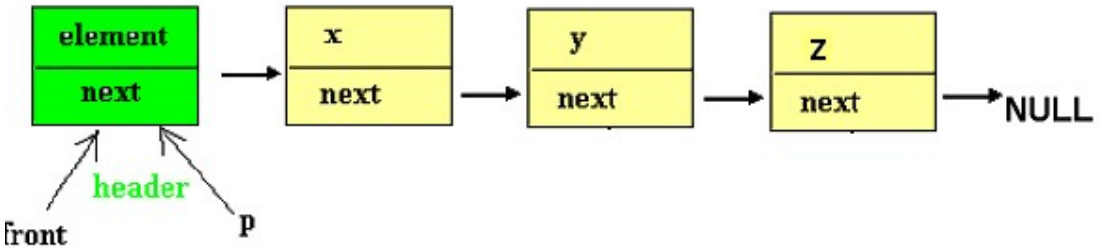
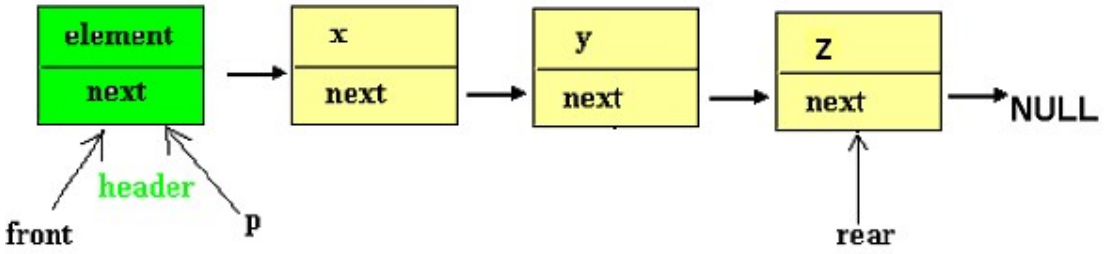
# Reprezentacja wskaźnikowa kolejki

(wstawiamy na koniec, usuwamy z początku)
Można wykorzystać listę wskaźnikową z głową.




Kolejkę można zaimplementować efektywniej pamiętając wskaźnik do ostatniego elementu kolejki (nie trzeba wtedy przechodzić przez całą listę przy operacji ENQUEUE)



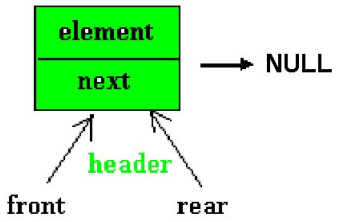
```
typedef int  elementtype;
struct celltype
{
  elementtype element;
  celltype * next;
};

typedef celltype * position;

class Queue
{
protected:
        position Front;      // wskaźnik do głowy listy
        position Rear;    // wskaźnik do ostatniego elementu

public:
   Queue();    //konstruktor
   ~Queue();  //destruktor
   void Enqueue(elementtype x);
   void Dequeue();
   elementtype FrontElem();
   bool Empty();
   };
```

**Kolejka pusta:**

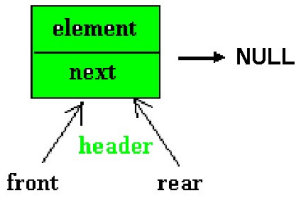



**Konstruktor:**

-Tworzy obiekt
-tworzy głowę
-aktualizuje wartości zmiennych *front* i *rear*

# Operacja Enqueue(x) –
# wstawia element x na koniec kolejki

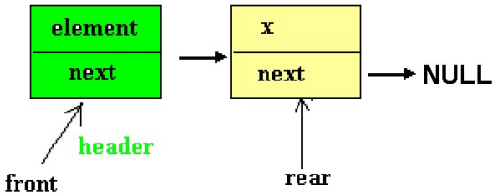
Załóżmy, że mamy kolejkę pustą:



