UNIVERSIDAD DE VALLADOLID **Departamento de Informática**
Informática Licenciatura en Matemáticas
Parcial Febrero de 2009

Apellidos : Nombre :

AVISOS:

- Es necesario entregar todas las hojas del enunciado del examen al finalizar, se hayan respondido o no a las preguntas.
- Es obligatorio firmar todas las hojas de respuesta del examen.
- En breve estará en la página web de la asignatura una copia del enunciado del examen y la solución del mismo.
- Debes usar el espacio debajo de cada cuestión para contestarla **de forma razonada**.
- **La duración de esta parte del examen es de 1 hora y 30 minutos. La siguiente parte del examen comenzará a las 11:45 h.**.

Cuestiones

1. **(0,5 puntos)** Explica qué es Internet. ¿Es lo mismo que la World Wide Web? Explica la diferencia si no es así.

   Internet es una red de redes, virtual, que conecta miles de redes ordinarias, ofreciendo servicios a todos los usuarios del mundo como si estuviesen conectados a una red física.
   La WWW es sólo uno de los servicios que se ofrecen sobre Internet (como un gran repositorio multimedia al cual es posible acceder gracias a protocolos como http, definiciones de documentos como HTML o conceptos como URL); por lo tanto, no son lo mismo.

2. **(0,5 puntos)** Enumera cuáles han sido las diferentes generaciones de lenguajes de programación y cuál era la principal característica de cada una de ellas.

   Primera generación, propia de los primeros ordenadores, donde los programas se escribían directamente en código máquina, en binario, y los ordenadores admitían un lenguaje propio, y por lo tanto los programas no eran intercambiables.
   Segunda generación, antes de 1955, se utilizan mnemónicos para las instrucciones; aparece el lenguaje Ensamblador. Los lenguajes aún son dependientes de las máquinas.
   Tercera generación, desde 1955, aparecen los lenguajes de alto nivel. Son lenguajes de propósito general (sirven para programar muchos tipos de problemas), son portátiles entre distintas máquinas. Ejemplos:Fortran, Pascal o C.
   Cuarta generación, paquetes de aplicaciones, que incorporan entornos de programación avanzados y son específicos para resolver ciertos tipos de problemas: bases de datos, simulación, programación científica, gráficos, estadística, etc. Ejemplo: matlab.

3. **(0,5 puntos)** Explica el concepto de FBFs equivalentes en lógica y qué relación tienen con las manipulaciones de expresiones lógicas que vemos en programación.

   Dos fórmulas bien formadas son equivalentes cuando proporcionan el mismo valor de verdad para cualquier interpretación. Por lo tanto, son intercambiables. Eso nos permite aplicar reglas de equivalencia y manipular fórmulas (como las expresiones lógicas de los lenguajes de programación) de forma sintáctica, esto es, sin cambiar su significado.
   Por ejemplo es lo mismo preguntar: (a = cierto) que (a), que (not a = falso).

4. **(0,5 puntos)** ¿Qué es la notación en punto o coma flotante? Explica brevemente qué es, para qué sirve y cuáles son las características principales de la notación IEEE 754 para reales de simple precisión?

   Una forma de representar los números reales mediante Mantisa, Base y Exponenete, que nos permite manipular la mantisa, moviendo la coma decimal -punto decimal- que supone multiplicar o dividir por la base, si se cambia el valor del exponente en una unidad -sumando o restando.
   La norma IEEE-754 es un estándar para los números reales dentro de un ordenador.
   Utiliza el esquema S-E-M para n bits. En el caso de simple precisión la distribución de bits es 1 para el signo, 8 para el exponente (en notación sesgada) y 23 para la mantisa (que se almacena normalizada y empaquetada).

5. **(0,5 puntos)** ¿En qué consiste el proceso de traducción de un programa? ¿Qué diferencia hay entre un lenguaje de programación compilado y otro interpretado?

   En la traducción del programa fuente, por ejemplo C, al código máquina entendible por el ordenador (programa ejecutable). Consta de un proceso de traducción propiamente dicho (análisis léxico, análisis sintáctico y generación de código objeto), que produce el programa objeto. Después hay una fase de enlazado (que introduce código objeto adicional, propio o del lenguaje de programación), y la fase de carga (listo para la memoria principal).
   Los lenguajes compilados realizan la fase de traducción, enlazado y carga sobre todo el programa fuente antes de ejecutarlo. Los lenguajes interpretados realizan el proceso línea a línea.

