Programación 1

Tema 1

Problemas de tratamiento de información, algoritmos y programas

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Problemas, algoritmos y programas

- Problemas de **tratamiento de información**
  - Objetivo: resolución **automática** del problema
  - ¿Quién? Un computador
  - Necesidad de **programarlo**

Algoritmo

- Conjunto de operaciones
  - ordenado,
  - finito,
  - carente de ambigüedades,

  que permite hallar la solución de un problema [de tratamiento de información]

«Deberes» para hoy

- Análisis de una receta para hacer tortilla de patata
  - Estructura
  - Modos de las formas verbales

Índice

- Problemas de tratamiento de información
- Algoritmos y programas
- Ejemplos de programas C++
- Funciones y especificación
- Propiedades de un algoritmo

Ejemplos de problemas de tratamiento de información

- Facilitar la escritura, edición, impresión y preservación digital de un texto
- Gestionar la información académica de los alumnos de la Universidad de Zaragoza
- Averiguar el número primo que sigue a 104743
- Permitir que una o varias personas jueguen en un entrono virtual persiguiendo un determinado objetivo
- Guiar el rayo láser que realiza queratectomía fotorrefractiva para corregir la miopía en ojos humanos
- Permitir que varias personas compartan entre sí en Internet información personal como noticias, fotografías, etc.

Problemas, algoritmos y programas

- **Problema** (de tratamiento de información)

- **Método para su resolución**

- **Algoritmo**

- **Programa**

Problemas, algoritmos y programas

- **Problema** (de tratamiento de información)

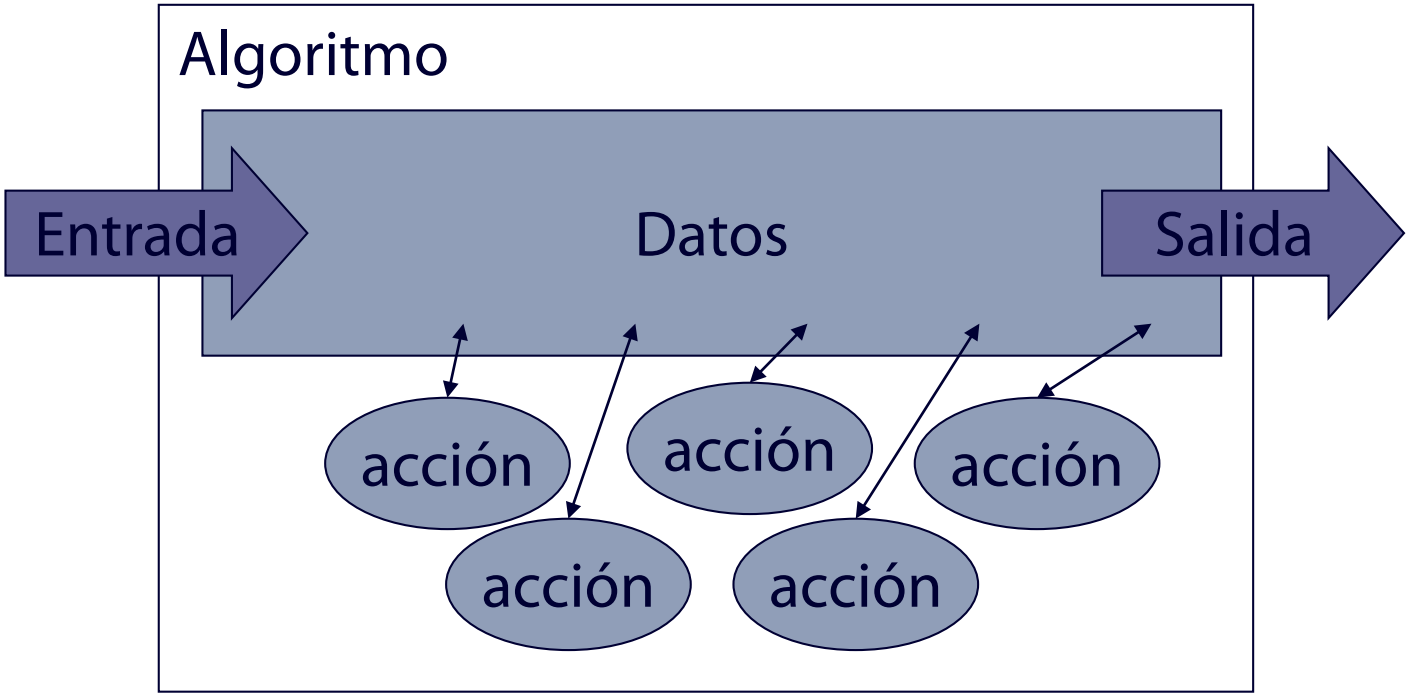
- **Método para su resolución**

- **Programa**

Algoritmo

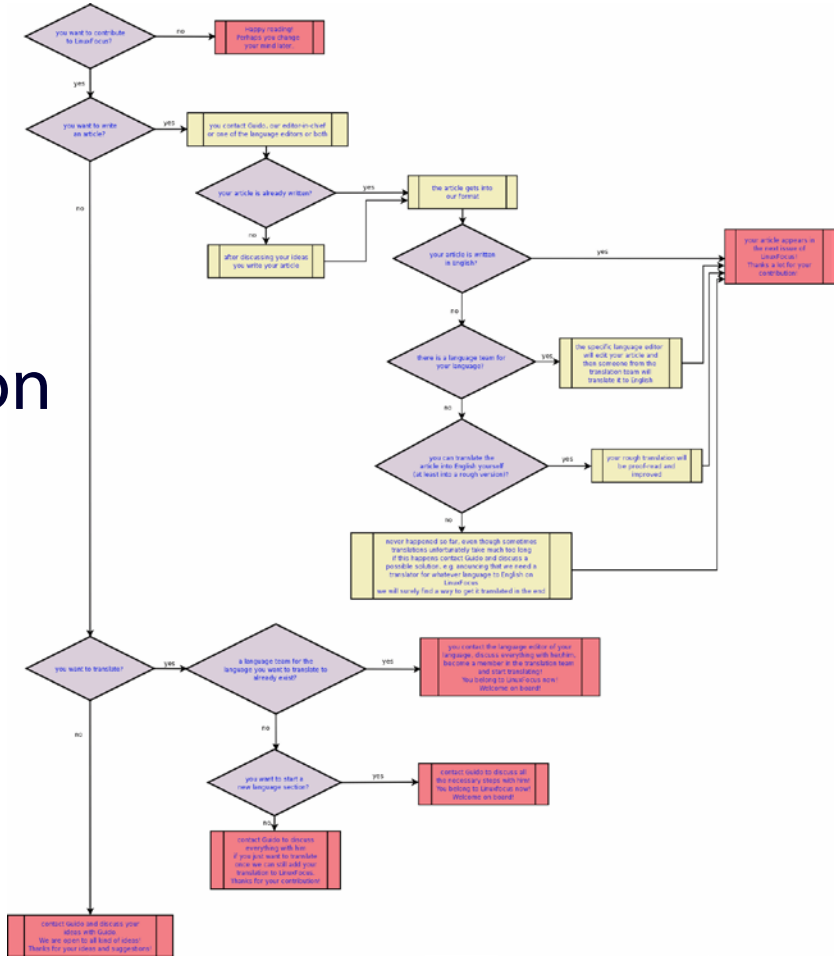
- Conjunto ordenado y finito de operaciones, carente de ambigüedades, que permite hallar la solución de un problema de tratamiento de información
- Consta de
  - **Descripción de la información** asociada al problema
  - **Descripción del modo de tratamiento** de esta información.

Esquema de algoritmo



Expresión de un algoritmo

- Lenguaje natural
- Notación algorítmica
- Notación gráfica
  - Diagramas de flujo
- Lenguaje de programación
  - Ada, Pascal, Módula-2, C
  - **C++**, Java
  - Lisp, Prolog
  - Fortran, Cobol



