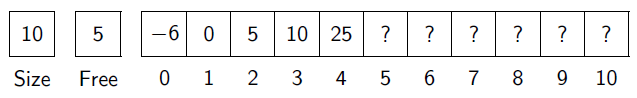
# Estudio de caso de una cola de prioridad (priority queue)

Este estudio de caso es la implementación de una cola de prioridad, primero usando arreglos y después usando apuntadores. Una cola de prioridad es una estructura de datos que almacena elementos de tal manera que la obtención del elemento de más alta prioridad pueda ser hecha eficientemente, aunque la inserción de nuevos elementos podría ser menos eficiente. En el estudio de caso, suponemos que los elementos son simplemente enteros y que los elementos de prioridad más alta tienen los valores más pequeños. Esta es una situación común: los clientes de una farmacia toman tikets numerados y el cliente con el tiket de número más pequeño es atendido primero.

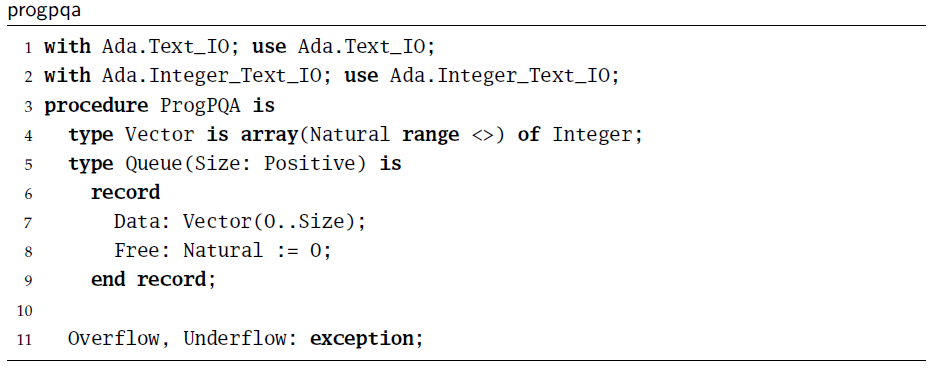
## Estudio de caso: cola de prioridad basada en arreglo

Los elementos de la cola de prioridad son almacenados en un arreglo cuyo índice inferior es 0 (la frontera inferior del subtipo Natural) y cuyo índice superior es determinado cuando se asigna memoria al arreglo. La cola misma es un registro con tres campos: (a) un array field Data, (b) Size, el cual contiene el tamaño del arreglo, y (c) Free, el cual guarda un índice a la primera posición libre en el arreglo:



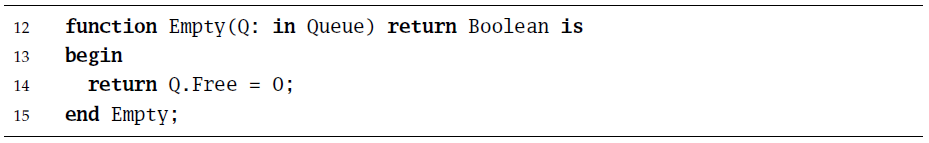
Los elementos de la cola son almacenados en orden creciente como se muestra en el diagrama. El elemento con la más alta prioridad (el valor más pequeño) es almacenado en la primera posición y puede ser obtenido en tiempo constante. Insertar un nuevo elemento es necesariamente menos eficiente porque se debe buscar el lugar correcto para insertar el elemento y los elementos que ya estaban presentes deben ser cambiados de lugar para liberar espacio.

La estructura de datos es declarada como un registro (record) con los tres componentes. El componente Size es un discriminante, el cual es usado para especificar la restricción de los índices del componente Data. Un discriminante es un componente de solo lectura cuyo valor es proporcionado por una restricción cuando se declara un objeto record.



El componente Free tiene una expresión por defecto, la cual es el valor inicial dado a un componente cuando un objeto record es declarado. Se han declarado dos excepciones: Underflow será lanzada cuando se intente obtener un elemento de una cola vacia y Overflow será lanzada cuando se intente poner un elemento en una cola llena.

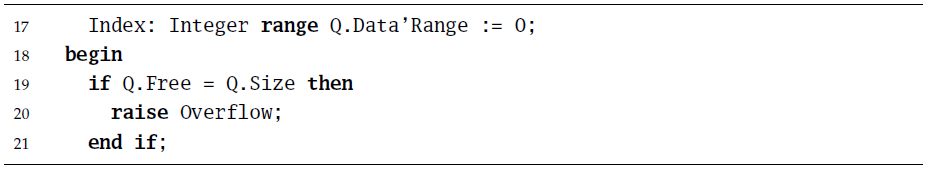
Las operaciones soportadas por la cola son: obtener el número mínimo, poner un nuevo número en la cola, y revisar si la cola está vacía. La función Empty es trivial:



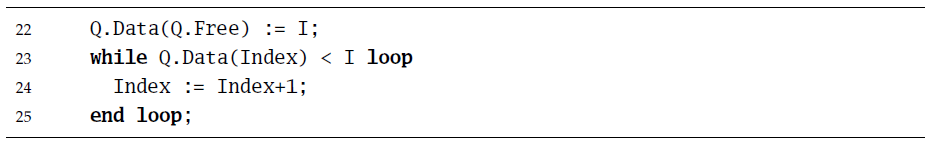
Dado que la Queue es modificada por la operación Put su modo de parámetro debe ser in out:



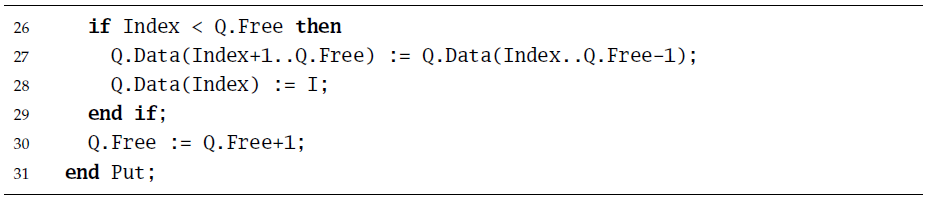
El procedimiento Put primero revisa si hay espacio en la cola para más elementos; si no es así, ejecuta la sentencia raise para la excepción Overflow. La excepción es propagada al procedimiento principal y es manejada, como se discute abajo.



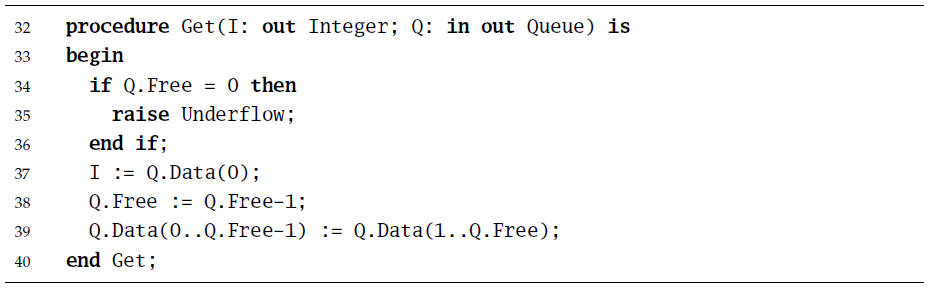
Los elementos existentes en la cola están almacenados en Q.Data(0..Q.Free - 1). Una búsqueda con centinela se realiza para encontrar el lugar donde poner el nuevo elemento. Data, el componente arreglo de la cola, tiene Size + 1 elementos, así que siempre hay un lugar para almacenar el centinela. El nuevo elemento I es colocado en Q.Data(Free) antes de comenzar la búsqueda de un elemento mayor o igual que I. El centinela asegura que incluso si I es más grande que todos los elementos existentes, el ciclo terminará.



Cuando el lugar correcto ha sido encontrado, los elementos subsecuentes son movidos a la derecha a fin de liberar una posición para el nuevo elemento. Se usan Slices y Sliding.

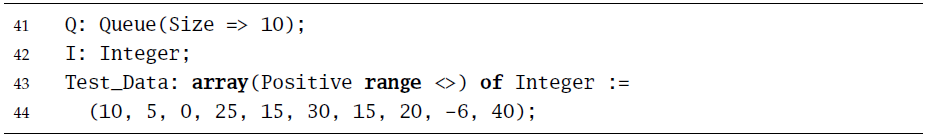


El elemento con la prioridad más alta está siempre en la primera posición del arreglo. Para removerlo, simplemente copie el último elemento de la izquierda y mueva los otros elementos un lugar hacia la izquierda:

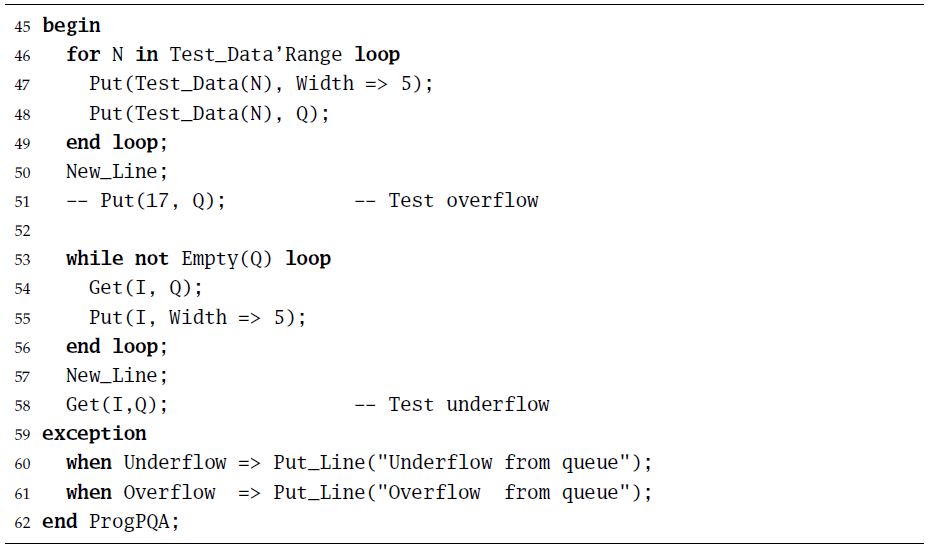


Get se implementa como un procedimiento y no como una función, porque una función solo tiene permitidos parámetros in. Una solución alternativa es usar parámetros access.

Para probar el programa, declaramos una cola con diez elementos y un arreglo con datos de prueba ordenados arbitrariamente:



Esos diez elementos son insertados en la cola y después son removidos:



El programa es correcto si los elementos son removidos en orden creciente.

El manejador de excepción al final del procedimiento imprime mensajes y termina. Para probar el overflow, remueva el comentario de la línea 51 y recompile.