Resumen de programacion

October 1, 2020

1 Recursion

definición inductiva: obtener conceptos nuevos empleando el mismo concepto que intenta describir.

Una **definición recursiva** dice cómo obtener conceptos nuevos empleando el mismo concepto que intenta definir

Un razonamiento recursivo tiene **dos partes:**

- La base, que no es recursiva y es el punto tanto de partida como de terminación de la definición. (son los elementos del conjunto se especifican explícitamente y aseguren el final o corte)
- regla recursiva de construcción. Los elementos del conjunto que se definen en términos de los elementos ya definidos

Recursividad de cola: Cuando una llamada recursiva es la última posición ejecutada del procedimiento

Recursion directa: Cuando un procedimiento incluye una llamada a si mismo

Recursion indirecta: Cuando un procedimiento llama a otro procedimiento y este causa que el procedimiento original sea invocado

Es menos peligrosa una recursión infinita que un ciclo infinito, ya que una recursividad infinita pronto **se queda sin espacio** (para cada llamada se crean copias independientes de las variables declaradas en el procedimiento) y termina el programa, mientras que la iteración infinita puede continuar mientras no se termine en forma manual.
el cálculo recursivo es mucho más caro que el iterativo.

Pasos de la recursion:

1. El procedimiento se llama a si mismo
2. El problema se resuelve, resolviendo el mismo problema pero de tamaño menor
3. La manera en la cual el tamaño del problema disminuye asegura que el caso base eventualmente se alcanzará

2 Principios de programacion

Nombres de variables y funciones

- Se deben elegir nombres significativos que sugieran claramente la finalidad de la funcion o variable en cuestion
- Las variables que se usan poco deben tener nombres mas sencillos (una letra basta).
- No cometer errores de ortografia.

Documentacion y formato

- Es conveniente documentar el programa **Antes** de empezar a codearlo.
- Se introduce cada seccion importante del programa con un comentario breve que explique su finalidad
- No se hacen comentarios que expliquen lo obvio ni se comenta cada linea.

Codificacion, prueba y refinamiento ulterior - Modularidad

- **Refinamiento descendente** Primero pensar en el programa entero y desde alli ramificarlo en subprogramas mas pequeños.
- Siempre revisar nuestro codigo para ver que se le puede mejorar. Para ello, es muy util probar el codigo, y existen 3 maneras conocidas para hacerlo:
    1. **caja negra** Se prueba el codigo con valores tipicos, realistas, genericos, etc.
    2. **caja de cristal** Se prueba el codigo con todos los valores de entrada posibles.
    3. **Caja de pandora** Dejamos que el cliente haga las pruebas y nos comunique los resultados.

Algunas otras recomendaciones:

- Procurar que los algoritmos sean lo mas simple y pequeños posibles.
- Analice las exigencias del tiempo y espacio al seleccionar un algoritmo.
- Comenzar de nuevo suele ser mas fácil que parchar un programa

3 Estructura de Datos

La pc solo entiende numeros binaros. Para mostrarnos todo lo que podemos ver en pantalla, todo se convierte a binario de distintas formas:

Enteros binarios 00100110 $\rightarrow$ 38// El primer dígito representa positivo o negativo, en caso de ser negativo se invierten todos los valores de los otros BITS 11011001 = −38

Notación de Punto Flotante El número real se representa por un número llamado MANTISA multiplicado por una base de elevada a una potencia entera, llamada EXPONENTE.
Ej. El número 387,53 se representa como $38753x10^{-2}$

Cadena de Caracteres Se pueden representar caracteres también.
Si 8 bits representan un carácter, se pueden representar hasta 256 caracteres diferentes.

Hardware y Software bit $\rightarrow$ 0 / 1
Los bits están agrupados en unidades más grandes, los **bytes**.

El concepto de implantación Implantación de Software: se escribe un programa que consta de instrucciones existentes en el hardware para interpretar en la forma deseada las cadenas de caracteres y ejecutar las operaciones requeridas. De acá en adelante cuando digamos Implantación hablaremos de IMPLANTACION DE SOFTWARE.

datos abstractos Una herramienta útil para especificar las propiedades lógicas de los tipos de datos, es el tipo de dato abstracto ADT, el cual es, una colección de valores y un conjunto de operaciones con esos valores.