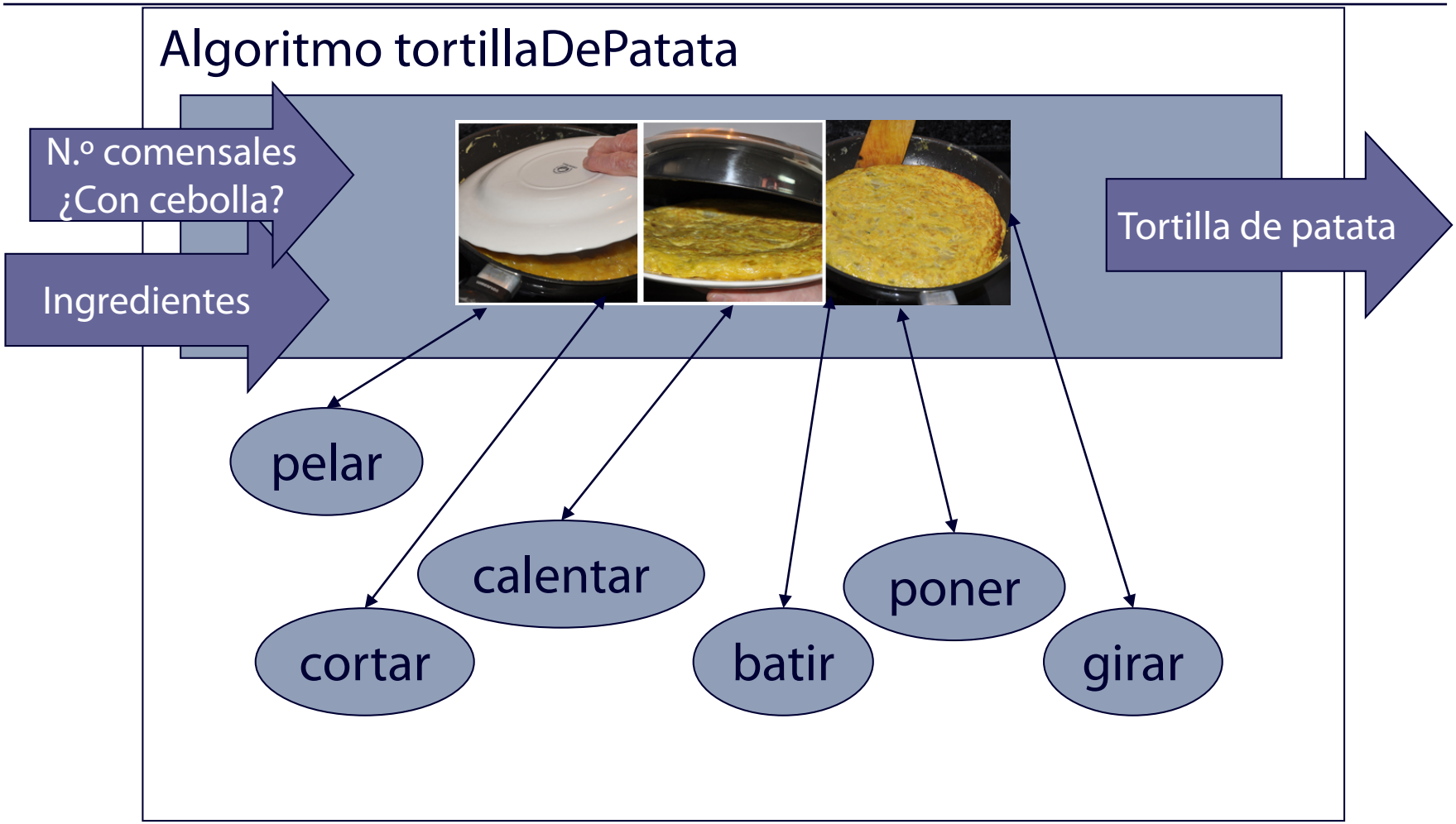
Algoritmo en lenguaje natural

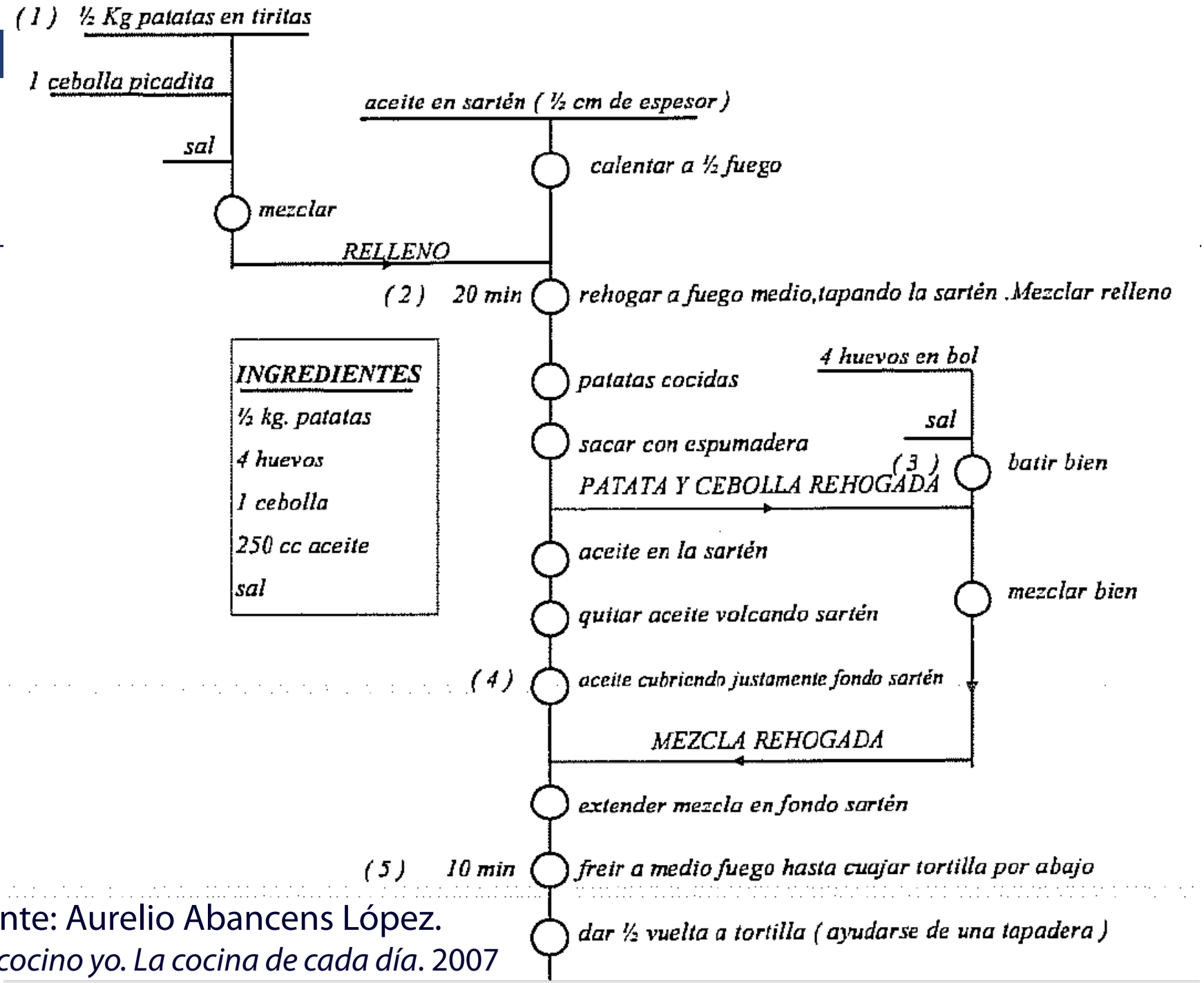
- Ingredientes para 4 comensales
  - 4 huevos
  - Medio kilo de patatas
  - Media cebolla
  - Aceite de oliva
  - Sal

- Elaboración:
  - Corte las patatas en trocitos bien finos. Ponga a calentar abundante aceite de oliva en la sartén. Ponga las patatas en la sartén cuando el aceite esté bien caliente (nunca debe humear). Añada un poco de sal. Si la quiere con cebolla, añada la cebolla picada. Cuando las patatas estén bien doraditas, sáquelas y escúrralas. Bata bien los huevos, con una pizca de sal. Añada las patatas ya fritas y mezcle bien. Retire el aceite sobrante de la sartén y vuelva a ponerla al fuego. Cuando la sartén esté bien caliente, eche la mezcla de huevo y patatas. Cuando ya está hecha o cuajada por debajo, darle la vuelta con un plato plano o una tapadera.

File:Tortilla de Patatas (Corte transversal).jpg. (9 de marzo de 2017). *Wikimedia Commons, the free media repository*. Accedido el 18 de septiembre de 2017. https://commons.wikimedia.org/w/index.php?title=File:Tortilla_de_Patatas_(Corte_transversal).jpg&oldid=236535592.

Algoritmo para cocinar una tortilla de patata





Fuente: Aurelio Abancens López.

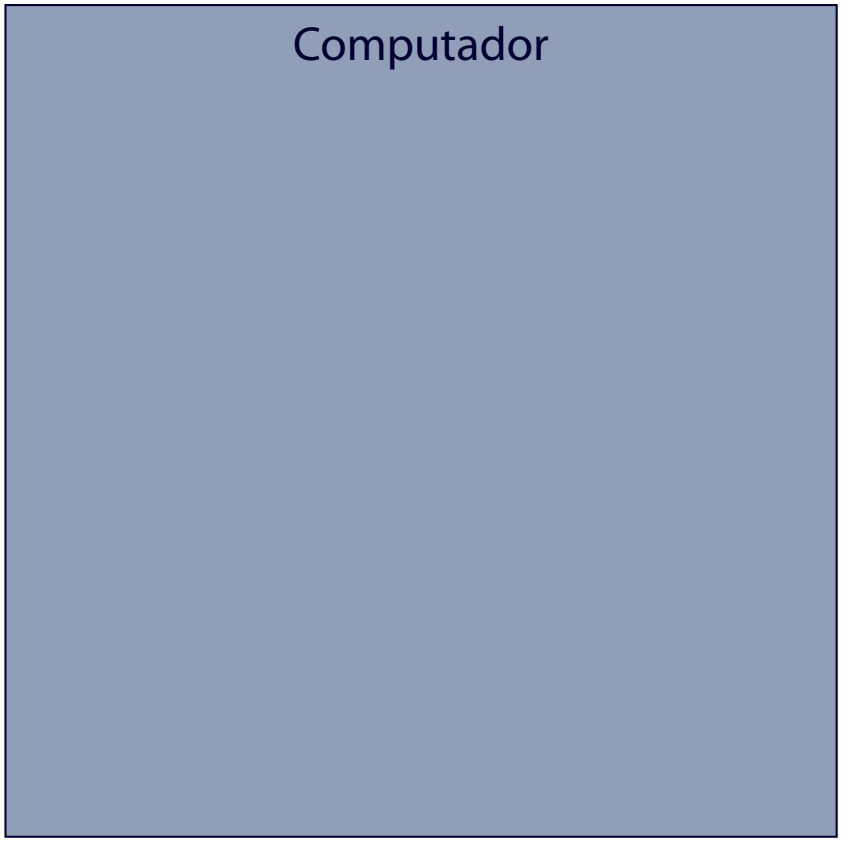
Así cocino yo. La cocina de cada día. 2007

