ABSTRACCIÓN POR ITERACIÓN

Contenedores.- son E.D que mantienen almacenados una colección de elementos de otro tipo de dato.

¿Existe una forma genérica para cualquier contenedor de acceder a los datos que lo forman? Sí. Con Iteradores.

ITERADORES. Es un T.D.A que abstrae la idea de acceder a los elementos de un contenedor como una forma parecida a los punteros.

COMO USARLOS

1. Inicializar el iterador a la primera posición (begin)
2. Saber como avanzar al siguiente elemento (++,--…)
3. Saber como acceder al elemento (*)
4. Saber cuando terminar (end)

ABSTRACCIÓN POR ITERACIÓN

```
class Lista{
  private:
    int *datos;
    int n , reservados;
  public:
   ....
  class iterator{
    private:
     int *it;
    public:
      iterator():it(nullptr){}
      int & operator*(){
         return *it;
      }
      iterator& operator ++(){
           ++it;
           return *this;
      }
iterator& operator --(){
       --it;
      return *this;
  }
bool operator ==(const iterator
&i)const{
      return it==i.it;
}
bool operator!=(const iterator
&i)const{
     return it!=i.it;
}
friend class Lista;
};//end iterator

//estamo en lista
iterator begin(){
   iterator i;
   i.it = &(datos[0]);
   return i;
}
iterator end(){
    iterator i;
    i.it = &(datos[n];
    return i;
} }; //end Lista
```

ABSTRACCIÓN POR ITERACIÓN

Iteradores Constantes

```
void Imprimir(const Lista &L){
  Lista :: iterator it;

  for (it=L.begin(); it!=L.end()++it)
    cout<<*it;
}
```

```
void Imprimir(const Lista &L){
  Lista :: const_iterator it;

  for (it=L.begin(); it!=L.end()++it)
    cout<<*it;
}
```

ABSTRACCIÓN POR ITERACIÓN

```
class Lista{
  private:
    int *datos;
    int n , reservados;
  public:
   ....
 //Dos iteradores

 class iterator{
    private:
     int *it;
    public:
      ...
     friend class Lista;

  };//end iterator
 //estamo en lista
  class const_iterator{
   private:
    const  int *it;
   public:
    const_iterator():it(nullptr){}
    const_iterator(const
 const_iterator &i):it(i.it){}
    const int & operator*(){
          return *it;
      }
      const_iterator& operator ++(){
            ++it;
            return *this;
      }
     const_iterator& operator --(){
        --it;
       return *this;
  }
 bool operator ==(const
 const_iterator &i)const{
      return it==i.it;}
 bool operator!=(const
 const_iterator &i)const{
     return it!=i.it;
 }
 friend class Lista;
 };//end const_iterator

 iterator begin(){
    iterator i;  i.it = &(datos[0]);
    return i;
 }
 iterator end(){
     iterator i; i.it = &(datos[n];
     return i;
 }
 const_iterator begin()const{
    const_iterator i;
    i.it = &(datos[0]);
    return i;
 }
 const_iterator end()const{
    const_iterator i;
    i.it = &(datos[n];
     return i;
 }
  }; //end Lista
```

ABSTRACCIÓN POR ITERACIÓN

Ejercicio.- Crear una clase Notas en la que se almacena pares (dni,nota) siendo nota la calificación de un alumno con un dni. Para almacenar el conjunto de pares usaremos la clase vector de la STL. Definir dentro de ella la clase iterator y la clase const_iterator. Además se pide

1. Crear una función que imprima los pares (dni,nota) usando un iterator (o const_iterator)
2. Crea una función para obtener la nota media.

Ejercicio. (continuación). Implementación de la clase

Ejercicio. (continuación). Implementación de la clase

Ejercicio. (continuación). Implementación de la clase

Ejercicio. (continuación). Implementación de la clase

Cuestiones.

1.- ¿Es lógico tener un iterador sobre la clase Pila?

2.- ¿Es lógico mantener un iterador sobre la clase Cola?

3.-Enumerar la diferencias entre un iterador y un const_iterator sobre un contenedor

4.-¿Por qué la clase contendora tiene que ser amiga de la clase iterator?

Cuestiones.

5.-Suponer que tenemos la clase vector dinámico de enteros sobre la que hemos definido un iterador. Se puede definir otro iterador (iterador_par) que itere solamente sobre los elementos pares. Si es afirmativo como se representa y como se implementan las funciones miembro

Cuestiones.

6.- Según la cuestión 5 como se implementaría las funciones begin y end para iterator_par.