# Pontificia Universidad Javeriana
# Departamento de Ingeniería de Sistemas
# Estructuras de Datos
# Taller 4: Árboles de Codificación, 2015-30

## 1. Objetivo

Utilizar árboles binarios para representar la codificación de mensajes usando el código Morse. En particular, se desea evaluar las habilidades del estudiante en el recorrido del árbol que permita realizar la codificación y decodificación de mensajes a partir del código Morse.

## 2. Recordatorio: compilación con g++

La compilación con g++ (compilador estándar que será usado en este curso para evaluar y calificar las entregas) se realiza con los siguientes pasos:

1. **Compilación**: de todo el código fuente compilable (**ÚNICAMENTE LOS ARCHIVOS CON EXTENSIONES** `*.c`, `*.cpp`, `*.cxx`)
   `g++ -c *.c *.cxx *.cpp`
2. **Encadenamiento**: de todo el código de bajo nivel en el archivo ejecutable
   `g++ -o nombre_de_mi_programa *.o`

   Nota: Estos dos pasos (compilación y encadenamiento) pueden abreviarse en un sólo comando:
   `g++ -o nombre_de_mi_programa *.c *.cxx *.cpp`
3. **Ejecución**: del programa ejecutable anteriormente generado
   `./nombre_de_mi_programa`

**ATENCIÓN**: Los archivos de encabezados (`*.h`, `*.hpp`, `*.hxx`) **NO SE COMPILAN**, se incluyen en otros archivos (encabezados o código). Así mismo, los archivos de código fuente (`*.c`, `*.cpp`, `*.cxx`) **NO SE INCLUYEN**, se compilan. Si el programa entregado como respuesta a este Taller no atiende estas recomendaciones, automáticamente se calificará la entrega sobre un $25\,\%$ menos de la calificación máxima.

## 3. Descripción del problema

En las primeras épocas del desarrollo de las comunicaciones, se desarrolló el código Morse (1836) para la transmisión de mensajes de manera rápida y eficiente. Cada letra, número y algunos caracteres especiales se codifican utilizando una secuencia estandarizada de señales cortas y largas conocidas como puntos y rayas, y esta codificación es única para cada caracter. Dado el avance de las telecomunicaciones, el código Morse se utiliza poco en la actualidad, sin embargo, aún se considera el método más simple y versátil de transmisión de mensajes [1].

La escogencia de la secuencia de puntos y rayas que debía representar cada caracter no se realizó de manera aleatoria. Como el objetivo era la transmisión rápida de mensajes, se analizó la frecuencia de ocurrencia de cada uno de los caracteres en los textos en inglés, y las letras más frecuentes se codificaron con secuencias cortas, haciendo así más rápida y eficiente la transmisión. La Figura 1 representa la codificación de las letras del alfabeto usando el código Morse, donde las letras más frecuentes se encuentran en niveles superiores del árbol, y las menos frecuentes se encuentran en las hojas del árbol.

En este árbol, cada nodo almacena un símbolo del alfabeto de 26 letras utilizado en la transmisión de un mensaje en clave Morse. Este árbol se utiliza para codificar y decodificar los símbolos de un mensaje. Cada arista entre un nodo padre y sus hijos se etiqueta con un ‘‘.’’ para el hijo izquierdo y con un ‘‘-’’ para el hijo derecho. Cada camino entre el nodo raíz y los nodos con símbolos se puede representar como la concatenación de las etiquetas de las aristas que lo conforman. Esta representación corresponde a la codificación para cada uno de los símbolos. En este caso, por ejemplo, para el símbolo ‘‘C’’ su codificación es ‘‘-.-.’’, el cual corresponde a las etiquetas de las aristas del camino desde la raíz hasta el nodo con el símbolo ‘‘C’’. Así mismo, la ‘‘S’’ se codifica como ‘‘...’’, la ‘‘O’’ se codifica como ‘‘---’’ y la ‘‘Z’’ se codifica como ‘‘--..’’.

