



# Estructuras de Datos

Grado en Informática  
Segundo Curso, segundo cuatrimestre  
Escuela Politécnica Superior de Córdoba  
Universidad de Córdoba  
Curso académico 2017-2018



---

## Práctica 3. Heap o montículo

- **Objetivo**
  - Implementación de un *heap* o **montículo** de mediciones de una estación meteorológica.
  - Se creará un montículo de **máximos** utilizando la **precipitación de lluvia** de cada día.
  - Se utilizará el montículo para cargar las mediciones de un fichero y luego escribirlas de forma ordenada en otro fichero diferente.
- **Desarrollo de la práctica número 3**
  - Duración de la práctica 3: tres sesiones de dos horas cada una.
  - **Plazo máximo de entrega**
    - **16:00 horas del 7 de mayo de 2018**
  - Se deberá subir un fichero comprimido denominado “**practica-3-usuario.zip**”, donde “usuario” es el *login* de cada estudiante.
  - El fichero comprimido contendrá
    - makefile
    - Doxyfile
    - ficheros hpp
    - ficheros cpp
    - ficheros txt de ejemplo
  - **Observación:**
    - Se debe usar el espacio de nombres de la asignatura: **ed**
- **Evaluación**
  - La calificación de la práctica se basará
    - en la calidad y completitud del trabajo realizado.
    - y en la **defensa presencial de cada estudiante**.
  - **Se valorará**
    - La correcta implementación de las **clases**
      - **Medicion**
      - **MonticuloMedicionesInterfaz**
      - **MonticuloMediciones**
    - La correcta codificación de las pre y post-**condiciones** mediante asertos.
    - La correcta codificación de las **funciones auxiliares** del programa principal.
    - El correcto funcionamiento del programa principal propuesto como ejemplo:
      - **principalMonticulo.cpp**
    - La codificación de **otro** programa principal con opciones diferentes.
    - La documentación del código con doxygen.

- La claridad del código.
- El uso de macros de pantalla para mejorar la visualización de la información
- **Y sobre todo**
  - Un profundo conocimiento de la práctica codificada.
- **Descripción de los ficheros**
  - **Introducción**
    - Se proporciona un fichero comprimido denominado “**practica-3-usuario.zip**” con los ficheros que se describen a continuación
    - **El Estudiante debe completar o revisar el código de los ficheros que se indican.**
  - **Fichero con datos de mediciones de precipitaciones de lluvia.**
    - Cordoba.txt
    - Este fichero es una modificación del fichero descargado desde la estación meteorológica de Córdoba, situada en la Alameda del Obispo:
      - [http://www.juntadeandalucia.es/agriculturaypesca/ifapa/ria/servlet/FrontController?action=Static&url=fechas.jsp&c\\_fecha=14&c\\_estacion=6](http://www.juntadeandalucia.es/agriculturaypesca/ifapa/ria/servlet/FrontController?action=Static&url=fechas.jsp&c_fecha=14&c_estacion=6)
  - **makefile**
    - Facilita la compilación de los ficheros, la generación de la documentación con doxygen y el borrado de ficheros que ya no sean necesarios (\*.o, \*~, etc.)
  - **Doxyfile**
    - Fichero de configuración para generar la documentación con doxygen
    - El Estudiante puede modificar este fichero para mejorar su documentación.
  - **Programas de prueba**
    - **principalMonticulo.cpp**
      - Programa principal utilizado como ejemplo para comprobar el funcionamiento de las clases **Medicion**, **MonticuloMediciones** y **MonticuloMedicionesInterfaz**.
    - **El Estudiante debe codificar otro programa de ejemplo.**
      - Debe utilizar macros de pantalla para mejorar la visualización de la información.
  - **macros.hpp**
    - Fichero con macros para mejorar la visualización de la información en la pantalla.
  - **funcionesAuxiliares.hpp**
    - Incluye los prototipos de funciones auxiliares utilizadas en el fichero **principalMonticulo.cpp**
    - **Importante**
      - El estudiante debe completar los comentarios de **doxygen**.
  - **funcionesAuxiliares.cpp**
    - Código complementario de las funciones auxiliares utilizadas en el programa principal.
    - **Importante**
      - El estudiante debe completar este código.

