Doble cola <deque>

La doble cola contiene secuencias de elementos que cambian de tamaño de forma dinámica. El tipo doble cola <deque> se puede expandir y contraer por los 2 extremos. Es similar al vector pero con una eficiencia diferente en la inserción y borrado de elementos.

A diferencia de los vectores las dobles colas no garantizan almacenar los elementos en posiciones contiguas de memoria

<deque>

Acceso elementos

operator[]
at
front
back

Modificadores

push_back || pop_back
push_front || pop_front

insert
erase
emplace || emplace_front || emplace_back
swap
clear

Iteradores

begin/end rbegin/rend

Ejemplo

```
#include <iostream>
#include <deque>
#include <vector>

int main()
{
    std::deque<int> mideque;

for (int i=1; i<6; ++i)
    mideque.push_back(i); //1 2 3 4 5

std::deque<int>::iterator it = mideque.begin();
++it;
it = mideque.insert(it, 10); //1, 10 2 3 4 5

mideque.insert(it, 2, 20); //1 20 20 10 2 3
                           //4 5
it = mideque.begin()+2;

std::vector<int> myvector(2, 30);
mideque.insert(it, myvector.begin, myvector.end());
                //1, 20 30 30 20 10 2 3 4 5

mideque.erase(mideque.begin(), mideque.begin()+3);
            //elimina los 3 primeros elementos

std::cout << "mideque contiene:";
for (it = mideque.begin(); it != mideque.end(); ++it)
    std::cout << " " << *it;
    std::cout << '\n';
return 0;
}
```

Crear con el tipo deque una clase pila o cola que permita a un objeto actuar como una pila o una cola dependiendo de cómo se inicialice una bandera

```
//pilaocola.h
#include <deque>
#include <iostream>
using namespace std;
template <class T>
class PilaoCola {
    private:
        deque <T> datos;
        bool is_cola;
            //true -> actua como cola
            //false -> actua como pila
    public:
        PilaoCola (bool tipo): is_cola(tipo){}
```

```
T & operator() () {
    if (is_cola)
        datos.front();
    else
        datos.back();
}
const T & operator() () const {
    if (is_cola)
        datos.front();
    else datos.back();
}
int size() const
    {return datos.size();
    }
bool empty() const{
    return datos.size()==0;
    }
```

```
void Pop () {
    if (!is_cola)
        datos.pop_front();
    else
        datos.pop_back();
    }

Void push (const T & v) {
    datos.push_back(v);
    }
```

Hemos implementado la función Frente cuando actúa como cola o Tope cuando actúa como Pila, usando el operador ().

Ejemplo de uso de la clase PilaoCola

```
#include "PilaoCola.h"

int main()
    PilaoCola<int> pila(false);
    PilaoCola<int> cola(true);

for(int i=10; i<100; i+=10)
        {
          pila.Push(i);
          cola.Push(i);
        }
std::cout<<"Los elementos de la pila son:";
  while(!pila.empty())
        {
          std::cout<<pila()<<' ';
          pila.Pop();
        }
std::cout<<std::endl;
 std:cout<<"Los elementos de la cola son:";
  while(!cola.empty())
        {
          std::cout<<cola()<<' ';
          cola.Pop();
        }
}
```