**Heap**

#include<stdio.h>

#include<malloc.h>

typedef struct heap {

int size; //the heap real size (number of items)

int capacity; //the heap total capacity

int\* data;

}heap;

void heapifyDown(heap\* h, int i) {

int largest = i;

int switched = 1;

while (i < h->size && switched) {

int right = i \* 2 + 2;

int left = i \* 2 + 1;

if (left < h->size && h->data[left] > h->data[i])

largest = left;

if (right < h->size && h->data[right] > h->data[i])

largest = right;

if (largest != i)

{

int temp = h->data[i];

h->data[i] = h->data[largest];

h->data[largest] = temp;

i = largest;

}

else

switched = 0;

}

}

void buildHeap(heap\* h)

{

for (int i = (h->size - 1 - 1) / 2; i >= 0; i--)

heapifyDown(h, i);

}

void heapInit(heap\* h) {

h->data = (int\*)malloc(sizeof(int) \* 10);

h->capacity = 10;

h->size = 0;

}

void heapInsert(heap\* h, int key) {

if (h->size == h->capacity)

{

h->capacity += 10;

h->data = (int\*)realloc(h->data,sizeof(int) \* h->capacity);

}

int i = h->size;

while (i>=1 && h->data[(i-1)/2] < key)

{

h->data[i] = h->data[(i - 1) / 2];

i = (i - 1) / 2;

}

h->data[i] = key;

h->size++;

}

void heapifyUp(heap\* h, int i) {

while (i && h->data[(i - 1) / 2] < h->data[i])

{

int temp = h->data[i];

h->data[i] = h->data[(i - 1) / 2];

h->data[(i - 1) / 2] = temp;

i = (i - 1) / 2;

}

}

**// החלק הזה עבור המיון המהיר בשביל HeapSort**

void swap(int\* i, int\* j) {

int temp = \*j;

\*j = \*i;

\*i = temp;

}

int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = low - 1;

for (int j = low; j <= high - 1; j++) {

if (arr[j] <= pivot) {

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]);

return (i + 1);

}

void quickSort(int arr[], int low, int high) {

if (low < high) {

int pi = partition(arr, low, high);

// נמיין את האיברים לפני ואחרי פרטישן

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

void heapSort(heap\* h) {

buildHeap(h);

int j = h->size;

quickSort(h->data, 0, h->size-1);

}

**main**

void main() {

heap h;

heap\* hptr = &h;

heapInit(hptr);

heapInsert(hptr, 6);

heapInsert(hptr, 4);

heapInsert(hptr, 18);

heapInsert(hptr, 11);

heapInsert(hptr, 85);

/\*-----------------------------\*/

heapInsert(hptr, 20);

heapInsert(hptr, 7);

heapInsert(hptr, 22);

heapInsert(hptr, 10);

heapInsert(hptr, 15);

heapInsert(hptr, 1);

heapInsert(hptr, 40);

heapInsert(hptr, 100);

for (int i = 0; i < hptr->size; i++)

{

printf("%d,", hptr->data[i]);

}

printf("\n");

heapSort(hptr);

for (int i = 0; i < hptr->size; i++)

{

printf("%d,", hptr->data[i]);

}