LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

UNIVERSIDAD NACIONAL DE TUCUMÁN
facet
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

PROGRAMACIÓN

Unidad 2: Lenguajes de programación. Programación estructurada. El lenguaje C. Elementos básicos de un programa. Análisis de programas. Estilo de programación

Lic. Mariela A. Velázquez

Lenguaje C

Un poco de historia…

- Los laboratorios Bell lo desarrollaron a principios de la década del 70.
- Los autores del lenguaje son: Brian Kernighan (Canadá) , y Dennis Ritchie(Estados Unidos).
- El objetivo de su creación fue para que los programadores de Bell pudiesen redactar su sistema operativo UNIX para una nueva computadora producida por DEC (Digital Equipment Corporation).
- Los laboratorios Bell diseñaron UNIX teniendo en cuenta que pudiese correr de manera eficiente en equipos pequeños; asimismo UNIX fue el primer sistema operativo totalmente escrito en un lenguaje de programación de alto nivel. Hasta entonces, el código fuente de los sistemas operativos se redactaban empleando lenguaje esamblador.
- Debido a que los otros lenguajes de alto nivel existentes en aquel tiempo (COBOL, FORTRAN, etc), eran demasiados lentos para ser utilizados en la codificación de un sistema operativo. Los programadores de laboratorios Bell decidieron desarrolla su propio lenguaje, basado en Algol y BCPL, dos eficientes lenguajes de alto nivel.
- Luego de algunas versiones (como era de esperar la primera fue denominada A).

Características de C

- Es un lenguaje para la programación estructurada.
- Es tipificado, aunque no tanto como poder ser Pascal. Un programa en C es una colección de funciones, que pueden devolver un valor o no (procedimiento), y que se encuentran distribuidas en varios ficheros o módulos.
- Contiene muy pocas palabras reservadas.
- No contiene órdenes para trabajar con objetos compuestos (cadenas, registros, etc).
- Distingue entre mayúsculas y minúsculas.

Ventajas

- Es el lenguaje más portado en existencia, habiendo compiladores para casi todos los sistemas conocidos.
- Proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes.
- Es un lenguaje muy flexible, muy veloz y potente, lo que permite un software efectivo.
- Posibilita una programación estructurada o modular.
- Acceso a memoria de bajo nivel mediante el uso de punteros.

Desventajas

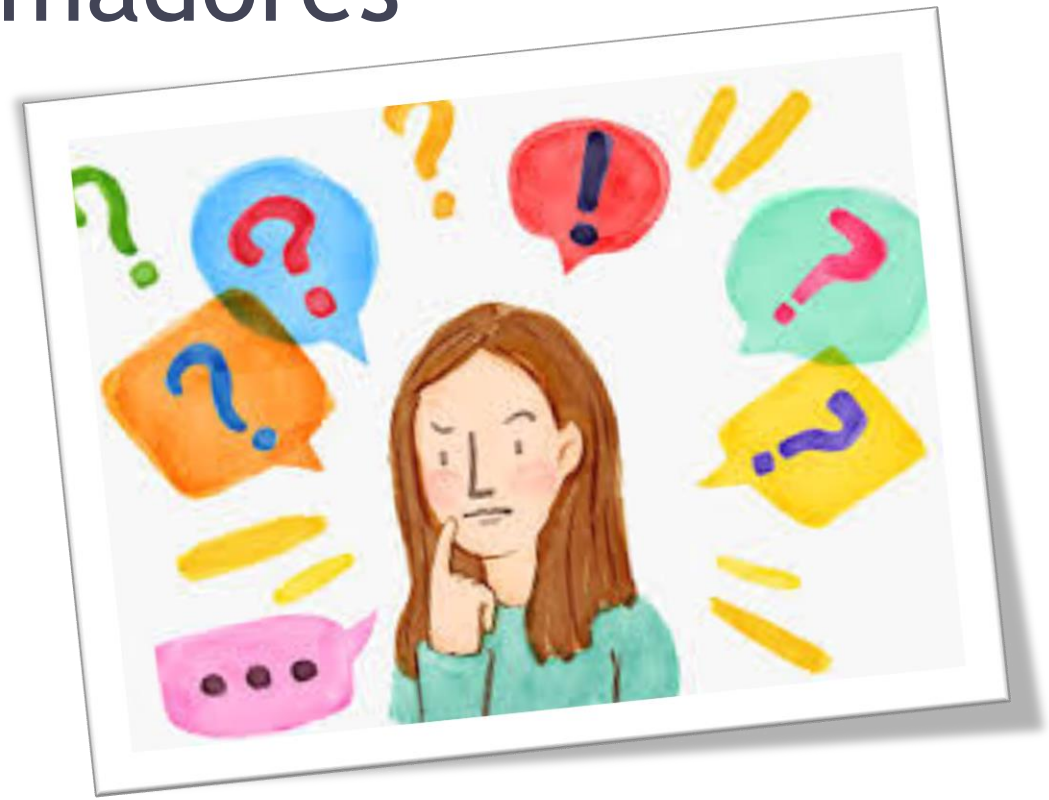
- No tiene instrucciones propias para la asignación dinámica de memoria ni instrucciones de entrada/salida. Todas estas operaciones de alto nivel pueden ser realizadas por funciones explícitamente.