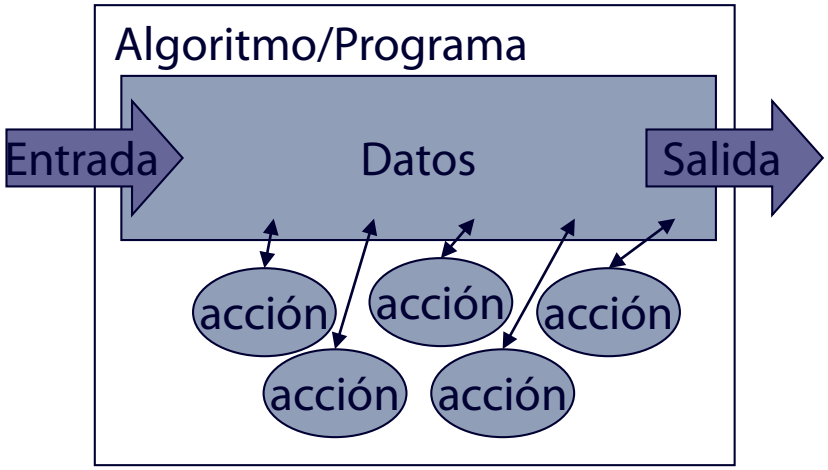
Algoritmo en notación algorítmica

```
algoritmo tortillaDePatata(
     personas: datoDeEntrada entero;
     seQuiereConCebolla:
          datoDeEntrada booleano);
{ Versión en una notación algorítmica
de la receta de tortilla de patata }

ingredientes
     1 huevo por persona
     125 g de patatas por persona
     1/8 de cebolla por persona
     Aceite de oliva
     Sal

menaje
     sartén
     tenedor
     plato
```

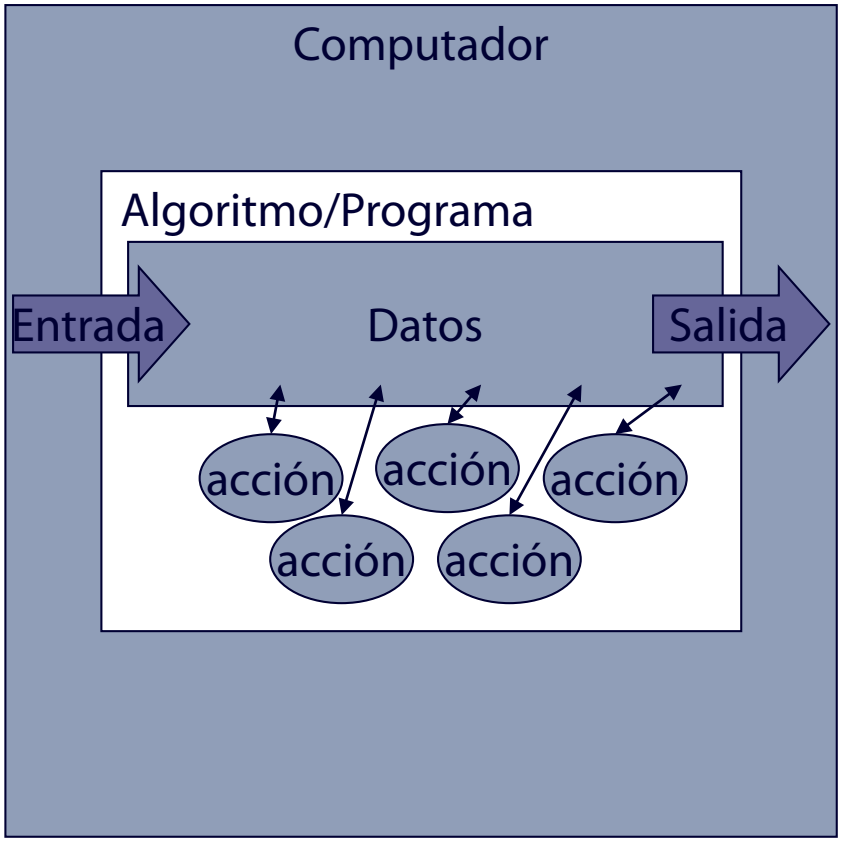
```
principio
     pelar(patatas);
     cortar(patatas);
     calentar(aceite, sartén);
     mientrasQue no estéBienCaliente(aceite) hacer
          esperar;
     finMQ;
     poner(patatas, sartén);
     poner(sal, sartén);

     si seQuiereConCebolla entonces
          picar(cebolla);
          poner(cebolla, sartén);
     finSi;
     mientrasQue no estén Doradas(patatas) hacer
          esperar;
     finMQ;
     batir(huevos);
     poner(sal, huevos);
     ...

fin.
```

