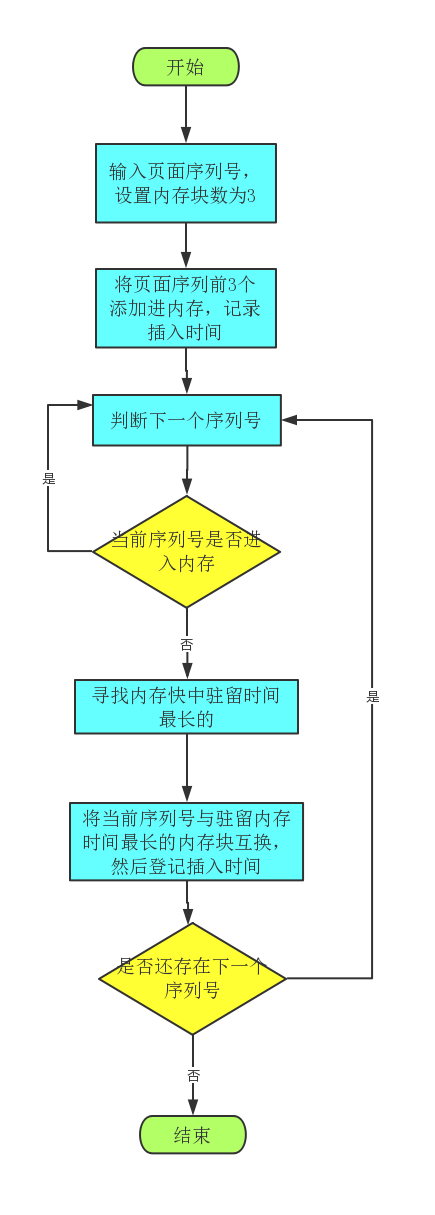
页面置换算法

一、先进先出（FIFO）：

一、流程图



二、源程序

package pageReplacement;  
  
import javafx.scene.control.ScrollToEvent;  
  
import java.util.Scanner;  
  
import static java.lang.System.exit;  
  
/\*\*  
 \* @author tanHongxia  
 \*/  
public class FIFO {  
 /\*\*内存块数\*/  
 private static int storeSize = 3;  
  
 private static int findFirstInIndex(int[][] saveState){  
 int firstIndex = 0;  
 for(int i = 0; i < storeSize; i++){  
 if(saveState[i][1] < saveState[firstIndex][1]){  
 firstIndex = i;  
 }  
 }  
 return firstIndex;  
 }  
  
 /\*\*  
 \* 计算缺页次数和缺页率  
 \* \*/  
 private static int[][] calculate(int[][] saveState, int[] pageOrder){  
 //缺页次数  
 float weackTime = 0;  
 //用于判断进入内存的次序  
 int time = 0;  
 //添加前storeSize块到内存中  
 for(int i = 0; i < storeSize; i++){  
 saveState[i][0] = pageOrder[i];  
 saveState[i][1] = time;  
 weackTime++;  
 time++;  
 }  
 for(int i = storeSize; i < 10; i++){  
 int j = 0;  
 //判断是否在内存中  
 for(; j < storeSize; j++){  
 if(saveState[j][0] == pageOrder[i]){  
 break;  
 }  
 }  
 if(j >= storeSize){  
 int firstInIndex = findFirstInIndex(saveState);  
 saveState[firstInIndex][0] = pageOrder[i];  
 saveState[firstInIndex][1] = time;  
 weackTime++;  
 }  
 time++;  
 }  
 System.out.println("缺页次数： " + weackTime);  
 System.out.println("缺页率：" + (weackTime/time)\*100 + "%");  
 return saveState;  
 }  
  
 private static void printPageOrder(int[] pageOrder) {  
 for (int i = 0; i < 10; ++i) {  
 System.out.print(pageOrder[i] + " ");  
 }  
 }  
  
 private static void saveSate(int[][] saveState) {  
 for(int i = 0; i < storeSize;i++) {  
 System.out.print(saveState[i][0] + " ");  
 }  
 }  
  
 @SuppressWarnings("InfiniteLoopStatement")  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.in);  
 // 获得10个页面序列号  
 int[] pageOrder = new int[10];  
 for(int i = 0; i < 10; i++){  
 System.out.print("输入页面序列号: ");  
 pageOrder[i] = input.nextInt();  
 }  
 //计算缺页次数和缺页率  
 int[][] saveState = new int[storeSize][2];  
 saveState = calculate(saveState,pageOrder);  
 while (true) {  
 System.out.println();  
 System.out.println("\*\*\*\*\*\*\*\*\*\*页面置换--FIFO算法\*\*\*\*\*\*\*\*\*\*");  
 System.out.println("1：显示已经被存入内存的页面");  
 System.out.println("2：显示页面序列号");

System.out.println("3：退出");  
 int choose = input.nextInt();  
 switch(choose)  
 {  
 case 1: saveSate(saveState);break;  
 case 2: printPageOrder(pageOrder);break;

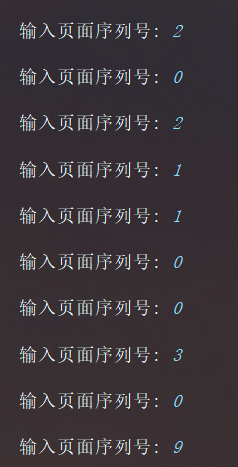
Case 3: exit(0); break;  
 default:exit(0);break;  
 }  
 }  
 }  
}

