DS Homework #3

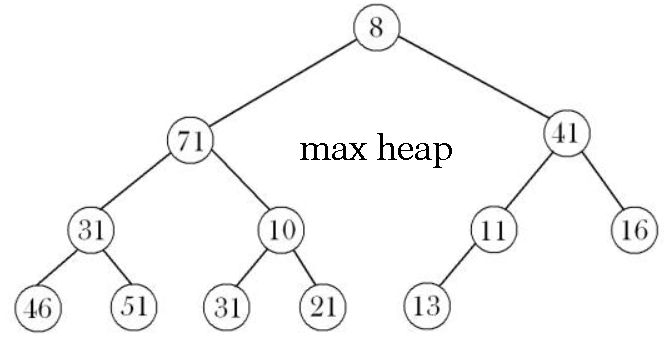
학 번 : 2015722025

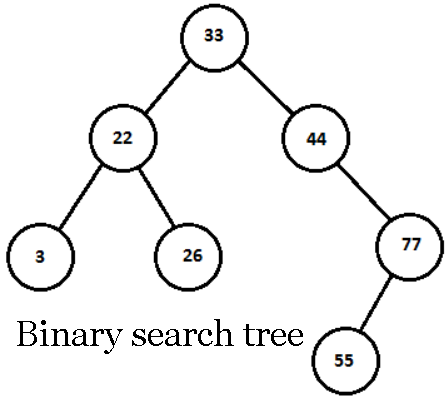
이 름 : 정용훈

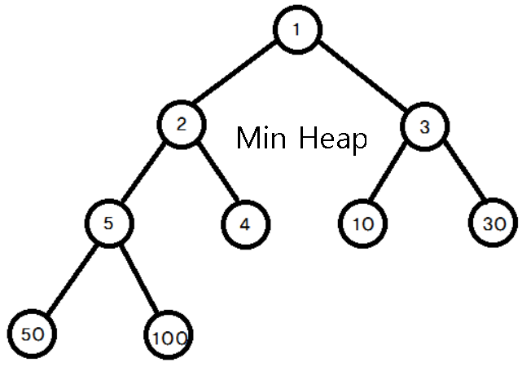
학 과 : 컴퓨터 정보공학부

1. Introduction

해당 과제는 Max Heap에 기본적인 구현과 각 함수 constructor, Push, pop 함수에 관련된 기능을 수행해보며 직접 구현해보는데 의의를 둡니다. Max heap이란 아래 그림과 같이 부모 노드의 값이 자식 노드의 값보다 크기만 하면 되는 것으로 Binary Search Tree와는 조금 다른 부분이 있습니다.

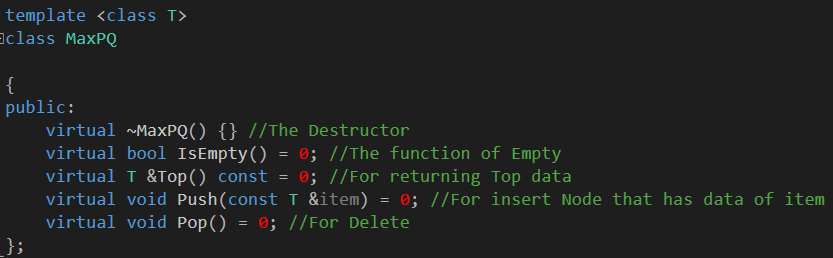


두 자료구조의 차이점을 보면 Binary Search Tree는 큰 값과 작은 값의 구분이 확실하게 되어 있습니다. 기준이 되는 노드에서 비교를 했을 때 기준 노드보다 작으면 삽입 함수를 수행할 때 왼쪽으로 노드가 이동하며 기준 노드보다 값이 크게 되면 오른쪽으로 노드가 이동하는 것을 볼 수 있습니다. 하지만 Max heap의 구조를 봤을 때는 값의 구분이 단순히 부모 노드보다 자식 노드의 KEY값이 작게 되면 해당 Tree의 구조를 성립하는 것을 이미지를 통해 확인할 수 있습니다. 또한 heap의 종류로 min heap이 있습니다. Min heap이란 max heap과 반대로 부모 노드의 KEY값이 자식 노드와 비교했을 때 값이 작으면 성립하는 heap을 말합니다. 아래 이미지를 확인 하면 쉽게 이해할 수 있습니다.