- Se requiere más tiempo en conseguir el ejecutable, porque cada vez compila todo el fichero.

- No dispone de sistemas de control automáticos y la seguridad depende casi exclusivamente de la experiencia del programador.


ENTONCES SATANAS DIJO,
USEMOS EL LENGUAJE C
Imagen creada en GeneradorMemes.com

Ejemplo: usaremos como ejemplo al programa mas famoso entre los programadores

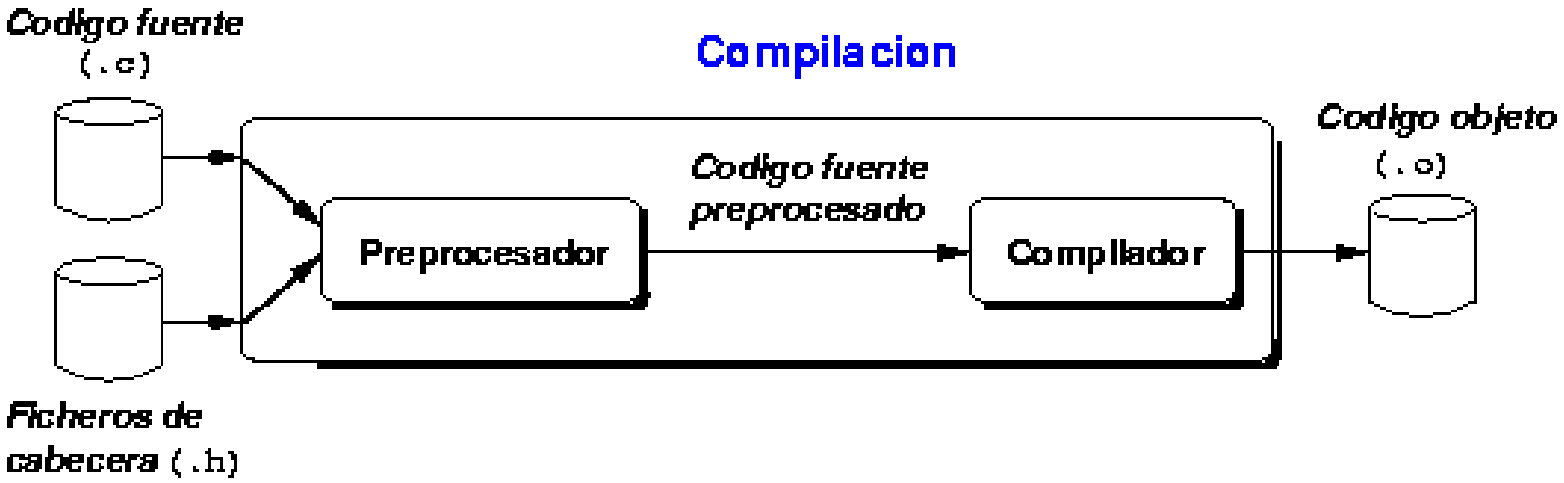


En un programa intervienen

- Ordenes para el preprocesador
- Variables
- Constantes
- Aritmética
- Funciones
- Funciones de entrada y salida
- Comentarios

Ordenes para el preprocesador

- Conceptualmente es un paso previo a la compilación.
- Las mas usadas son: # include
  # define
- Para inclusión de archivos y substitución de macros.



