Unidad Nº6 bis: STL

Preguntas orientadoras

1) ¿Qué es la STL?

2) ¿Cuáles son los componentes de la STL?

3) ¿Cuáles son las características generales de los contenedores? ¿Explique cuales son y en qué categorías se dividen?

4) ¿Existen 4 elementos que se conocen como "casi contenedores"? Investigue cuáles son.

5) ¿Investigue sobre el significado de contenedor (deep) y contenedor (shallow)? ¿Los contenedores en C++ son deep o shallow?

6) ¿Qué son los iteradores? Mencione las diferentes categorías de iteradores y arme una tabla que muestre la abreviatura asignada en la STL a cada una de esas categorías.

7) Si se almacenan valores a partir de la posición apuntada por un iterador, ¿se sobreescriben los valores almacenados en el contenedor? ¿Qué problemas podrían surgir y cómo se pueden solucionar? Investigue.

8) Investigue sobre los iteradores: istream_iterator y ostream_iterator. Describa qué hace el siguiente programa:

```
#include <iostream>
#include <string>
#include <conio.h>
#include <algorithm>

using namespace std;

int main()
{
    string a1,a2,x;

    cout<<"\nIntroduce una frase, palabra a palabra, 0 para finalizar: \n";
    do{
    cin>>x;
    if (x!="0")
    {if (a1!="\0") a1+=" ";
     a1+=x;
    }
    } while (x!="0");

    ostream_iterator<char> salida(cout,"");
    cout<<"La frase introducida es: \n"<<a1;
    cout<<"\nLa fase al reves es:\n";
    copy(a1.rbegin(),a1.rend(),salida);

    return 0;
}
```

9) ¿Qué son los allocators? ¿Cuál es el allocator por defecto de la STL? ¿Cuál es su definición y que debe incluir en el main para utilizarlo?

10) ¿Qué es un objeto-función?

11) Los objetos-funciones incluídos en la STL se dividen en: predicados y aritméticos. Explique qué tipo de operaciones permiten realizar ambas categorías.

12) Compile el programa que se muestra a continuación, si no compila corrija el problema, y luego, verifique el funcionamiento. Estudie los objetos-función y el algoritmo utilizados.

```cpp
#include <vector>
#include <iostream>
#include <functional>
using namespace std;

template <class T> void mostrar(const vector <T> &v)
{
    ostream_iterator<T> salida(cout," ");
    cout<<endl;
    copy(v.begin(),v.end(),salida);
    cout<<endl;
}

int main()
{
    int c1[] = {1, 2, 3, 4, 16}, r, tam;
    tam = sizeof(c1)/sizeof(int);
    vector<int> v1(c1, c1+tam);
    cout << "Valores de v1";
    mostrar(v1);
    r = accumulate(v1.begin(), v1.end(), 1, multiplies<int>());
    cout << "El resultado de multiplicar todos los elementos de v1 es:" << r;
    return 0;
}
```

Ejercicios

1) Implementar un programa que permita introducir una frase, escribiendo palabra por palabra, escribir la frase introducida y escribirla al revés.

Ejemplo:
Ingrese una frase palabra por palabra, 0 para finalizar:
hola,
cómo
estás?
La frase introducida es:
hola, cómo estás?
La frase al revés es:
¿sátse omóc ,aloh

2) Crear una función que devuelva un par de valores; en el primero debe devolver la letra mayúscula que corresponde con el código ASCII del número pasado como argumento; por su parte, el segundo debe devolver la letra minúscula. Un ejemplo sería el que se muestra a continuación:

Ingrese un número entre 65...90 ó 97… 122:
65
Mayúscula A
Minúscula a

Nota: utilizar *pair* y *make_pair* para resolver el ejercicio.

