Proyecto 2: Estructuras de Datos Lineales usando vectores de STL.

Prof. Luis Ernesto Garreta U.

17 de octubre de 2017

1. Objetivo

El objetivo del proyecto es utilizar la clase vector de la librería STL para implementar tres tipos de estructuras lineales: pilas, colas y listas. De está manera aplica los conocimientos de POO a través del uso de la clase *vector* de STL y también empieza a familiarizarse con estas tres estructuras que más adelante tendrá que implementarlas completamente.

2. Conceptos

Los vectores (o arreglos) son secuencias de elementos donde cada componente tiene un único sucesor y un único predecesor con excepción del último y el primero. El acceso se realiza a través de un índice que indica la posición del elemento dentro de la estructura. Dependiendo de si el arreglo es dinámico o estático, se podrán adicionar elementos al final o no, sin gran costo computacional. Algunas implementaciones de *vector* manejan operaciones de adición de elementos tanto al inicio como al final, pero internamente adicionar al inicio tendrá un costo alto.

2.1. Operaciones Básicas Estructuras Lineales

Las operaciones básicas para estas estructuras son:

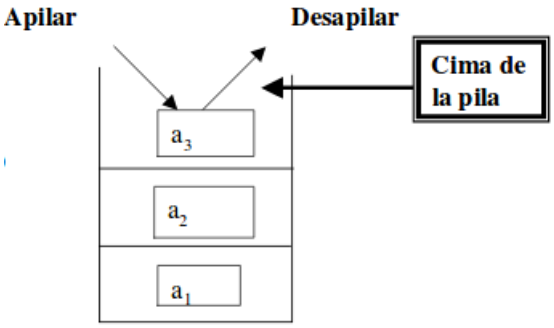
- crear la secuencia vacía
- destruir la secuencia (vaciarla y recuperar la memoria asignada)
- adicionar un elemento a la secuencia
- eliminar un elemento a la secuencia
- consultar un elemento de la secuencia
- comprobar si la secuencia está vacía

La diferencia entre las tres estructuras vendrá dada por la posición del elemento a añadir, borrar y consultar:

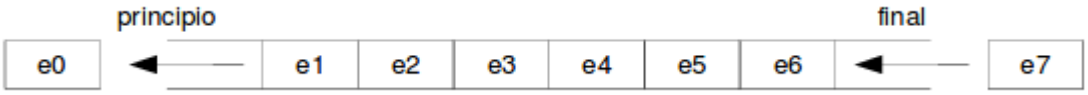
- **Pilas:** las tres operaciones actúan sobre el final de la secuencia
- **Colas:** se añade por el final y se borra y consulta por el principio
- **Listas:** las tres operaciones se realizan sobre una posición privilegiada de la secuencia, la cual puede desplazarse

2.2. Pilas

- Una pila es un contenedor de objetos que son insertados y eliminados de acuerdo con el principio de que el último en entrar es el primero en salir (LIFO, Last Input First Output)
- Los elementos se insertan de uno en uno (apilar)
- Se sacan en el orden inverso al cual se han insertado (desapilar)
- El único elemento que se puede observar dentro de la pila es el último insertado (cima o tope)

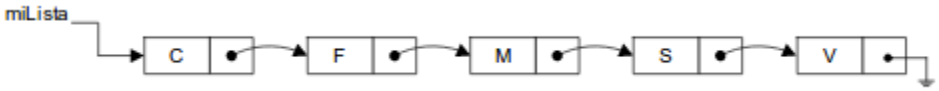


2.3. Colas



- Estructura lineal de datos compuesta por un conjunto de elementos en la que la adición de nuevos elementos se hará por un extremo de la cola, final (back), y la salida de elementos por el contrario, principio (front).
- Estructura de datos de tipo FIFO (first in-first out), es decir el primer elemento en entrar es el primero en salir.
- En aplicaciones informáticas se utiliza para contro lar procesos que tengan que realizarse en un cierto orden (colas de impresión, colas de prioridades, etc.

2.4. Listas



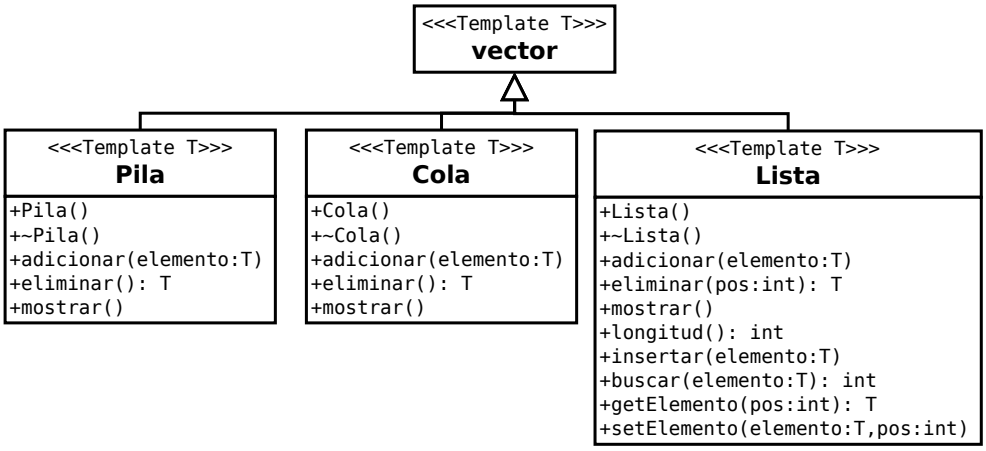
- Estructura lineal de datos compuesta por un conjunto de nodos que ocupan posiciones no contiguas de memoria.
- Cada nodo contiene la información del componente y, al menos, un puntero que indica la posición del siguiente nodo.
- En el último nodo, la posición del siguiente elemento será un puntero nulo.
- Al no ocupar posiciones contiguas en memoria y ser variable la posición del primer nodo de la estructura, la lista tendrá otro puntero que indicará la dirección del primer nodo de la estructura.