Funciones

“ Un programa escrito en código C es una reunión de funciones “.

main → función principal debe estar , presente en todos los programas escritos en C

main → puede convocar a otra funciones .

Función main

- Un método para comunicar datos entre las funciones, es que la función que llama proporciona una lista de valores, llamados argumentos.

- Los paréntesis después del nombre de la función, están para encerrar una lista de valores que serán argumentos.

Función main

- Puede ocurrir que una función este definida para ser una función que no espera argumentos, tal es el caso del main, lo cual se indica con una lista de argumentos vacía.

main()

Función main

- Las sentencias de la función main están encerradas entre llaves:
  - Llamado a una función
  - Sentencia return

Funciones de Entrada y Salida

La biblioteca estándar stdio.h provee al programador de una extensa gama de funciones para lectura y escritura.

Es necesario escribir una orden para el preprocesador para usar dichas funciones.
#include <stdio.h>

De las variables y constantes

- Objetos sobre los que actúan las instrucciones que componen el programa.
- Deben declarase como tales.
- Deben tener un identificador asociado, formado por una secuencia de letras y números. El 1°carácter debe ser una letra. Letras mayúsculas y minúsculas son diferentes.

De las variables y constantes

- Debe indicarse que tipo de datos pueden contener (rango y operaciones permitidas).
- Las variables pueden inicializarse en forma grupal.
- La declaración de la/s variables reúne todas estas exigencias.
- Las constantes (constantes simbólicas) permiten hacer modificaciones sistemáticas.

De las variables y constantes

- Las constantes se definen en una línea #define , que es una directiva del preprocesador.
- *#define nombretexto de reemplazo*
- Las constantes pueden ser enteras, reales y de carácter.
- Deben definirse previo a su uso.

Aritmética

- Interacción entre los operadores aritméticos y las variables y/o constantes declaradas (si el tipo es numérico).

- El tipo de operación permitido está ligado tipo de dato con que fue declarada la variable y/o constante.

- Tipos de datos básicos: int, float, char, short, long, double.

Comentarios

Dos modos de comentar las acciones del código escrito:

// comentarios de una sola línea

/*comentarios de varias líneas

..... */

Luego de este análisis podemos…

Holamundo.c

```c
#include<stdio.h>

int main(){

    printf("Hola Mundo ");

    return 0;
}
```

Compilador

- El compilador para C del proyecto GNU se llama “gcc” y su nombre proviene de “GNU C Compiler”[1]
- Un compilador es un programa informático que traduce un programa de un lenguaje a otro, generando un lenguaje equivalente que la maquina será capaz de interpretar.
- Este proceso de traducción se conoce como compilación.

[1] Hoy en día “gcc” también quiere decir “GNU Collection Compiler” porque también representa una colección de compiladores

Código Fuente

- El compilador gcc es capaz de compilar cualquier programa en el lenguaje C escrito en un archivo de texto convencional.

- El código fuente escrito en un archivo de texto lleva el sufijo ".c" para identificar que su contenido corresponde al código de un programa escrito en el lenguaje C.

- También existen herramientas más especializadas para escribir y editar código fuente de manera más eficiente que un simple editor de texto.

Compilando un programa en C

```
$ gcc –Wall primerPrograma.c –o primerPrograma
```

El comando anterior compila el código fuente a código maquina y lo almacena el archivo ejecutable.

-Wall : opción para activar las advertencias del compilador . El compilador puede generar advertencias sobre los elementos del código fuente que pueden llegar a ser errores en la codificación.

-o: permite especificar el archivo de salida. Usualmente es la única opción en la línea de comando, si se omite esta opción, el archivo de salida por defecto es a.exe.

Compilando un programa en C

```
$ gcc –Wall primerPrograma.c –o primerPrograma
```

primerPrograma.c: nombre del archivo fuente.

primerprograma: nombre del archivo de salida.


VAMOS PROGRAMADOR, DI LO TUYO
NO COMPILA

Etapas de compilación

Cuando invocamos el comando “gcc”, normalmente se realiza **preprocesamiento**, **compilación**, **ensamblado** y **enlazado**.

Cada etapa tiene un código fuente como entrada y un código objeto como resultado, y en las etapas intermedias el código objeto sirve de fuente para la etapa siguiente.

