Examen convocatoria de Febrero de 2001

Nombre:_____

1. Señala la opción que sea verdadera de los enunciados siguientes

La diferencia entre un bit y un byte es:

Histórica. Primero se le llamó bit y tras el desarrollo de los primeros computadores electrónicos se le llamo byte

Un bit es la unidad mínima de información y un byte es el mínimo espacio que puede direccionarse en la RAM (celdas de memoria)

Un bit es una celda de memoria compuesta por 8 bytes

Un byte codifica hasta 256 bits diferentes

De las diversas definiciones vistas de información concluímos:

Es un concepto difícil de definir y por tanto no hay una teoría válida aceptada.

la más antigua vista (Shannon) ya no se usa pues ha sido sustituida por la de Kolmogorov.

Las dos definiciones son iguales solo que enunciado de manera diferente

Cada teoría surge para aplicar el concepto de información a un contexto diferente. Al contexto de la transmisión Shannon) y al contexto de la compresión (Kolmogorov)

Un lenguaje de Alto Nivel

Requiere, como su nombre indica, un alto grado de especialización para trabajar con él.

Usa instrucciones parecidas al lenguaje humano aparte de favorecer el uso de estructuras de datos como vectores o registros.

Usa instrucciones parecidas a las del juego de instrucciones de microprocesador

Incluye sentencias de lenguajes de Medio y de Bajo Nivel

Un algoritmo es:

Un programa ejecutado por un ordenador

Un conjunto de instrucciones que se aplican por orden sobre unos datos

Un proceso mecánico y efectivo, finito en su ejecución y bien definido (no ambiguo)

Un programa finito que ha necesitado de una tarea intelectual para ser diseñado.

El diseño descendente o "Top-Down"

Es la metodología usada por la programación estructurada cuyo resultado es la descomposición del problema en otros más sencillos directamente codificables en subprogramas

Es la metodología usada por la programación estructurada cuyo resultado es definir el ámbito de las variables usadas en el programa

Es la metodología usada por la programación estructurada cuyo resultado es proporcionar algoritmos que resuelvan un mismo problema de diferentes formas sencillas.

No es usado por la Programación estructurada en absoluto. Es una herramienta del compilador

Un tipo de datos es

Un espacio reservado en la memoria RAM para almacenar algún dato de un tamaño fijo.

Una característica que tienen en común un conjunto de datos, por ejemplo, que todos se puedan sumar

Un espacio en memoria RAM para almacenar un dato y la definición de las operaciones permitidas sobre ese dato.

Una manera de darles nombres a las variables de un programa.

La máquina de Turing:

Fue construida en la década de los 30 del S. XX para realizar cálculos matemáticos intensivos

Es un modelo de computación, y como tal, es una abstracción intelectual que sirve para explorar las propiedades y límites de lo que es y no es computable.

Hace operaciones aritméticas rápidamente y es barata de construir(sólo hace falta una cabeza escritora/lectora y una cinta donde escribir y leer)

Fue superada en su diseño rápidamente por el modelo de Von Newmann

Los 27 caracteres del alfabeto castellano pueden ser codificados con un número mínimo de

5 bits

2^5 bits

$\log_2$ (128) bits

una cadena de 2 caracteres (el carácter considerado y el '\0')

El nivel físico de un fichero: (o los aspectos de los que se ocupa el nivel físico de un fichero)

Es transparente al programador y está controlado por el sistema operativo

es el que nos permite abstraernos de los detalles de cómo se almacena realmente un fichero en el D.A.S.

es el conjunto de propiedades físicas (tamaño, peso,...) del fichero

Controla directamente la posición de la ventana del fichero.