- **Fecha.hpp**
  - Definición de la clase **Fecha**
- **Fecha.cpp**
  - Código auxiliar de la clase **Fecha**.
- **Medicion.hpp**
  - Definición de la clase **Medicion**
  - **Importante**
    - El estudiante debe completar este código.
- **Medicion.cpp**
  - Código auxiliar de la clase **Medicion**
  - **Importante**
    - Cada estudiante debe completar este código.
- **MonticuloMedicionesInterfaz.hpp**
  - Definición de la clase abstracta **MonticuloMedicionesInterfaz**
  - **Importante**
    - Cada estudiante debe completar este código.
- **MonticuloMediciones.hpp**
  - Definición de la clase **MonticuloMediciones**
  - **Importante**
    - Cada estudiante debe completar este código
- **MonticuloMediciones.cpp**
  - Código de la clase **MonticuloMediciones**
  - **Importante**
    - Cada estudiante debe completar este código
- **Clases**
  - **Fecha**
    - Clase codificada por el profesor
  - **Medicion**
    - Código que debe codificar cada estudiante
  - **MonticuloMedicionesInterfaz**
    - Clase abstracta que debe ser codificada por cada estudiante
  - **MonticuloMediciones**
    - Clase que debe codificar cada estudiante
- **Especificación de las clases**
  - Clase **Fecha**
    - **Atributos**
      - día, mes y año de tipo Entero.
    - **Constructores**
      - **Constructor parametrizado con valores por defecto**
        - *Fecha(día= 1:Entero; mes= 1:Entero; año= 1:Entero)*
          - Postcondición

- *getDía() == día*
  - *getMes() == mes*
  - *getAño() == año*
- **Constructor de copia**
  - ***Fecha(fecha: Fecha)***
    - Postcondición
      - *getDía() == fecha.getDía()*
      - *getMes() == fecha.getMes()*
      - *getAño() == fecha.getAño()*
- **Observador privado**
  - Lógico ***esBisiesto()***
    - Comprueba si es bisiesto el año de la fecha actual.
- **Observadores públicos**
  - Entero ***getDía()***
    - Devuelve el día de la fecha.
  - Entero ***getMes()***
    - Devuelve el mes de la fecha.
  - Entero ***getAño()***
    - Devuelve el año de la fecha.
  - Lógico ***esCorrecta()***
    - Comprueba si la fecha es correcta.
- **Modificadores públicos**
  - ***setDía(día: Entero)***
    - Asigna un nuevo día a la fecha.
    - Postcondición
      - *getDía() == día*
  - ***setMes(mes: Entero)***
    - Asigna un nuevo mes a la fecha.
    - Postcondición
      - *getMes() == mes*
  - ***setAño(año: Entero)***
    - Asigna un nuevo año a la fecha.
    - Postcondición
      - *getAño() == año*
- **Operadores públicos**
  - **Operador de igualdad**
    - Lógico ***operador == (fecha: Fecha)***
      - Compara dos fechas.
      - Postcondición
        - *valorDevuelto == (getDía() == fecha.getDía()) y (getMes() == fecha.getMes()) y (getAño() == fecha.getAño())*
  - **Operador de asignación**

