LECCION 8.- (PARTE I)

VECTOR DISPERSO de TIPO T

ESPECIFICACION: Es un array 1-D de parejas formadas por un índice o clave y un valor de tipo T. Si el índice que se consulta no está en el array se devuelve un valor por defecto (d).

$\{(i_0, v_{i_0}), (i_1, v_{i_1}), \ldots, (i_{n-1}, v_{i_{n-1}}), (*, d)\}$

OPERACIONES

- T GetDefault() const: devuelve el valor por defecto
- T Get(int ind) const: obtiene el elemento con índice ind
- void Set(int ind, const T& v): modifica un elemento con índice ind le pone el valor v.
- int NumNoDefault() const: devuelve el nº de elementos no asignados.
- Void DatosPosicion(int i, int &indice, T &valor) const: Obtiene el índice y valor del elemento en la posición i del array.
- void Insertar(int ind, const T&v)

TIPO rep

```
template <class T>
struct Elemento{
    int indice;
    T valor;
};
```

```
template <class T>
class Vdisperso{
  private:
    VD<Elemento> datos;
    T valor-por-defecto
```

FUNCION de ABSTRACCION

r un objeto de tipo rep

$f_A(r) = \{(r.datos[0].indice, r.datos[0].valor), (r.datos[1].indice, r.datos[1].valor), \ldots, (r.datos[r.datos.size()-1].indice, r.datos[r.datos.size()-1].valor)\}$

$(\forall 0 \le i \le r.datos.size()-1$

$\wedge \forall i \notin [0, r.datos.size()-1]$

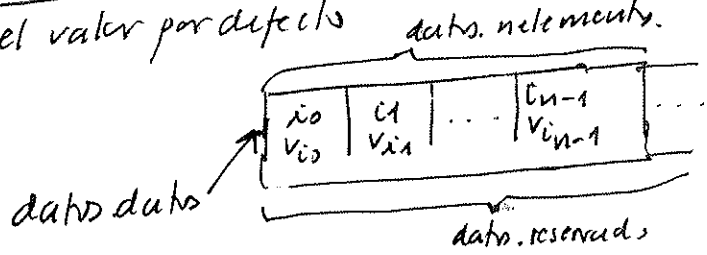
$(-1, valor\_por\_defecto)$

INVARIANTE de la representacion.

valor_por_defecto != r.datos[i].valor

$\forall 0 \le i \le r.datos.size()-1 \wedge r.datos[i].indice \neq r.datos[j].indice$ (No se usa en los datos almacenados)

el valor por defecto



LECCION 8: (PARTE I)

```
// FICHERO Vdispersa.h
#include "VD.h"
template <class T>
struct Elemento{
    int indice;
    T valor;
};

template <class T>
class Vdispersa{
  private:
    VD<Elemento> datos
    T valor_por_defecto;
    bool PosicionIndice(int &pos, int ind)const;

  public:
    T GetDefault()const{ return valor_por_defecto; }

    T Get(int ind)const;
    void Set(int ind, const T &v);
    int NumNoDefault()const{ return datos.size(); }
    void DatosPosicion(int i, int &indice, T &valor)const;

    Vdispersa(const T &vdef=T()){
        valor_por_defecto = vdef;
    }
    void Insertar(int indice, const T &v);
};
#include "Vdispersa.cpp"
```

LECCION 8- PARTE I

```
// Vdisperso.cpp
template <class T>
bool Vdisperso<T>::PosicionIndice(int &pos, int ind)const{
        for(int K=0; K<datos.size(); K++){
            if (datos[K].indice==ind){
                    pos=K
                    return true;
             }
        }
        return false;
}
template <class T>
T Vdisperso<T>::Get(int ind) const{
        int pos;
        if (PosicionIndice(pos,ind))
              return datos[pos].valor;
        else  return valor_por_defecto
}
template <class T>
void Vdisperso<T>::Set(int ind, const T &v){
     int pos;
     if (PosicionIndice(pos, i)){ //esta
         if (v!= valor_por_defecto){
                  datos[pos].valor=v;
         }else datos.Borrar(pos);
     }else{ //no esta
           if (v!= valor_por_defecto){
                  Elemento<T> a={i,v};
                  datos.Insertar(a, datos.size());
           }
     }
}
```

LECCION 8 - PARTE I

```
template <class T>
void Vdisperso<T>::DatosPosicion(int i, int &indice, T &valor) const
{
    if (i>=0 && i<datos.size()){
        indice = datos[i].indice;
        valor = datos[i].valor;
    }
    else {
        indice = -1;
        valor = valor_por_defecto;
    }
}
template <class T>
void Vdisperso<T>::Insertar(int indice, const T &v){
    if (v!=valor_por_defecto){
        int pos;
        if (PosicionElemento(pos, indice)){
            Elemento<T> a = {indice, v};
            datos.Insertar(a, datos.size());
        }
    }
}
```

EJEMPLO de USO

```
#include "Vdisperso.h"
int main(){
    Vdisperso<float> vdis(-1.0);
    for(int i=0; i<5; i++)
        vdis.Insertar(i, i*0.5);

    while(true){
        int ind;
        cout << "Dime un indice";
        cin >> ind;
        float valor = vdis.Get(ind);
        if (valor != vdis.GetDefault()){
            cout << "Valor asociado: " << valor
        }else
            cout << "Indice no valido" << endl;
    }
}
```