DataStructure

|  |  |
| --- | --- |
| 프로젝트명 | 자료구조 구현 |
| 제작자 | 김성원 |
| 플랫폼 | Windows |
| 개발 언어 | C++ |
| 개발 도구 | Visual Studio 2017 |
| 세부 구현내역 | Stack, Queue, List, Vecotr |

목차

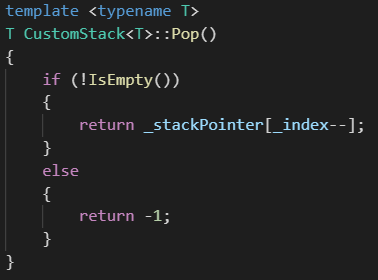
1. Stack
2. Stack의 개념
3. Stack의 연산
4. Stack의 실행
5. Queue
   1. Queue의 개념
   2. Queue의 연산
   3. Queue의 실행
6. List
   1. List의 개념
   2. List의 연산
   3. List의 실행
7. Vector
   1. Vector의 개념
   2. Vector의 연산
   3. Vector의 실행
8. Stack
9. Stack의 개념

Stack은 한 쪽 끝에서만 데이터를 넣고 뺄 수 있는 LIFO 형식의 자료구조이다.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| front |  |  |  |  |  |  |  |  | end |

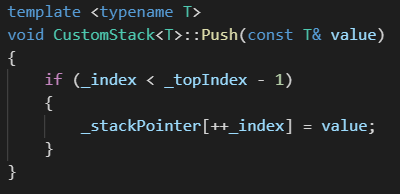
1. Stack의 연산 및 구현

Pop() : 스택에서 가장 위 항목을 제거한다.



* 윗 항목 제거 후 스택의 길이를 1 감소시킨다.

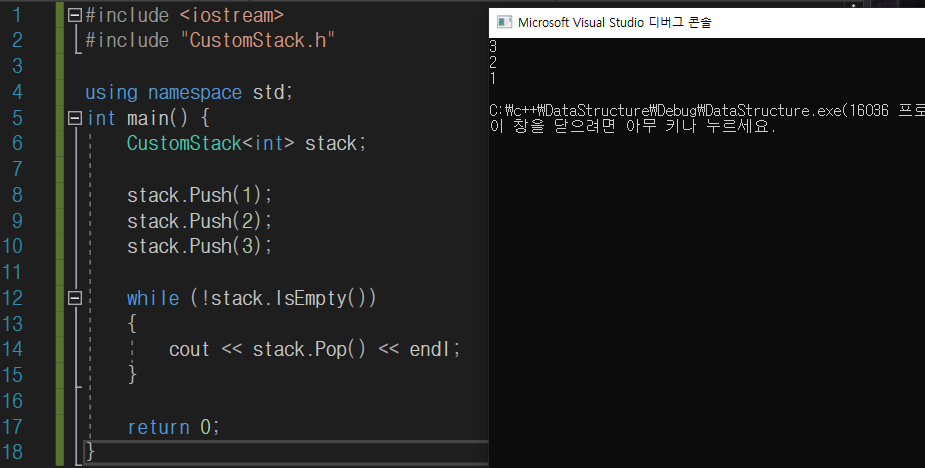
Push(value) : value 값을 스택의 가장 윗부분에 추가한다.



* 큐의 가장 윗부분에 value 값을 넣고 길이를 1 증가시킨다.

isEmpty() : 스택이 비어 있을 때 true를 반환한다.

1. Stack의 실행

Stack 전체 코드 :

<https://github.com/Ho-rang/DataStructure>/DataStructure/DataStructure/CustomStack.h

1. Queue
2. Queue의 개념

Queue는 먼저 넣은 데이터가 먼저 나오는 FIFO 형식의 자료구조이다.

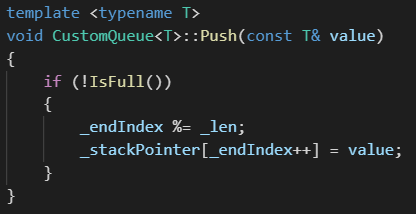
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| front |  |  |  |  |  |  |  |  | end |

-선형 큐의 문제점을 해결하기 위해 원형 큐를 구현하였다.

