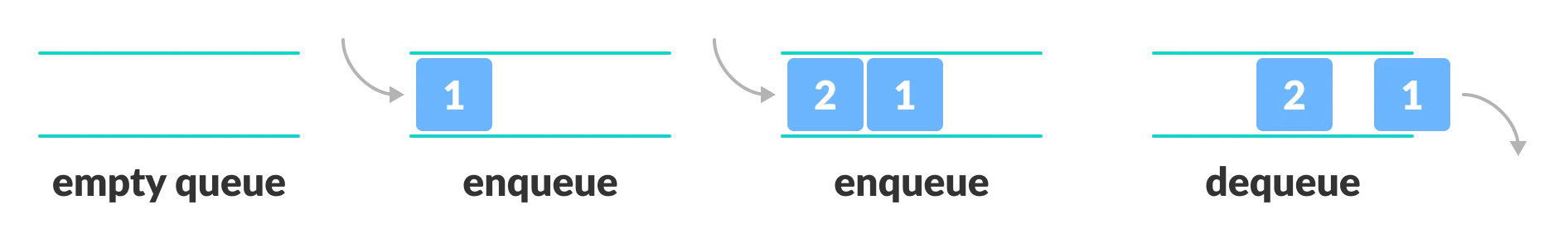
**Queue**

يعتبر الqueue أو ما يسمى بالرتل هي بنية معطيات خطية تعتبر مفيدة جدا في البرمجة وهي تشبه طابور التذاكر في قاعة السينما

حيث يكون أول شخص يدخل قائمة الانتظار هو أول شخص يحصل على التذكرة

حيث تتبع قاعدة FIFO العنصر الذي يدخل أولا هو العنصر الذي يخرج أولا



في الصورة الموجودة في الأعلى عندما تم إضافة العنصر الأول 1 قبل العنصر الثاني 2 فنجد أن العنصر الأول هو الذي سوف تتم ازالته أولا من الرتل

في البرمجة وضع العناصر في الرتل يدعى enqueue وإزالة العناصر من قائمة الانتظار يدعى dequeue

العمليات الأساسية في الارتال :

إضافة عنصر إلى نهاية الرتل : **enqueue**

حذف عنصر من بداية الرتل : **dequeue**

التحقق إذا كان الرتل فارغ:**IsEmpty**

التحقق إذا كان الرتل ممتلئ:**IsFull**

الحصول على قيمة أخر عنصر بدون حذفه:**Peek**

طريقة عمل الارتال :

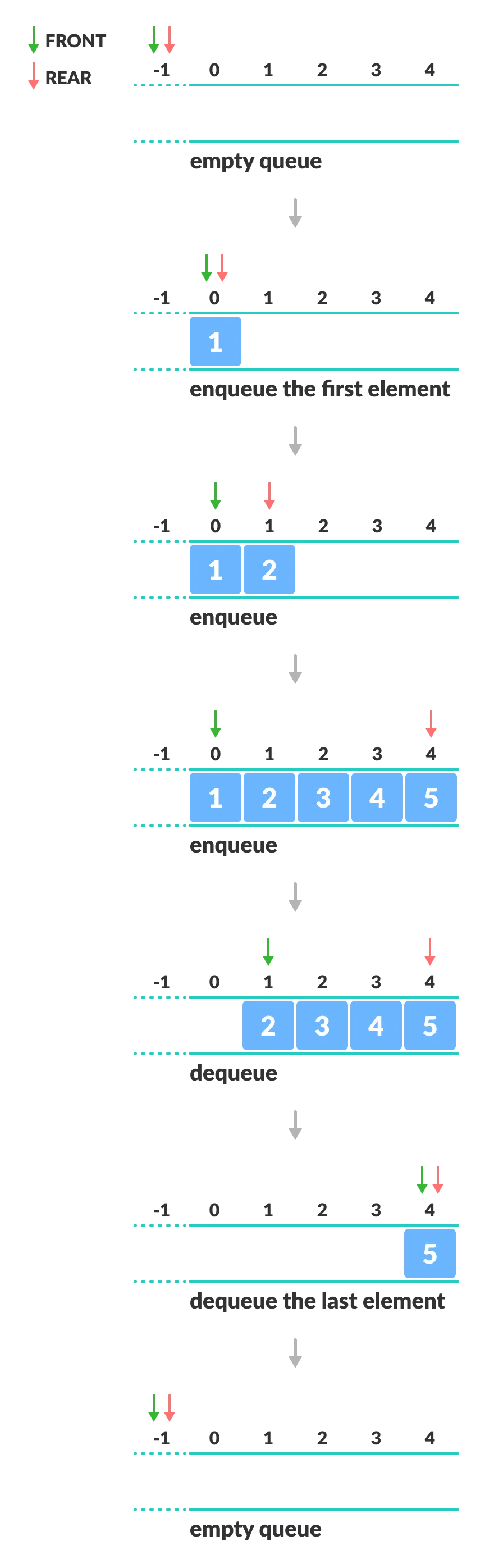
* يتم إنشاء مؤشرين FRONT و REAR
* FRONT يقوم بتتبع العنصر الأول في الرتل
* REAR يقوم بتتبع العنصر الأخير في الرتل
* نقوم بإعطاء قيمة 1- لكل من FRONT و REAR

