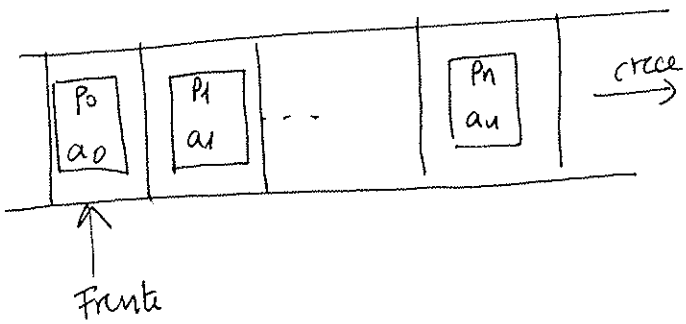
(4) LECCION 10: COLAS CON PRIORIDAD

COLA CON Priondad

Contienen una secuencia de valores especialmente diseñados para realizar los accesos y borrados por el Frente y las inserciones se realizan en cualquier pto de acuerdo a un criterio de prioridad.



Operaciones

+Frente: devuelve el elemento en el frente.

+Prioridad_Frente: devuelve la prioridad en el elemento del frente.

+Quitar: Elimina el elemento del frente

+Vacia: indica si la cola esta vacia.

+Poner: Inserta un nuevo elemento con una prioridad

```
#ifndef COLA_PRIO_H
#define COLA_PRIO_H
template <class Tpno, class T>
struct info{
    Tpno prior; // prioridad
    T    elemento; // dato almacenado

};
template <class Tpno, class T>
struct celda{
   info <Tpno,T> dato;
   celda * sig;
};
template <class Tpno, class T>
class Cola_Pno{
  private:
    celda <Tpno,T> * primera;
    void Copiar(const Cola_Pno
    void Borrar();
  public:
    Cola_Pno();
    Cola_Pno(const Cola_Pno<Tpno,T> & CP);
    ~Cola_Pno();
    Cola_Pno<Tpno,T> & operator=(const Cola_Pno
                                 <Tpno,T> &CP);

    T Frente() const;
    Tpno Prioridad_Frente() const;
    void Poner(const Tpno & Tp, const T &E);
    void Quitar();
    int size() const;
    bool Vacia() const;

};
#endif
```

⑤

LECCION : COLAS. Prioridad

```
template <class Tpno, class T>
T Cola_Pno <Tpno,T>::Frente() const{
        assert (primera!=0);
        return primera->dato.elemento;
}
template <class Tpno, class T>
Tpno Cola_Pno <Tpno,T>::Prioridad_Frente() const{
        assert (primera!=0);
        return primera->dato->prior;
}
template <class Tpno, class T>
void Cola_Pno <Tpno,T>::Poner(const Tpno &Tp, const T &e){
    Celda <Tpno,T> *aux = new Celda<Tpno,T>;
    aux->dato.pno = Tp;
    aux->dato.elemento = e;
    if (primera ==0){
        primera =aux;
        primera->sig=0;
    }
    else{
      if (primera->dato.prior <pno){
          aux->sig=primera
          primera=aux;
      }else{ // buscamos donde ponerlo
        Celda <Tpno,T> *p = primera;
        while (p->sig->dato.pno>=Tp && p->sig!=0)
              p=p->sig;
        aux->sig=p->sig;
        p->sig= aux;
    }
  }
}
```

```
template <class Tpno, class T>
void Cola_Pno<Tpno,T>::Quitar(){
 Celda <Tpno,T>* aux=primera;
 primera = primera->sig;
 delete aux;
}
```

Cola con prioridad en la STL

Biblioteca ⇒ #include <queue>

Clase ⇒ class priority_queue

Operaciones: emplace, empty, pop, push, size, swap, top ← elemento más prioritario

EJEMPLO 1

```
#include <queue>
#include <iostream>
using namespace std;
int main(){
    priority_queue<int> mipq, otra;
    mipq.push(10);
    mipq.push(20);
    mipq.push(15);

    cout << mipq.top() << endl; // 20

    mipq.swap(otro);
    while(!mipq.empty()){
        cout << mipq.top();
        mipq.pop();
    }
}
```

EJERCICIO: Dado un conjunto de frutas y precios en un vector ordenarlas de mayor a menor precio.

```
#include <string>
#include <queue>
#include <vector>
using namespace std;

struct fruta{
    string f;
    float precio;
    fruta(string name, float p):f(name), precio(p){}
    bool operator<(const fruta& f){
        return this->precio < f.precio;
    }
};
ostream& operator<<(ostream& os, const
                    fruta& f){
    os<< f.f << " " << f.precio <<endl;
    return os;
}
vector<fruta> ordena_frutas(const vector<fruta>& fs){
    priority_queue<fruta> mipq;
    for(int i=0; i<fs.size(); i++){
        mipq.push(fs[i]);
    }
    vector<fruta> vout;
    while(!mipq.empty()){
        vout.push_back(mipq.top());
        mipq.pop();
    }
    return vout;
}
```

Colas con prioridad

Esquema de cola

Una posible *clase ColaPri* para almacenar datos de tipo *string* con una prioridad indicada por un valor entero puede tener la siguiente sintaxis.

```
#ifndef __COLAPRI_H__
# define __COLAPRI_H__

  class ColaPri{
        ...                    // La implementación que deseemos
    public:
        ColaPri();
        ColaPri(const ColaPri& p);
        ~ColaPri();
        ColaPri& operator= (const ColaPri& p);

        bool vacia() const;
        void poner (int pri, string c)
        void quitar();
        string frente() const;
        int prioridad_frente () const;
  };

#endif
```

Uso de una cola con prioridad

```
#include <iostream>
#include <string>
#include <colapri.h>
using namespace std;
int main()
{
  ColaPri q;
  int nota;
  string dni;

  cout << "Escriba una nota" << endl;
  cin >> nota;
  while (0<=nota && nota<=10) {
    cout << "Escriba un dni" << endl;
    cin >> dni;
    q.poner(nota,dni);
    cout << "Escriba una nota" << endl;
    cin >> nota;
  }
  cout << "Los elementos en el orden de las notas son:"<<endl;
  while (!q.vacia()) {
    cout << "Nota: " << q.prioridad_frente()
         << " DNI:" << q.frente() << endl;
    q.quitar();
  }
  return 0;
}
```