□ 프로그램 구현 환경

windows 10 운영체제에서 visual studio 2017를 이용해 c언어로 작성.

프로젝트는 1차 과제 때 만든 더블링크드와 배열리스트가 각각 포함된 스택과 큐를 구현한 소스파일로 구성.

□ 함수 설명 및 시간 복잡도

|  |  |  |
| --- | --- | --- |
| **자료구조**  **함 수** | **Stack** | |
| StackInit | 스택 초기화 함수 : 스택의 리스트를 새로 할당하고  그 리스트를 LisiInit메서드로 초기화 | |
| Empty | 스택이 비어있는지 확인하는 함수 :  numOfData가 0이면 TRUE 출력, 아니면 FALSE출력. | |
| Top | 가장 나중에 들어간 데이터를 반환. 스택이 비어있으면 반환할 데이터가  없으므로 -1을 반환. 리스트의 head로 데이터가 삽입되기 때문에  가장 먼저 삽입이 된 데이터는 계속해서 밀리게 되고 tail이 가리키게 되고,  가장 최근에 삽입된 데이터는 head가 가리키게 되므로  head 포인터가 가리키는 데이터를 반환한다. | |
| front | head 포인터 조회 O(1) |
| back | - |
| k | - |
| Pop | 가장 나중에 삽입된 데이터를 삭제 및 반환 : 스택이 비어있으면 꺼낼  데이터가 없으므로 -1을 반환. head가 가리키는 데이터가 가장 최근에  삽입된 데이터 이므로 head 포인터를 통해서 데이터를 Erase함수를 통해서  데이터를 삭제하고 반환한다. | |
| front | head포인터 조회 후 삭제. 링크드리스트 맨 앞 삭제 시간은 상수 시간이므로 O(1) |
| back | - |
| k | - |

|  |  |  |
| --- | --- | --- |
| Push | 리스트의 head가 가리키는 곳 앞에 데이터를 삽입. Insert\_Head함수를  호출해서 데이터를 헤드에 삽입한 후 head포인터가 최근에 삽입한 데이터를  가리키도록 한다. | |
| front | head 앞에 삽입할 때 head 포인터를 바로 조회가능하므로 상수시간 O(1) |
| back | - |
| k | - |
| Size | 스택에 들어있는 데이터의 개수 확인 | |
| Reverse\_Polish\_  Notation | 계산할 행을 문자배열 array에 담고 이 배열에 0번째부터 차례대로 하나씩 문자를 가져와 해당 문자가 공백, 숫자, 연산자인지 조사해 문자가 어떤 형태인지에 따라 적절한 분기로 나눠 처리하였다. 우선 스택에 데이터가 존재한다면 해당 데이터를 모두 Pop해서 Empty상태로 만든다. 해당 문자가 숫자면 스택에 저장한다. 숫자가 아닌 연산자라면 스택에서 두 데이터를  Pop해와 연산자 종류에 맞게 연산을 수행하고 다시 Stack에 Push한다.  수행시간은 while문이 반복되는 횟수에 비례되는데 계산할 데이터의 수에 따라 반복되므로 수에 의해 결정된다. 숫자를 골라내기 위한 while문은 반복되는 횟수가 데이터의 수에 비하면 적으므로 따라서 수행복잡도는 O(n)이다.(n:데이터 수) while문 내부에 Stack 함수들은 상수시간이므로 미치지 않는다. | |

□ 함수 설명 및 시간 복잡도

|  |  |  |
| --- | --- | --- |
| **자료구조**  **함 수** | **Queue** | |
| QueueInit | 큐 초기화 함수 : 사용할 큐를 초기화 하는 메서드  삭제할 때 쓸 front와 삽입할 때 쓸 rear 포인터들을 0으로 초기화.  데이터를 저장해놓을 리스트를 ListInit으로 초기화 | |
| Empty | 큐가 비어있는지 확인하는 함수 :  numOfData가 0이면 TRUE 출력, 아니면 FALSE출력. | |
| Front | 가장 최근에 삽입된 데이터를 반환한다. 데이터가 존재하지 않으면  -1 반환. front 포인터를 한 칸 옮겨서 참조하는데 원형배열이므로 front가 배열크기 index를 넘어갈 시에 배열크기로 모듈러해서 적절히 조정한다. | |
| front | front index를 통한 배열 참조는 상수시간 O(1) |
| back | - |
| k | - |
| Back | 가장 최근에 삽입된 데이터를 반환한다. 데이터가 존재하지 않으면  -1 반환. rear 포인터를 통해 참조. | |
| front | - |
| back | rear index를 통한 배열 참조는 상수시간 O(1) |
| k | - |
| Pop | front가 가리키는 곳에서의 데이터를 꺼낸다. 큐가 비어있을시 -1을 반환  front 포인터를 한 칸 옮겨서 참조한 데이터를 임시 데이터 변수에  저장해 놓고 그 곳을 -1로 설정한 후 개수를 하나 줄인다. | |
| front | front index를 통한 배열 참조는 상수시간 O(1) |
| back | - |
| k | - |
| Push | rear가 가리키는 곳으로 데이터를 삽입한다. 큐가 꽉차게 되면 아무것도  하지 않는다. rear 포인터를 한 칸 옮겨서 그곳에 넣는다.  개수를 하나 증가시킨다. | |
| front |  |
| back | rear index를 통한 배열 참조는 상수시간 O(1) |
| k | - |

|  |  |
| --- | --- |
| Breath\_First\_  Traversal | 큐가 비어있지 않으면 큐를 비운다. 입력받은 노드의 수가 0개면 종료하고  1개면 루트노드의 데이터만 출력 후 종료한다. parent변수로 차례로  parent데이터와 일치하는 데이터를 갖고있는 arrParent에서의  인덱스를 찾고 이 인덱스를 이용해 2차원 배열에서 자식노드들을 찾는다. 자식노드들을 큐에 Push한 후 (-1을 제외한) 큐에 넣은 데이터 중 가장 먼저 넣어진 노드를 출력하고 꺼내면서  다음에 찾을 parent로 설정해 놓고 반복한다.  while문 안에서 반복문은 for문으로 numOfNode만큼 반복하고  큐의 메서드들은 상수시간에 이뤄지므로 O(n)(n:노드의 수) |