|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』실습 보고서 | | | |
| 제목 | 9장 실습( O ) 과제( ) | 제출일자 | 2022.  11 .    23 . |
| 학번 | 201911608 | 이름 | 김 지환 |

|  |
| --- |
| 1. 프로그램 9.3 - 히프트리 테스트 프로그램  #include <stdio.h>  #include <stdlib.h>  #define MAX\_ELEMENT 200  typedef struct {  int key;  } element;  typedef struct {  element heap[MAX\_ELEMENT];  int heap\_size;  } HeapType;  HeapType\* create() {  return (HeapType\*)malloc(sizeof(HeapType));  }  void init(HeapType\* h) {  h->heap\_size = 0;  }  void insert\_max\_heap(HeapType\* h, element item) {  int i;  i = ++(h->heap\_size);  while ((i != 1) && (item.key > h->heap[i / 2].key)) {  h->heap[i] = h->heap[i / 2];  i /= 2;  }  h->heap[i] = item;  }  element delete\_max\_heap(HeapType\* h) {  int parent, child;  element item, temp;  item = h->heap[1];  temp = h->heap[(h->heap\_size)--];  parent = 1;  child = 2;  while (child <= h->heap\_size) {  if ((child < h->heap\_size) &&  (h->heap[child].key) < h->heap[child + 1].key)  child++;  if (temp.key >= h->heap[child].key) break;  h->heap[parent] = h->heap[child];  parent = child;  child \*= 2;  }  h->heap[parent] = temp;  return item;  }  int main(void) {  element e1 = { 10 }, e2 = { 5 }, e3 = { 30 };  element e4, e5, e6;  HeapType\* heap;  heap = create();  init(heap);  insert\_max\_heap(heap, e1);  insert\_max\_heap(heap, e2);  insert\_max\_heap(heap, e3);  e4 = delete\_max\_heap(heap);  printf("< %d > ", e4.key);  e5 = delete\_max\_heap(heap);  printf("< %d > ", e5.key);  e6 = delete\_max\_heap(heap);  printf("< %d > \n", e6.key);  free(heap);  return 0;  }  실행결과    -> 10, 5, 30 순서로 삽입하였지만 heap 구조로 최대 힙 정렬이 된다.  -> 출력시 30 -> 10 -> 5로 출력이 된다. |
| 2. Quiz(340쪽) 01 삽입 전과 후의 히프 트리를 실행하고 히프 그리기  01. 최대 히프가 다음과 같이 배열에 저장되어 있다. 여기에 11을 삽입하였을 경우에, 재구성된 히프를 그려라.  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #define MAX\_ELEMENT 200  #define logB(x, base) log(x)/log(base)  typedef struct {  int heap[MAX\_ELEMENT];  int heap\_size;  } HeapType;  HeapType\* create() {  return (HeapType\*)malloc(sizeof(HeapType));  }  void init(HeapType\* h) {  h->heap\_size = 0;  }  void insert\_max\_heap(HeapType\* h, int item) {  int i;  i = ++(h->heap\_size);  while ((i != 1) && (item > h->heap[i / 2])) {  h->heap[i] = h->heap[i / 2];  i /= 2;  }  h->heap[i] = item;  }  int delete\_max\_heap(HeapType\* h) {  int parent, child;  int item, temp;  item = h->heap[1];  temp = h->heap[(h->heap\_size)--];  parent = 1;  child = 2;  while (child <= h->heap\_size) {  if ((child < h->heap\_size) &&  (h->heap[child]) < h->heap[child + 1])  child++;  if (temp >= h->heap[child]) break;  h->heap[parent] = h->heap[child];  parent = child;  child \*= 2;  }  h->heap[parent] = temp;  return item;  }  int main(void) {  HeapType\* heap;  heap = create();  init(heap);  int arr[8] = { 6, 2, 12, 4, 5, 3, 10, 8 };    for (int i = 0; i < 8;i++)  insert\_max\_heap(heap, arr[i]);  printf("====기존 heap====\n");  for (int i = 1; i <= heap->heap\_size;i++) {  printf("%d ", heap->heap[i]);  }    printf("\n\n===========================재구성 heap========================\n");  insert\_max\_heap(heap, 11);  int i = 0, h = 1, node = 1;  while (heap->heap\_size != i++) {  if (h == 1) printf("\t\t\t\t");  if (h == 2) {  if (node == 1) printf("\t\t");  else printf("\t\t\t\t");  }  if (h == 4) {  if (node == 1) printf("\t");  else printf("\t\t");  }  if (h == 8) {  if (node != 1) printf("\t\t");  }  printf("%d ", heap->heap[i]);  if (node == h) {  printf("\n");  h <<= 1;  node = 0;  }  node++;  }  printf("\n");  free(heap);  return 0;  }  실행결과 |
