TEMA 7
ALGORITMOS Y PROGRAMAS. ESTRUCTURA DE UN PROGRAMA

7.1. Algoritmos

7.2. Compilación

7.3. Estructura de un programa

7.1. Algoritmos

Objetivo: Resolver problemas mediante computadoras (usando el lenguaje C)

PROBLEMA → ALGORITMO → PROGRAMA

Los algoritmos son más importantes que los lenguajes de programación (lenguaje que permite transmitirlo de forma efectiva a un computador) o que los computadores (herramienta que lo lleva a cabo)

Características de los algoritmos:

- Son precisos e indican de manera estricta su orden de ejecución
- Sus resultados solo dependen de los datos de entrada, i. e., están definidos
- Terminarán en un número finito de pasos, es decir, son finitos

Partes de un Algoritmo

- Entrada: Quedará completamente definida
- Proceso: Se describirá completamente (dependiendo de su nivel de abstracción)
- Salida: Quedará completamente definida

Diseñar algoritmos para modelar:

Un cliente realiza un pedido a una fábrica. La fábrica examina la información que tiene sobre el cliente y solo cuando este es solvente, se acepta y cursa el pedido

Mostrar por pantalla el resultado de sumar todos los numeros enteros del 1 al 1000

Identificar si un número es primo

Sumar todos los primos menores que 1000

7.2. Compilación

Interprete $\neq$ Compilador

Fases en la ejecución de un programa en C:

1. Compilación:

   Fichero fuente .C $\rightarrow$ Fichero objeto .OBJ

2. Linkado:

   Fichero objeto .OBJ $\rightarrow$ Fichero ejecutable .EXE

Fases en la ejecución de un programa en C

1. Escritura del programa fuente con un editor, el que tiene *Turbo C 3.0* o el de *Borland C++ 5.0*, y almacenado del mismo en disco.

2. Introducir el programa fuente en memoria

3. Compilar el programa

4. Verificar y corregir errores de compilación

5. Obtención del programa objeto

6. Obtención del programa ejecutable mediante el montador o enlazador

7. Ejecución del mismo

7.3. Estructura de un programa

Todo programa en C consta de una o mas funciones, una de las cuales se llama *main*.

El programa siempre comenzará por la ejecución de la función *main*.

Las definiciones de las funciones adicionales pueden preceder o seguir a *main*.

Cada definición de función debe contener:

- Una cabecera de la función (nombre + argumentos/parámetros)

- Una lista de declaración de argumentos de la cabecera

- Una sentencia compuesta que contiene el resto de la función

Estructura de un programa II

Las sentencias compuestas se encierran entre llaves.

Estas llaves pueden contener otras sentencias compuestas o combinaciones de sentencias elementales (llamadas sentencias de expresión).

Cada sentencia de expresión termina con ;

Los comentarios pueden aparecer en cualquier parte del código, han de estar encerrados por unas marcas especiales:

// *comentario*

/* *comentario* */

Cuando utilicemos alguna función propia de alguna librería, será necesario que añadamos delante de *main* la declaración de inclusión de la librería.

Estructura de un programa III

```
#include <stdio.h>

int primo (int n)

{ int i = 2;

  if (n < 4) return 1;

  while (n%i != 0) i++;

  if (n == i) return 1;

     else  return 0; }

void main (void)

{ int x;

  int primo (int);

  printf("Introduce un entero: ");

  scanf("%d", &x);

  if (primo(x)) printf("El numero %d es PRIMO", x);

    else  printf("El numero %d es COMPUESTO", x); }
```

Funciones I

Es necesario dividir los programas grandes en subprogramas o funciones más pequeños que serán llamados por el principal.

Ventajas:

- Modularización
- Ahorro de memoria y tiempo de desarrollo
- Independencia de datos y ocultación de la información

Una función de C es una porción de código o programa que realiza una determinada tarea

Cada función estará asociada a un identificador o nombre

Funciones II

En toda función distinguiremos:

- Definición (cabecera, argumentos, sentencia)
- Declaración (especificación de tipos)
- Llamada (con su nombre y los argumentos instanciados)

Tras ser llamada devuelve un *valor de retorno* (especificado en la declaración)

En su definición no será necesario declarar los argumentos, ya referidos en la cabecera

Una línea de su código comenzará por *return* y a continuación la expresión cuya evaluación proporciona el *valor de retorno*