**Practica de Laboratorio de Objetos y Abstracción de Datos**

**Listas de Prioridad – HEAP**

Implemente su cola de prioridad con un heap binario, con el código iniciador mostrado abajo. Luego construir un heap para la siguiente lista: 18 25 41 34 14 10 52 50 48

heap.h

|  |
| --- |
| #ifndef PQueue\_Heap\_Included  #define PQueue\_Heap\_Included  using namespace std;  class Heap {  public:  /\* Construye una nueva cola de prioridad vacía respaldada por un heap binario. \*/  Heap();  /\* Limpia toda la memoria asignada por esta cola de prioridad. \*/  ~Heap();  /\* Retorna el número de elementos de la cola de prioridad. \*/  int size();  /\* Retorna si la cola de prioridad está o no vacía. \*/  bool isEmpty();  /\* Encola una nuevo elemento en la cola de prioridad. \*/  void enqueue(int value);  /\* Devuelve, pero no elimina, el primer elemento en la cola de prioridad.\*/  int peek();  /\* Devuelve y elimina el primer elemento de la cola de prioridades. \*/  int dequeue();  private:  /\* Almacena la longitud del array que contiene el heap.\*/  int allocatedLength;  /\* Almacena el número real de elementos en el heap.\*/  int n;  /\* Array que representa el heap.\*/  int\* T;  /\* Aumenta el tamaño del array cuando se necesita más espacio.\*/  void grow();  /\* Metodo para subir un nodo recién colocado para obtener un heap válido. \*/  void bubbleUp(int childPos);  /\* Metodo para descender para obtener un heap válido. \*/  void bubbleDown(int parentPos);  };  #endif |

heap.cpp

|  |
| --- |
| #include "heap.h"  #include <iostream>  /\* Starting size for the heap.\*/  const int DEFAULT\_SIZE = 9;  Heap::Heap() {  allocatedLength = DEFAULT\_SIZE;  n = 0;  T = new int[allocatedLength];  }  Heap::~Heap() {  delete[] T;  }  /\* Retorna el número de elementos de la cola de prioridad. \*/  int Heap::size() {  return n;  }  bool Heap::isEmpty() {  return (size() == 0);  }  /\* Aumenta el tamaño del array cuando se necesita más espacio. \*/  void Heap::grow() {  allocatedLength\*=2;  int\* newElems = new int[allocatedLength];  for (int i = 0; i < n; i++) {  newElems[i] = T[i];  }  delete[] T;  T = newElems;  }  /\* Devuelve, pero no elimina, el primer elemento en la cola de prioridad. \*/  int Heap::peek() {  if (isEmpty()) {  cout<<"Cola esta vacia.";  }  return T[0];  }  /\* Encola una nuevo elemento en la cola de prioridad. \*/  void Heap::enqueue(int value) {  }  /\* Devuelve y elimina el primer elemento de la cola de prioridades. \*/  int Heap::dequeue() {  }  /\* Metodo para subir un nodo recién colocado y obtener un heap válido. \*/  void Heap::bubbleUp(int childPos) {  }  /\* Metodo para descender y obtener un heap válido. \*/  void Heap::bubbleDown(int parentPos) {  } |