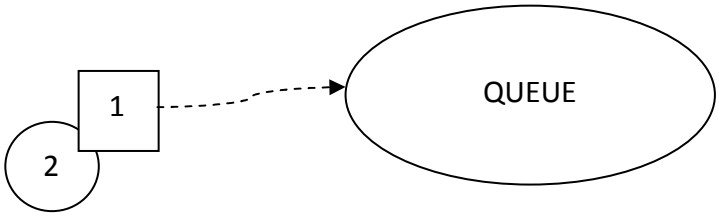
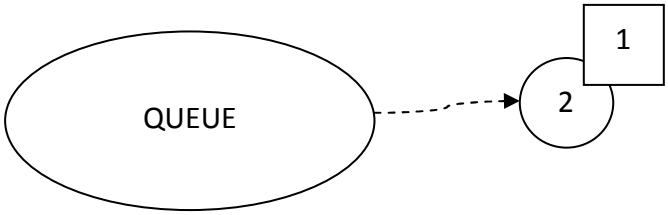
Ex. 1 以陣列實作"儲列"來儲存資料.

儲列原理: 先進先出



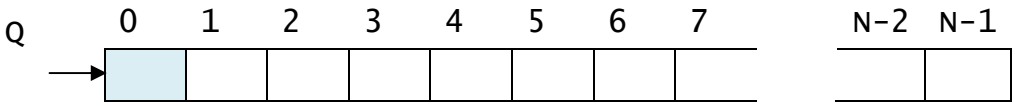
進入順序 1, 2



退出順序 1, 2

使用陣列實作, 新增資料在 rear(隊伍後面), 退出資料在 front(隊伍前面)

空儲列, 最多存 N 個

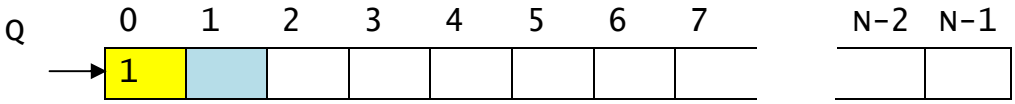


front=0

rear=0

count=0

新增 (enqueue) "1"

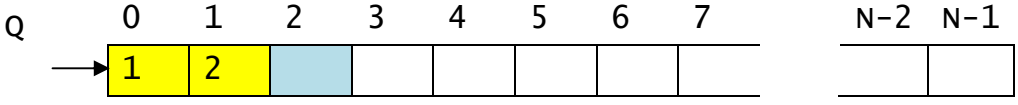


front=0

rear=1

count=1

新增 (enqueue) "2"

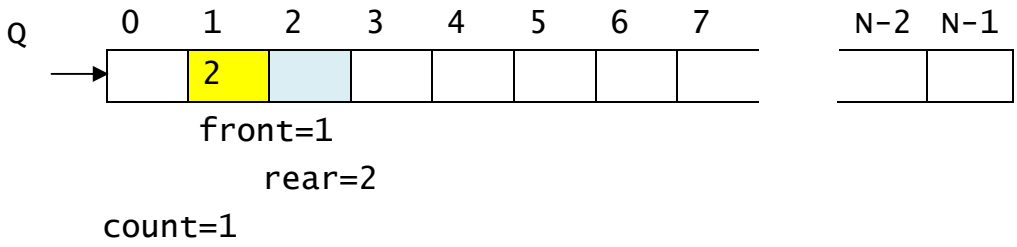


front=0

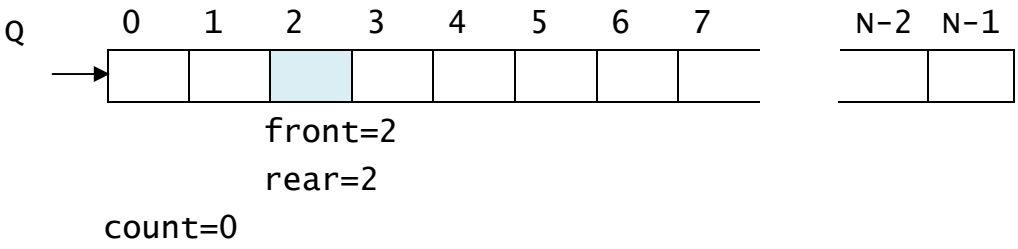
rear=2

coutnt=2

退出 (dequeue) "1"