Nuestro modelo de computador

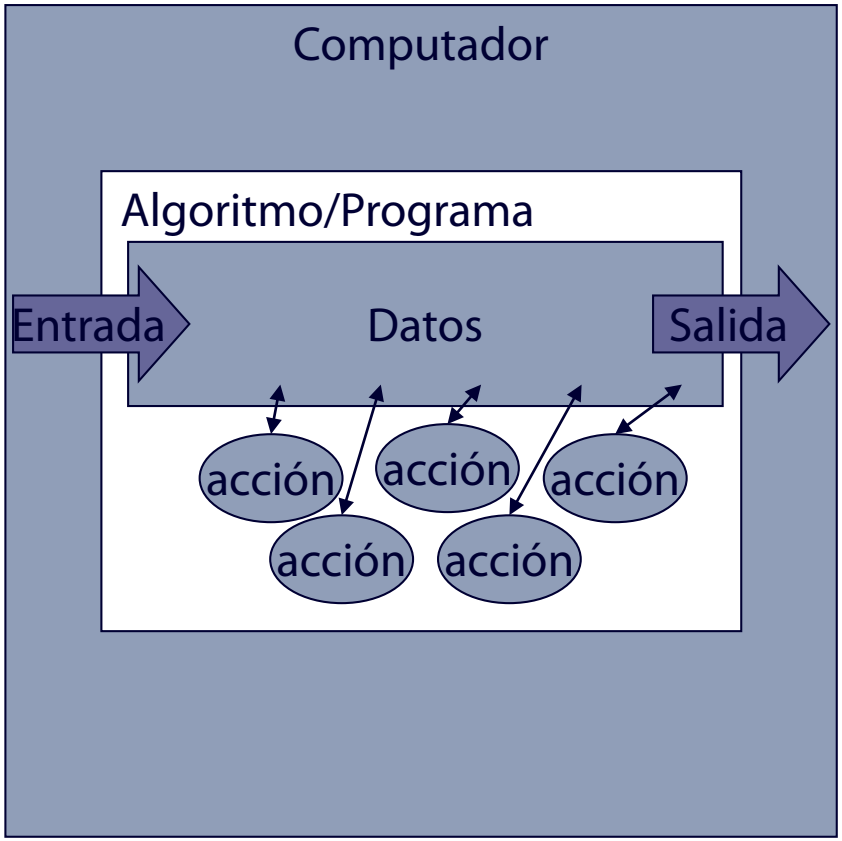




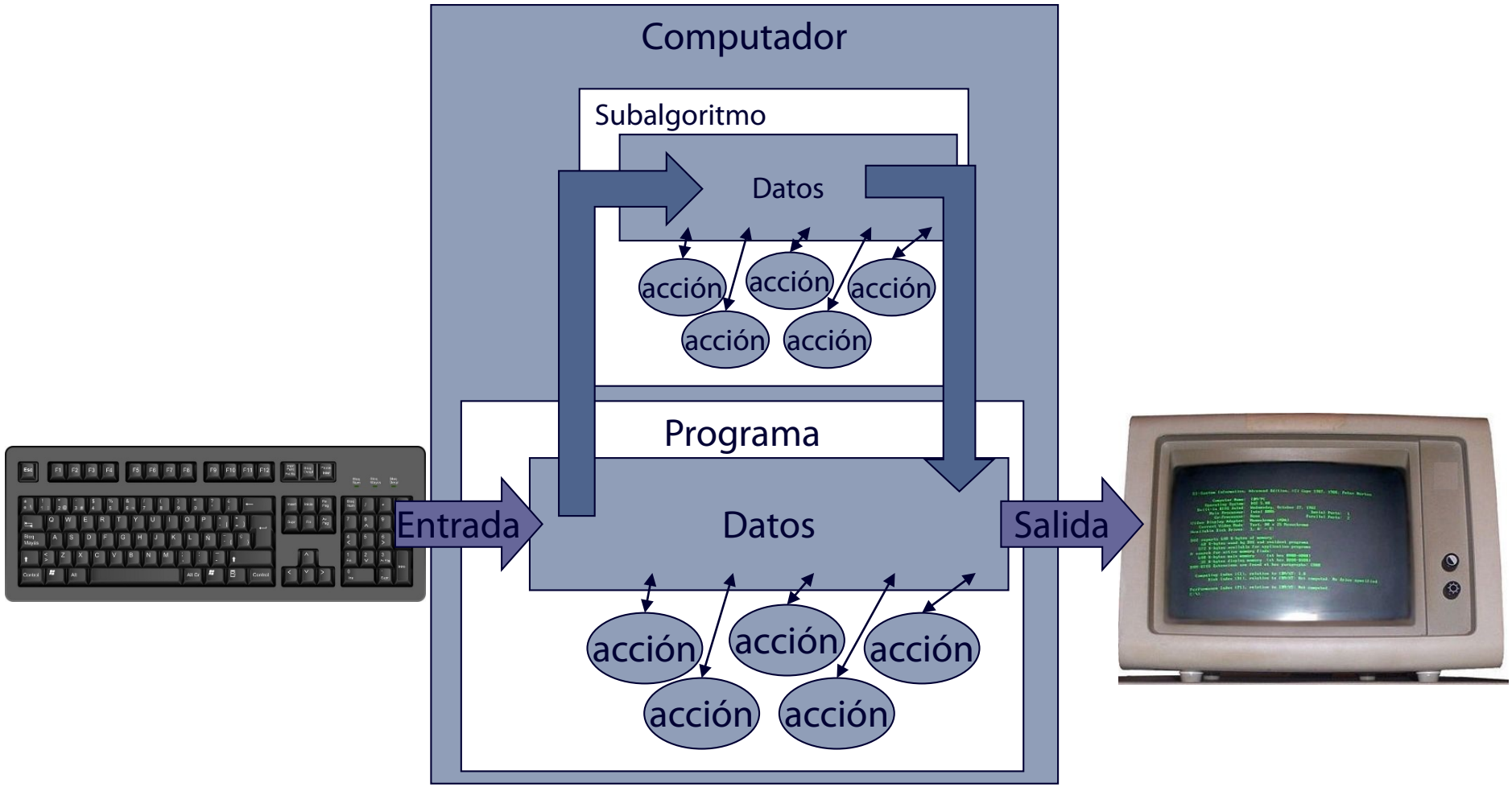
Nuestro modelo de computador



Nuestro modelo de computador



Nuestro modelo de computador



Un algoritmo en una notación algorítmica

```
algoritmo bienvenida;
{ Pre:  --- }
{ Post: Aparece escrita una línea en pantalla
        con un mensaje de bienvenida }

principio
   { Acciones a ejecutar cuando sea invocado }
   escribir(pantalla, "Bienvenidos a UNIZAR");
fin.
```

Un programa en Ada

```
with ada.text_IO;

procedure bienvenida is
-- Pre:  ---
-- Post: Aparece escrita una línea en pantalla
--       con un mensaje de bienvenida


begin
   -- Acciones que ejecutará el programa cada vez
   -- que sea invocado
   ada.text_IO.put("Bienvenidos a UNIZAR");
   ada.text_IO.new_line;
end bienvenida;
```

Un programa en Java

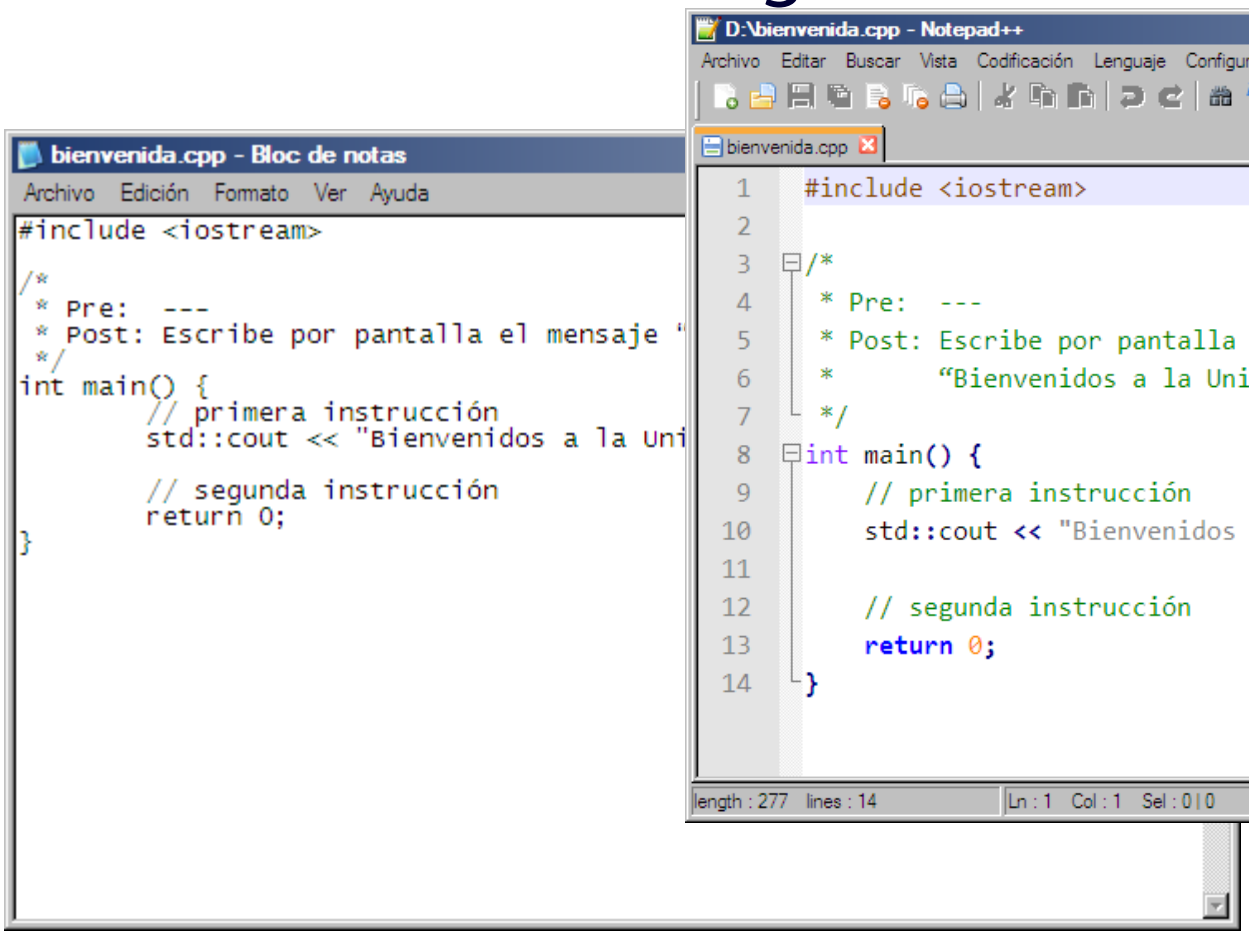
```
package es.unizar.eina.prog1.cap1;

/**
  * Al construir un programa Java alrededor de esta clase se
  * ejecuta su método «main» que escribe un mensaje de
  * bienvenida a la Universidad
  */
public class Bienvenida {

   /**
     *  Pre: ---
     *  Post: Escribe por pantalla una línea con  el mensaje
     *        "Bienvenidos a la Universidad"
     */
   public static void main(String[] argumentos) {
       // El código a ejecutar se limita a una sola instrucción
       System.out.println("Bienvenidos a la Universidad");
   }
}
```

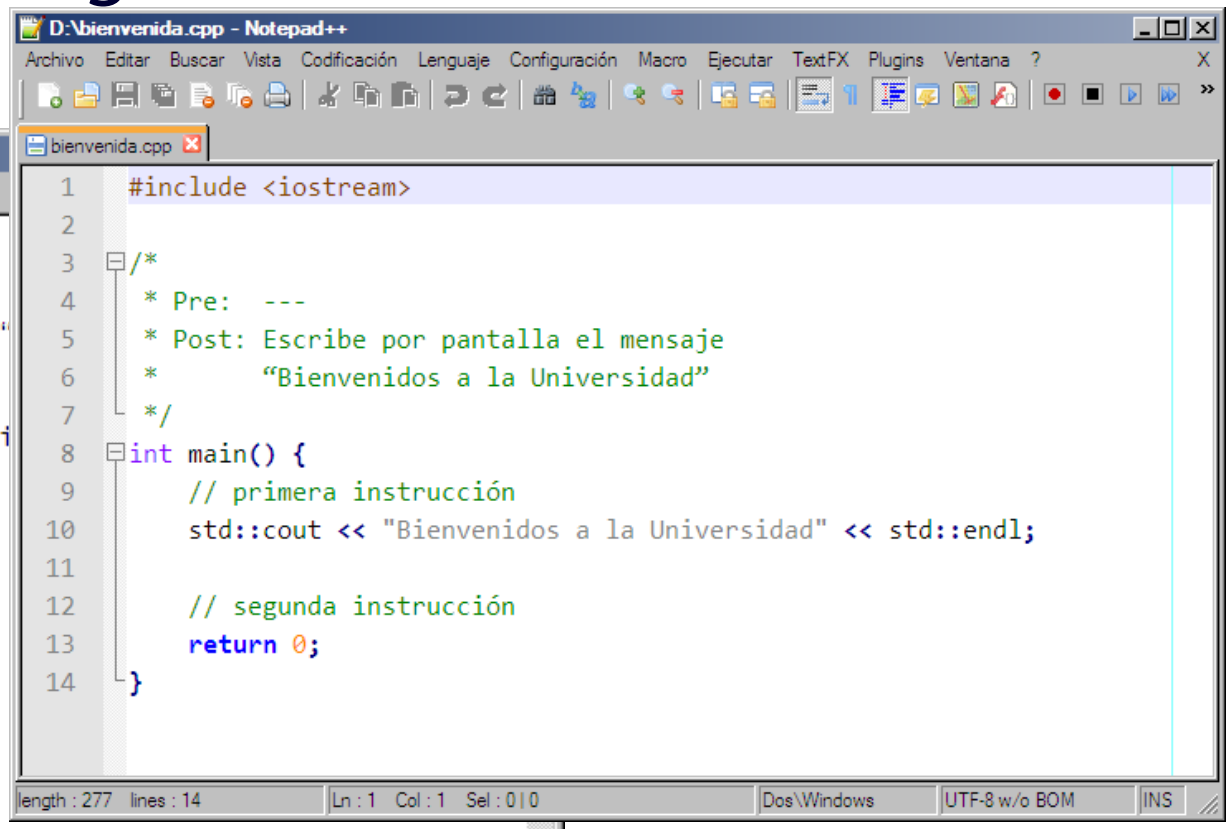
Un primer programa en C++

```
#include <iostream>

/*
 * Pre:  ---
 * Post: Escribe por pantalla el mensaje
 *       "Bienvenidos a la Universidad"
 */
int main() {
   // primera instrucción
   std::cout << "Bienvenidos a la Universidad" << std::endl;

   // segunda instrucción
   return 0;
}
```

¿Cómo se ejecuta el código C++?

