 - 과제 #3 : 프로그램 4.6과 Chain, ChainIterator 구현하기

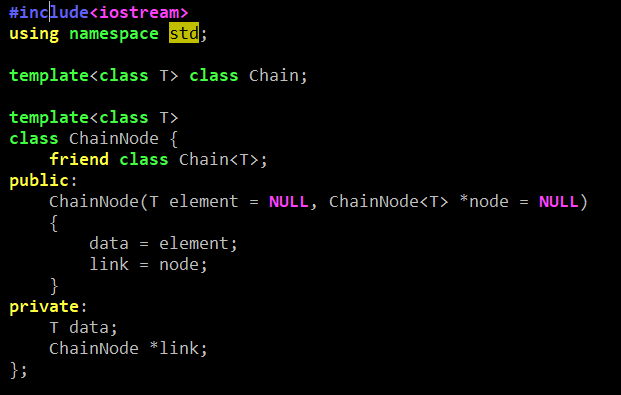
|  |
| --- |
|  |
| 자료구조 |
|  |
| 과제 #3 |

2017.05.18 제출

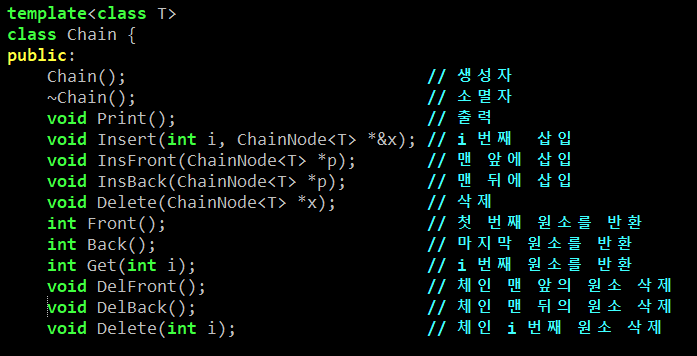
컴퓨터학부 나반

20162481 안승훈

1. 소스코드 설명

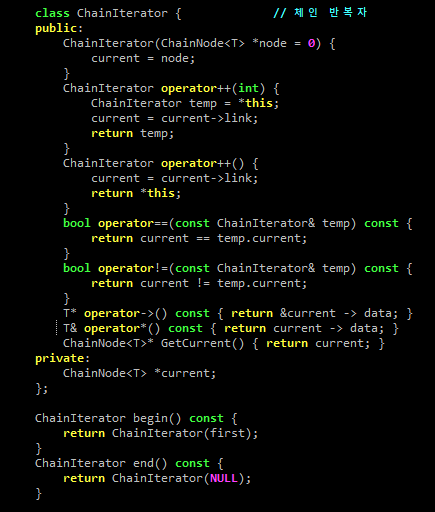


- ChainNode를 구성하기 위한 Class 이다. ChainNode에 필요한, Data 항목과 다음으로 이어지기 위한 link를 가지고 있다.



- Chain Class에 적용되어 있는 함수의 목록이다.

- 자세한 함수의 구성은 후에 서술되어있다.

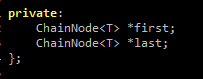


- Chain Class 내에 선언되어 있는 Chain Iterator(반복자) Class이다.

- 여러 가지 연산자의 오버로딩과 begin() end()함수가 선언되어있다.

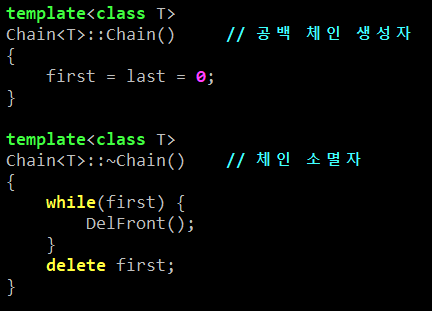
- 전, 후위 연산자와 등호, 부등호, ->, \* 등의 연산이 오버로딩 되어있다.