退出 (dequeue) "2"



總結:

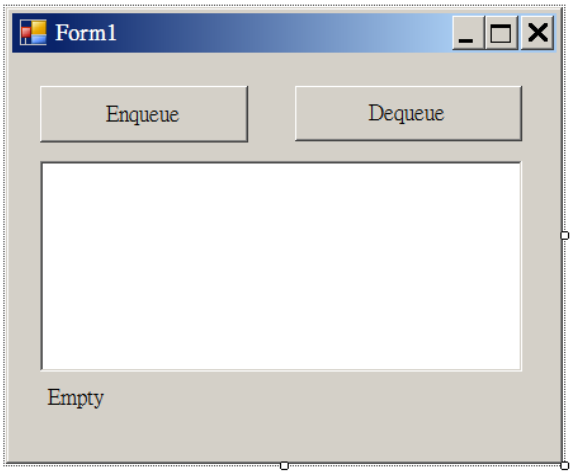
1. 空儲列情況 count=0
2. 儲列已滿 count=N
3. 當 rear, front 指標超過 N-1 時, 設為零
   (因此又名循環陣列 circular array)

（程式設計）

button1

textBox1

label1



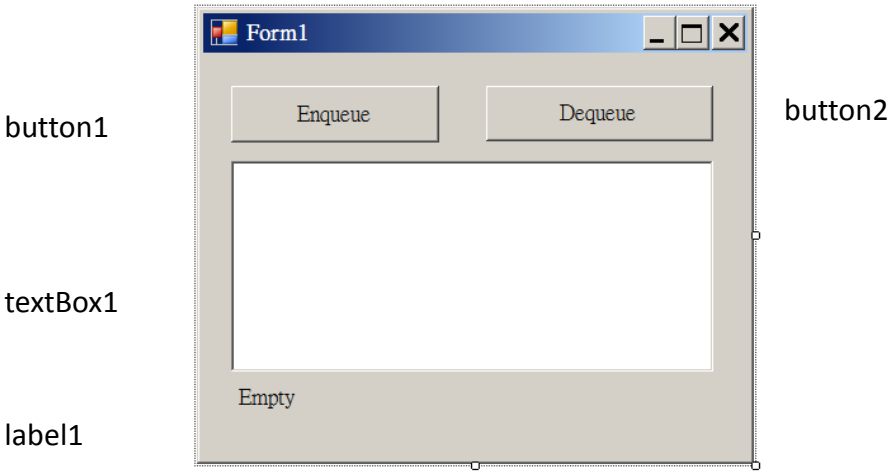
button2

(程式碼)

```
ref class QUEUE {
  private:
    int N;
    array<String^> ^name;
    int rear,front,count;
  public:
    QUEUE(){
        N=5;
        rear=0; front=0; count=0;
        name = gcnew array<String^>( N );
    }
    void Enqueue( String ^x ){   //新增資料
        if( ! IsFull() ){
            name[rear]=x;
            if( rear<N-1 ) ++rear; else rear=0;
            ++count;
        }
     }
    String^ Dequeue(){      //退出資料
        String ^x;
        if( ! IsEmpty() ) {
            x = name[front];
            name[front]="";
            if( front<N-1 ) ++front; else front=0;
            --count;
        } else x = "";
        return x;
    }
    void Clear(){
         rear=0; front=0; count=0;
    }
    bool IsEmpty(){
        if( count == 0 ) return true; else return false;
    }
    bool IsFull(){
        if( count == N ) return true; else return false;
    }
```