| 3. Quiz(340쪽) 02 삭제 전과 후의 히프 트리를 실행하고 히프 그리기  02. 위의 최대 히프에서 우선 순위가 가장 높은 요소를 삭제하였을 경우에, 재구성된 히프 그리기  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #define MAX\_ELEMENT 200  #define logB(x, base) log(x)/log(base)  typedef struct {  int heap[MAX\_ELEMENT];  int heap\_size;  } HeapType;  HeapType\* create() {  return (HeapType\*)malloc(sizeof(HeapType));  }  void init(HeapType\* h) {  h->heap\_size = 0;  }  void insert\_max\_heap(HeapType\* h, int item) {  int i;  i = ++(h->heap\_size);  while ((i != 1) && (item > h->heap[i / 2])) {  h->heap[i] = h->heap[i / 2];  i /= 2;  }  h->heap[i] = item;  }  int delete\_max\_heap(HeapType\* h) {  int parent, child;  int item, temp;  item = h->heap[1];  temp = h->heap[(h->heap\_size)--];  parent = 1;  child = 2;  while (child <= h->heap\_size) {  if ((child < h->heap\_size) &&  (h->heap[child]) < h->heap[child + 1])  child++;  if (temp >= h->heap[child]) break;  h->heap[parent] = h->heap[child];  parent = child;  child \*= 2;  }  h->heap[parent] = temp;  return item;  }  int main(void) {  HeapType\* heap;  heap = create();  init(heap);  int arr[8] = { 6, 2, 12, 4, 5, 3, 10, 8 };    for (int i = 0; i < 8;i++)  insert\_max\_heap(heap, arr[i]);  printf("====기존 heap====\n");  for (int i = 1; i <= heap->heap\_size;i++) {  printf("%d ", heap->heap[i]);  }  printf("\n삭제된 요소 = %d\n", delete\_max\_heap(heap));    printf("\n\n===========================재구성 heap========================\n");  int i = 0, h = 1, node = 1;  while (heap->heap\_size != i++) {  if (h == 1) printf("\t\t\t\t");  if (h == 2) {  if (node == 1) printf("\t\t");  else printf("\t\t\t\t");  }  if (h == 4) {  if (node == 1) printf("\t");  else printf("\t\t");  }  if (h == 8) {  if (node != 1) printf("\t\t");  }  printf("%d ", heap->heap[i]);  if (node == h) {  printf("\n");  h <<= 1;  node = 0;  }  node++;  }  printf("\n");  free(heap);  return 0;  }  실행결과 |
| 4. 연습문제 13번 - 연산들 수행후의 최소 히프트리의 모습 그리기  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #define MAX\_ELEMENT 200  #define logB(x, base) log(x)/log(base)  typedef struct {  int heap[MAX\_ELEMENT];  int heap\_size;  } HeapType;  HeapType\* create() {  return (HeapType\*)malloc(sizeof(HeapType));  }  void init(HeapType\* h) {  h->heap\_size = 0;  }  void insert\_min\_heap(HeapType\* h, int item) {  int i;  i = ++(h->heap\_size);  while ((i != 1) && (item < h->heap[i / 2])) {  h->heap[i] = h->heap[i / 2];  i /= 2;  }  h->heap[i] = item;  }  int delete\_min\_heap(HeapType\* h) {  int parent, child;  int item, temp;  item = h->heap[1];  temp = h->heap[(h->heap\_size)--];  parent = 1;  child = 2;  while (child < h->heap\_size) {  if ((child <= h->heap\_size) &&  (h->heap[child]) > h->heap[child + 1])  child++;  if (temp < h->heap[child]) break;  h->heap[parent] = h->heap[child];  parent = child;  child \*= 2;  }  h->heap[parent] = temp;  return item;  }  void is\_empty(HeapType\* h) {  if (h->heap\_size == 0) printf("empty");  else printf("not empty");  }  int main(void) {  HeapType\* heap;  heap = create();  init(heap);  insert\_min\_heap(heap, 20);  insert\_min\_heap(heap, 12);  insert\_min\_heap(heap, 3);  insert\_min\_heap(heap, 2);  int d1, d2;  d1 = delete\_min\_heap(heap);  insert\_min\_heap(heap, 5);  insert\_min\_heap(heap, 16);  d2 = delete\_min\_heap(heap);  insert\_min\_heap(heap, 1);  is\_empty(heap);  printf("\n=========================== 최소 heap ========================\n");  int i = 0, h = 1, node = 1;  while (heap->heap\_size != i++) {  if (h == 1) printf("\t\t\t\t");  if (h == 2) {  if (node == 1) printf("\t\t");  else printf("\t\t\t\t");  }  if (h == 4) {  if (node == 1) printf("\t");  else printf("\t\t");  }  if (h == 8) {  if (node != 1) printf("\t\t");  }  printf("%d ", heap->heap[i]);  if (node == h) {  printf("\n");  h <<= 1;  node = 0;  }  node++;  }  printf("\n");  free(heap);  return 0;  }  실행결과 |