**Solución del problema**

Para diseñar e implementar la aplicación, se tomaron en cuenta algunas características básicas que debían realizarse inicialmente. En esta sección explicaremos cronológicamente las fases que se fueron implementando y la manera en que se hicieron, para dar a entender más claramente cómo se llegó al resultado esperado.

Inicialmente partimos del hecho de la necesidad de leer un archivo3, para lo cual era necesario plantear primeramente la forma en que estaría estructurado. Para que el archivo fuera de texto plano y más entendible para las personas que diseñaran o vieran el formato, fue necesario seleccionar varios caracteres especiales, los cuales funcionarían como separadores de secciones. Finalmente determinamos que una buena opción sería utilizar las comas (,) y los punto y comas (;) para separar atributos y actividades, respectivamente. Cada actividad debía almacenar información sobre su nombre, tiempo de duración y actividades predecesoras. Considerando que la cantidad de actividades que podían haber antes de cualquiera era indefinida, utilizamos el siguiente formato:

nombreActividad,tiempo,nomrePredecesor1,nombrePredecesor2,...,nombrePredecesorN;

Para señalar cuales son los predecesores de cada actividad se utiliza el nombre de cada una de ellas. El símbolo de punto y coma indica que ahí termina la actividad. Tomando esta estructura en consideración, diseñamos un algoritmo que leyera el contenido de texto del archivo dado y que lo subdividiera por caracteres separadores y que se creara una estructura de actividad para ser insertada en la lista.

Ahora bien, necesitamos implementar varias estructuras1 dentro de un grafo, las cuales fueron la de vértice, arco y subarco. El vértice marca los eventos, es decir, el inicio o fin de una o más actividades. El arco describe una actividad con sus propiedades y los subarcos son simples contenedores utilizados para crear sublistas de arcos.

|  |  |
| --- | --- |
| Estructura | Propiedades o atributos |
| Grafo | Vértice inicial, Primer actividad, Total de vértices, Ruta crítica |
| Vértice | ID, ET, Arcos Predecesores, Arcos Antecesores. |
| Arco | Nombre, Tiempo, Destino |
| Subarco | Arco Destino |

Estructuras de datos utilizados

Una vez que fue posible reconocer la información representada por los archivos y estructurarla como actividades (arcos) procedimos a realizar un algoritmo que permitiera insertar la actividad según la posición correspondiente, la cual dependía de los predecesores que tuviera la actividad. Esta función fue un tanto extensa ya que era necesario considerar varias opciones cuando se insertaba en el grafo. Algunos ejemplos: Era la primera actividad, no tenía predecesores, los tenía pero eran compartidos, etc.

Eventualmente fue posible crear la estructura correctamente, por lo que restaba hacer uso de los algoritmos de PERT y CPM, y de los recorridos por Anchura (Amplitud) y Profundidad. Estos primeros debían calcular el Early Time de cada evento, variando solamente en el cálculo del tiempo de cada actividad. CPM es directo y utiliza el tiempo dado para cada actividad5, por lo que no fue necesario agregar más que un método que recorriera el grafo por cada ruta posible y marcara los eventos con los tiempos menores con los que se llegaba a cada uno. Por otro lado, PERT necesita utilizar una fórmula que considera el tiempo optimista y pesimista4, así como el tiempo estimado. Como referencia, mostramos la fórmula:

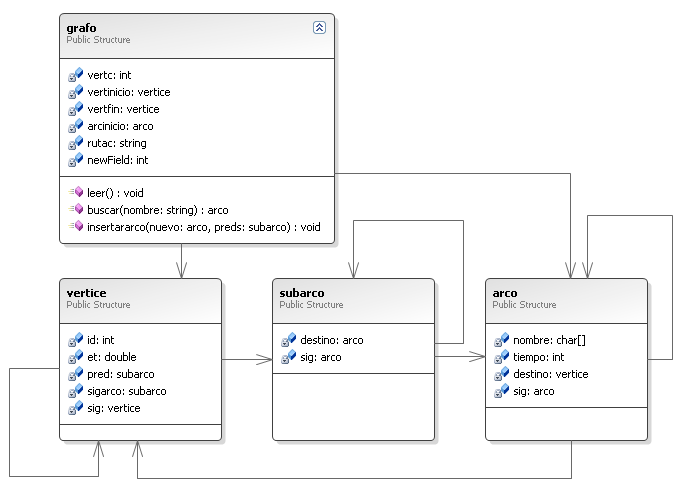
Donde a y b son los tiempos optimista y pesimista y m el tiempo más probable.

Junto a estos método, se encuentra el de calcular la ruta crítica de cada grafo. Para implementarla decidimos recorrer por profundidad cada vértice pero solo por los arcos que, sumando su tiempo y el tiempo acumulado, obtuvieran el ET del vértice asociado a su destino. Al final, solo podrá llegarse al final de una forma y se almacena la información de los arcos visitados.

Luego debíamos implementar los métodos de recorrido e impresión del grafo. Iniciamos con el método por amplitud, pues era más sencillo. Simplemente recorremos la lista general de eventos o vértices y en cada uno revisamos sus arcos sucesores. Imprimimos tanto la información de cada evento como la de cada actividad.

Después del primer método de recorrido, implementamos el método por profundidad. Un algoritmo recursivo que se ramifica por cada ruta posible del grafo. Por cada arco visitado se muestra la información pertinente al mismo, tomando en cuenta el evento que lo origina, el vértice destino, el tiempo de duración y, por supuesto, el nombre de la actividad.

**Diagrama de clases**



**Literatura citada**

1. Estructuras de datos, Salvador Pozo Coronado. Consultado el 10 de abril de 2011. Explicación general y superficial de varias estructuras de datos. <http://c.conclase.net/edd/>.
2. Operator New. Consultado el 11 de abril de 2011. Explica el almacenamiento dinámico de instancias, utilizando el operador “new” del lenguaje.

<http://www.cplusplus.com/reference/std/new/operator%20new/>

1. Input/Output with files. Consultado el 12 de abril de 2011. Manejo de archivos con la librería “fstream”, operaciones de posicionamiento, lectura y escritura. <http://www.cplusplus.com/doc/tutorial/files/>
2. El método PERT. Consultado el 24 de abril de 2011. Manejo de grafos utilizando el algoritmo PERT. <http://ocw.upm.es/proyectos-de-ingenieria/proyectos-de-desarrollo-rural-i/Materiales-de-cada-tema/Planificacion-de-proyectos.-Metodo-PERT.pdf>
3. Project Management FAQ. Consultado el 24 de abril de 2011. Varios métodos de recorrido y cálculo de tiempos usando los algoritmos PERT y CPM, entre otros. <http://www.codeproject.com/KB/aspnet/ProjectManagementFAQ.aspx>