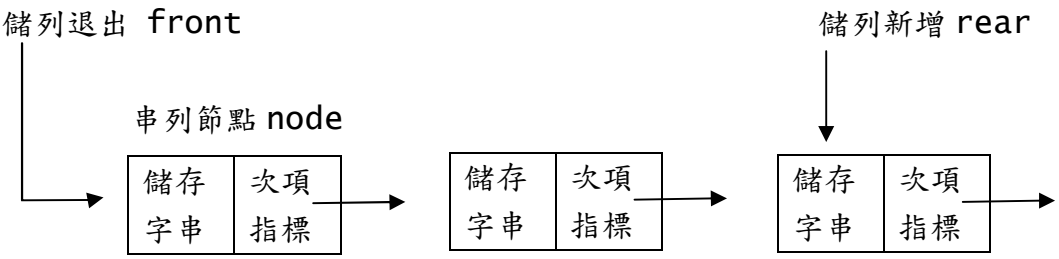
```
        int Total(){     return count;       }
        String^ Show(){
            String ^s="";
            int k;
            for(k=0; k< N; k++ ){
                s += "[" + name[k] + "]";
            }
            return s;
        }
    };
    QUEUE ^A;
    void report(){
        if( A->IsEmpty() ) label1->Text = "Empty";
        else if ( A->IsFull() ) label1->Text = "Full";
        else label1->Text = "Ready";
    }
    int B;
private: System::Void Form1_Load(System::Object^  sender,
                System::EventArgs^  e) {
        A=gcnew QUEUE;
        textBox1->Text = A->Show();
        B=0;
    }
private: System::Void button1_Click(System::Object^
                sender, System::EventArgs^  e) {
        ++B;
        A->Enqueue( B.ToString() );
        textBox1->Text = A->Show();
        report();
    }
private: System::Void button2_Click(System::Object^
                sender, System::EventArgs^  e) {
        String ^x = A->Dequeue();
        textBox1->Text = A->Show();
        report();
        if( x!="" ) label1->Text += " Got " + x;
    }
```

Ex.2 同上，以串列實作儲列，並列出相關資訊。



(資料結構)

與一般串列操作不同，改成模擬陣列型儲列的操作，新增資料在 rear，退出資料在 front。




```
class list {
        String  ^name; //儲存字串
        list    ^next; //次項指標
};

class QUEUE {
        list ^rear, ^front;
        int count; //儲列已儲存總數
}
```

開始時 front 指向無效指標(null)，rear 指向無效指標(null)，代表空串列。

加入 "1", front 指向"1", rear 指向"1"
"1" -> (null)

加入 "2", front 指向"1", rear 指向"2"
"1" -> "2" -> (null)

加入 "3", front 指向"1", rear 指向"3"
"1" -> "2" -> "3" -> (null)

退出 "1", front 指向"2", rear 指向"3"
"2" -> "3" -> (null)

退出 "2", front 指向"3", rear 指向"3"
"3" -> (null)

退出 "3", front 指向(null), rear 指向(null)
(null)

(程式碼)

```
ref class list {
 public:
   String ^name;
   list   ^next;
   list( String ^s ){
       name=s;
       next=nullptr;
   }
};
ref class QUEUE {
  private:
   list ^rear, ^front;
   int count;
  public:
   QUEUE(){ rear=nullptr; front=nullptr; count=0; }
   void Enqueue( String ^s ){        //新增資料
       list ^x = gcnew list( s );
       if( IsEmpty() ) { rear=x; front=x; }
       else {
           rear->next = x;
           rear = x;
       }
       ++count;
   }
   String^ Dequeue(){              //退出資料
       String ^s;
       if( !IsEmpty() ){
           s = front->name;
           front=front->next;
           --count;
           if( IsEmpty() ) rear=nullptr;
       } else s="";
       return s;
   }
   void Clear(){
        rear=nullptr;   front=nullptr; count=0;
   }
```

```
        bool IsEmpty(){
            if( count == 0 ) return true;
            else return false;
        }
        int Total(){
            return count;
        }
        String^ Show(){
            String ^s = "";
            list ^x=front;
            while( x !=nullptr ){
                s += x->name + "->";
                x=x->next;
            }
            s += "(null)";
            return s;
        }
    };

    QUEUE ^A;
    void report(){
        if( A->IsEmpty() ) label1->Text = "Empty";
        else label1->Text = "Ready";
    }
    int B;
private: System::Void Form1_Load(System::Object^  sender,
                System::EventArgs^  e) {
        A=gcnew QUEUE;
        textBox1->Text = A->Show();
        B=0;
    }
private: System::Void button1_Click(System::Object^
                sender, System::EventArgs^  e) {
        ++B;
        A->Enqueue( B.ToString() );
        textBox1->Text = A->Show();
        report();
    }
```

```
private: System::Void button2_Click(System::Object^
                    sender, System::EventArgs^  e) {
           String ^x = A->Dequeue();
           textBox1->Text = A->Show();
           report();
           if( x!="" ) label1->Text += " Got " + x;
         }
```