- 반복자는 Chain을 이용하기 위한 일종의 Cursor의 역할을 수행한다.



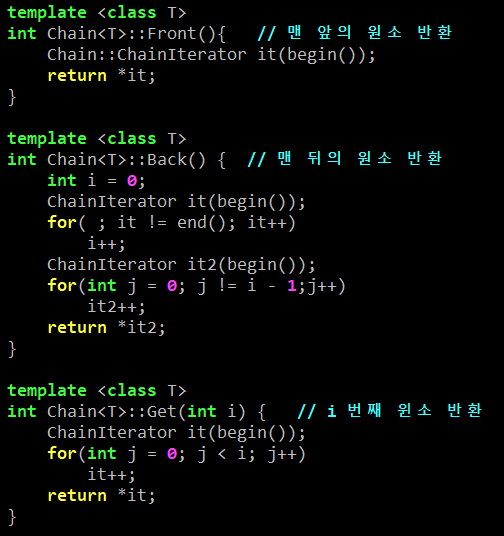
- Chain Class의 private원소로 first와 last를 선언하였다.

- 이 두 개의 ChainNode<T>를 통하여 체인의 시작과 끝을 알 수 있게 하였다.



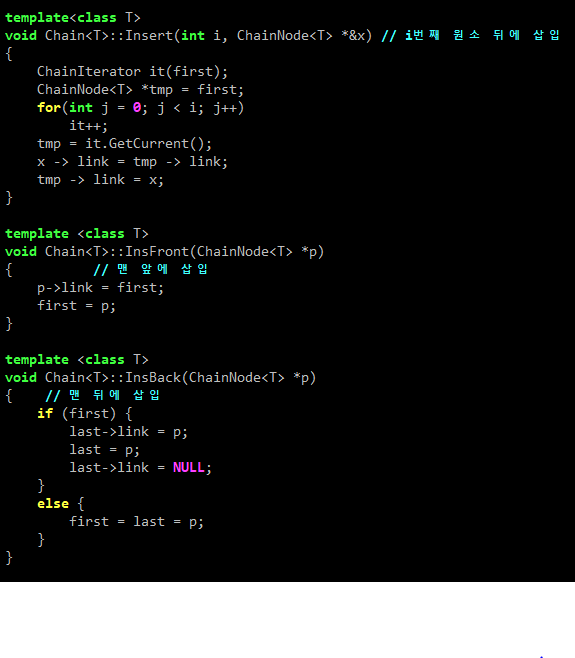
- 공백 체인을 생성하는 생성자와 Chain을 삭제하기 위한 소멸자이다.

- 소멸자의 DelFront() 는 체인의 가장 앞 노드를 삭제하는 함수이다. 이 함수를 통해 마지막 노드까지 삭제하며 진행한 다음, 마지막으로 남은 노드를 삭제하여 Chain을 소멸시킨다.



- Front(), Back(), Get() 함수의 정의부이다.

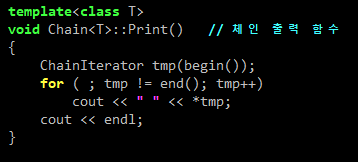
- 각각의 역할은 주석에 쓰여있는 것과 같다.



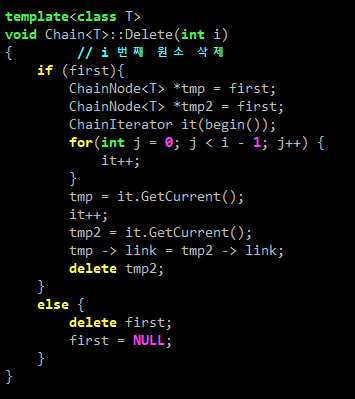
- Insert()는 체인의 i번째에 삽입하기 위한 함수이다.

- InsFront()는 체인의 맨 앞에 삽입하기 위한 함수이다.

