
Estructura de Datos y de la Información I, Abril 2003

Apellidos:
Grupo:

Nombre:
Aula:

Bloque:

1	2	3	T

Observaciones y advertencias: leáanse detenidamente antes del inicio del examen

1. El alumno escribirá su nombre en **TODAS** las hojas de examen que se le entreguen y deberá entregar **TO-DAS ELLAS** al terminar el examen, separando cuidadosamente las hojas a corregir de las de borradores (ha de extremarse el cuidado en dicha separación, pues los borradores no se corregirán en ningún caso).
El no hacerlo así se considerará como indicio de posible participación en copia.
2. Se recuerda que, como es obvio, el alumno **TIENE LA OBLIGACIÓN** de custodiar **ACTIVAMENTE** las hojas y otros materiales suyos con los que trabaje en el examen, manteniéndolos fuera del alcance visual o físico de otros estudiantes.
El no hacerlo así se considerará como indicio de participación en copia.
3. De detectarse casos de copia, los mismos supondrán de entrada el suspenso de todos los implicados, bien sean fuentes o receptores, sin perjuicio de otras medidas disciplinarias que puedan aplicarse.
4. Las incidencias de copia detectadas durante el examen o en su corrección se pondrán en conocimiento de la Dirección de la EPS, así como del resto de los profesores de otras asignaturas en las que estén matriculados los implicados.
5. No se considerarán aquellas respuestas que no estén razonablemente explicadas.

Preguntas

1. a. Enumerar las fases del ciclo de vida del software. ¿Qué tipos de pruebas se realizan habitualmente sobre un programa? ¿Qué datos de prueba se usan?
b. Enumerar y describir brevemente las componentes básicas de un algoritmo. ¿Qué se entiende por programación estructurada?
c. Dar el pseudocódigo de un algoritmo de prototipo `ent evalSufijo(CC S)` que reciba una cadena de caracteres `S` con una expresión sufijo con datos del tipo entero `ent` y devuelva su valor evaluado mediante una pila (usar las primitivas de pila definidas más adelante, suponer definidas otras funciones que se puedan necesitar y NO considerar posibles errores).
d. Definir qué se entiende en C por un puntero. Si se tiene una tabla `C` declarada como `int t[10]`, ¿qué se debe hacer para acceder a sus datos mediante índices que vayan de 1 a 10?
2. a. Se ha definido un tipo de dato un poco raro de nombre `tipoR` cuyos elementos se almacenan mediante una estructura de la forma inferior, donde `dim` es la dimensión de una tabla de datos `char` apuntada por `pc`:

```
typedef struct TIPO_RARO {  
    char *pc ;  
    int dim ;  
} tipoR ;
```

Dar el prototipo y el código de una función de nombre `swapTR` que reciba de manera adecuada dos datos tipo `R` `d1`, `d2` que tengan un mismo valor en el miembro `dim` e intercambien los datos almacenados en sus correspondientes tablas.

Si se declaran dos datos como tipo `R` `d1`, `d2`, ¿cómo habría que llamar a la función anterior sobre los mismos?

b. Para la expresión inferior, dar la evolución sobre una pila de la aplicación a la misma del algoritmo de traducción de infijo a sufijo (expresar dicha evolución en tres columnas que en cada paso contengan respectivamente el carácter que se lee, la traducción parcial y el estado de la pila):

$(A - B) * (C - (D + E) \wedge (F - G))$;

3. a. Sobre la cola circular `Q` del esquema inferior se efectúan las siguientes operaciones: a. `ColaIns(3, Q)`; b. `ColaRem(A, Q)`; c. `ColaIns(4, Q)`; d. `ColaIns(5, Q)`. Dar para cada una el estado de la cola y las posiciones de su `front` y `rear`.

B			A
---	--	--	---

b. Una implementación de pilas tiene como único interfaz las funciones

`status PIni(pila P)`,

`bool PVacia(pila P)`,

`status Push(dato D, pila P)`, que inserta el dato `D` en la pila, y

`status Pop(dato D, pila P)`, que extrae el elemento `D` desde la pila.

Dar razonadamente el pseudocódigo de una función de prototipo `status juntaPilas(pila A, pila B)` que reciba una tales pilas y sitúe los elementos de `B` encima de los elementos de `A` y en el mismo orden que seguían en `B`. Si la función devuelve `Ok`, `B` debe quedar vacía, pero ninguna de las dos pilas debe cambiar si hay algún error. Para ello, indicar brevemente el método a seguir y

a. dar primero una versión sin control de excepciones;

b. identificar sobre dicha versión la ubicación de las posibles excepciones, y

c. dar finalmente para cada excepción un pseudocódigo que incorpore su detección y recuperación.

Suponer para ello que `Pop` devuelve `error` si la pila está vacía y que `k pops` sobre una pila pueden seguirse por $p \leq k$ pushes sobre la misma pila sin causar errores.