Etapas de compilación: Preprocesamiento

El preprocesamiento es realizado por el preprocesador. Esta primera etapa traduce el archivo fuente que es una forma ampliada del lenguaje.

Ejemplo: Calcular el área de un circulo.

Preprocesamiento

```
$ gcc –E circulo.c –o circulo.i
```

Que puede observar en el archivo circulo.i?,

Encontró algún cambio respecto al código fuente?

Preprocesamiento

```
$ gcc –E circulo.c –o circulo.i
```

-E: opción para detener el proceso de compilación luego de realizado el preprocesamiento. La salida es en la forma de un archivo preprocesado.

Compilación

La compilación transforma el código C preprocesado en el lenguaje ensamblador propio del procesador de nuestra maquina.

```
$ gcc –S circulo.c
```

Con este comando se realizan las dos primeras etapas y se crea un archivo circulo.s

-S: opción para detener luego de realizada la etapa de compilación.

Ensamblado

En la etapa de ensamblado se traduce el programa escrito en el lenguaje ensamblar, de la etapa anterior, a código binario en lenguaje de maquina entendible por el procesador.

```
$ gcc –c circulo.c
```

Con este comando se realizan las tres primeras etapas y se crea un archivo circulo.o

-c: opción para detener luego de realizada la etapa de ensamblado.

Enlazado

Las funciones como printf, se encuentran compiladas y ensambladas en librerías existentes. Para que el archivo resultado del proceso de compilación completo sea ejecutable es necesario incorporar el código binario de estas funciones en el lenguaje final. De esto se trata la etapa de enlace o linking.

```
$ gcc  circulo.c –o circulo
```

Corrección y pruebas

Retomemos el ejemplo del área del circulo.

```
#include <stdio.h>
#define PI 3,1416

int main()
{
    float area, radio=10;
    area = PI* (radio * radio);
    printf("Area circulo = %d", area);
    return 0;

}
```

Corrección y pruebas

El formato del mensaje producido por el compilador gcc tiene la forma: “archivo:linea:mensaje”

Repasamos!

Características de C

- Es un lenguaje para la programación estructurada.
- Contiene muy pocas palabras reservadas.
  Distingue entre mayúsculas y minúsculas.

Ventajas y Desventajas

- Posibilita una programación estructurada o modular.
  Flexible, veloz y potente, lo que permite un software efectivo
- Cada vez compila todo el fichero.
  No tiene instrucciones propias de entrada/salida.

Repasamos!


Ordenes para el preprocesador
Partes de un programa en C
Funciones. Funciones de entrada/salida
Variables y constantes
Comentarios

Repasamos!


Preprocesamiento:
$ gcc –E circulo.c –o circulo.i
Compilación
$ gcc –S circulo.c
Etapas de Compilación
Ensamblado
$ gcc –c circulo.c
Enlazado
$ gcc circulo.c –o circulo

Entrada y salida básicas

- C no contiene instrucciones de entrada ni de salida.
- El lenguaje C interpreta que la entrada proviene de stdin(dispositivo estándar de entrada).
- La salida es dirigida a la stdout(dispositivo estándar de salida).
- Ambas pueden redirigirse a otros dispositivos.

Función de Salida: printf

- Permite la presentación de valores numéricos, caracteres y cadenas de texto por el archivo estándar de salida (pantalla) .El prototipo de la función printf es el siguiente:

printf(control,arg1,arg2…);

- **Control:** la cadena de control en la cual se indica la forma en que se mostrarán los argumentos posteriores (si los hubiere) .También se puede escribir una cadena de texto (sin necesidad de argumentos), o combinar ambas posibilidades, así como secuencias de escape.

Función de Salida: printf

printf(control, arg_1, arg_2...);

arg_i: argumentos cuyos valores habrán de ser mostrados en la línea de salida. Si se utilizan argumentos se debe indicar en la cadena de control tantos caracteres de conversión (o modificadores) como argumentos se van a presentar.

El formato completo de los modificadores: % [signo] [longitud] [.precisión] [l/L] conversión

Entre el % y el carácter de conversión puede haber:

- **Signo:**

  - : indica si el valor se ajustará a la izquierda.

  +: establece que el número siempre será impreso con signo.