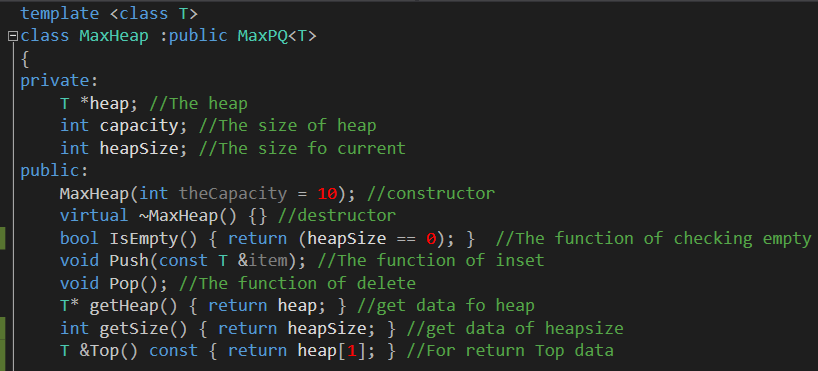
해당 이미지는 Min heap으로 부모 노드보다 자식 노드의 값이 비교했을 때 더 작은 것을 확인 할 수 있습니다.

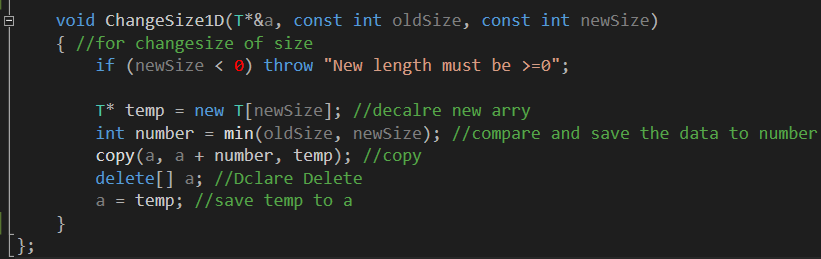
해당 자료구조를 구현하기 위해서는 함수로 Push, Pop을 사용하여 자료구조를 구현하게 됩니다. Heap구조를 구현할 때 Push와 Pop의 함수를 사용하는 이유는 우선순위 큐를 구구 할 때 Heap을 자주 사용하기 때문입니다. 우선순위 큐 란 평범한 큐나 스택과 비슷한 자료 형입니다. 그러나 각 원소들이 우선순위를 가지고 있으며 우선순위 큐에서, 높은 우선순위를 가진 원소는 낮은 우선순위를 가진 원소보다 먼저 처리가 됩니다. 만약 두 원소가 같은 우선순위를 가지게 된다면 그들은 큐에서 그들의 순서에 의해 처리가 됩니다. 또한 많은 사람들이 헷갈릴 수 있는 사항으로 “우선순위 큐를 Heap이다.” 라는 오류를 범할 수 있습니다. 우선순위 큐는 ‘리스트’, ‘맵’과 같이 추상적인 개념이며 ‘리스트’는 연결리스트나 배열로 구현될 수 있는 것과 같이, 우선순위 큐는 heap이나 다양한 다른 방법으로 구현될 수 있기 때문에 위에 해당하는 개념은 오류가 나는 것입니다. 지금 까지는 우선순위 큐, Max heap과 Min heap에 대한 개념과 프로그램을 구현하기전 알아야 할 사항을 정리하고 소개하는 문구였습니다. 다음 문항으로 직접 Max heap을 구현하고 프로그램의 결과 화면을 보면서 Heap에 대하여 조금 더 자세하게 알아보겠습니다.

1. 프로그램 결과화면 및 소스코드 설명



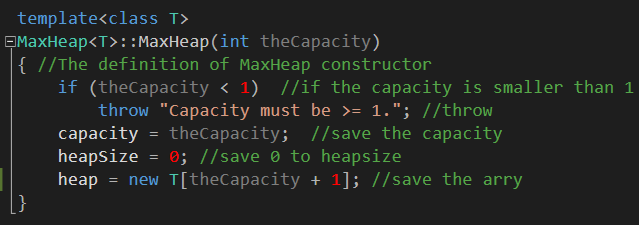
해당 코드는 구현하게 될 함수의 대한 정의입니다. 가상함수로 선언하였으며 MaxPQ를 상속받아 자식 클래스에서 해당 함수에 대한 내용을 자세히 정의합니다.