[1] https://en.wikipedia.org/wiki/Morse_code

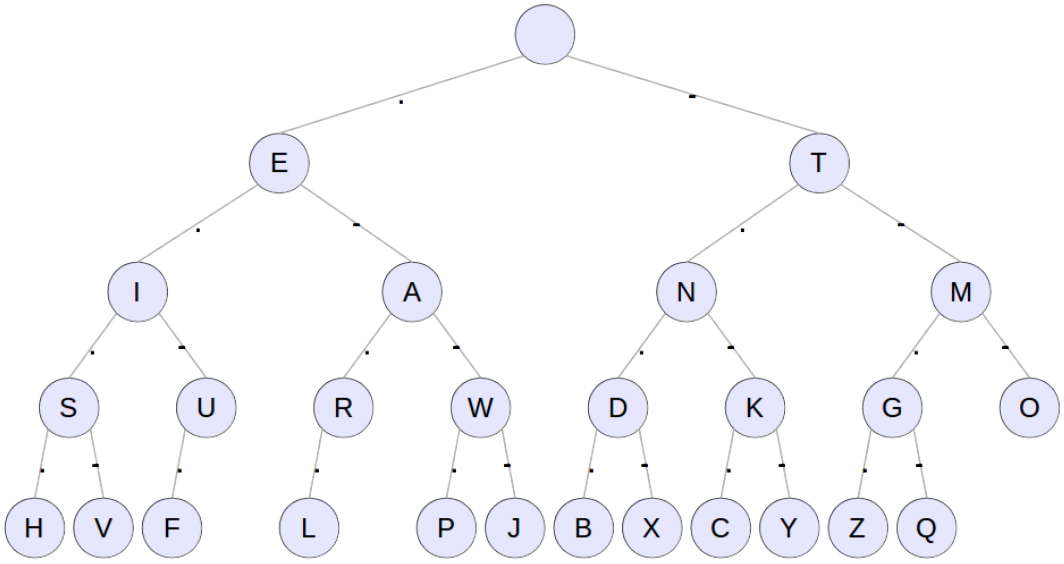


Figura 1: Árbol binario con la representación del código Morse

## 4. Descripción de la implementación

Para resolver el problema de codificación y decodificación de mensajes utilizando el código Morse, se propone una implementación contenida en los archivos fuente ‘‘`taller_4_codificacion.cxx`’’, ‘‘`MorseTree.h`’’ y ‘‘`MorseTree.cxx`’’. El programa implementado provee al usuario los siguientes comandos:

- `load <archivoMorse.txt>`
  El sistema abre el archivo `<archivoMorse.txt>`, que contiene la tabla de equivalencias entre los 26 símbolos del alfabeto y su representación en el código Baudot, y construye el árbol binario correspondiente que permitirá hacer posteriormente la codificación de los mensajes. Si la carga del archivo fue exitosa, el sistema genera una imagen PNG con un dibujo del árbol generado, el cual puede usarse para una inspección visual de la información almacenada en el árbol. Si se presenta algún problema, ya sea en la lectura del archivo o en la construcción del árbol, el programa genera en pantalla un mensaje de error.

- `encode <cadena_mensaje>`
  El sistema toma la cadena de caracteres `<cadena_mensaje>` (la cual puede incluir espacios como parte del mensaje), y realiza el proceso de codificación caracter a caracter. Si el caracter no se encuentra en el árbol (como por ejemplo el caracter espacio), no se codifica. Para organizar la presentación en pantalla de la codificación del mensaje, las letras de una misma palabra se separan con un espacio, mientras que las palabras se separan con un cambio de línea. Por ejemplo, si el mensaje a codificar es ‘‘`Hola buenos dias`’’, su codificación en pantalla debe aparecer así:
  ```
  .... --- .-.. .-
  -... ..- . -. --- ...
  -.. .. .- ...
  ```
  En caso de que el comando de codificación se invoque antes de cargar el árbol en memoria, el programa genera en pantalla un mensaje de error.

- `decode <cadena_codigos>`
  El sistema toma la cadena de caracteres `<cadena_codigos>` (la cual contiene códigos de puntos y guiones separados por espacios), y realiza el proceso de decodificación código a código. Por ejemplo, si la codificación dada es ‘‘`-... ..- . -. --- ...`’’, su decodificación en pantalla corresponde a:
  `BUENOS`
  En caso de que el comando de decodificación se invoque antes de cargar el árbol en memoria, el programa genera en pantalla un mensaje de error.