Cuestiones teórico-prácticas de programación

1. **(0,5 puntos)** ¿Cuáles son las características que definen un objeto? Enumera los tipos básicos que existen en código algorítmico y cuáles son sus equivalentes en C.

   Un objeto viene definido por su nombre, que lo identifica de forma única, por su tipo, que indica qué valores puede tomar y qué operaciones se pueden hacer, y su valor.
   Los tipos básicos en lenguaje algorítmico son: numéricos, carácter, lógicos y cadenas.
   Los tipos básicos en C son int, float y double (para numéricos), char para carácter, vector de char para cadenas. No hay lógicos.

2. **(0,5 puntos)** ¿Qué dos preguntas nos tenemos que hacer para decidir qué esquema repetitivo tenemos que usar? ¿Qué esquemas se eligen en función de los resultados?

   Primero: cuántas veces se repetirá la iteración. Si lo conozco a priori, debo elegir el esquema para. Si no lo sé, hay que preguntarse si se entrará en el bucle cero o más veces, entonces se elegirá un esquema mientras, o una o más veces, entonces se elegiría repetir.

3. **(0,5 puntos)** Explica brevemente en qué consiste un programa propio y da dos ejemplos de prácticas de programación que vayan en contra de los principios de la programación estructurada.

Un programa propio es un lenguaje correcto que respeta la programación estructurada (tiene un único punto de entrada y de salida), es correcto para todos los casos, no contiene sentencias inútiles ni bucles infinitos.
Los programas propios se pueden construir sin instrucciones de salto, usando sólo la composición secuencial, los condicionales y los esquemas repetitivos.
Ejemplos de malas prácticas: utilización de saltos, presencia de bucles infinitos, sentencias que no modiquen el entorno, o tener varios puntos de salida en un esquema repetitivo.
Otras malas prácticas: modificar parámetros dato, crear bucles infinitos o modificar el valor de una variable índice en varios puntos de un esquema para.

4. **(1,5 puntos)**

   a) Propón un algoritmo que nos muestre por pantalla el nombre de un tipo de polígono a partir de su número de lados. Por ejemplo, si lados es 3 debe mostrar Triángulo, si es 4 Cuadrado, y así sucesivamente.
   El algoritmo debe respetar la siguiente cabecera:

```
PBR. Analizo cada caso del valor lados y en función de su valor pongo el nombre adecuado.
Son valores discretos y enteros, entonces puedo usar un esquema condicional generalizado.
Además, compruebo que los valores estén en los límites permitidos.
Algoritmo nombre_poligono (lados, ok) es
lados: numérico; {parámetro dato; entero, valores válidos entre 3 y 8}
ok: lógico; {parámetro resultado; cierto si el número de lados era correcto;
falso en caso contrario}
INICIO
si lados < 3 or lados > 8 entonces
  ok := falso;
sino
  ok := cierto;
  según lados hacer
    3: Escribir "Triángulo";
 4: Escribir "Cuadrado";
 5: Escribir "Pentágono";
 6: Escribir "Hexágono";
 7: Escribir "Heptágono";
 8: Escribir "Octógono";
  fin según
fin si
FIN
```

b) Traduce el algoritmo anterior a C. Explica qué tipo de función vas a utilizar y cómo se transforman los parámetros del código algorítmico al C.

Hay un parámetro resultado, que es el valor lógico, asociado al parámetro ok. Como en C no hay valores lógicos, se utiliza el tipo int, dando 0 para falso y 1 para cierto.

```
int nombre_poligono (int lados)
{
 int ok;
 if (lados < 3 || lados > 8)
  ok = 0;
 else {
  ok = 1;
  switch (lados) {
  case 3: printf("\nTriangulo"); break;
  case 4: printf("\nCuadrado"); break;
  case 5: printf("\nPentágono"); break;
  case 6: printf("\nHexágono"); break;
  case 7: printf("\nHeptágono"); break;
  case 8: printf("\nOctógono"); break;
  } /* fin switch */
 };  /* fin if - else*/
 return ok;
}
```