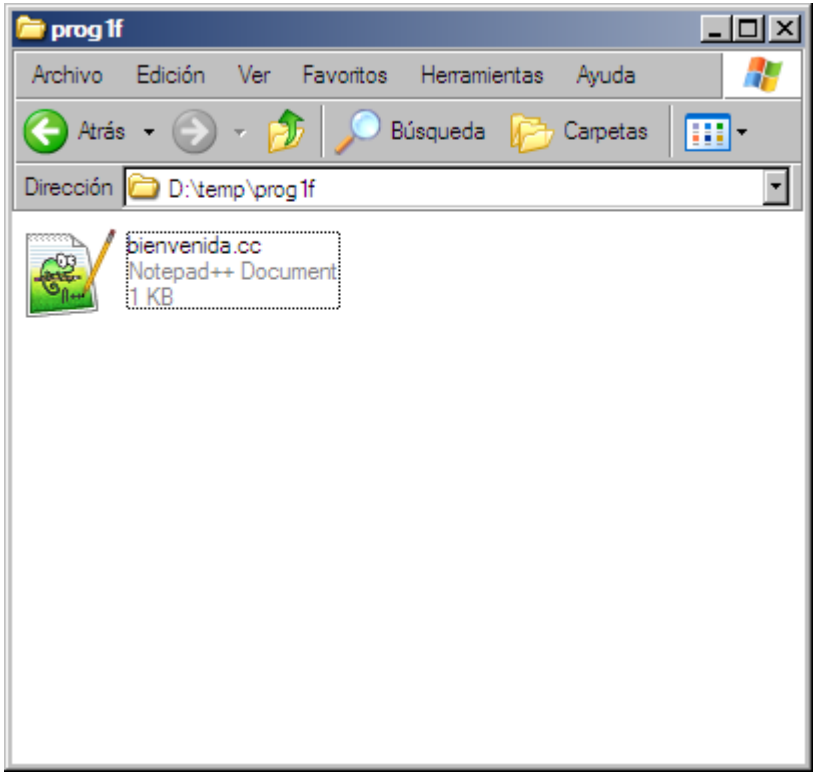
- Edición del código fuente





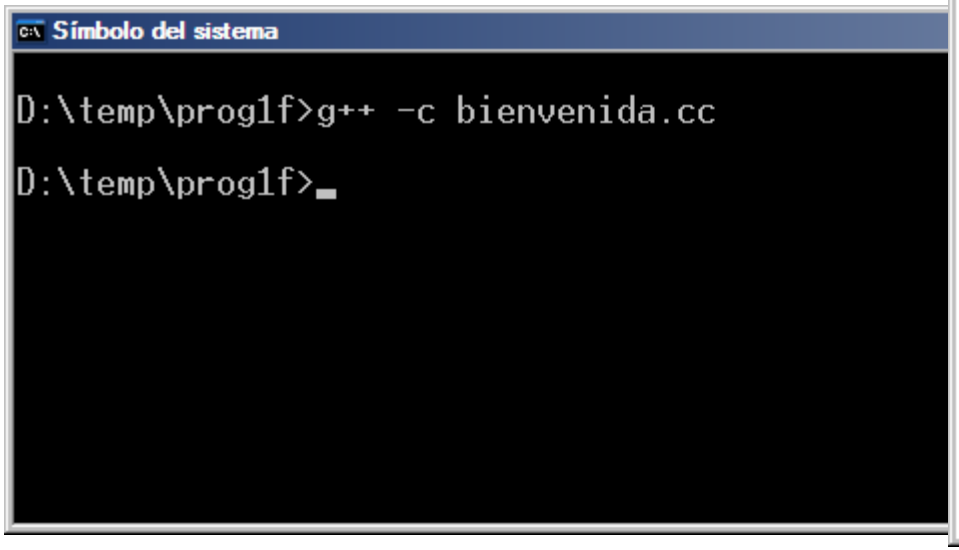
¿Cómo se ejecuta el código C++?

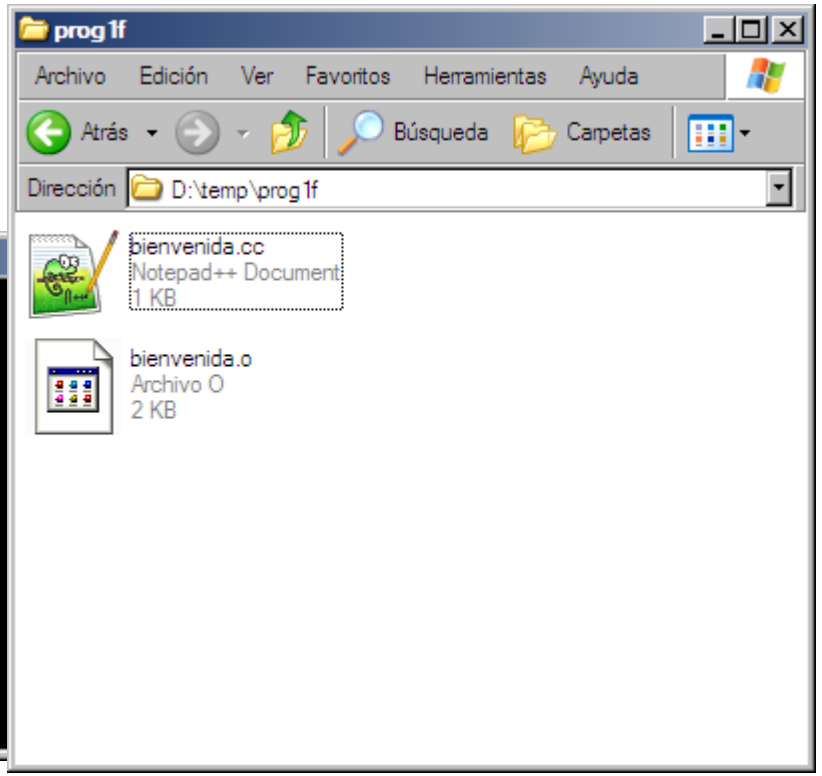
- Edición del código fuente



¿Cómo se ejecuta el código C++?

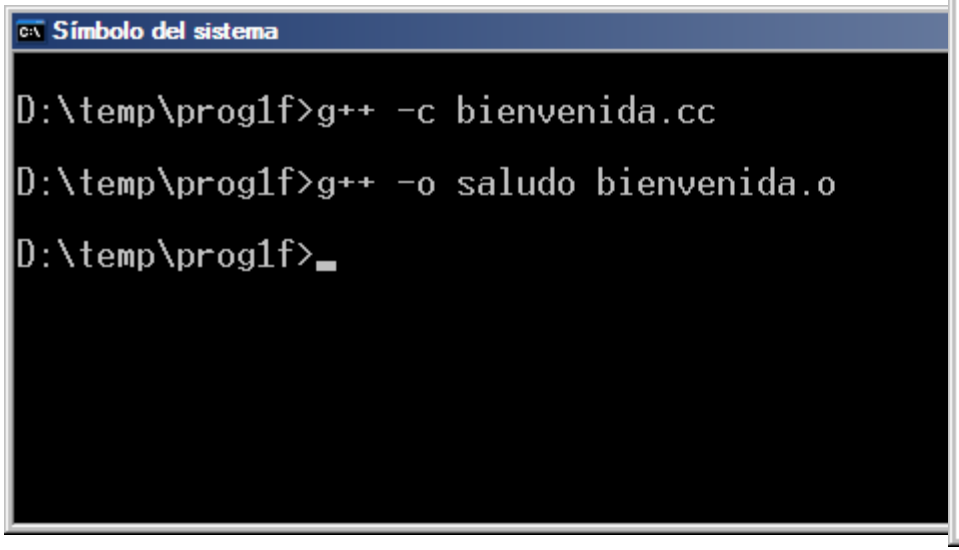
- Edición del código fuente
- Compilación del código fuente

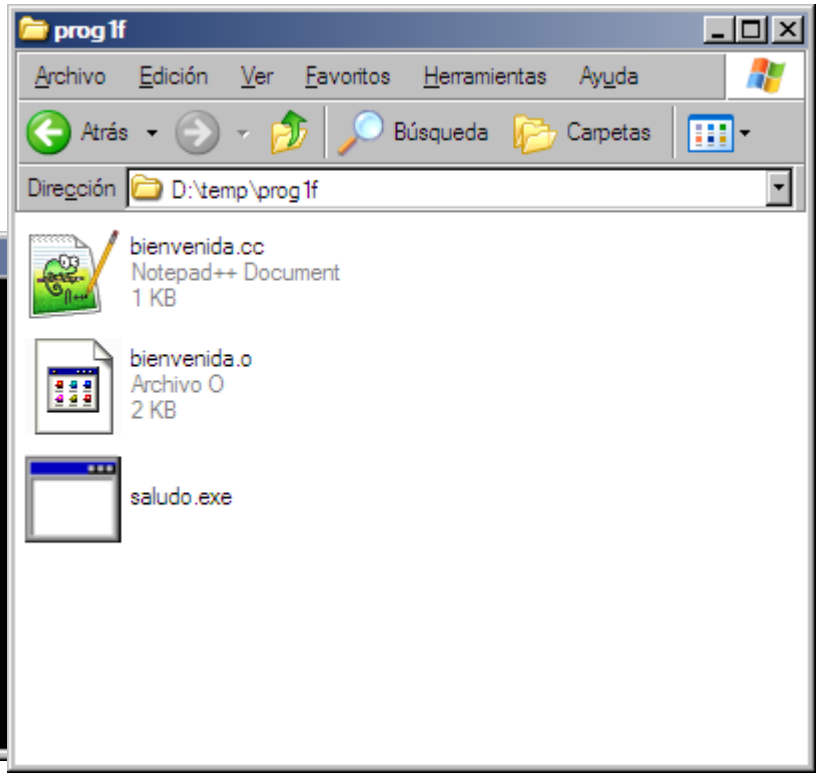




¿Cómo se ejecuta el código C++?