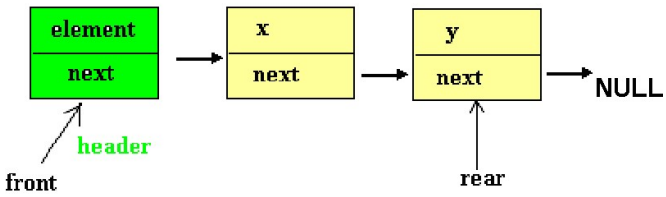

Tworzymy obiekt klasy Queue:

Q = new Queue;

Po wywołaniu metody Q->Enqueue(x) mamy:




Po wywołaniu metody Enqueue(y) mamy:



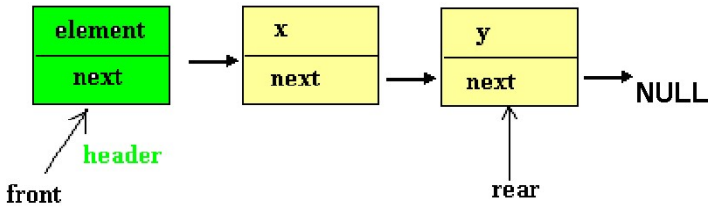

**void Queue::Enqueue(elementtype x)**

**Operacja Enqueue polega na:**

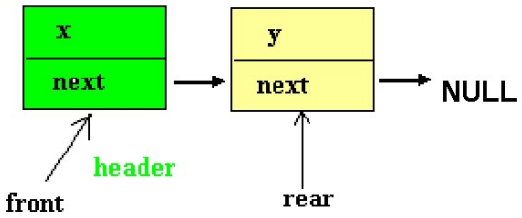
- stworzeniu nowej komórki:
- wstawieniu nowostworzonej komórki na koniec kolejki
- uaktualnieniu wartości zmiennej rear tak, aby wskazywała na nowo utworzoną komórkę

## Operacja Dequeue (usuwa element z początku kolejki)

Załóżmy, że mamy kolejkę:

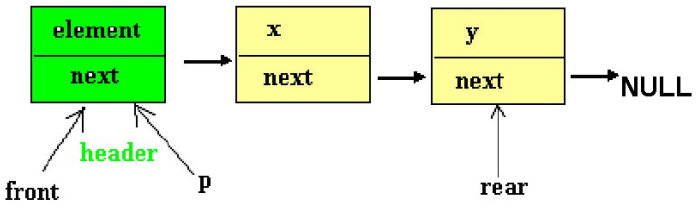



Po wywołaniu metody Q.Dequeue mamy:

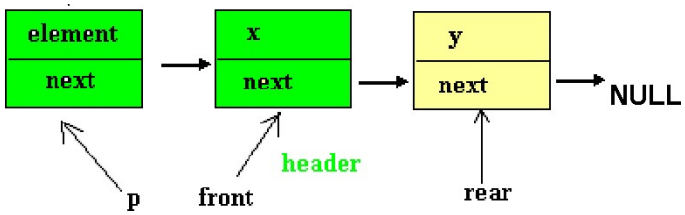


**Operacja Dequeue polega na:**

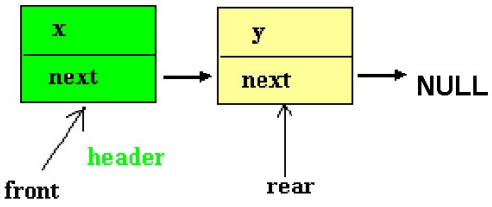
- sprawdzeniu, czy kolejka nie jest pusta
- jeżeli nie jest pusta, to
  - zapamiętujemy dotychczasową głowę w zmiennej pomocniczej p




  - aktualizujemy pole front tak, aby wskazywało na dotychczasowy pierwszy element, który od tej pory będzie głową




  - usuwamy z pamięci (zwalniamy pamięć) element wskazywany przez p (wcześniej będący głową)




// Zwróć pierwszy element z kolejki.
**elementtype Queue::FrontElem(){}**

// Sprawdź, czy kolejka jest pusta
**bool Queue::Empty()**

**Queue::~Queue()**

**ZADANIE:** Zaimplementuj podstawowe operacje na kolejsce w reprezentacji wskaźnikowej: Enqueue(x), Front(), Dequeue(), IsEmpty() oraz konstruktor i destructor.