**عملية الإضافة :**

* نتحقق إذا كان الرتل ممتلئ
* للعنصر الأول نجعل قيمة FRONT 0
* نزيد قيمة REAR بمقدار 1
* إضافة العنصر الجديد في الموقع الذي يشير إليه REAR

**عملية الحذف:**

* التحقق إذا كان الرتل فارغ
* إعادة القيمة التي يشير إليها FRONT
* زيادة قيمة FRONT بمقدار 1
* اذا كنا نريد حذف اخر عنصر في الرتل نجعل قيمة كل من FRONT و REAR 1-



عادة نقوم بتطبيق الارتال بواسطة المصفوفات :

// Queue implementation in C++

#include <iostream>

#define SIZE 5

using namespace std;

class Queue {

private:

int items[SIZE], front, rear;

public:

Queue() {

front = -1;

rear = -1;

}

bool isFull() {

if (front == 0 && rear == SIZE - 1) {

return true;

}

return false;

}

bool isEmpty() {

if (front == -1)

return true;

else

return false;

}

void enQueue(int element) {

if (isFull()) {

cout << "Queue is full";

} else {

if (front == -1) front = 0;

rear++;

items[rear] = element;

cout << endl

<< "Inserted " << element << endl;

}

}

int deQueue() {

int element;

if (isEmpty()) {

cout << "Queue is empty" << endl;

return (-1);

} else {

element = items[front];

if (front >= rear) {

front = -1;

rear = -1;

} /\* Q has only one element, so we reset the queue after deleting it. \*/

else {

front++;

}

cout << endl

<< "Deleted -> " << element << endl;

return (element);

}

}

void display() {

/\* Function to display elements of Queue \*/

int i;

if (isEmpty()) {

cout << endl

<< "Empty Queue" << endl;

} else {

cout << endl

<< "Front index-> " << front;

cout << endl

<< "Items -> ";

for (i = front; i <= rear; i++)

cout << items[i] << " ";

cout << endl

<< "Rear index-> " << rear << endl;

}

}

};

int main() {

Queue q;

//deQueue is not possible on empty queue

q.deQueue();

//enQueue 5 elements

q.enQueue(1);

q.enQueue(2);

q.enQueue(3);

q.enQueue(4);

q.enQueue(5);

// 6th element can't be added to because the queue is full

q.enQueue(6);

q.display();

//deQueue removes element entered first i.e. 1

q.deQueue();

//Now we have just 4 elements

q.display();

return 0;

}

**التعقيد الزمني ل Queue:**

تعقيد عمليات الإضافة والحذف باستخدام المصفوفات يستغرق زمنا ثابتا O(1)