- *Fecha* **operador** = (*fecha: Fecha*)
      - Devuelve la fecha actual modificada con los atributos de otra fecha
      - Postcondición
        - *getDía()* == *fecha.getDía()*
        - *getMes()* == *fecha.getMes()*
        - *getAño()* == *fecha.getAño()*
  - **Funciones de lectura y escritura públicas**
    - *leerFecha()*
      - Lee una fecha desde el teclado.
    - *escribirFecha()*
      - Escribe la fecha por la pantalla con el formato día-mes-año.
  - **Funciones externas a la clase Fecha**
    - **Sobrecarga del operador de entrada**
      - Lee desde el flujo de entrada los atributos de la fecha
      - Prototipo de C++
        - *istream &operator>>(istream &stream, Fecha &fecha);*
    - **Sobrecarga del operador de salida**
      - Escribe en el flujo de salida los atributos del municipio:
      - Prototipo de C++
        - *ostream &operator<<(ostream &stream, Fecha const &fecha);*
- **Clase Medición**
  - **Atributos**
    - *fecha*: dato de tipo *Fecha*
    - *precipitación de lluvia*: dato de tipo *Real* no negativo.
  - **Constructores**
    - **Constructor parametrizado con valores por defecto**
      - *Medición(fecha=Fecha(1,1,1): Fecha; precipitación=0.0:Real)*
        - Postcondición
          - *getFecha()* == *fecha*
          - *getPrecipitación()* == *precipitación*
    - **Constructor de copia**
      - *Medición(medición:Medición)*
        - Postcondición
          - *getFecha()* == *medición.getFecha()*
          - *getPrecipitación()* == *medición.getPrecipitación()*
  - **Observadores públicos**
    - *Entero getFecha()*
      - Devuelve la fecha de la medición.
    - *Entero getPrecipitación()*
      - Devuelve la precipitación de lluvia de la medición.
  - **Modificadores públicos**

- **setFecha**(fecha: Fecha)
  - Asigna una nueva fecha a la medición.
  - Postcondición
    - *getFecha() == fecha*
- **setPrecipitación**(precipitación: Real)
  - Asigna una nueva precipitación de lluvia a la medición.
  - Postcondición
    - *getPrecipitación() == precipitación*
- **Operadores públicos**
  - **Operador de igualdad**
    - Lógico **operador** == (medición: Medición)
      - Compara las fechas de dos mediciones.
      - Postcondición
        - *valorDevuelto == (getFecha() == medición.getDía())*
  - **Operador de asignación**
    - Fecha **operador** = (medición: Medición)
      - Devuelve la medición actual modificada con los atributos de otra medición.
      - Postcondición
        - *getFecha() == medición.getFecha()*
        - *getPrecipitación() == getPrecipitación()*
- **Funciones de lectura y escritura públicas**
  - **leerMedición()**
    - Lee una medición desde el teclado.
  - **escribirMedición()**
    - Escribe la medición en la pantalla con el formato:
      - *día-mes-año precipitación*
- **Funciones externas a la clase Medición**
  - **Sobrecarga del operador de entrada**
    - Lee desde el flujo de entrada los atributos de la medición
    - Prototipo de C++
      - *istream &operator>>(istream &stream, **Medición** &medicion);*
  - **Sobrecarga del operador de salida**
    - Escribe en el flujo de salida los atributos del municipio:
    - Prototipo de C++
      - *ostream &operator<<(ostream &stream, **Medición** const &medicion);*
- **Clase MonticuloMedicionesInterfaz**
  - **Observación**
    - Clase abstracta porque tiene métodos virtuales puros
  - **Observadores públicos**
    - Lógico **isEmpty()**
      - Postcondición

- *Devuelve verdadero si no tiene mediciones; falso, en caso contrario.*
- **Medición *top()***
  - Devuelve la cima del montículo
  - Precondición
    - *isEmpty() == falso*
- **Modificadores públicos**
  - ***insert(medición: Medición)***
    - Inserta una nueva medición en el montículo
    - Postcondición
      - *isEmpty() == falso*
  - ***remove()***
    - Borra la medición que ocupa la cima
    - Precondición
      - *isEmpty() == falso*
- **Clase MonticuloMediciones**
  - **Observación**
    - Clase que hereda de forma pública de la clase abstracta [MonticuloMedicionesInterfaz](#)
  - **Atributo**
    - Atributo que almacena las mediciones
  - **IMPORTANTE**
    - Cada estudiante deberá elegir la representación que desee para almacenar las mediciones en el montículo.
    - alguna de las opciones son:
      - Versión acotada:
        - vector (array) de mediciones con un tamaño fijo preestablecido (el tamaño deberá ser superior o igual a 366, número de días de un año bisiesto).
      - Versión no acotada:
        - vector STL de mediciones.
  - **Observadores privados**
    - ***Medición getElement(i:Entero)***
      - *Devuelve el elemento que ocupa la posición i en el vector que almacena el montículo*
      - Precondición
        - *(i >= 0) y (i < size())*
    - ***setElement(i:Entero; medición: Medición)***
      - *Modifica el elemento que ocupa la posición i en el vector que almacena el montículo*
      - Precondición
        - *(i >= 0) y (i < size())*
      - Postcondición
        - *getElement(i) == medición*