- **longitud:** especifica un ancho mínimo de campo. El argumento será impreso en por lo menos esta longitud. Si tiene menos caracteres que el ancho del campo, será rellenado a la izquierda. El carácter de relleno normalmente es espacio.

- **precisión:** indica el número máximo de decimales que tendrá el valor.

- **l/L:** utilizamos l cuando se trata de una variable de tipo long y L cuando es de tipo double.

Listado de los modificadores o caracteres de conversión más utilizados

- %c Un único carácter
- %d Un entero con signo, en base decimal
- %u Un entero sin signo, en base decimal
- %o Un entero en base octal
- %x Un entero en base hexadecimal
- %e Un número real en coma flotante, con exponente
- %f Un número real en coma flotante, sin exponente
- %s Una cadena de caracteres
- %p Un puntero o dirección de memoria

Ejemplos, un poco de reflexión y de ejercitación

```
...
int longi = 25 ;
printf ( "\n longi = %-3d \n", longi); // ¿cómo es la salida que produce esta línea?
...
```

```
  ...
float prom = 1258.32;
printf  ( "longi=%-5d\t\nprom = %5.2f\n", longi, prom); // ¿y  esta otra?
  ...
```

Secuencias de Escape

Nombre	Secuencia de Escape
Nulo	\o
Retroceso	\b
Tabulador	\t
Salto de línea	\n
Salto de página	\f
Retorno de carro	\r
Comillas	\"
Signo de interrogación	\?
Barra invertida	\\

Las secuencias de escape permiten expresar ciertos caracteres no imprimibles, así como la barra inclinada hacia atrás (\) y la comilla simple (´) a través de la combinación adecuada de algunos caracteres alfabéticos. Una secuencia de escape siempre comienza con una barra inclinada hacia atrás (barra invertida) y es seguida por uno o más caracteres especiales. Por ejemplo, un salto de línea (LF), que representa el carácter de nueva línea, se representa con un \n. Las secuencias de escape son interpretadas por el compilador como un solo carácter.

Función de Entrada: scanf

Función scanf(): Permite ingresar la información o datos en posiciones de memoria de la computadora a través del archivo estándar de entrada (teclado).El prototipo de la función scanf es el siguiente:

scanf(control,arg1, arg2…);

Control:cadenadecontrolqueindica,porreglageneral,losmodificadoresqueharánreferenciaaltipodedatodelosargumentos.

Función de Entrada: scanf

scanf(control,arg_1, arg_2...);

arg_i: son los nombres o identificador es de los argumentos de entrada cuyos valores se incorporarán por teclado.

Scanf "necesita conocer" la posición de la memoria en que se encuentra la variable para poder almacenar la información ingresada. Por eso se utiliza un operador unario de dirección: **&(ampersand)**, que se coloca delante del nombre de cada variable.

scanf(“%d %d %d”, &var1, &var2, &var3);

Si Ud. no indica a **scanf()** cual es la dirección interna (la posición de memoria) de estas variables, el programa en cuestión no podría acceder luego a los valores tipiados desde el teclado.

Sentencia While

La sentencia while ejecuta una sentencia, simple o compuesta, cero o más veces, dependiendo de una condición.

Su sintaxis es:

while (condición) sentencia ;

Donde:

condición es cualquier expresión numérica, relacional o lógica.

sentencia es una sentencia simple o compuesta.

Sentencia While

while (condición) sentencia ;