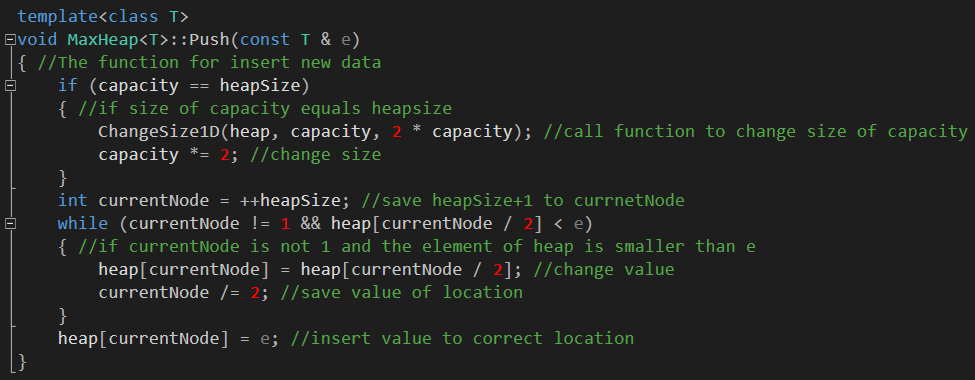
Maxheap의 관련된 함수의 정의입니다. MaxPQ클래스를 상속받았으며 재정의를 위해 함수를 재선언해준 모습이 보입니다. 사용될 변수로는 heap으로 사용될 heap 변수가 있으며 전체 크기를 나타내는 capacity변수와 현재 heap의 크기를 반환 받기 위한 heap Size 변수가 있습니다. 특히 프로그램을 사용하기 위해 사용된 함수로는 Constructor, Push, Pop이 보입니다. 또한 inline으로 선언된 Change Size함수는 heap의 크기보다 데이터가 더 많이 들어왔을 경우 heap의 크기를 현재부도 2배 크게 늘려주는 역할을 하는 함수입니다. IsEmpty 함수는 heap의 크기가 0인지 아닌지를 판단해주는 함수이며 Top함수는 현재 heap의 root의 정보를 반환해서 보여주는 함수입니다. 다음으로는 가장 중요하게 쓰일 Constructor와 Push, Pop함수에 관련된 설명입니다

1. Program 5.15 Constructor



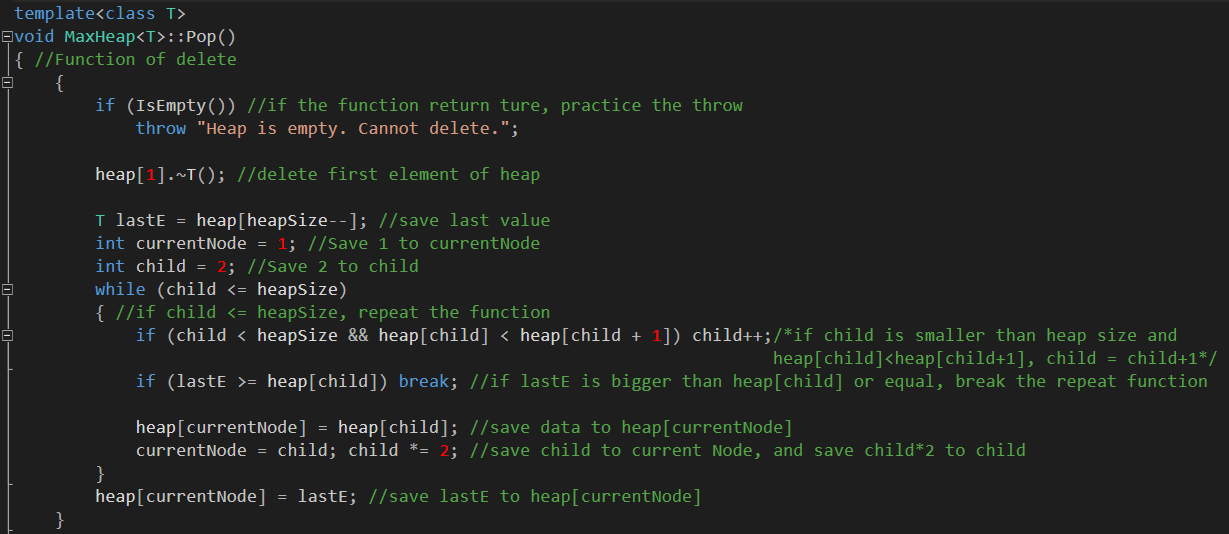
해당 소스코드는 Constructor입니다. 인자로는 integer값을 받으며 만약 1보다 작을 경우에는 throw의 해당 명령어를 실행 후 함수가 끝납니다. 변수 capacity에 인자로 받아온 변수를 저장하며 정보가 없기 때문에 Heap size는 0으로 초기화하며 생성합니다. 또한 배열의 0번째가 아닌 1번째부터 값을 저장하기 때문에 받아온 인자에 1을 더하여 배열을 선언하며 생성자를 종료합니다.

