Prob1-1.cube

|  |
| --- |
| I\*LR'FFBBLR'DD=P |

이 과제는 무리없이 수행할 수 있었다. I 인자를 통해 값을 입력받고 \*인자를 통해 LOAD의 과정을 거쳐 데이터 셀 주변 비트 셀에 입력한 값의 이진수를 한 바이트씩 시계방향으로 입력한다. 그 후 반대편 면으로 순서가 바뀌지 않게 옮긴 후, =인자를 통해 SAVE의 과정을 거쳐 비트 셀의 값을 데이터 셀로 입력하여 P를 통해 데이터 셀의 값을 출력하는 과정을 거치는 코드이다.

Prob1-2.cube

|  |
| --- |
| I\*DU'LR'D'UL'RX\*DU'LR'D'UL'R=P |

이 과제는 1의 보수를 구하는 문제였다. 1의 보수는, 어떤 수를 이진법으로 나타내었을 때 0은 1, 1은 0으로 NOT 게이트를 통하여 나타낸 수이다. 위에서 작성한 코드는 입력을 받고, 큐브의 NOT 게이트 연산을 할 수 있는 옆면으로 옮기고, 다시 옆면으로 이동하여 출력하는 기능을 한다.

visual studio를 텍스트 편집기로 이용하여 Queue 알고리즘을 구현하는 코드를 작성하였다.

1. Add 함수

큐에서 데이터를 추가하는 함수의 연산은 다음과 같이 진행된다.

1) 큐의 size를 증가시킨다.

2) 데이터가 들어갈 head 인덱스를 다음 번호로 점점 늘린다.

3) Head에 해당하는 배열에 데이터를 추가한다.

|  |
| --- |
| size++;  head++;  CircularQueue[head] = value;  return 0; |

2. pop 함수

큐에서 데이터를 추가하는 함수의 연산은 다음과 같이 진행된다.

1) 삭제되는 데이터를 remove\_data 변수에 저장한다.

2) 데이터가 나올 tail 인덱스를 다음 번호로 점점 늘린다.

3) 큐의 사용 사이즈를 줄인다.

4) remove\_data 변수를 반환한다.

|  |
| --- |
| int remove\_data = CircularQueue[tail];  tail++;  size--;  return remove\_data; |

- 첫 번째 문제점은 예외처리할 경우를 생각하지 못했 다는 점이다. 원형 큐 같은 경우 값을 할당하다가 capacity보다 size가 커지게 되면 다시 처음 메모리로 돌아와야 한다. 하지만 무한한 선형 큐의 함수를 짰고, 그 결과 원형 큐를 구현할 수 없었다. 포인터와 배열에 관련한 지식이 부족한 점을 깨달아, 조원 친구에게 자세한 설명을 부탁했다.

- 두 번째 문제점은 python에 비해 C 관련 지식이 부족함이 있다는 것이다. 남은 자기 관리 시간에 C++을 집중적으로 공부하기로 계획을 세울수 있었다.