Queue

큐

1) 큐(Queue)는 뒤쪽으로 들어가서 앞쪽으로 나오는 자료 구조(Data Structure)입니다.
2) 이러한 특성 때문에 스케줄링, 탐색 알고리즘 등에서 다방면으로 활용됩니다.

- PUSH: 큐에 데이터를 넣습니다.
- POP: 큐에서 데이터를 빼냅니다.

1. 배열

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 100000
#define INF 999999999

int queue[SIZE];
int nFront = 0;
int nRear = 0;

void Push(int nData)
{
        if(nRear >= SIZE)
        {
                printf("Queue Overflow가 발생했습니다.\n");
                return;
        }

        queue[nRear++] = nData;
}
```

```
int Pop()
{
        if (nFront == nRear)
        {
                printf("Queue Underflow가 발생했습니다. \n");
                return -INF;
        }

        return queue[nFront++];
}

void Show()
{
        printf("--- Queue Front ---\n");
        for (int i = nFront; i < nRear; ++i)
                printf("%d\n", queue[i]);
        printf("--- Queue Rear ---\n");
}

int main(void)
{
        Push(7);
        Push(5);
        Push(4);
        Pop();
        Push(6);
        Pop();
        Show();
        system("pause");
        return 0;
```

```
}
```

1.1 결과

```
--- Queue Front ---
4
6
--- Queue Rear ---
```

2. Linked list

```
#include <stdio.h>
#include <stdlib.h>

#define INF 99999999

typedef struct _Node
{
        int nData;
        struct _Node *pNext;
} Node;

typedef struct
{
        Node *pFront;
        Node *pRear;
} Queue;

void Push(Queue *pQue, int nData)
{
        Node *pNode = (Node*)malloc(sizeof(Node));
        pNode->nData = nData;
        pNode->pNext = NULL;
```

```
        if (NULL == pQue->pFront)
        {
                pQue->pFront = pNode;
                pQue->pRear = pNode;
        }
        else
        {
                pQue->pRear->pNext = pNode;
                pQue->pRear = pNode;
        }

        printf("Push : %d\n", nData);
}

int Pop(Queue *pQue)
{
        if (NULL == pQue->pFront)
        {
                printf("Queue underflow");
                return -INF;
        }

        Node *pFront = pQue->pFront;
        pQue->pFront = pFront->pNext;

        int nData = pFront->nData;
        free(pFront);
```

```
        printf("Pop : %d\n", nData);
        return nData;
}

void Show(Queue* pQue)
{
        Node *pCur = pQue->pFront;

        printf("--- Queue Front ---\n");
        while (NULL != pCur)
        {
                printf("%d\n", pCur->nData);
                pCur = pCur->pNext;
        }
        printf("--- Queue Rear ---\n");
}

int main(void)
{
        Queue que;
        que.pFront = que.pRear = NULL;

        Push(&que, 7);
        Push(&que, 5);
        Push(&que, 4);
        Pop(&que);
        Push(&que, 6);
        Pop(&que);

        Show(&que);
```

```
        system("pause");
        return 0;
}
```

2.1 결과

```
Push : 7
Push : 5
Push : 4
Pop : 7
Push : 6
Pop : 5
--- Queue Front ---
4
6
--- Queue Rear ---
```