***BÀI THỰC HÀNH BUỔI 4***

Cài đặt Stack tổng quát, Sử dụng stack để:

1. Đổi số nguyên dương ra dạng nhị phân (có trong bài giảng)
2. Đảo ngược chuổi theo từ

“anh yêu em” đảo thành “em yêu anh”

“cấm không được đứng đây” đảo thành “đây đứng được không cấm”

1. Kiểm tra biểu thức hợp lệ về mở ngoặc và đóng ngoặc
2. Khử bỏ đệ quy bài toán tháp Hà nội
3. Khử bỏ đệ quy Quick Sort

***Thử thách:*** *Viết chương trình nhập vào biểu thức dạng chuổi, in ra kết quả nếu không lỗi, ngược lại thông báo lỗi.*

***BÀI THỰC HÀNH BUỔI 5***

Cài đặt hàng đợi, sử dụng để sắp thứ tự dãy số nguyên dương bằng thuật toán radix sort

import java.util.\*;

public class Bai1 {

private int maxSize; // Kích thước tối đa của hàng đợi

private int front; // Vị trí phần tử đầu hàng đợi

private int rear; // Vị trí phần tử cuối hàng đợi

private int[] queueArray; // Mảng lưu trữ hàng đợi

private int nItems; // Số phần tử hiện tại trong hàng đợi

// Constructor khởi tạo hàng đợi với kích thước maxSize

public Bai1(int maxSize) {

this.maxSize = maxSize;

this.queueArray = new int[maxSize];

this.front = 0;

this.rear = -1;

this.nItems = 0;

}

public boolean isEmpty() {

return (nItems == 0);

}

public void insert(int x) {

if (rear == maxSize) rear = -1;

queueArray[++rear]=x;

nItems++;

}

public int remove() {

int temp = queueArray[front++];

if(front==maxSize)

front=0;

nItems--;

return temp;

}

public int peekFront() {

return queueArray[front];

}

public boolean isFull() {

return (nItems==maxSize);

}

public int size() {

return nItems;

}

static void ngaunhien(int[] a, int n) {

Random rd = new Random();

for (int i=0; i<n; i++)

a[i] = rd.nextInt(1000);

}

static void xuat(int[] a, int n) {

for (int i=0; i<n; i++)

System.out.print(" " + a[i]);

System.out.println();

}

static int scs(int n) {

int d = 0;

while (n>0)

{

d++;

n=n/10;

}

return d;

}

static void radixsort(int[] a, int n) {

int i, k, m, lt, s, d, t, v;

Bai1[] Q;

Q = new Bai1[20];

for (i=0; i<20; i++)

Q[i] = new Bai1(100);

k = a[0];

for (i=0; i<n; i++) {

if (k<a[i]) k = a[i];

Q[a[i]%10].insert(a[i]);

}

m = scs(k);

lt = 1;

s = 0;

d = 10;

for (i=1; i<m; i++) {

lt = lt\*10;

for (k=0; k<10; k++)

while (!Q[s+k].isEmpty()) {

v = Q[s+k].remove();

Q[d+v/lt%10].insert(v);

}

t = s;

s = d;

d = t;

}

n=0;

for (k=0; k<10; k++)

while (!Q[s+k].isEmpty())

a[n++]=Q[s+k].remove();

}

public static void main(String[] args) {

int[] a;

int n;

a = new int[100];

Scanner x;

System.out.print("Nhap so phan tu: ");

x = new Scanner(System.in);

n = x.nextInt();

ngaunhien(a,n);

System.out.print("\nDay so ban dau:\n");

xuat(a,n);

radixsort(a,n);

System.out.print("\nDay so da sap xep:\n");

xuat(a,n);

}

}

***BÀI THỰC HÀNH BUỔI 6***

1. Cài đặt danh sách liên kết đơn
2. Sử dụng danh sách liên kết đơn để viết chương trình:
   1. Nhập danh sách dãy số nguyên
   2. In danh sách liên kết
   3. Sắp thứ tự danh sách
   4. Đếm số phần tử mang giá trị x trong danh sách
   5. Xóa tất cả các phần tử mang giá trị x trong danh sách
   6. Đếm số dãy con tăng dần trong danh sách.

Ví dụ: 1 6 8 2 5 9 4 8

Có 3 dãy con tăng dần

1. Làm tương tự bài 2 (từ a đến e) với danh sách sinh viên (MSSV, Ho, Ten, Diem)

class Link {

public int dData;

public Link next;

public Link(int d) {

dData = d;

}

}

class LinkedList {

private Link first;

private Link last;

private Link pos;

private int count;

public LinkedList() {

first = null;

last = null;

pos = null;

count = 0;

}

public boolean isEmpty() {

return first == null;

}

public void insert(int dd, Link p) {

Link newLink = new Link(dd);

if (p==null) {

newLink.next = first;

first = newLink;

}

else {

newLink.next = p.next;

p.next = newLink;

}

if (newLink.next == null) {

last = newLink;

count++;

}

}

public void insertFirst(int dd) {

insert(dd, null);

}

public void insertPos(int dd) {

insert(dd, pos);

}

public void insertLast(int dd) {

insert(dd, last);

}

public void delete(Link p) {

Link t;

if (p==null) {

t = first;

first = t.next;

}

else {

t = p.next;

p.next = t.next;

}

if (t.next == null)

last = p;

count--;

}

public void deleteFirst() {

delete(null);

