学习情况表

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 张蕊晗 | **学号** | 2020905198 |
| **学院** | 信息工程学院 | **专业** | 电子信息工程 |

（请在下面表格本周学习情况）

|  |
| --- |
| 学习情况简述 |
| 复习了堆的相关知识 |
| 本周练习过的代码（例） |
| （本周练习写过的代码，如实填写，疑似抄袭会单独测试）  *代码1：*  *#include<stdio.h>*  *int main()*  *{*  *printf("HelloWorld\n");*  *return 0;*  *}*  *代码2：*  …….. |
| 本周练习过的代码 |
| //heap.h  #pragma once  #include <stddef.h>  #include <stdio.h>  //如果a和b满足比较关系，返回1  //如果a和b不满足比较关系，返回0  //所谓的比较关系，对于小堆来说，就是a<b  //对于大堆来说，就是a>b  #define HeapMaxSize 1000  typedef int HeapType;  typedef int (\*Compare)(HeapType a, HeapType b);  typedef struct Heap{  HeapType data[HeapMaxSize];  size\_t size;  Compare cmp;  }Heap;  //heap.c  #include "heap.h"  //如果a小于b，就返回1，否则返回0  int Less(HeapType a, HeapType b){  return a < b;  }  void HeapInit(Heap\* heap, Compare compare){  if(heap == NULL || compare == NULL){  return;//非法输入  }  heap -> size = 0;  heap -> cmp = compare;  return;  }  void Swap(HeapType\* a, HeapType\* b){  HeapType tmp;  tmp = \*a;  \*a = \*b;  \*b= tmp;  return;  }  void AdjustUp(HeapType data[], size\_t size, Compare cmp, size\_t index){  if(index >= size){  return;  }  //1.先找到当前节点对应的父节点  size\_t child = index;  size\_t parent = (child - 1)/2;  while(child > 0){  //2.比较父节点和子节点的大小关系，如果子节点值比父节点小，交换父子节点的值，如果子节点的值比父节点的大，说明调整也完成了  if(cmp( data[child] , data[parent] )){  Swap(&data[child], &data[parent]);  }else{  break;  }  //3.将当前父节点作为新的子节点，再去找子节点的父节点，循环进行比较和交换  child = parent;  parent = (child - 1)/2;  }  //4.子节点下标等于0，循环结束  return;  }  void HeapInsert(Heap\* heap, HeapType value){  if(heap == NULL){  return;  }  if(heap -> size >= HeapMaxSize){  return;//堆满了  }  heap -> data[heap -> size++] = value;  AdjustUp(heap -> data, heap -> size, heap -> cmp, heap -> size - 1);  }  int HeapRoot(Heap\* heap, HeapType\* value){  if(heap == NULL){  return 0;  }  \*value = heap -> data[0];  return \*value;  }  void AdjustDown(HeapType data[], size\_t size, Compare cmp, size\_t index){  //1.设定parent指向开始的位置，找到对应的子树节点  size\_t parent = index;  //2.设定一个child指向parent的左子树  size\_t child = parent \* 2 + 1;  //3.判定child和child+1的大小关系，如果child+1的值比child小，就让child = child + 1  while(child < size){  if(child + 1 < size && cmp(data[child + 1], data[child])){  child = child + 1;  }  //4.判定parent和child的值的大小关系，如果parent比child的值打，就进行交换  if(cmp(data[child], data[parent])){  Swap(&data[child], &data[parent]);  }else{  //否则就说明调整已经完成  break;  }  //5.parent赋值为child，child再重新复制成parent的做孩子节点  parent = child;  child = parent \* 2 + 1;  }  }  void HeapErase(Heap\* heap){  if(heap == NULL){  return;//非法输入  }  if(heap -> size == 0){  return;//堆为空  }  Swap(&heap -> data[0], &heap -> data[heap -> size - 1]);  --heap -> size;  AdjustDown(heap -> data, heap -> size, heap -> cmp, 0);  }  //void HeapErase(Heap\* heap){  // if(heap == NULL){  // return;  // }  // heap -> data[0] = heap -> data[heap -> size - 1];  // heap -> size--;  // size\_t parent = 0 ;  // size\_t child = parent \* 2 + 1;  // while(child < heap -> size){  // if(child < heap -> size && heap -> data[child ] < heap -> data[child - 1]){  // child ++;  // }  // if(Less(heap -> data[child], heap -> data[parent])){  // Swap(&heap -> data[child], &heap -> data[parent]);  // parent = child;  // child = 2 \* parent + 1;  // }else{  // break;  // }  //  // }  //}  int HeapEmpty(Heap\* heap){  if(heap == NULL){  return 0;  }  return heap -> size == 0 ? 