

Estructuras de Datos

Práctica 4: TDA no lineales

Sesión 2: Iteradores

Cristina Zuheros Montes

czuheros@ugr.es

Universidad de Granada

November 23, 2020



Outline

1. Iteradores
2. Iteradores y Vector dinámico
3. Práctica 4

Outline

1 Iteradores

2 Iteradores y Vector dinámico

3 Práctica 4

Motivación Iteradores

Iterador:

Tipo de dato abstracto que permite recorrer los elementos de un contenedor

Contenedor:

Tipo de dato compuesto por una colección de elementos de algún otro tipo

Ejemplos de contenedores:

- vector dinámico
- polinomio
- lista

Iteradores

Definiremos una clase iterador en cada contenedor que los requiera

Son necesarios dos tipos de iteradores

- iterador: para contenedores que se van a modificar
- const_iterador: para contenedores que se no se modifican

Son necesarias dos funciones para cada iterador que se definen en el contenedor

- begin: devuelve un iterador al primer elemento
- end: devuelve un iterador al elemento después del último

Outline

1 Iteradores

2 Iteradores y Vector dinámico

3 Práctica 4

Vector dinámico con iteradores - I

```
int main(){
    Vector_Dinamico<float> vec;
    vec.resize(10);
    cargar_indices(vec);
    cout << "El vector vec contiene:"<< endl;
    imprimir(vec);
    cout << "Maximo de "<< vec.size() << " elementos: " << maximo(vec) << endl;
    cout << endl;

    Vector_Dinamico<string> cadenas(4);
    cadenas[0]="vector";
    cadenas[1]="de";
    cadenas[2]="tipo";
    cadenas[3]="string";
    cout << "El vector cadenas contiene:"<< endl;
    imprimir(cadenas);
}
```

```
czuheros@czuheros-ubuntu:~/Escritorio/VD$ ./usovd
El vector vec contiene:
0 1 2 3 4 5 6 7 8 9
Maximo de 10 elementos: 9

El vector cadenas contiene:
vector de tipo string
Maximo de 4 elementos cadena: vector
```

Vector dinámico con iteradores - II

```
template <typename T>
T maximo(const Vector_Dinamico<T>& v){
    assert (v.begin()!=v.end());
    typename Vector_Dinamico<T>::const_iterador max = v.begin();
    typename Vector_Dinamico<T>::const_iterador p;
    for (p=v.begin(); p!=v.end();++p)
        if (*max < *p)
            max = p;
    return *max;
}

template <typename T>
void imprimir(Vector_Dinamico<T>& v){
    assert (v.begin()!=v.end());
    typename Vector_Dinamico<T>::iterador it;
    for (it=v.begin();it!=v.end();++it)
        cout<<*it<<" ";
    cout << endl;
}
```


Ej. Vector dinámico - VD.h

```

...
class Vector_Dinamico{
    private:
        T * datos;
        int nelementos;
    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;
        void resize(int n);
        Vector_Dinamico<T> operator=(const Vector_Dinamico<T>& original);
        ...
};
...

```

Ej. Vector dinámico con iteradores - VD.h

```

...
class Vector_Dinamico{
    private:
        T * datos;
        int nelementos;
    public:
        ...
        class iterador {...};
        class const_iterador {...};
        iterador begin(){...};
        iterador end(){...};
        const_iterador begin() const {...};
        const_iterador end() const {...};
};
...

```

Ej. Vector dinámico con iteradores - Clase iterador I

```
class iterador{  
    private:  
        T * iter;  
    public:  
        iterador(): iter(0) {}  
        ~iterador(){}  
        T& operator*() const{  
            return *iter;  
        }  
}
```

Ej. Vector dinámico con iteradores - Clase iterador I

```

class iterador{
private:
    T * iter;
public:
    iterador(): iter(0) {}
    ~iterador(){}
    T& operator*() const{
        return *iter;
    }

    iterador& operator++(){...}
    iterador& operator--(){...}
    bool operator!=(const iterador& i) const {...}
    bool operator==(const iterador& i) const {...}

```

Ej. Vector dinámico con iteradores - Clase iterador I

```

class iterador{
private:
    T * iter;
public:
    iterador(): iter(0) {}
    ~iterador(){}
    T& operator*() const{
        return *iter;
    }

    iterador& operator++(){...}
    iterador& operator--(){...}
    bool operator!=(const iterador& i) const {...}
    bool operator==(const iterador& i) const {...}

    friend class Vector_Dinamico<T>;
    friend class const_iterador;
};

```

Ej. Vector dinámico con iteradores - Clase const_iterador

```
class const_iterador{  
    private:  
        T * iter;  
        ...  
    public:  
        ...  
        friend class Vector_Dinamico<T>;  
};
```

Ej. Vector dinámico con iteradores - begin y end

```

iterador begin(){
    iterador i;
    i.iter = datos;
    return i;
}
iterador end(){
    iterador i;
    i.iter = datos+nelementos;
    return i;
}
const_iterador begin() const{
    ...
}
const_iterador end() const{
    ...
}

```

Actividad voluntaria

Considerar el TDA Vector dinámico proporcionado en la P3:

- Pasarlo a template (con instanciación implícita)
- Crear las clases iteradora e iteradora constante
- Crear programa de prueba

Outline

1 Iteradores

2 Iteradores y Vector dinámico

3 Práctica 4

Iteradores para la Práctica 4

- Hay que definir dos clases para trabajar con iteradores constantes y no constantes para cada contenedor
- Podéis apoyaros en los iteradores de Vector Dinámico para ver las operaciones básicas de iteradores
- Ambas clases harán uso de los objetos `iterator` y `const_iterator` de la STL

Actividad obligatoria

Considerar el TDA Diccionario

- `list<data<T,U>> datos;`
- Crear las clases iteradora e iteradora constante

Considerar el TDA Guia_Tlf

- `map<string,string> datos;`
- Crear las clases iteradora e iteradora constante