**تطبيقات ال Queue:**

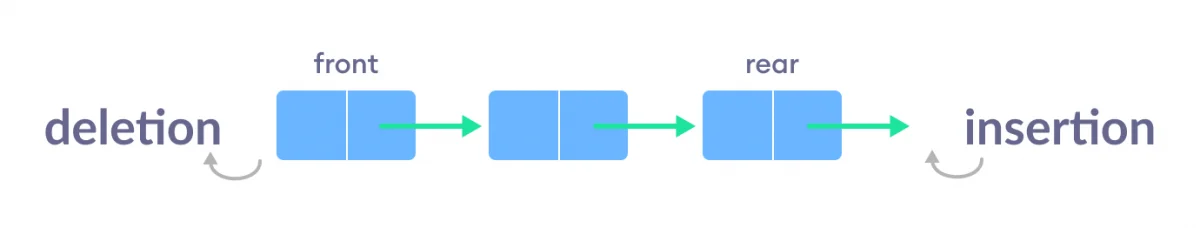
* جدولة وحدة المعالجة المركزية والاقراص
* عندما يتم نقل البيانات بشكل غير متزامن بين عملتين يتم استخدام الأرتال للمزامنة على سبيل المثال IO Buffers, pipes, file IO
* معالجة المقاطعات في أنظمة الوقت الحالي
* أنظمة الهاتف في مراكز الاتصال تستخدم الأرتال لتنظيم دور الاتصال

**أنواع ال Queue:**

**هناك أربعة أنواع مختلفة من الارتال :**

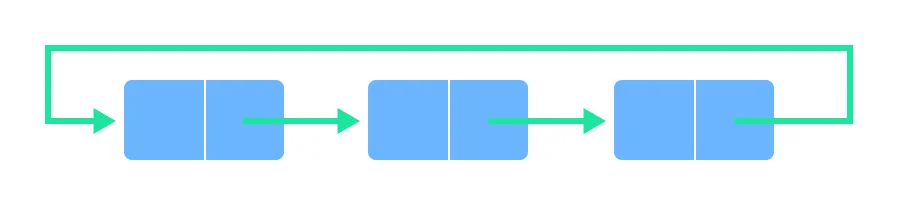
* **الارتال البسيطة:**

في هذا النوع يتم إضافة العناصر في الخلف rearوتقوم بحذف العناصر من الامامFront حيث يستخدم قاعدة FIFO



* **الارتال الدائرية:**

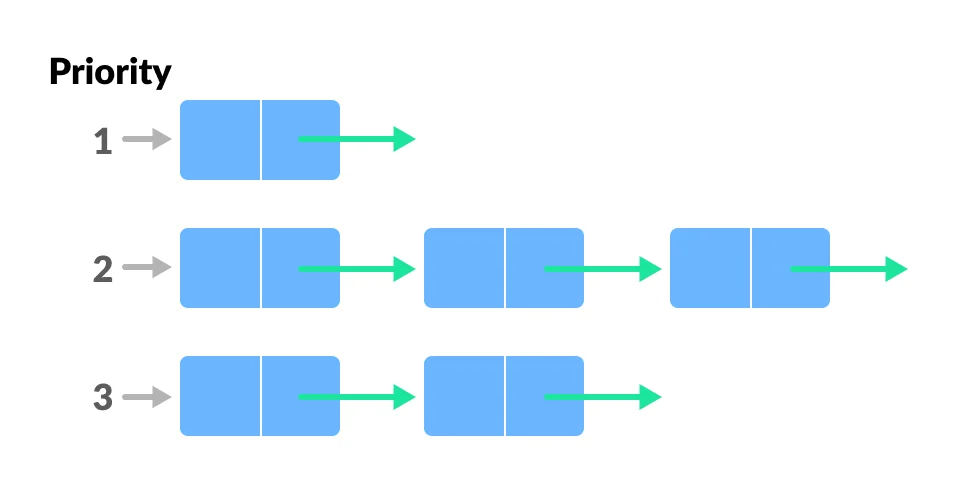
في هذا النوع يشير العنصر الأخير إلى العنصر الأول الذي يقوم بعمل ارتباط دائري ويتميز هذا النوع عن الارتال البسيطة في استخدام الذاكرة بشكل أفضل إذا كان الموقع الأخير ممتلئ والموضع الأول فارغ يمكننا أضافة عنصر في الموقع الأول وهذا الاجراء غير ممكن في النوع البسيط



* **الارتال الأولوية:**

يعتبر نوع خاص من الارتال التي يقترن كل عنصر فيها بالأولوية ويتم تقديمه وفقا للأولوية الخاصة به إذا كان للعناصر نفس الأولوية يتم تقديمها وفقا لترتيبها في الرتل

الإضافة تحدث اعتمادا على الوصول للقيم والحذف يحدث تبعا للأولوية



* **الارتال المزدوجة المنتهية:**

في هذا النوع يمكن إضافة العناصر أو حذفها من الأمام والخلف ولا يتبع قاعدة FIFO