三、程序运行环境、使用工具说明

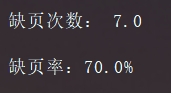
Jdk1.8 ideal

四、程序运行结果

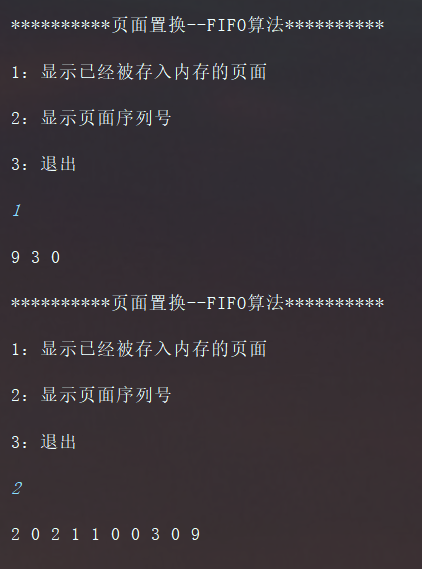
输入页面序列号



显示缺页次数和缺页率：



菜单页：

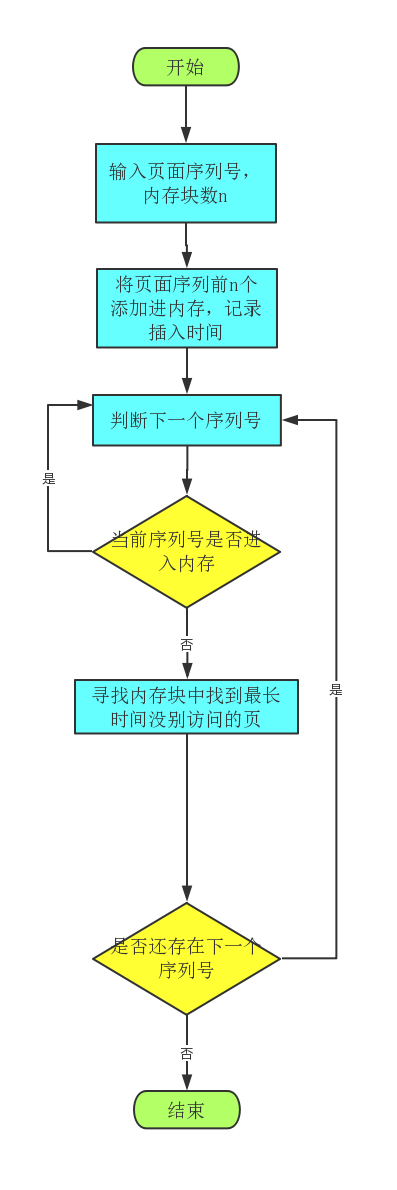


五、结果的简单说明或分析

掌握了先进先出置换算法的概念和处理方法

二、最近最久未使用（LRU）：

一、流程图



二、源程序

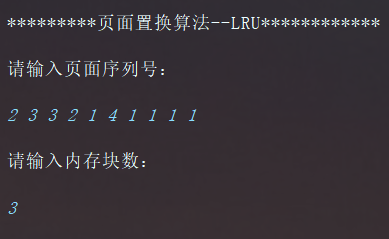
*package* pageReplacement;  
  
*import* java.util.ArrayList;  
*import* java.util.Iterator;  
*import* java.util.LinkedList;  
*import* java.util.Scanner;  
  
*/\*\*  
 \* @author tanHongxia   
 \*/  
public class* LRU{  
 *private int* times;  
 */\*\*缺页次数\*/  
 private int* fail;  
 *private int* cap;  
 ArrayList<Integer> list=*new* ArrayList<Integer>();  
 *private* LRU(){  
 Scanner in=*new* Scanner(System.in);  
 fail=0;  
 System.out.println("\*\*\*\*\*\*\*\*\*页面置换算法--LRU\*\*\*\*\*\*\*\*\*\*\*\*");  
 System.out.println("请输入页面序列号：");  
 String s=in.nextLine();  
 String sarr[]=s.split(" ");  
 times=sarr.length;  
 *for*(*int* i=0;i<times;i++){  
 list.add(Integer.*parseInt*(sarr[i]));  
 }  
 System.out.println("请输入内存块数：");  
 cap=in.nextInt();  
 run();  
 }  
  
 *private void* run(){  
 LinkedList<Integer> linkedList=*new* LinkedList<Integer>();  
 *for*(*int* i=0;i<times;i++){  
 *int* t=list.get(i);  
 *if*(!linkedList.contains(t)){  
 fail++;  
 *if*(linkedList.size()<cap){  
 linkedList.addLast(t);  
 }  
 *else*{  
 linkedList.removeFirst();  
 linkedList.addLast(t);  
 }  
 }  
 *else*{  
 linkedList.remove((Integer)t);  
 linkedList.addLast(t);  
 }  
 System.out.print("第"+(i+1)+"次 ：");  
 Iterator<Integer> it=linkedList.iterator();  
 *while*(it.hasNext()){  
 Integer itg=it.next();  
 System.out.print(itg+" ");  
 }  
 System.out.println();  
 }  
 System.out.println("缺页次数："+fail);  
 System.out.println("缺页率："+(*float*)fail/times);  
 }  
  
 *public static void* main(String[] args){  
 LRU lru=*new* LRU();  
 }  
}

三、程序运行环境、使用工具说明

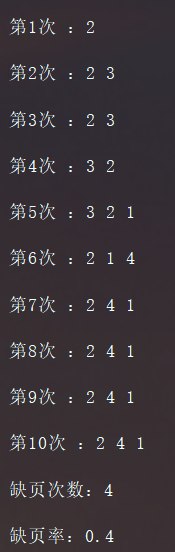
Jdk1.8 ideal

四、程序运行结果

输入页面序列号、内存块数：



输出:



五、结果的简单说明或分析

对最近最久未使用算法（LRU）了解得更加深入。