1. Queue의 연산

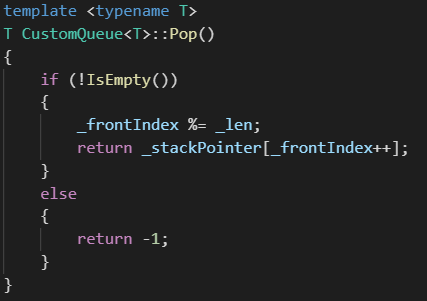
Pop() : 큐에서 첫번째 항목을 제거한다.

Push(value) : 큐의 마지막에 value 값을 추가한다.



* 큐가 가득 차 있지 않을 때 value를 넣고 끝 지점을 1 증가시킨다.

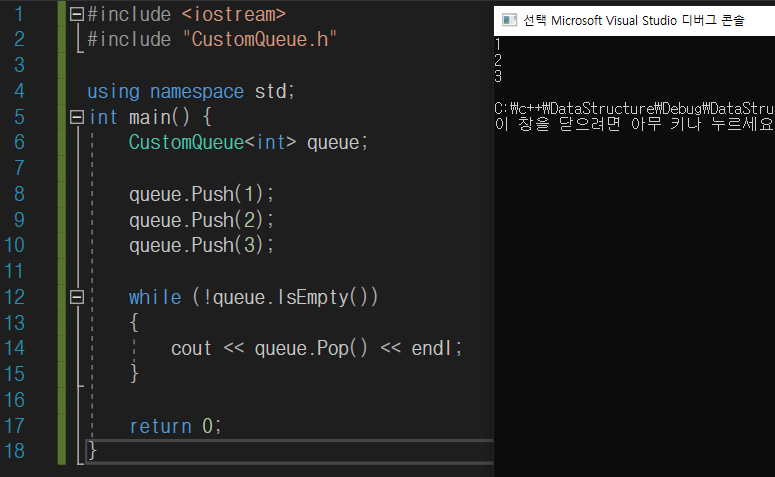
isEmpty(): 큐가 비어 있을 때에 true를 반환한다.



* 큐가 비어 있지 않을 때 맨 앞의 값을 꺼내고 시작 지점을 1증가시킨다.

isFull() : 큐가 가득 차 있을 경우 true를 반환한다.

1. Queue의 실행



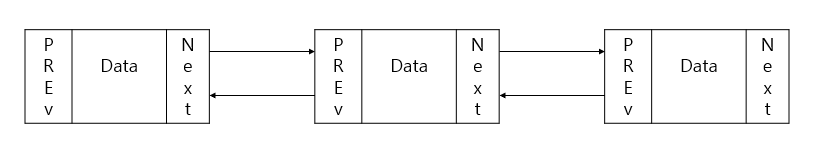
Queue 전체 코드 :

<https://github.com/Ho-rang/DataStructure>/DataStructure/DataStructure/CustomQueue.h

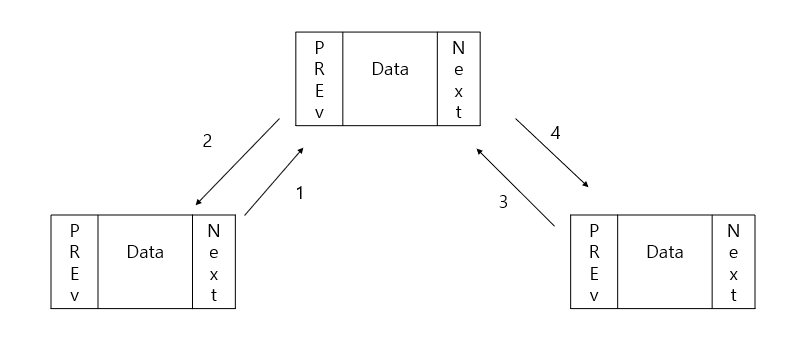
1. List(double linked list)
2. List의 개념

불연속한 메모리 공간을 효율적으로 사용하기 위한 자료구조이다.

노드들은 변수와 함께 이전 노드의 포인터와 다음 노드의 포인터를 갖고 있어야 한다.



