```cpp
#include <iostream>
#include <string>
#include <list>
#include <vector>

using namespace std;

template <typename T> using Par_INT_T = pair<int,T>;

template <typename T>
class vdisperso
{
private:
    list<pair<int, T>> coefs;
    int n;
    T nulo;

public:
    vdisperso(const vector<T> &v,T nulo=T());
    void asignar_coeficiente(int i, const T &x);
    vector<T> convertir() const;
    void mostrarVectorDis() const;
    void cambiarNulo(const T &nuevo_nulo);
};

template <typename T>
void vdisperso<T>::mostrarVectorDis() const{
    for(typename list<pair<int,T>>::const_iterator
it=coefs.cbegin();it!=coefs.cend();it++)
    {
        cout << "POSICION:" << (*it).first << "->" << (*it).second << endl;
    }
}


template <typename T>
vdisperso<T>::vdisperso(const vector<T> &v, T nulo)
{
    int i;
    this->nulo=nulo;
    pair<int,T> aux;
    this->n=v.size();
    for(i=0;i<v.size();i++)
    {
        if(v[i]!=nulo)
        {
            aux.first=i;
            aux.second=v[i];
            this->coefs.push_back(aux);
        }
    }
}

template <typename T>
void vdisperso<T>::asignar_coeficiente(int i, const T &x)
{
    typename list<pair<int, T>>::iterator it;
    it=this->coefs.begin();
    while(it!=this->coefs.end() && (*it).first!=i)
    {
        it++;
    }
    if(it!=this->coefs.end())
    {
        this->coefs.erase(it);
    }
    pair<int, T> aux;
    aux.first=i;
    aux.second=x;
    this->coefs.push_back(aux);

}

template <typename T>
vector<T> vdisperso<T>::convertir() const
{
    vector<T> ret;
    typename list<pair<int,T>>::const_iterator it;
    it=this->coefs.begin();
    for(int i=0;i<this->n;i++)
    {
        if(i==(*it).first)
        {
            ret.push_back((*it).second);
            it++;
        }else
        {
            ret.push_back(nulo);
        }
    }
    return ret;
}

template <typename T>
void vdisperso<T>::cambiarNulo(const T &nuevo_nulo)
{
    this->nulo=nuevo_nulo;
}

template <typename T>
void mostrar_vector(const vector<T> & v)
{
    for(typename vector<T>::const_iterator it=v.cbegin();it!=v.cend();it++)
    {
        cout << (*it) << " ";
    }
    cout << endl;
}

int main()
{
    vector<int> aux(10,1);
    aux[0]=104353;
    aux[2]=3;
    aux[4]=9;
    aux[7]=99;
    aux[9]=81;
    vdisperso<int> prueba(aux,1);
    prueba.mostrarVectorDis();
    aux=prueba.convertir();
```

```cpp
    mostrar_vector(aux);

    prueba.cambiarNulo(8);
    prueba.mostrarVectorDis();
    aux=prueba.convertir();
    mostrar_vector(aux);
}
```