig;
   Dig = 0;
   SonEquivalentes = true;
   while ((Dig<=9) && (SonEquivalentes==true))
    {
    if(ContarVeces(Num1, Dig) != ContarVeces(Num2, Dig))
                    SonEquivalentes = false;
      else
                    Dig = Dig+1;
    }
    return(SonEquivalentes);
}
```

Esto podría estar en el *main*

```
int main( )
{
   int N1, N2;

   printf( "Ingrese un numero entero: " ); scanf( "%i", &N1 );
   printf( "Ingrese un numero entero: " ); scanf( "%i", &N2 );

   if (NumerosEquivalentesEnDigitos(N1, N2) == true)
         printf("Son equivalentes en digitos\n");
     else
        printf("No son equivalentes en digitos\n");

return(0);
}
```


Código en C

Código en C


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


int ContarVeces( int Numero, int Digito )
{
    int Cantidad;
    Cantidad = 0;
    while (Numero > 0)
    {
        if ((Numero%10) == Digito)
                Cantidad = Cantidad+1;
        Numero = Numero/10;
    }
     return(Cantidad);
}
```

versión 2.0

```
int main( )
{
   int Num1, Num2, Digito;
   bool SonEquivalentes;

   printf( "Ingrese un numero entero: " ); scanf( "%i", &Num1 );
   printf( "Ingrese un numero entero: " ); scanf( "%i", &Num2 );

Digito = 0;
SonEquivalentes = true;
while ((Digito<=9) && (SonEquivalentes==true))
 {
  if   (ContarVeces(Num1, Digito) != ContarVeces(Num2, Digito))
              SonEquivalentes = false;
      else
    Digito = Digito+1;
 }

   if (SonEquivalentes == true)
          printf("Son equivalentes en digitos\n");
       else
            printf("No son equivalentes en digitos\n");
return(0);
}
```


GOOD MORNING...
...ENJOY A
BEAUTIFUL DAY