Double linked list의 노드 삽입 방법



1. List의 연산

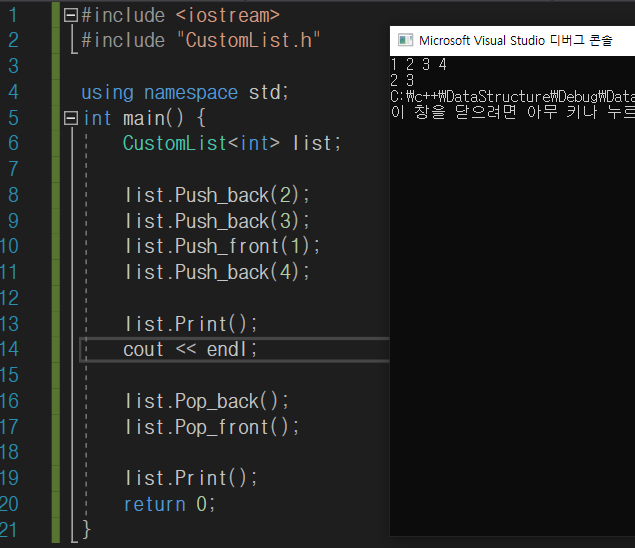
Push\_back(value) : value 값을 리스트의 마지막에 추가시킨다.

Push\_front(value) : value 값을 리스트의 앞에 추가시킨다.

Pop\_back() : 리스트의 마지막 항목을 제거한다.

Pop\_front() : 리스트의 처음 항목을 제거한다.

1. List의 실행



List 전체 코드 :

<https://github.com/Ho-rang/DataStructure>/DataStructure/DataStructure/CustomList.h

1. Vector
2. vector의 개념

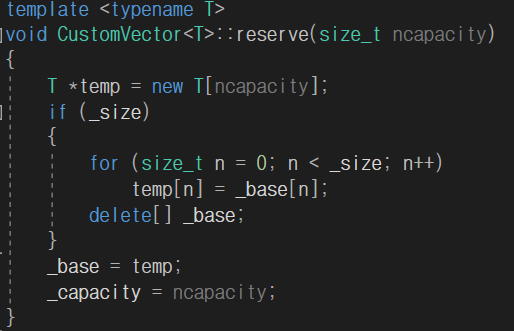
STL의 대표적인 라이브러리로 유동적으로 크기 조절이 가능하고 스택과 구조가 비슷하며 원소가 메모리 블록에 연속하게 저장되어 임의접근이 가능하다.

1. vector의 연산

push\_back(value) : vector의 마지막에 value 값 추가

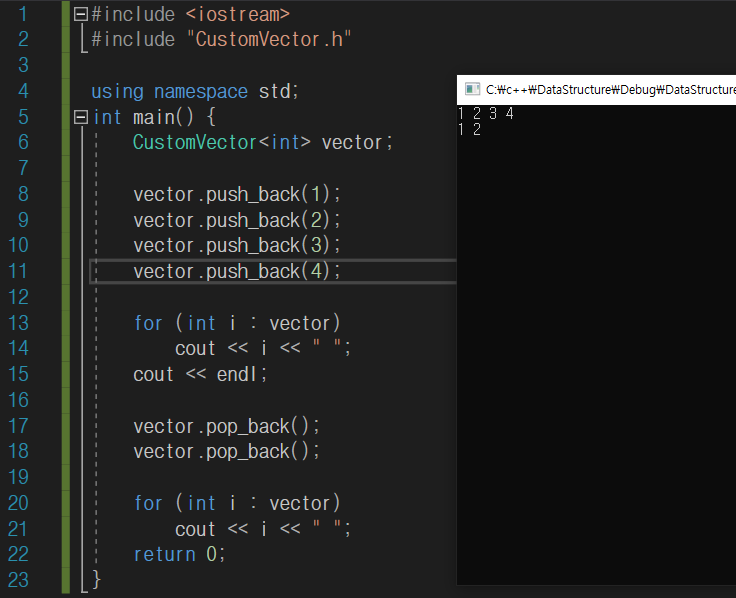
pop\_back() : vector의 마지막 원소 삭제

reserve(size) : size의 크기로 다시 메모라 할당



* 새로운 메모리를 할당해 기존 요소들을 복사 후 새로운 메모리 대입

1. Vector의 실행



Vector 전체 코드 :

<https://github.com/Ho-rang/DataStructure>/DataStructure/DataStructure/CustomVector.h