- `exit`
  El comando termina la ejecución del programa, liberando la memoria asociada al árbol de codificación y a cualquier otra estructura o variable creada con memoria dinámica.

# 5. Desarrollo del taller

El desarrollo del taller consistirá en diseñar e implementar las funcionalidades de construcción del árbol de códigos Morse, así como la codificación y decodificación de un mensaje dado usando ese código. El diseño puede consistir de diagramas de flujo, conjuntos de pasos, esquemáticos, etc. En cualquier caso, debe quedar evidencia física de que analizó y diseñó cada proceso previamente a la implementación en C++. La implementación implica traducir el proceso diseñado a código C++, completando las secciones marcadas con el comentario `// TODO #n` en el archivo **‘‘**`MorseTree.cxx`**’’**.

Para completar el taller, realice las siguientes actividades:

1. Estudie los archivos de código fuente entregados. Asegúrese de entender completamente los procesos allí descritos. Si encuentra alguna porción de código que no entienda, pregunte al profesor o al monitor de la clase.
2. Compile el programa y ejecútelo. Verifique que no se generan errores ni advertencias al momento de compilar.
3. **(17 %)** Diseñe el algoritmo o proceso necesario para construir el árbol de códigos Morse a partir del archivo **‘‘**`morseCode.txt`**’’**.
4. **(18 %)** Diseñe el algoritmo o proceso necesario para la decodificación de un mensaje utilizando el árbol de códigos Morse.
5. **(21 %)** Diseñe el algoritmo o proceso necesario para la codificación de un mensaje utilizando el árbol de códigos Morse.
6. **(13 %)** Implemente en C++ el algoritmo o proceso diseñado para la construcción del árbol de códigos Morse dentro de la función `AddCharacter` en el archivo **‘‘**`MorseTree.cxx`**’’** (marcada con el comentario `// TODO #1`).
7. Verifique que la carga del árbol, utilizando el archivo **‘‘**`morseCode.txt`**’’**, es exitosa. Para esto, compile y ejecute el programa, haga el llamado al comando `load` con el archivo dado, y verifique la imagen PNG generada del árbol.
8. **(14 %)** Implemente en C++ el algoritmo o proceso diseñado para la decodificación de un mensaje dentro de la función `Decode` en el archivo **‘‘**`MorseTree.cxx`**’’** (marcada con el comentario `// TODO #2`).
9. **(17 %)** Implemente en C++ el algoritmo o proceso diseñado para la codificación de un mensaje dentro de la función `Encode` en el archivo **‘‘**`MorseTree.cxx`**’’** (marcada con el comentario `// TODO #3`).
10. Compile nuevamente el programa para incluir las nuevas implementaciones y ejecútelo. Verifique con diferentes mensajes que la codificación y decodificación usando el código Morse funciona correctamente.

# 6. Evaluación

Como parte del desarrollo del taller, se debe entregar:

1. Informe escrito que incluya el diseño de los procesos de carga de información, codificación y decodificación de mensajes usando el código Morse.
2. Código fuente modificado y funcional, que implementa los procesos de carga de información, codificación y decodificación diseñados.

La entrega se hará a través de la correspondiente actividad de UVirtual, antes de finalizar la sesión de clase del viernes 2 de octubre (11:00a.m.). Se debe entregar un único archivo comprimido **(únicos formatos aceptados: .zip, .tar, .tar.gz, .tar.bz2, .tgz)** que contenga dentro de un mismo directorio **(sin estructura de carpetas interna)**, documentos **(único formato aceptado: .pdf)** y código fuente **(únicos formatos aceptados: .h, .hxx, .hpp, .c, .cxx, .cpp)**. Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

La evaluación del taller tendrá la siguiente escala para cada diseño o secciones de código a completar:

- **Excelente** (**5.0/5.0**): El código es correcto o el diseño tiene una calidad adecuada.
- **Bueno** (**3.0/5.0**): El código es parcialmente correcto o el diseño no tiene una calidad suficiente para ser un trabajo de ingeniería.
- **Necesita mejoras sustanciales** (**2.0/5.0**): El código no es correcto o el diseño no es adecuado.
- **No entregó** (**0.0/5.0**).