Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Organización de Computadoras

Práctica para el Final Regular

Segundo Cuatrimestre de 2024

Ejercicios

1. Considerar los siguientes planteos en el contexto del lenguaje C:

   a) ¿Cómo se definen las variables de tipo puntero?

   b) ¿Qué estructuras de datos pueden ser apuntadas por una variable de tipo puntero?

   c) ¿Es posible declarar una variable de tipo puntero sin especificar a qué estructura de datos apuntará?

   d) ¿La constante `NULL`, de qué tipo es?

   e) ¿Qué diferencia una estructura de datos estática de una dinámica? Encontrar al menos dos ejemplos de cada categoría.

   f) ¿Una variable de tipo puntero en sí misma, constituye una estructura de datos estática o dinámica?

   g) ¿Cuál es la utilidad de las funciones de librería `malloc()` y `free()`?

   h) ¿Cómo hace el compilador para saber cuanta memoria debe reservar en el `HEAP` ante cada invocación a la función `malloc()`?

2. Diseñar un TDA `EnteroLargo`, el cual debe ser capaz de almacenar números enteros de hasta 100 dígitos. Definir e implementar todas las operaciones necesarias para poder crear, eliminar, inicializar, mostrar por pantalla, comparar y sumar enteros largos.

3. Diseñar un TDA `ListaEnterosLargos`, el cual haciendo uso del TDA `EnteroLargo` y de alguno de los conceptos de posición conocidos, sea capaz de modelar listas simplemente enlazadas. Definir e implementar todas las operaciones necesarias para poder crear, eliminar, consultar la cantidad de elementos, recorrer la lista, agregar elementos y determinar la posición de un elemento dado.

4. Haciendo uso del TDA `ListaEnterosLargos`, implementar sendas funciones que a partir de dos listas de enteros largos, retorne lo siguiente:

   a) Una lista que sea la concatenación de las listas recibidas.

   b) Una lista que contenga los elementos de las listas recibidas, pero donde los elementos en común aparezcan una única vez.

   Pista: para cada inciso, escribir una función que tenga como entrada a las dos listas y retorne una nueva lista, eliminando las listas recibidas como entrada durante el procesamiento de las mismas.

5. Extender el TDA `ListaEnterosLargos` agregando dos funciones, una recursiva y la otra no recursiva, que permitan liberar todas las celdas de una lista de enteros, con la restricción de recorrer la lista una única vez.

6. Diseñar un TDA `PilaEnteros`, el cual haciendo uso de alguno de los conceptos de posición conocidos, sea capaz de modelar pilas convencionales. Definir e implementar todas las operaciones necesarias para poder crear, eliminar, apilar, desapilar y consultar si se trata de una pila vacía.

7. Dada una pila de enteros, se desea intercambiar los elementos ubicados en el fondo y en el tope. Implementar una función que realice esta tarea apelando a una pila auxiliar, con la condición de sólo hacer uso de las funciones provistas por el TDA `PilaEnteros`.

8. Rehacer el ejercicio anterior, esta vez sin hacer uso de la pila auxiliar.

9. Escribir una función `invertir()` que reciba una pila de enteros arbitraria y que retorne otra pila conteniendo los mismos enteros, pero en orden inverso. En este caso, sólo se pueden utilizar las operaciones del TDA `PilaEnteros`, haciendo uso de una o más estructuras auxiliares, según se requiera.

10. Implementar un programa que reciba el nombre de un archivo en su línea de comandos y muestre por pantalla su contenido.

11. Definir un programa que reciba el nombre de dos archivos en la línea de comandos, uno de entrada y otro de salida, tal que el archivo de salida sea un copia del archivo de entrada, con la salvedad de haber expandidos cada caracter de tabulación por una secuencia de cuatro espacios en blanco.

12. Escribir un programa que al recibir el nombre de un archivo de texto en la línea de comandos le agregue el número de línea actual a cada una de las líneas que componen al archivo.

13. Implementar un programa que dado un archivo de texto y una palabra en la línea de comando muestre por pantalla la primer línea que contenga la palabra buscada.

Referencias

[KR88] Kernighan, B., and Ritchie, D. *The C Programming Language.* Prentice-Hall, 1988.