- InsBack()은 체인의 맨 뒤에 삽입하기 위한 함수이다.



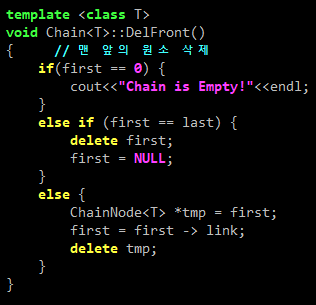
- Chain의 출력을 위한 함수이다. 반복자는 통해 순차적으로 출력한다.



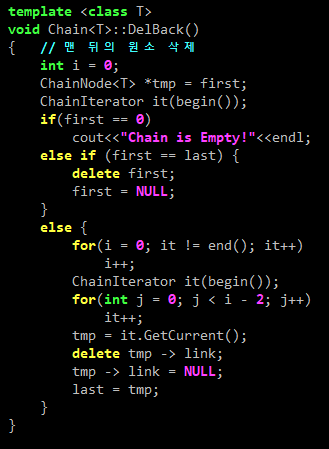
- Delete함수는 Chain의 i번째 원소를 삭제하기 위한 함수이다.

- ChainIterator의 GetCurrent함수를 통하여 Node의 정보를 불러온 다음, 삭제를 진행한다.

- i번째의 삭제 이후, i의 전 노트에서 i의 다음 노드로 link를 연결한다.

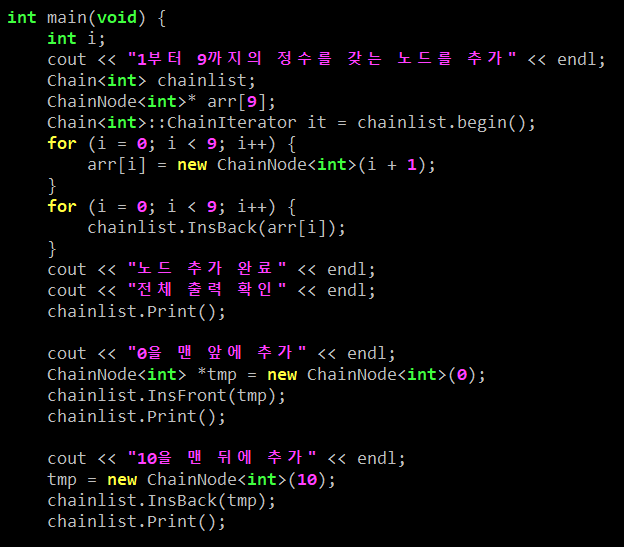
  
- DelFront() 함수는 Chain의 가장 앞 원소를 삭제한다.

- 체인의 원소가 없을 경우, 메시지를 출력하고, Node가 1개일 경우 그 원소를 삭제하고, Node가 2개 이상일 경우 가장 앞에 있는 Node를 삭제한다.



- DelBack() 함수는 맨 뒤의 원소를 삭제하는 원소이다.

- 반복자를 통해서 맨 뒤의 원소까지의 개수를 계산하고, 계산한 개수를 통하여 마지막 노드까지 이동한다. 이후 마지막 노드를 삭제하고, last를 재설정하는 것으로 마무리 한다.