3) Implementar un programa utilizando el contenedor *vector* que permita ingresar una serie de números hasta que el usuario introduzca 0 y que al final muestre los valores introducidos. Ejemplo:

Ingrese un número (fin=0) 3
Ingrese un número (fin=0) 6
Ingrese un número (fin=0) 8
Ingrese un número (fin=0) 2
Ingrese un número (fin=0) 9
Ingrese un número (fin=0) 100
Ingrese un número (fin=0) 0
Valores ingresados:
3 6 8 2 9 100

4) Modificar el programa del ejercicio 3 para que la salida ahora sea *100 9 2 8 6 3,* si los valores ingresados son los mismos.

5) Implementar el ejercicio 3 y 4 pero usando el contenedor *list*.

6) Implementar el ejercicio 3 y 4 pero usando el contenedor *deque*.

7) Escribir un programa que permita almacenar el primer apellido y el nombre de los empleados de una empresa: lógicamente los apellidos y nombres pueden estar repetidos. Luego de ingresar los datos, el programa debe solicitar el apellido de los empleados a buscar, y mostrar todos aquellos cuyo apellido sea igual al ingresado. Si no hay empleados con ese apellido se debe informar al usuario.

8) Escribir un programa que permita crear una pila con el primer apellido y el nombre de los empleados de una empresa. Luego, deberá mostrar los datos en ella incluidos e ir borrandolos a medida que se van mostrando.

9) Escribir un programa similar al ejercicio 8 pero usando una cola.

10) Implementar un programa, que utilizando el contenedor más adecuado, permita almacenar los empleados de una empresa, **ordenados** por DNI. Para ello escriba la clase Empleado, cuya información es la siguiente: DNI, nombre, primer apellido, segundo apellido, sueldo y años de antigüedad. Luego se debe mostrar toda la información ingresada.

11) Modifique el programa del ejercicio 12 de la parte teórica para que **sume** todos los elementos del vector.

12) Implemente un programa en el que haya una función que genere los números primos. El programa debe solicitar el número de elementos a incluir en un vector, se debe cargar el vector con los números primos y al final debe mostrar dicha información.

13) Modifique el programa anterior para que compruebe cuántos elementos del vector son menores o iguales a 11, cuántos son múltiplos de 5 y cuántos son iguales a 2.

14) Cargue un vector de complejos, cuyo módulo sea igual a 1 y sus ángulos múltiplos de pi/3 dentro del intervalo [0, 2pi]. Verifique que los algoritmos que cuentan elementos funcionan para este tipo de datos.

15) Estudie la función main que sigue, agregue los *#include* necesarios para que compile y explique el funcionamiento de los algoritmos que permiten realizar operaciones entre conjuntos de datos.

```
int main()
{
    int c1[] = {1,2,4,5,6,8,9,10,12,15};
    int c2[] = {1,3,4,5,7};
    int t1,t2;
    t1=sizeof(c1)/sizeof(int);
    t2=sizeof(c2)/sizeof(int);
    vector <int> v1(c1,c1+t1),v2(c2,c2+t2),v3(t1);
    vector <int>::iterator final;
    ostream_iterator<int> salida(cout," ");
    cout << "Valores de v1"; muestra(v1);
    cout << "Valores de v2"; muestra(v2);
    if (includes(v1.begin(),v1.end(),v2.begin(),v2.end()))
     cout << "v2 está incluido en v1\n";
    else cout << "v2 no está incluido en v1\n";
    final= set_union(v1.begin(),v1.begin()+6,
                     v2.begin(),v2.end(), v3.begin());
    cout << "Valores de v3. Unión de v1..v1+5 y v2\n";
    copy(v3.begin(),final,salida);
    final= set_intersection(v1.begin(),v1.begin()+6,
                            v2.begin(), v2.end(),v3.begin());
    cout << "\nValores de v3. Intersección de v1..v1+5 y v2\n";
    copy(v3.begin(),final,salida);
    final= set_difference(v1.begin(),v1.end(),
                          v2.begin(),v2.end(), v3.begin());
    cout << "\nValores de v3. Diferencia de v1 y v2\n";
    copy(v3.begin(),final,salida);
    return 0;
}
```