Las sentencias de tipo iterativo

Se ejecutan siempre un número fijo de veces, siendo determinado éste antes de entrar en ellas

No se pueden anidar si son de diferentes tipos (por ej. un while dentro de un for) ya que rompen la estructuración del código

pueden provocar el mal funcionamiento del programa si no se determina bien en su construcción la condición de salida de ellas

siempre provocan que el programa funcione mal cuando dentro de su código aparecen sentencias condicionales

2. **Encuentra los errores de estos códigos en C. (Puede haber más de un error en cada uno, e incluso ningún error en alguno). Nota: no es necesario que encuentres un sentido práctico a lo que hace el código, sólo fíjate en los errores conceptuales:**

CODIGO I

```
1 void main()
2 {
3 int i;
4 float resultado, suma;
5 resultado = 0.0;
6 for(i=0; i< 100; i++)
7  {
8  resultado = (resultado + suma) / 4.1;
9  if( resultado = = (float) i)
10   resultado = 0.0;
11 }
12 }
```

CODIGO II

```
1 int suma(int *a, int *b)
2 {
3    int resultado;
4    resultado = a + b;
5    return resultado;
6 }
7 void main()
8 {
9 int h, m;
10 printf("Introduce dos enteros");
11 scanf("%d%d", &h, &m);
12 printf("%d", suma(h, m));
13 }
```

CODIGO III

```
1 struct cosa{
2 int ag1;
3 float ag2;
4 char ag3;
5 }

6 void imprime_cosa(void){
7 printf("Entero %d\n", cosa.ag1);
8 printf("Real %f\n", cosa.ag2);
9 printf("Carácter %c\n", cosa.ag3);
10 }
11 void main(){
12 struct cosa;

13 scanf("%d", &(cosa.ag1);
14 scanf("%f", &(cosa.ag2);
15 scanf("%c", &(cosa.ag3);

16 imprime_cosa();
17 }
```

CODIGO IV

```
1 int inicializa(int x, int vector[]){
2 vector[x] = x;
3 x = x + 1;
4 return x;
5 }

6 void main(){
7 int valor;
8 int V[100];
9 int i;
10 valor = 0;
11 while(valor < 100){
12   valor = inicializa(valor, V);
13   if((valor % 3) = =0)
14      V[valor] = 1000; /*marca de grupo*/
15 for (i=0;i<100; i++){
16   if(v[i] != 1000)
17       printf("%d\n", V[i]);
18   else
19       printf("-----------\n");
20 }
21}
```

3. A)Indica lo que hace la siguiente función .(No lo que hace paso a paso, sino para que sirve)

```
1  #include <stdio.h>
2  #include <math.h>
3  #define  MAX  100

4  void main(){
5  int i,j;
6  int V[MAX];
7  int raiz;
/* sqrt calcula la raiz cuadrada y devuelve un double*/
8  raiz = (int) sqrt(MAX);
9  for(i=1;i<MAX; i++)
10   V[i] = 1;

11 for(i=2; i<= raiz; i++){
12   j = i;
13   while(i * j < MAX){
14       V[i*j] = 0;
15       j++;
16   }
17 }

18 for(i=1; i<MAX; i++)
19   if(V[i] !=0)
20      printf("%d\n", i);
21 }
```

B) Supón que ahora MAX = 10. Escribe el vector V:
Cuando el programa se encuentra en ejecución en la línea 11
Cuando el programa se encuentra en ejecución y i=2 j = 10
Cuando el programa está ejecutando la línea 18

4 Escribir el código C de un programa que haga lo siguiente:
Dado un fichero supuestamente existente llamado "datos.txt" que contiene EN FORMATO TEXTO números reales separados por el carácter '\n' (fin de línea),

Leer los datos de dicho fichero e imprimir por pantalla el número mínimo, el máximo y la media de todos los números de dicho fichero.

5 **Responde razonadamente y brevemente estas preguntas**:

¿Porqué es conveniente modularizar los programas medianamente grandes?

¿Por qué no es conveniente usar variables globales ?

¿Por qué en el código del ejercicio 3 definimos una constante (MAX) y no ponemos directamente donde lo necesitemos el valor 100?