4 Listas ligadas

Una lista ligada es una estructura de datos en la que cada elemento (nodo) tiene dos partes, uno con **información** (un valor de tipo genérico, dato, info, etc.) y el otro con la **dirección del nodo siguiente**.

Clasificación de Listas

- **Listas simplemente Enlazadas:** cada nodo contiene un único enlace que conecta al nodo siguiente o sucesor. (Recorrer hacia adelante)
- **Listas doblemente Enlazadas:** cada nodo contiene 2 enlaces, uno al nodo siguiente y uno al anterior. (Recorrer hacia delante y hacia atrás)
- **Lista circular simplemente enlazada:** similar a la primera sólo que el último nodo enlaza con el primero. (Recorrer hacia adelante en forma circular)
- **Lista circular doblemente enlazada:** similar a la segunda con el último nodo enlazado al primero y el primero al último.

5 Pilas

La propiedad anterior se conoce como **LIFO** (Last In First Out), es decir, el último en entrar será el primero en salir y al que se tenga acceso.

Esta estructura se aplica en multitud de supuestos en el área de informática debido a su simplicidad y capacidad de dar respuesta a numerosos procesos.

Para el manejo de los datos cuenta con dos operaciones básicas: apilar (push), que coloca un objeto en la pila, y su operación inversa, retirar (o desapilar, pop), que retira el último elemento apilado.

En cada momento solamente se tiene acceso a la parte superior de la pila, es decir, al último objeto apilado (denominado **TOS**, **Top of Stack**). La operación retirar permite la obtención de este elemento, que es retirado de la pila permitiendo el acceso al anterior (apilado con anterioridad), que pasa a ser el último, el nuevo TOS.

Las pilas suelen emplearse en Implementación de recursividad.

En un sistema operativo cada proceso tiene un espacio de memoria (pila) para almacenar valores y llamadas a funciones.

Una **pila acotada** es una pila limitada a un tamaño máximo impuesto en su especificación.

6 Colas

Una **COLA** es una colección ordenada de elementos de la que se pueden borrar elementos en un extremo (**FRENTE** de la cola) o insertarlos en el otro (**FINAL** de la cola).

FIFO (First In First Out), es decir, el primero primero en entrar será el primero primero en salir.

La cola de prioridad Es una estructura de datos en la que el ordenamiento de los elementos se determina por el resultado de operaciones básicas. Puede ser **ascendente** o **descendente**

7 Arboles

Un nuevo arbol a partir de un nodo n_r y K arboles de raices n_1, n_2, n_k con N_1, N_2, N_k elementos cada uno, puede construirse estableciendo una relacion padre e hijo entre n_r y cada una de las raices de los demas k arboles. El arbol resultante de $N = N_1 + N_2 + N_k$ nodos tiene como raiz el nodo n_r, los nodos n_1, n_2, n_k son nodos hijos de n_r y el conjunto de nodos hoja esta formado por la union de los k conjuntos de hojas iniciales. A cada uno de estos arboles A_i se los denota **subarboles** de la raiz.

Arbol binario Un árbol binario binario es un conjunto conjunto finito de elementos que o está vacío o esta dividido en tres subconjuntos. El primer subconjunto contiene un solo elemento llamado **raíz** del árbol. Los otros dos son en sí mismos árboles binarios, y se llaman **subárbol izquierdo y derecho**. Cualquiera de estos puede estar vacío.

Arbol estrictamente binario Tiene $2n - 1$ nodos

Nivel de un nodo la raiz es de nivel 0, cada nodo del arbol es un nivel mas que su padre

Arbol binario completo es un arbol binario estricto con todas sus hojas en un mismo nivel d

Algunas funciones de los arboles

- devolver info de un nodo
- devuelve el puntero al hijo izq o der de un nodo
- Crea un arbol
- asigna un hijo izq o der a un padre sin el determinado hijo.