- **Entero *getLeftChild*(i:Entero)**
  - Devuelve el índice del hijo izquierdo del índice recibido.
  - *Precondición*
    - $i \geq 0$
  - *Postcondición*
    - $\text{valorDevuelto} == 2 * i + 1$
- **Entero *getRightChild*(i:Entero)**
  - Devuelve el índice del hijo derecho del índice recibido.
  - *Precondición*
    - $i \geq 0$
  - *Postcondición*
    - $\text{valorDevuelto} == 2 * i + 2$
- **Entero *getParent*(i:Entero)**
  - Devuelve el índice del padre del índice recibido.
  - *Precondición*
    - $i \geq 1$
  - *Postcondición*
    - $\text{valorDevuelto} == (i - 1) / 2$
- ***shiftUp*(i:Entero)**
  - El elemento indicado por el índice es subido en el montículo hasta que se verifica la ordenación de máximos.
  - *Precondición*
    - $(i \geq 0)$  y  $(i < \text{size}())$
  - *Postcondición*
    - Si no es la cima, el elemento actual es menor o igual que su padre
    - Si tiene hijo izquierdo, el elemento actual es mayor o igual que él y, además, si tiene hijo derecho, es mayor o igual que él.
    - **Importante**
      - Cada estudiante debe incluir el código de los asertos
- ***shiftDown*(i:Entero)**
  - El elemento indicado por el índice es bajado en el montículo hasta que se verifica la ordenación de máximos.
  - *Precondición*
    - $(i \geq 0)$  y  $(i < \text{size}())$
  - *Postcondición*
    - Si no es la cima, el elemento actual es menor o igual que su padre
    - Si tiene hijo izquierdo, el elemento actual es mayor o igual que él y, además, si tiene hijo derecho, es mayor o igual que él.
    - **Importante**
      - Cada estudiante debe incluir el código de los asertos
- **Lógico *has*(medición: Medición)**
  - Se comprueba si la medición está incluida en el montículo.



- **Constructor público**
  - **Constructor sin argumentos**
    - *MonticuloMediciones()*
      - Postcondición
        - *isEmpty() == verdadero*
- **Observadores públicos**
  - Lógico *isEmpty()*
    - Postcondición
      - *valorDevuelto == (size() == 0)*
  - Entero *size()*
    - Devuelve el número de mediciones del montículo
  - Medicion *top()*
    - Devuelve la cima del montículo
    - Precondición
      - *isEmpty() == falso*
    - Postcondición
      - *valorDevuelto == getElement(0)*
- **Modificadores públicos**
  - *insert(medición: Medicion)*
    - Inserta una nueva medición en el montículo
    - Postcondición
      - *isEmpty() == falso*
      - *has(medición) == verdadero*
  - *remove()*
    - Borra la medición que ocupa la cima
    - Precondición
      - *isEmpty() == falso*
  - *removeAll()*
    - Borra todas la mediciones del montículo
    - Postcondición
      - *isEmpty() == verdadero*
  - *modify(medición: Medicion)*
    - Modifica la medición que ocupa la cima y actualiza el montículo para que esté ordenado.
    - Precondición
      - *isEmpty() == false*
    - Postcondición
      - *has(medición) == true*
- **Operadores públicos**
  - **Operador de asignación**
    - *MonticuloMediciones operador = (m: MonticuloMediciones)*
      - Devuelve el montículo actual que ha sido modificado con las mediciones del montículo “m”.

- Precondición
  - Los montículos actual y “m” no son el mismo objeto
- **Función de escritura pública**
  - *print()*
    - Escribe las mediciones tal y como están almacenadas en el vector que representa el montículo.