Pilas y colas son un tipo especial de listas enlaza das con la entrada y la Universidad Pontificia de Salamanca (Campus Madrid) Luis Rodríguez Baena, Escuela Superior de Ingenierí a y Arquitectura, 2012 61

Pilas y colas son un tipo especial de listas enlaza das con la entrada y la salida limitadas. ● En el caso de las listas enlazadas, las insercione s y eliminaciones se podrán hacer por cualquier punto de la estructura.

3. Diagramas de Clases

- Todas las clases derivan de la clase *vector* de la STL de C++



4. Entregables y Fechas

- Para cada estructura (Pila, Cola, Lista), realice un archivo con *main* donde construya un objeto de la clase respectiva y pruebe que funcionan los métodos.

De ahora en adelante todos sus proyectos van a estar en la nube, tanto el proyecto de candy como este de STL. Para esto:

- Usted debe tener una cuenta en un servidor de código (bitbucket: privado, o github: público)
- Después de cada sesión de trabajo, ya sea en su casa o en la clase, usted debe subir o actualizar el repositorio con lo último que haya realizado.
- Los ambientes de desarrollo van a ser dos: uno durante la clase y otro en casa o cualquier otro sitio:
  - Durante la clase, el desarrollo y la presentación de los trabajos se va a realizar en un servidor linux con las herramientas necesarias para editar (editor *nano*), compilar (compilador *g++*), y ejecutar (shell de *linux*).
  - En su casa, puede utilizar cualquier ambiente (windows, linux, iOS) y cualquier IDE (dev, codeblocks, etc.), pero al final tiene subirlos al repositorio de código para que los pueda descargar al inicio de la clase.
- La evaluación se hará con herramientas linux consola (git, nano, g++).

5. Recursos

- Tutorial STL vector: http://www.codeguru.com/cpp/cpp/cpp_mfc/stl/article.php/c4027/C-Tutorial-A-Beginners-Guide-to-stdvector-Part-1.htm
- Métodos de *vector:* http://www.yolinux.com/TUTORIALS/LinuxTutorialC++STL.html
- Dirección servidor local durante la clase (se dará ese mismo día)
- Dirección servidor remoto fuera de clase:
  - Nombre del equipo: *eisc.univalle.edu.co*
  - Puerto: 2224
  - Login: el que se le asignó el clase
  - Password (escondido): el mismo login
- Para acceder desde windows, descarge la aplicación de terminal *putty*
- Para acceder desde linus o iOS, utilize *ssh:*

$ ssh -l <nombre de usuario> <nombre de la máquina>

6. Algunos comandos desde el shell de linux:

6.1. Git

- Para clonar el repositorio de la nube (ejemplo: github) a su cuenta en el servidor. La dirección la copia del sitio web del repositorio:

```
$ git clone <dirección de clonación>
```

- Para actualizar el repositorio desde su cuenta en el servidor al repositorio de la nube (ejemplo: github). El mensaje describe lo ultimo que se hizo de forma corta. La instrucción de *push* le solicitará el nombre del usuario y la clave.

```
$ git commit -a -m "Mensaje"
$ git push
```

- Para ver el estado del *git:*

```
$ git status
```

6.2. Manejo de Archivos Shell de Linux:

- Para listar los archivos y directorios (carpetas) de donde está ubicado actualmente:

```
$ pwd                 # Muestra la ruta o directorio actual donde está ubicado
$ ll                  # Detallado
$ ls -l               # Detallado
$ ls                  # Solo los nombres
```

- Para acceder a directorios desde donde está ubicado actualmente:

```
$ cd <nombre del directorio>      # se ubica en el directorio
$ cd ..                           # se ubica en el directorio anterior
$ cd                              # se ubica en el directorio raíz de su cuenta
```

- Para manipular archivos:

```
$ rm <nombre del archivo>           # Remueve o elimina el archivo
$ cp <nombre archivo> <directorio>  # Copia el archivo al directorio
```

- Para manipular directorios:

```
$ mkdir <nombre directorio>    # crea un nuevo directorio
# rmdir <nombre directorio>    # Borra el directorio SOLO si está vacio
# rename <nombre del directorio> <nombre nuevo del directorio nuevo>
```