LECCIÓN 6.- ABSTRACCIÓN: GENERALIZACIÓN

- Abstracción por generalización
  - A partir de varios objetos extraer características comunes que definan una generalización más fácil de manejar.

- Abstracción funcional
  [es una abstr. por generalización

- Los parámetros de una función constituyen una generalización en la que se integran todas las posibles ejecuciones sobre los distintos parámetros.

GENERALIZACION: Abstracción por parametrización.

FUNCIONES PATRÓN

```
template <class T>
void Intercambiar (T &a, T &b) {
    T aux=a;
    a=b;
    b=aux;
}
```

C++ → template → template <class T>
(plantilla)

Ejemplo: Ordenar un vector.
El vector puede ser de enteros, caracteres flotante, etc. El procedimiento es el mismo independiente del tipo base del vector.

```
template <class T>
void ordenar_seleccion (T *vector, int n)
{
  int i, minimo;
  for (i=0; i<n-1; i++){
      minimo=i;
      for (int j=i+1; j<n; j++)
          if (vector[j] < vector[minimo])
              minimo=j;
      Intercambiar(vector[i], vector[minimo]);
  }
}
.
.
.

int main() {
  int vint[]={5,7,9,10,14,15,1};
  int nint=7;
  ordenar_seleccion(vint, nint);

  char vch[]={'e','n','d','$','g','a'};
  ordenar_seleccion(vch, 7);

  float vf[7]={3.1, 4.9, 0.1, -3.2};
  ordenar_seleccion(vf, 4);

}
```

LECCION 6: continuación

CLASES PATRÓN

Establecemos para las clases el mismo mecanismo de generalización que para las funciones.

```
// file VD.h
template <class T>
class Vector_Dinamico {
    private:
      T* datos;
      int n;
      int reservados;
      void Resize(int nuevotam);
    public:
      Vector_Dinamico (int n=0);
      Vector_Dinamico (const Vector_Dinamico<T>
                       & original);

      ~Vector_Dinamico ();
      int size() const;
      T & operator [](int i);
      const T & operator [](int i) const;
      Vector_Dinamico <T> & operator=(const Vector_Dinamico<T> & vd);
};
#include "VD.cpp"

:

int main() {
    Vector_Dinamico <int> vint;
    Vector_Dinamico <string> nombres;
    Vector_Dinamico <polinomio> polinomios

}
```

③

LECCION 6- continuación

```
template <class T>
Vector_Dinamico<T>::~Vector_Dinamico(){
        if (datos!=0)
            delete[] datos;

}
template <class T>
int Vector_Dinamico<T>::size() const{
        return n;

}

template <class T>
T & operator[](int i){
        return datos[i];

}
template <class T>
const T & operator[](int i)const{
        return datos[i];

}

template <class T>

Vector_Dinamico<T> & Vector_Dinamico<T>::operator=(const Vector_Dinamico<T>&v)
{
        if (this!=&v){
            if (datos!=0)
                delete[] datos;

            reservados = v.reservados;
            n =   v.n;
            if (v.datos!=0){
                datos= new T[reservados];
            }
            for (int i=0; i<n; i++)
                datos[i]=v.datos[i];

        }
        return *this;
}
```

LECCION 7.- continuacion

```
//file "VD.cpp"
template <class T>
void Vector_Dinamico<T>::Resize(int nuevotam){
    T *aux= new T[nuevotam];
    int minimo = (n<nuevotam)?n:nuevotam;
    for (int i=0;i<minimo;i++)
        aux[i]= datos[i];

    reservados = nuevotam;
    n=minimo;
    delete [] datos;
    datos=aux;
}
template <class T>
Vector_Dinamico<T>::Vector_Dinamico(int n){
    if (n==0){
        reservados=this->n=0;
        datos=0;
    }
    else{
        reservados = n;
    this->n=0
        datos=new T[reservados];
    }
}template <class T>
Vector_Dinamico<T>::Vector_Dinamico(const Vector_Dinamico<T> &original){
    reservados = original.reservados;
    n = original.n;
    if (reservados>0)
        datos= new T[reservados];
    else datos=0;
    for (int i=0; i<n; i++)
        datos[i] = original.datos[i];
}
```

3.