Facultad de Ingeniería y Ciencias Hídricas
FICH
UNL

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Ingeniería Informática

PROGRAMACIÓN ORIENTADA A OBJETOS

UNIDAD 8
Standart Template Library

2011

UNIDAD 8

Standart Template Library

Ejercicio 8.1

Escriba un programa que defina un vector dinámico de 30 enteros aleatorios menores que 50. Luego:

a) Muestre el vector generado.
b) Ordene en forma ascendente los elementos ubicados entre las posiciones 10 y 20 inclusive. Luego muestre el vector.
c) Quite del vector los elementos que fueron ordenados en el apartado anterior e insértelos en un nuevo vector (ayuda: utilice los algoritmos *copy* y *remove*).
d) Muestre el nuevo vector sin elementos repetidos.
e) Elimine uno por uno los elementos del nuevo vector a partir de su última posición. Muestre el vector resultante luego de cada eliminación.

Ejercicio 8.2

Cree un programa que lea 10 valores flotantes menores a 50 ingresados por teclado y los inserte en una lista. Luego:

a) Muestre dicha lista.
b) Inserte en medio de cada par de elementos contiguos el promedio de dichos elementos y guarde la lista resultante en un archivo de texto llamado "*lista-float.txt*".
c) Genere otra lista de 10 reales con valores aleatorios y combínela con la primera utilizando el algoritmo *merge*. Muestre el resultado.
d) Responda: ¿es posible ordenar solamente una porción de la lista de la misma manera que se hizo con el vector en el ejercicio 8.1?
e) Elimine de la lista resultante los valores máximo y mínimo y muestre la lista sin dichos valores.

Ejercicio 8.3

Cree una lista de 15 enteros aleatorios entre 0 y 10. Muestre la lista antes y después de quitar los elementos repetidos. Haga una búsqueda bibliográfica sobre la clase *ostream_iterator* y utilice el algoritmo *copy* para mostrar la lista en pantalla y luego guardarla en un archivo.

Ejercicio 8.4

Genere un arreglo estático (no un vector STL) de 20 elementos enteros aleatorios entre 0 y 10. Luego:

- Utilice los algoritmos genéricos de la STL para ordenarlo y eliminar los elementos repetidos.
- Utilice el algoritmo *copy* para mostrar el arreglo resultante.
- Responda: ¿Pueden los algoritmos de la STL eliminar realmente los elementos y liberar la memoria del vector? Justifique su respuesta.

Ejercicio 8.5

El archivo *"datos.txt"* contiene una lista de flotantes, uno por línea. Lea el contenido del archivo y almacénelo en una lista utilizando el algoritmo copy. Para esto haga lo siguiente:

- Abra el archivo para lectura y cree un iterador de lectura para leer valores flotantes que apunte al principio del archivo de la siguiente forma: *istream_iterator <float> p(archi)*.
- Cree otro iterador a punte al final del archivo indique el fin del archivo.
- Cree un contenedor para almacenar lo que se leerá.
- Utilice el algoritmo copy para copiar los datos del archivo al contenedor.

Luego, realice lo siguiente:

a) Calcule el promedio de los elementos del contenedor (ayuda: utilice el algoritmo *accumulate*).
b) Restar dicho valor a cada elemento.
c) Guarde los datos modificados en el archivo utilizando el algoritmo *copy*.

Ejercicio 8.6

Genere un vector de 20 elementos y asígnele valores aleatorios utilizando el algoritmo *generate*. Muestre los elementos en pantalla utilizando el algoritmo *copy*. Luego pídale al usuario que ingrese un valor númerico y utilice el algoritmo *find* para buscar dicho valor e informar la posición del vector en la que se encuentra o si no el valor no existe.

Ejercicio 8.7

Diseñe y programe una clase para manipular datos del tipo que se muestra en el recuadro. La clase será utilizada para administrar los pacientes de una clínica. La clínica está conformada por 6 médicos y cada uno de ellos atiende a un número variable de pacientes.
La clase debe proveer funciones para:

```
struct FichaMedico{
    string nombreMedico;
    vector <long> dniPacientes;
};
```

- Incorporar nuevos pacientes indicando el número de médico y el DNI del paciente.
- Listar el nombre de cada médico y los DNI de los pacientes que atiende.
- Eliminar un paciente indicando solamente su DNI.

Utilice la clase desarrollada desde un programa cliente.

Ejercicio 8.8 (OPCIONAL)

Diseñe una clase que permita manejar un archivo de configuración con el formato que se muestra en el recuadro. Las opciones deberán ser almacenadas en un mapeo de tipo *string* a *string*, y los valores de las opciones serán convertidos a otros formatos (int, float, etc) cuando el usuario lo solicite.

```
unaopcion=5
otraopcion=hola
gravedad=9.8
```

La clase debe proveer funcionalidades para:

- Leer un archivo de texto dado por el usuario y poblar el mapa con las opciones encontradas .
- Agregar una opción con su respectivo valor, si dicha opcion ya existe su valor debe actualizarse. Se recomienda que esta función tenga varias sobrecargas para soportar los diversos tipos de datos (int, float, etc) y los convierta a string para almacenarlos en el mapa.
- Devolver el por referencia el valor asociado a una determinada opción y un booleano indicando si dicho valor se encuentró o no en el mapa. Se recomienda que esta función tenga varias sobrecargas para soportar los diversos tipos de datos (int, float, etc).
- Guardar la configuración modificada en un archivo de texto.

Se propone el siguiente prototipo para la clase:

```
class Config {
    private:
    map<string,string> entries;

    public:
    // constructores
    Config();
    Config(string filename);
    // cargar y guardar en archivo de texto
    int Load(string filename);
    int Save(string filename);
    // pedir un valor
    bool GetValue(string option, float &value,
                  float default_value=0);
    bool GetValue(string option, int &value,
                  int default_value=0);
    bool GetValue(string option, string &value,
                  string default_value="");
    // agregar o modificar un valor
    bool SetValue(string option, float newValue);
    bool SetValue(string option, int newValue);
    bool SetValue(string option, string newValue);
};
```

Cuestionario

- ¿Qué es la STL?¿Para qué sirve?
- ¿Qué es un contenedor?¿Cuáles son los tipos de contenedores?
- ¿Qué es un iterador?¿Para qué sirve?
- ¿En qué se diferencian los algoritmos de la STL de otros algoritmos convencionales?
- ¿Cuál es la diferencia entre un vector y una lista?¿En que casos conviene usar uno o el otro?
- ¿Por qué la clase *list* implementa sus propios algoritmos en lugar de utilizar los algoritmos genéricos de la STL?