}

public void deletePost() {

delete(pos);

}

public boolean searchList(int dd) {

Link c;

c = first;

pos = null;

while (c!=null && c.dData!=dd) {

pos = c;

c = c.next;

}

return c!=null;

}

public boolean searchOrder(int dd) {

Link c;

c = first;

pos = null;

while (c!=null && c.dData<dd) {

pos = c;

c = c.next;

}

if (c!=null && c.dData>dd)

return false;

return c!=null;

}

public void startList() {

pos = first;

}

public void nextLink() {

if (pos == null)

pos = first;

pos = pos.next;

}

public boolean endList() {

return pos == null;

}

public int getData() {

return pos.dData;

}

public int nItems() {

return count;

}

}

import java.util.Scanner;

public class Bai2 {

static void nhap(LinkedList l) {

Scanner sc = new Scanner(System.in);

int x;

do {

x = sc.nextInt();

if (x>0)

l.insertLast(x);

} while (x>0);

}

static void in(LinkedList a) {

int x;

a.startList();

while (!a.endList()) {

x = a.getData();

System.out.print(x + " ");

a.nextLink();

}

System.out.println();

}

static LinkedList saptt(LinkedList lt) {

int x;

LinkedList l = new LinkedList();

while (!lt.isEmpty()) {

lt.startList();

x = lt.getData();

l.searchOrder(x);

l.insertPos(x);

lt.deleteFirst();

System.out.print(x + " ");

}

System.out.println();

return l;

}

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

LinkedList linkedList = new LinkedList();

int choice;

do {

System.out.println("Menu:");

System.out.println("1. Nhap day: ");

System.out.println("2. In day: ");

System.out.println("3. Sap thu tu day: ");

System.out.println("0. Thoát");

System.out.print("Chọn: ");

choice = sc.nextInt();

switch (choice) {

case 1:

System.out.print("Nhap day: ");

nhap(linkedList);

break;

case 2:

System.out.print("Day da nhap: ");

in(linkedList);

break;

case 3:

System.out.print("Day da sap xep: ");

saptt(linkedList);

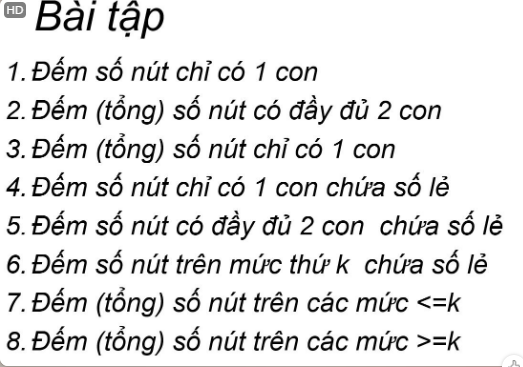
break;

}

} while (choice != 0 );

}

}



import java.util.Scanner;

class Node {

public int dData;

public Node left, right;

public Node(int dd) {

dData = dd;

left = right = null;

}

}

class Tree {

private Node root;

public Tree() {

root = null;

}

public boolean isEmpty() {

return root == null;

}

public int getData() {

return root.dData;

}

public void setData(int dd) {

root.dData = dd;

}

public Tree getLeft() {

Tree t = new Tree();

t.root = root.left;

return t;

}

public Tree getRight() {

Tree t = new Tree();

t.root = root.right;

return t;

}

}

public class DemSoNut {

static Scanner sc = new Scanner(System.in);

public static int soNutChiCo1Con(Tree t) {

int sonut = 0;

if (t.isEmpty() && t.getLeft() != null && t.getRight() == null || t.getLeft() == null && t.getRight() != null) {

sonut++;

}

return sonut;

}

public static int tongSoNutCoDayDu2Con(Tree t) {

int sonut = 0;

if (t.isEmpty() && t.getLeft() != null && t.getRight() != null) {

sonut++;

}

return sonut += tongSoNutCoDayDu2Con(t.getLeft()) + tongSoNutCoDayDu2Con(t.getRight());

}

public static int tongSoNutChiCo1Con(Tree t) {

int sonut = 0;

if (t.isEmpty() && t.getLeft() != null && t.getRight() == null || t.getLeft() == null && t.getRight() != null) {

sonut++;

}

return sonut += tongSoNutChiCo1Con(t.getLeft()) + tongSoNutChiCo1Con(t.getRight());

}

public static int soNutChiCo1ConChuaSoLe(Tree t) {

int sonut = 0;

if (t.isEmpty() && t.getLeft() != null && t.getRight() == null || t.getLeft() == null && t.getRight() != null) {

if (t.getData() % 2!= 0) {

sonut++;

}

}

return sonut;

}

public static int soNutCoDayDu2ConChuaSoLe(Tree t) {

int sonut = 0;

if (t.isEmpty() && t.getLeft() != null && t.getRight() != null) {

if (t.getData() % 2 != 0) {

sonut++;

}

}

return sonut;

}

public static int soNutTrenMucThuKChuaSoLe(Tree t, int k) {

if (t.isEmpty())

return 0;

if (k>1)

return soNutTrenMucThuKChuaSoLe(t.getLeft(), k-1) + soNutTrenMucThuKChuaSoLe(t.getRight(), k-1);

if (k==1)

return t.getData() % 2; // + soNutTrenMucThuKChuaSoLe(t.getLeft(), k-1) + soNutTrenMucThuKChuaSoLe(t.getRight(), k-1);

}

}