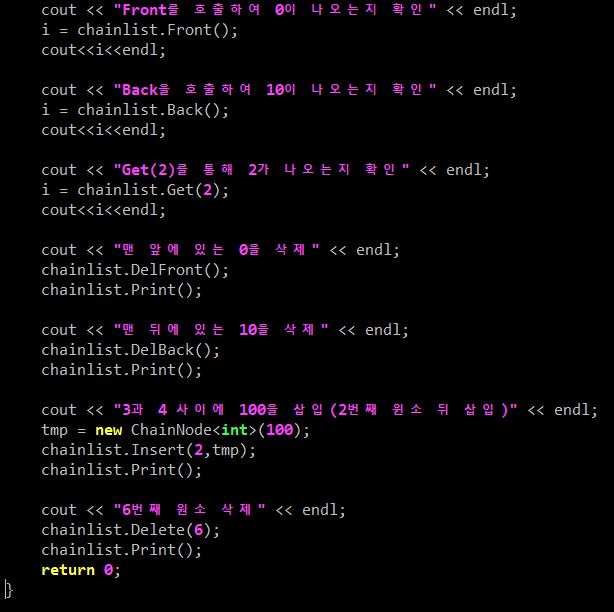


- Main 함수의 전반부이다. 1~9까지의 숫자가 data로 저장된 ChainNode를 배열로 구성한 다음 이 배열을 Chain에 추가하여 초기 Chain을 생성하였다.

- 이후에 진행되는 ChainNode의 추가, 삭제는 배열을 통하지 않고 직접 ChainNode에 관한 연산을 통해 이루어진다.

- tmp로 0을 원소로 갖는 ChainNode와 10을 윈소로 갖는 ChainNode를 생성하여 문제의 조건에 따라 Chain에 입력한다.

- 문제의 조건에 따라 Print() 함수를 통하여 노드의 확인이 필요한 곳에서 ChainNode들의 상황을 출력을 통해 확인 할 수 있다.



- 앞에서 이어지는 main함수의 뒷부분이다.

- Front(), Back(), Get()함수로부터 반환 받아 문제의 조건에 맞는 ChainNode를 반환하였다.

- DelFront(), DelBack() 함수를 통하여 Chain앞 뒤에 있는 ChainNode를 삭제할 수 있는지를 테스트 한다.

- 2번째 원소 뒤에 또다른 데이터를 삽입할 수 있는지를, Insert() 함수를 실행시킴으로써 확인한다.

- 마지막으로 Delete() 함수가 원하는 데이터를 삭제할 수 있는지 테스트 한다.

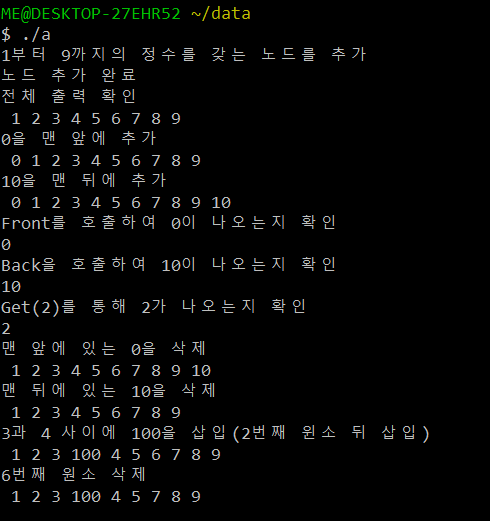
- 문제의 조건에 따라 Print() 함수를 통하여 노드의 확인이 필요한 곳에서 ChainNode들의 상황을 출력을 통해 확인 할 수 있다.

2. 컴파일 및 실행 방법 매뉴얼

- 20162481\_3.cpp 파일을 컴파일하면 프로그램을 사용할 수 있다.

- 별다른 실행 방법 없고, 컴파일 결과로 생성된 exe 파일을 실행하면 자동적으로 연산결과를 출력하고 프로그램이 종료된다.

3. 실행결과



- 문제의 제시되어있는 테스트 방법에 따라 결과를 확인 할 수 있다.

- 실행 결과로 보아 구성된 Chain이 잘 작동하였음을 알 수 있다.

4. 미 구현 부분

- 문제의 조건인 1) ~12)까지의 함수 중에 구현되지 못한 부분은 없다.

- Chain Iterator를 사용함에 있어 미흡한 부분이 존재함에 따라, 비효율적인 코드가 존재하는 부분이 있지만, 프로그램의 실행과 사용에 지장을 초래하는 문제는 없다.

- 따라서, 문제에서 요구하는 1) ~12)의 함수는 모두 구현되어있다.