1 : 0;  }  size\_t HeapSize(Heap\* heap){  if(heap == NULL){  return 0;  }  return heap -> size;  }  void HeapDestroy(Heap\* heap){  if(heap == NULL){  return;  }  heap -> size = 0;  return;  }  //需要开辟额外的空间  void HeapSort1(HeapType array[], size\_t size){  Heap heap;  HeapInit(&heap, Less);  //1.先将数组里的所有元素插入到一个堆里面  size\_t i = 0;  for(; i < size; ++i){  HeapInsert(&heap, array[i]);  }  //2.然后依次进行取堆顶元素，放回原数组，并删除堆顶元素  size\_t output\_index = 0;  while(HeapEmpty(&heap) != 1){  HeapType root = 0;  HeapRoot(&heap, &root);  array[output\_index] = root;  ++output\_index;  HeapErase(&heap);  }  }  //不需要开辟额外的空间，和浪费额外的时间  void HeapSort2(HeapType array[], size\_t size, Compare cmp){  if(size == 0 || size == 1){  return;  }  //1.将数组的所有元素放在一个堆中，循环结束之后，就调整成一个小堆  size\_t i = 0;  for(; i < size;){  AdjustUp(array, i, cmp, i);  ++i;  }  //2.依次去堆顶元素  while(i > 0){  Swap(&array[0], &array[i - 1]);  --i;  AdjustDown(array, i, cmp, 0);  }  }  ///////////////////////////////////////////////////////////  //////////////////////TEST////////////////////////////////  /////////////////////////////////////////////////////////  #if 1  #include <stdio.h>  #define TESTHEARD printf("\n==========================%s=============================\n",\_\_FUNCTION\_\_)  void HeapPrintChar(Heap\* heap, const char\* msg){  printf("%s\n", msg);  size\_t i = 0;  for(; i < heap -> size; ++i){  printf("[%lu] : %d ", i, heap -> data[i]);  }  printf("\n");  }  void TestInit(){  TESTHEARD;  Heap heap;  HeapInit(&heap, Less);  printf("head -> size expect 0, actual %lu\n", heap.size);  printf("head -> cmp expect %p, actual %p\n", Less, heap.cmp);  }  void TestInsert(){  TESTHEARD;  Heap heap;  HeapInit(&heap, Less);  HeapInsert(&heap, 10);  HeapInsert(&heap, 12);  HeapInsert(&heap, 16);  HeapInsert(&heap, 8);  HeapInsert(&heap, 4);  HeapInsert(&heap, 2);  HeapPrintChar(&heap, "插入6个元素");  }  void TestRoot(){  TESTHEARD;  Heap heap;  HeapInit(&heap, Less);  HeapInsert(&heap, 10);  HeapInsert(&heap, 12);  HeapInsert(&heap, 16);  HeapInsert(&heap, 8);  HeapInsert(&heap, 4);  HeapInsert(&heap, 2);  HeapType value;  HeapRoot(&heap, &value);  printf("root expect 2, actual %d\n", value);  }  void TestErase(){  TESTHEARD;  Heap heap;  HeapInit(&heap, Less);  HeapInsert(&heap, 10);  HeapInsert(&heap, 12);  HeapInsert(&heap, 16);  HeapInsert(&heap, 8);  HeapInsert(&heap, 4);  HeapInsert(&heap, 2);  HeapErase(&heap);  HeapPrintChar(&heap, "Erase 1 个元素");  }  void TestSort(){  TESTHEARD;  int array[] = {0,1,4,2,54,32};  HeapSort2(array, sizeof(array)/sizeof(array[0]),Less);  size\_t i = 0;  for(; i < sizeof(array)/sizeof(array[0]); ++i){  printf("%d ", array[i]);  }  printf("\n");  }  int main(){  TestInit();  TestInsert();  TestRoot();  TestErase();  TestSort();  return 0;  }  #endif |

1. 该表信息将会作为你报名申请的重要依据，请认真仔细填写。
2. 培训班有严格的制度，请认真阅读规则并结合自身情况填写该表。
3. 完成该表填写后以“xxx学习情况”命名，及时上传作业。

最后希望大家能够加入我们，一起努力，共同进步！