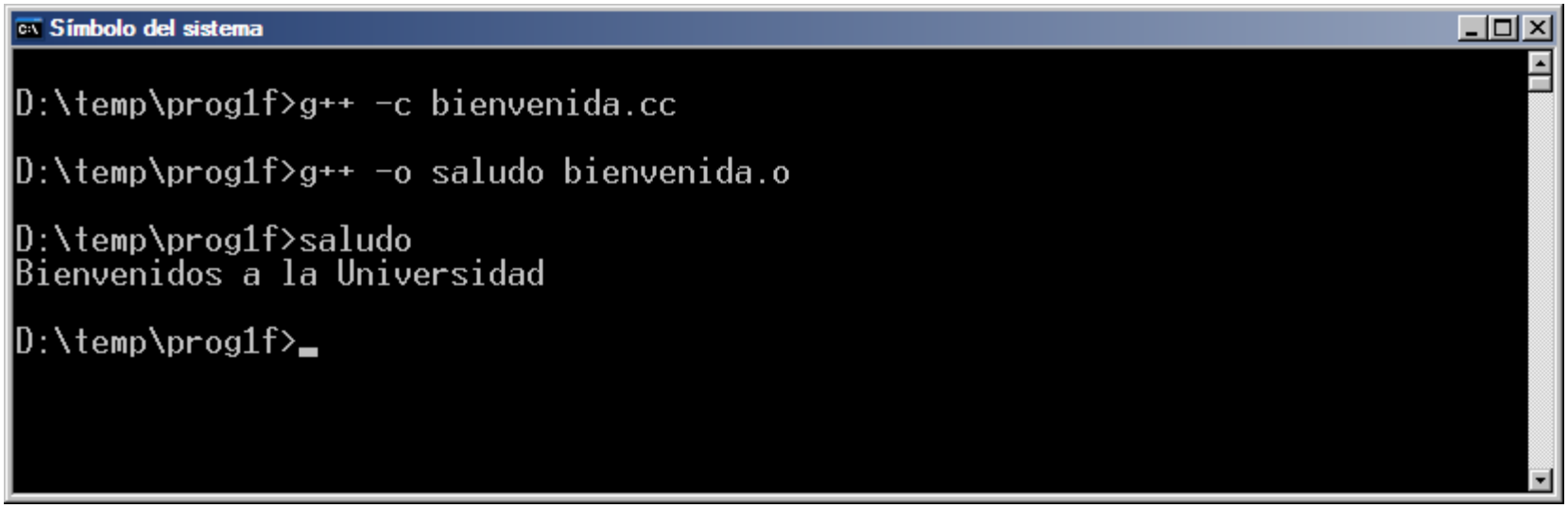
- Edición del código fuente
- Compilación del código fuente



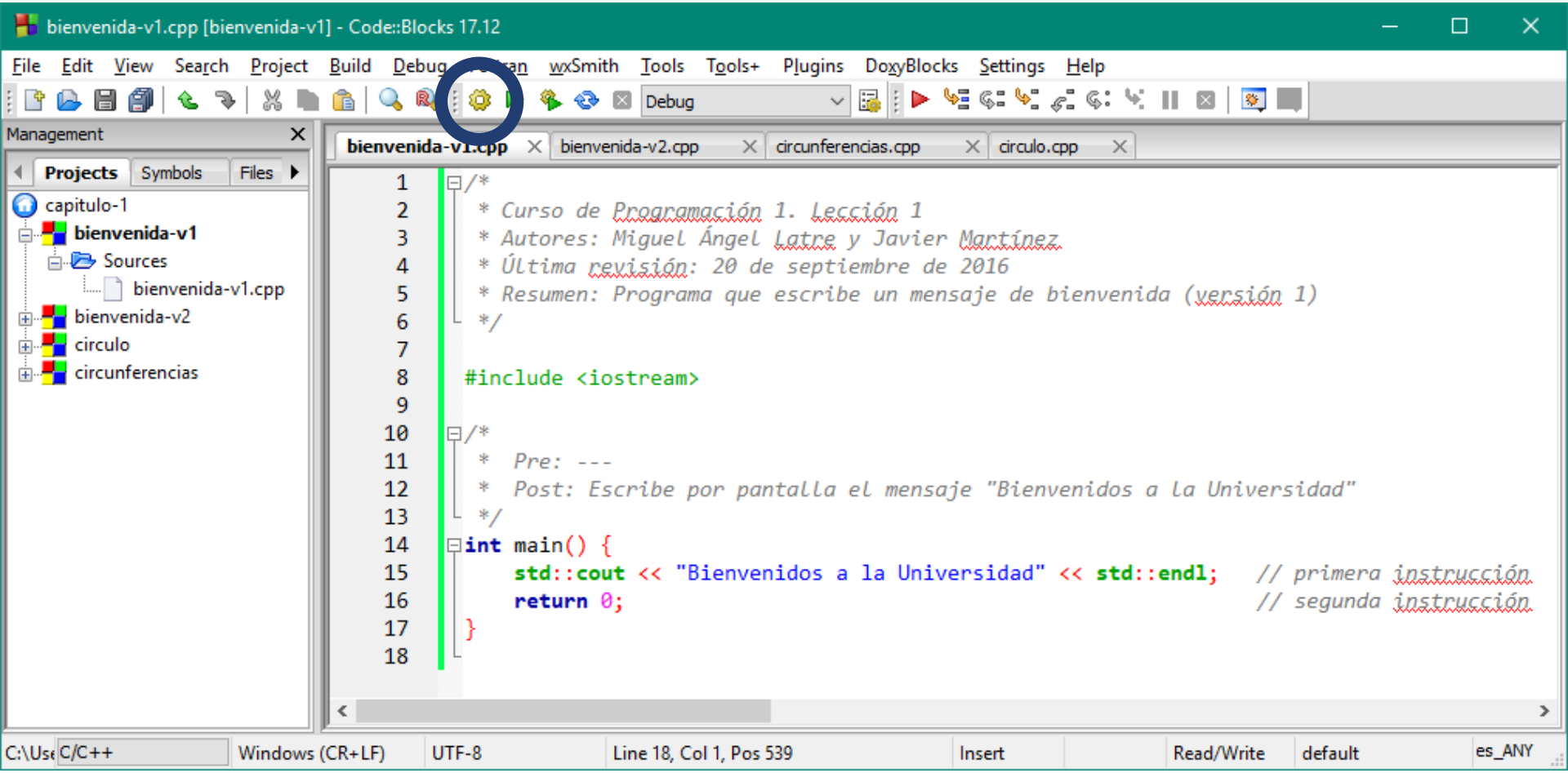


¿Cómo se ejecuta el código C++?

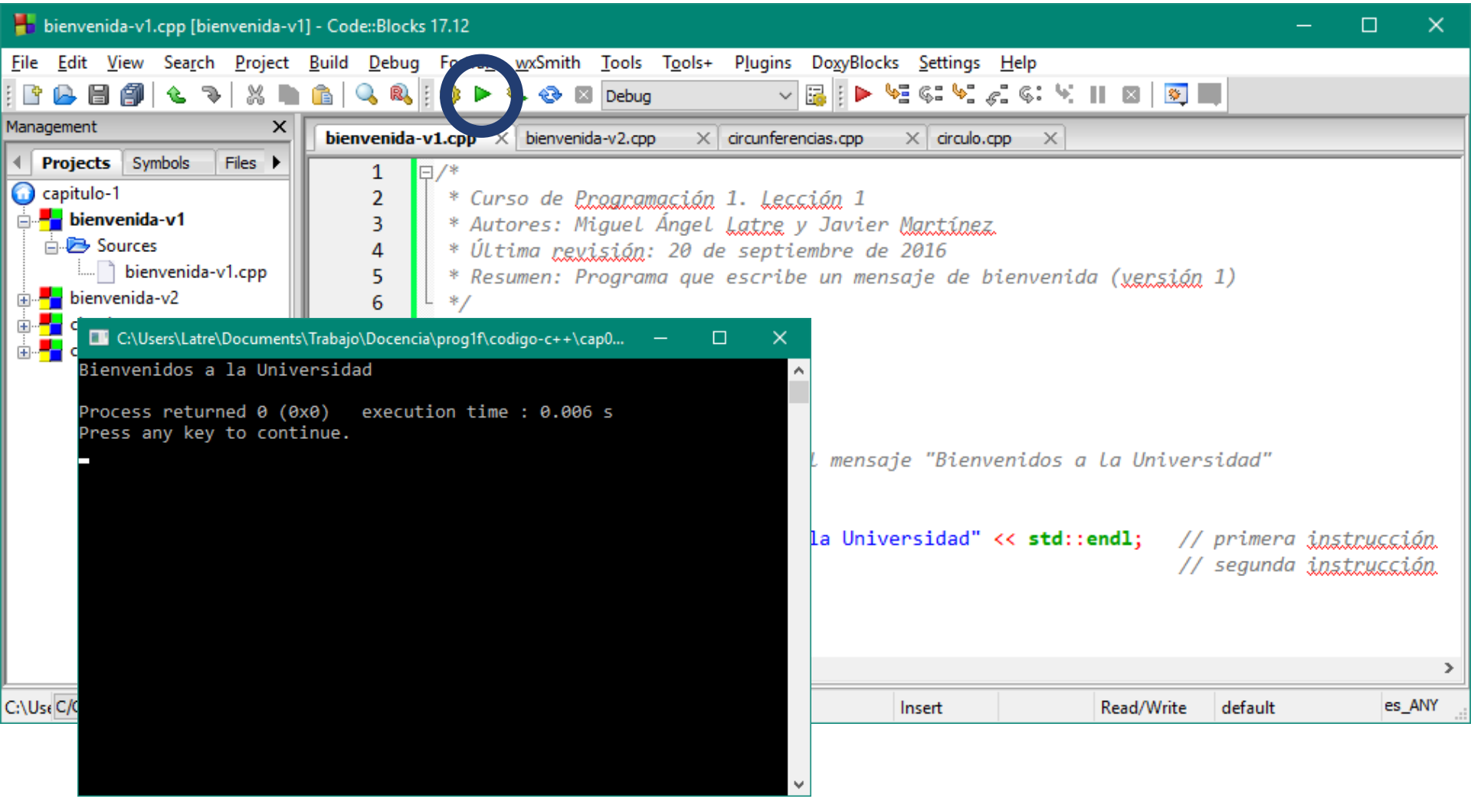
- Edición del código fuente
- Compilación del código fuente
- Ejecución del código ejecutable