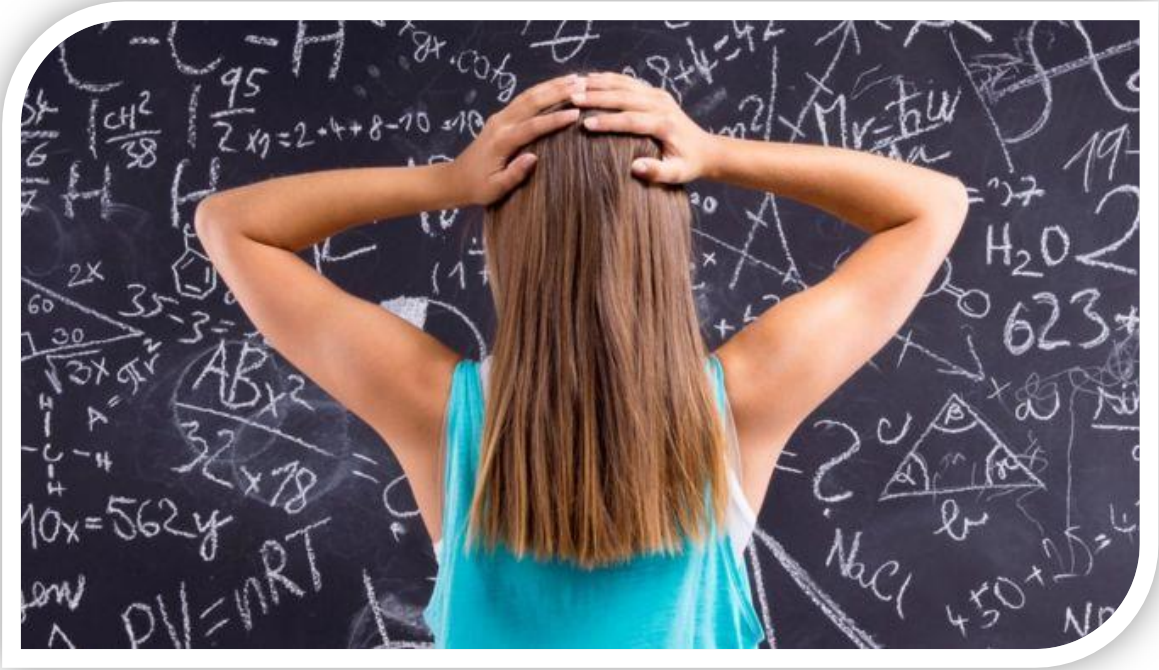
La ejecución de la sentencia while sucede así:

Se evalúa la condición . Si el resultado de la evaluación es 0 (falso), la sentencia no se ejecuta y se pasa el control a la siguiente sentencia en el programa. Si el resultado de la evaluación es distinto de 0 (verdadero), se ejecuta la sentencia y el proceso descrito se repite desde el Inicio.

Ejercicio

Diseñar un algoritmo que muestre la tabla de multiplicar de un número determinado.

Además se solicita que muestre por pantalla hasta que número se desea ver la tabla. Implemente en C.



Sentencia do-while

En el lazo do-while tiene la comprobación relacional al final, en vez de tenerla al inicio como es en la estructura while. La sintaxis es:

```
do {
        sentencia;
}while ( condición );
```

La sentencia se ejecuta y después se evalúa la condición. Si es verdadera la proposición se evalúa de nuevo. Cuando la expresión se hace falsa el ciclo termina.

Sentencia do-while

- Retomar el ejemplo anteriormente realizado usando la estructura do-while.

- ¿Qué diferencias nota entre while y do-while?.

Sentencia For

A ver Marta, pase y escriba 500 veces no debo tirar papeles en clases...

```
#include <stdio.h>

int main()
{
     int cont;

      for(cont=1;  cont<=500;  cont++)
     {
          printf("no debo tirar papeles en clases");
      }
      return 0;
}
```

desmotivaciones.es

Cosas de programadores.

Sentencia for

La sentencia for permite ejecutar una sentencia simple o compuesta, repetidamente un número de veces conocido.

Su sintaxis es la siguiente:

for (expresión-comienzo; condición; progresión-condición)

Sentencia;

Expresión-comienzo: representan variables de control que serán iniciadas con los valores de las expresiones.

Condición: es una expresión booleana que si se omite, se supone verdadera.

Progresión-condición: es una o más expresiones separadas por comas cuyos valores evolucionan en el sentido de que se cumpla la condición para finalizar la ejecución de la sentencia for.

Sentencia: es una sentencia simple o compuesta.

Bibliografía

- Articulo “Una Introducción al Compilador C de GNU”, Mg. Hector A. Valdecantos.
- Greg Perry. C con ejemplos (1ra ed.).
- Brian W. Kernighan, Dennis M. Ritchie. El lenguaje de programación C (2da ed.).
- Ceballos, Sierra, Francisco Javier. *C/C++ curso de programación (3a. ed.)*, RA-MA Editorial, 2007.

```c
#include <stdio.h>

int main()
{
    printf("Hasta la próxima clase!!\n");

    return 0;
}
```