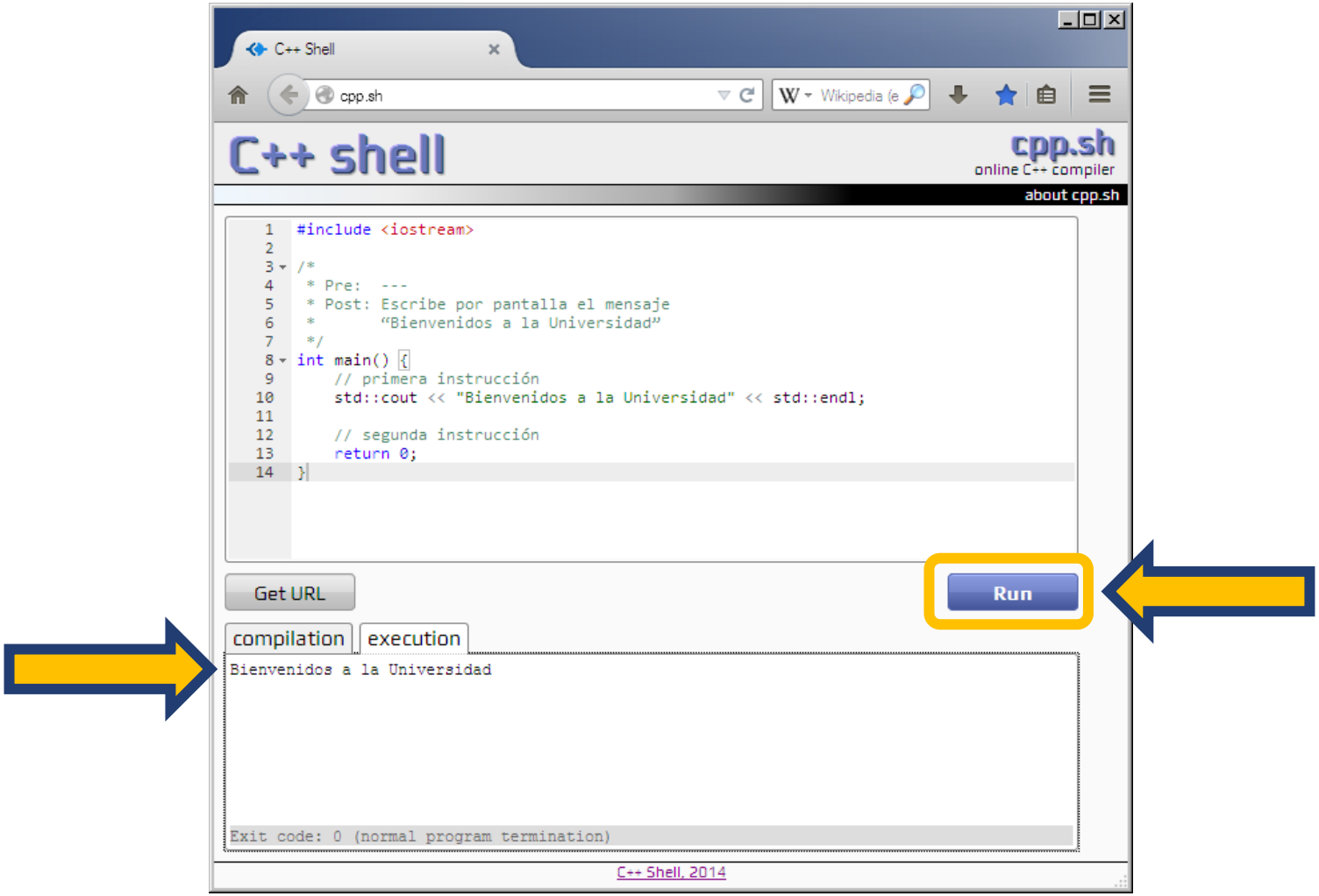
Compilación en Code::Blocks



Ejecución en Code::Blocks



Compilación y ejecución en cpp.sh



Un primer programa en C++

```
#include <iostream>

/*
 * Pre:  ---
 * Post: Escribe por pantalla el mensaje
 *       "Bienvenidos a la Universidad".
 */
int main() {
   // primera instrucción
   std::cout << "Bienvenidos a la Universidad" << std::endl;

   // segunda instrucción
   return 0;
}
```

Un primer programa en C++

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

/*
 * Pre:  ---
 * Post: Escribe por pantalla el mensaje
 *       "Bienvenidos a la Universidad".
 */
int main() {
   // primera instrucción
   cout << "Bienvenidos a la Universidad" << endl;

   // segunda instrucción
   return 0;
}
```

Un programa que realiza algunos cálculos

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

/*
 * Pre:  r >= 0.0
 * Post: Escribe por pantalla, en una misma línea, el valor del
 *       radio «r» y de la longitud de una circunferencia con
 *       ese radio.
 */
void circunferencia(double r) {
   const double PI = 3.14159265358979323846;
   cout << r << " " << 2.0 * PI * r << endl;
}
```

Un programa que realiza algunos cálculos

```
/*
 * Pre:  ---
 * Post: Escribe por pantalla el radio y la longitud de tres circunferencias
 */
int main() {
    // Escribe la cabecera de una tabla en la pantalla
    cout << "Radio      Circunferencia" << endl;
    cout << "=====      ==============" << endl;

    // Escribe por pantalla el radio y la longitud de tres circunferencias
    circunferencia(1.234);
    circunferencia(5.0112);
    circunferencia(11.5178);

    //Añade una línea en blanco adicional antes de finalizar
    cout << endl;

    // El programa termina normalmente devolviendo el valor 0
    return 0;
}
```

Ejecución del segundo programa

```
  Radio        Circunferencia
  =====        ==============
   1.23              7.75
   5.01             31.49
  11.52             72.37
```

Un programa interactivo

```
#include <iostream>

using namespace std;

/*
 * Pre:  r ≥ 0.0
 * Post: Escribe por pantalla en una línea el valor del radio y del área
 *       de un círculo de radio «r».
 */
void circulo(double r) {
   const double PI = 3.14159265358979323846;
   cout << "El área de un círculo de radio " << r << " es igual a "
        << PI * r * r << endl;
}
```

Un programa interactivo

```
/*
 * Pre:  ---
 * Post: Pregunta al operador por el «Radio del círculo:» y le informa
 *       en la línea siguiente del valor del radio y del área del
 *       círculo
 */
int main() {
   // Pregunta por el radio y lo almacena en la variable «radio»
   cout << "Radio del círculo: " << flush;
   double radio;
   cin >> radio;

   // Presenta por pantalla los datos del círculo de radio r
   circulo(radio);

   // Concluye normalmente y devuelve un 0
   return 0;
}
```

Ejecución

```
Radio del círculo:
```

Ejecución

```
Radio del círculo: 23.0754
```

Ejecución

```
Radio del círculo: 23.0754
El área de un círculo de radio 23.0754 es igual a 1672.82
```

Un programa que utiliza el resultado de una función

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

/*
 * Pre:  agnoNacimiento <= agnoActual
 * Post: Ha devuelto el valor entero que representa el número de años
 *       cumplidos en el año «agnoActual» por una persona nacida en el
 *       año «agnoNacimiento».
 */
int edad(int agnoNacimiento, int agnoActual) {
   return agnoActual - agnoNacimiento;
}
```

Un programa que utiliza el resultado de una función

```
const int AGNO_ACTUAL = 2017;

/*
 * Pre:  ---
 * Post: Ha preguntado al operador por el año de su nacimiento y
 *       le ha informado de la edad que ha cumplido o cumplirá
 *       este año.
 */
int main() {
   int agnoNacimiento;
   cout << "Escribe el año de tu nacimiento: " << flush;
   cin >> agnoNacimiento;

   cout << "En el " << AGNO_ACTUAL
        << " has cumplido o cumplirás "
        << edad(agnoNacimiento, AGNO_ACTUAL) << " años."
        << endl;
   return 0;
}
```

Especificación de funciones

- **Función**: **algoritmo** que resuelve un problema concreto de tratamiento de información
  - Datos de entrada
    - Parámetros
    - Datos leídos de teclado
  - Datos de salida
    - Valor devuelto
    - Datos escritos en la pantalla
- Especificación
  - Precondición
  - Postcondición

Especificación de funciones

```
/*
 * Pre:  P
 * Post: Q
 */
void f() {
   ...
}
```

Si se cumple la precondición *P* inmediatamente antes de invocar a la función f, entonces f se ejecuta, termina y se alcanza un estado en el que se cumple la postcondición *Q*.

Especificación de funciones

```
/*
 * Pre:  ---
 * Post: Ha devuelto el valor del polinomio
 *       ax² + bx + c
 */
double calcular(double a, double b, double c,
                double x) {
  return ((a*x+b)*x)+c;
}
```

Especificación de funciones

```
/*
 * Pre:  1 ≤ dia ≤ 31, 1 ≤ mes ≤ 12, anyo > 0
 * Post: Ha escrito en la pantalla una línea con
 *       la fecha definida por los valores de
 *       los parámetros «dia», «mes» y «anyo»
 *       con el siguiente formato: dia/mes/anyo.
 *       Por ejemplo: 12/1/2014
*/
void escribirFecha(int dia, int mes, int anyo) {
   cout << dia << "/" << mes << "/" << anyo << endl;
}
```

Especificación de funciones

```
/*
 * Pre:  ---
 * Post: Ha presentado por pantalla una línea
 *       con el texto “En esta asignatura se
 *       aprende a programar”
 */
void anunciar() {
   cout <<
      "En esta asignatura se aprende a programar"
      << endl;
}
```

Especificación de funciones

```
/*
 * Pre:  n >= 0
 * Post: Ha devuelto el valor de n!
 */
int factorial(int n) {
    ...
}
```

Propiedades de un algoritmo

- **Imprescindibles**
  - Corrección
  - Legibilidad
- **Deseables**
  - Generalidad
  - Reusabilidad
  - Eficiencia
  - Independencia de la máquina y del lenguaje
  - Simplicidad
  - Fiabilidad

Programa sintácticamente incorrecto

```
#include <iostream>

/*
 * Pre:  ---
 * Post: Escribe por pantalla el mensaje
 *       "Bienvenidos a la Universidad"
 */
{
   // primera instrucción
   cout << "Bienvenidos a la Universidad" << endl;

   // segunda instrucción
   return 0;
}
```

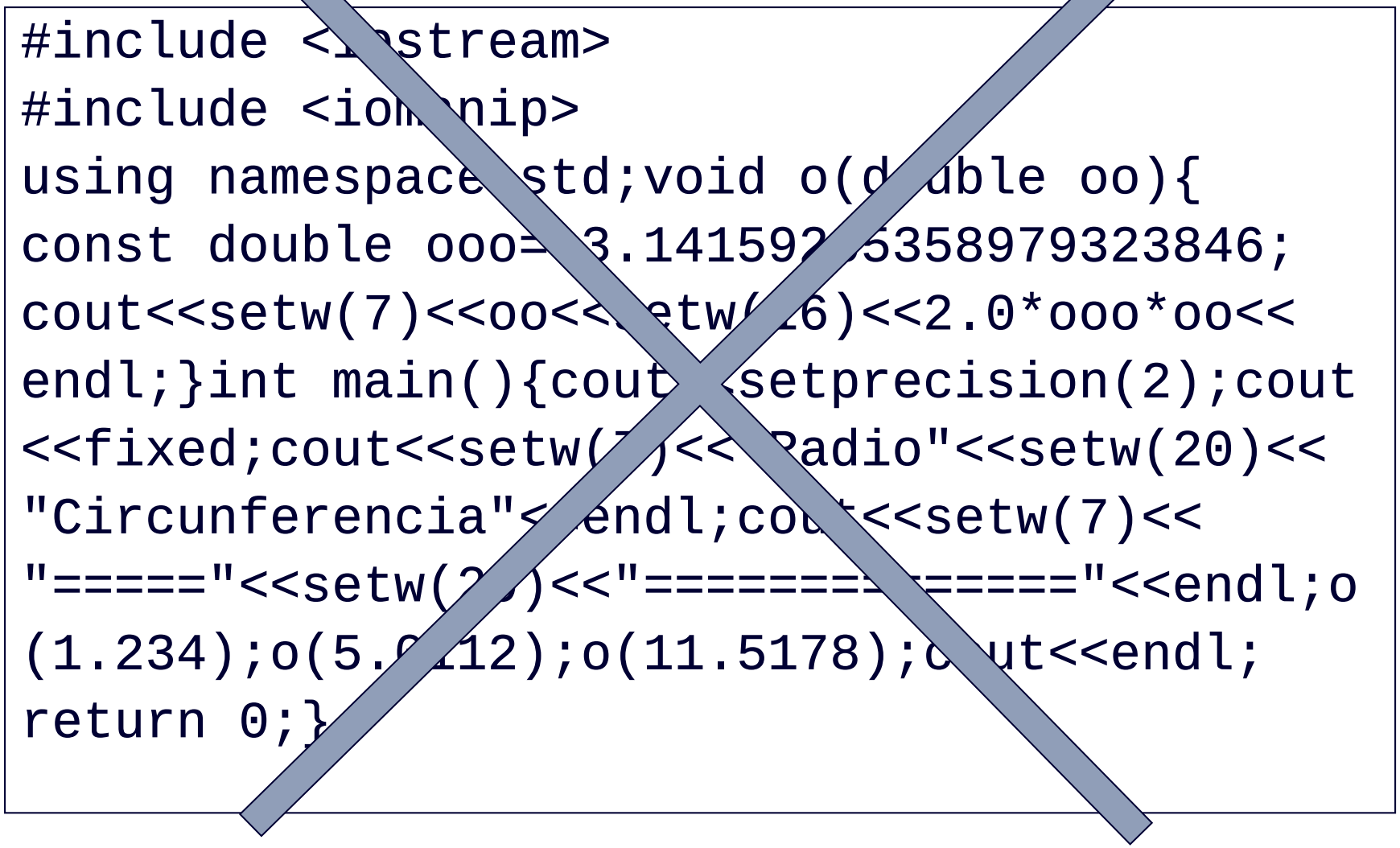
Programa formalmente incorrecto

```
#include <iostream>

using namespace std;

/*
 * Pre:  ---
 * Post: Ha escrito en pantalla la suma de los números
 *       del 1 al 5
 */
int main() {
   cout << 1 + 2 + 3 + 4 << endl;
   return 0;
}
```

Propiedades de un algoritmo

- **Imprescindibles**
  - Corrección
  - **Legibilidad**
- **Deseables**
  - Generalidad
  - Reusabilidad
  - Eficiencia
  - Independencia de la máquina y del lenguaje
  - Simplicidad
  - Fiabilidad

Programa C++ ilegible



```
#include <iostream>
#include <iomanip>
using namespace std;void o(double oo){
const double ooo=3.14159265358979323846;
cout<<setw(7)<<oo<<setw(16)<<2.0*ooo*oo<<
endl;}int main(){cout<<setprecision(2);cout
<<fixed;cout<<setw(7)<<"Radio"<<setw(20)<<
"Circunferencia"<<endl;cout<<setw(7)<<
"======"<<setw(20)<<"==================="<<endl;o
(1.234);o(5.0112);o(11.5178);cout<<endl;
return 0;}
```

Propiedades de un algoritmo

- **Imprescindibles**
  - Corrección
  - Legibilidad
- **Deseables**
  - **Generalidad**
  - **Reusabilidad**
  - Eficiencia
  - Independencia de la máquina y del lenguaje
  - Simplicidad
  - Fiabilidad

Generalidad

```
/*
 * Pre:  ---
 * Post: Devuelve la suma de los enteros
 *       comprendidos en el intervalo [1, 100]
 */
int sumaDe1A100() {
  int resultado = 0;
  for (int i = 1; i <= 100; i++) {
    resultado = resultado + i;
  }
  return resultado;
}
```

Generalidad

```
/*
 * Pre:  inicial <= final
 * Post: Ha devuelto la suma de los enteros
 *       comprendidos en el intervalo [inicial, final].
 */
int suma(int inicial, int final) {
  int resultado = 0;
  for (int i = inicial; i <= final; i++) {
    resultado = resultado + i;
  }
  return resultado;
}
```

Propiedades de un algoritmo

- **Imprescindibles**
  - Corrección
  - Legibilidad
- **Deseables**
  - Generalidad
  - Reusabilidad
  - **Eficiencia**
  - Independencia de la máquina y del lenguaje
  - Simplicidad
  - Fiabilidad

Eficiencia

```
/*
 * Pre:  inicial <= final
 * Post: Ha devuelto la suma de los enteros
 *       comprendidos en el intervalo
 *       [inicial, final].
 */
int sumaEficiente(int inicial, int final) {
  return (inicial + final)
     * (final – inicial + 1) / 2;
}
```

Propiedades de un algoritmo

- **Imprescindibles**
  - Corrección
  - Legibilidad
- **Deseables**
  - Generalidad
  - Reusabilidad
  - Eficiencia
  - **Independencia** de la máquina y del lenguaje
  - **Simplicidad**
  - **Fiabilidad**

Resumen

- Problemas de tratamiento de información
- Algoritmos y programas
- Ejemplos de programas C++
- Funciones y especificación
- Propiedades de un algoritmo

¿Cómo se estudia este tema?

- Repasando los apuntes
- Ejecutando los programas presentados en un entorno de ejecución en línea, como http://cpp.sh/
- Ejecutando los programas en el entorno Code::Blocks:
  - Instalándolo según las instrucciones de «Instalación de Code::Blocks» en la web de la asignatura
  - Leyendo la descripción del entorno Code::Blocks en la sección 1.3 de la 1.ª práctica de la asignatura.