1. Program 5.16 Push



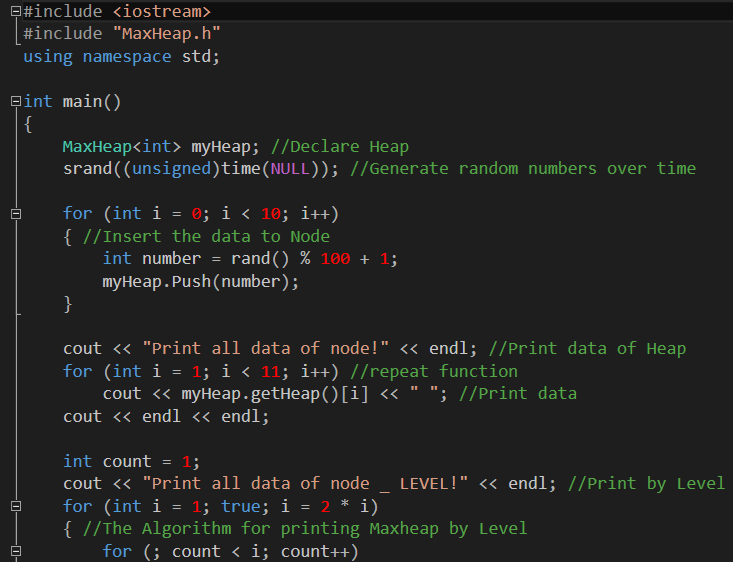
Maxheap의 삽입과 관련된 함수입니다. 만약 heap의 전체 크기와 현재 heap의 크기가 같게 되면 change size함수를 불러와 heap의 사이즈를 늘려준 후 함수를 순서데로 실행하게 되어있습니다. 인자로는 데이터의 값인 e를 받으며 받은 인자의 값이 각각의 부모 노드보다 값이 작아야 하기 때문에 해당 값을 비교하며 배열의 위치를 바꿔주어 받는 currentNode의 값이 되면 e의 값을 저장하며 함수를 끝내는 모습을 볼 수 있습니다.

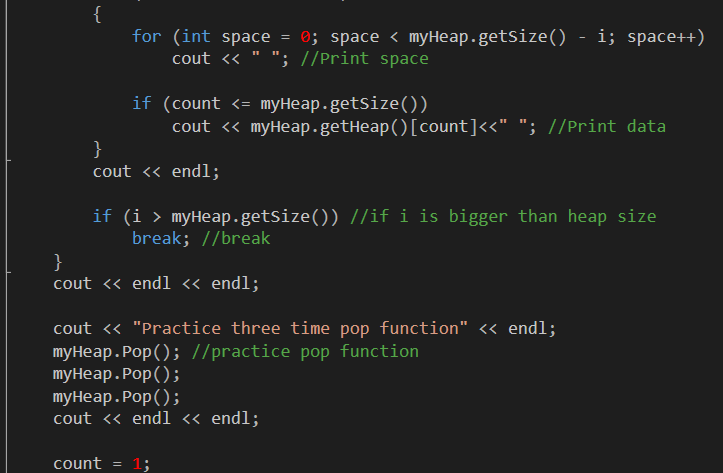
1. Program 5.17 Pop

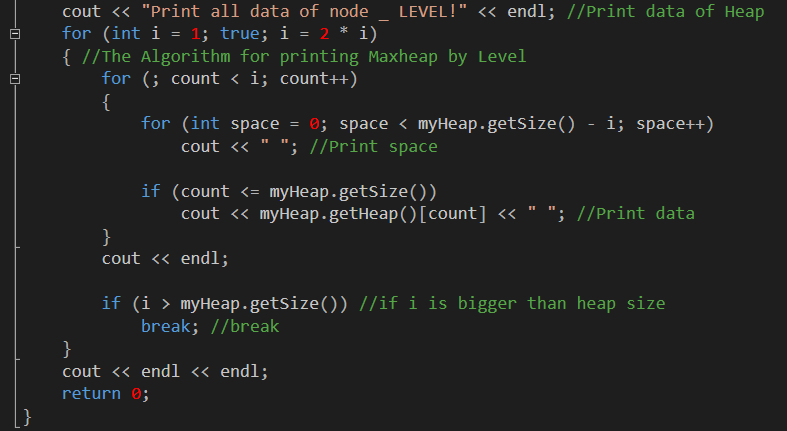


Heap의 data를 지우는 pop명령어입니다. 우선 함수가 실행되기 전에 IsEmpty 함수를 통하여 해당Heap의 크기를 확인하여 size가 0이면 throw를 통하여 알맞은 문구를 반환합니다. Queue의 FIFO방식을 사용하기 때문에 해당 함수가 실행되면 Max heap의 root가 지워지게 되며 다음 root가 될 Node를 반복문을 통하여 찾게 되고 찾은 노드는 해당 Heap의 root에 위치하게 됩니다. 특징으로는 해당 자료구조를 배열로 생성하기 때문에 값의 교환이 이루어지는 것이고 흔히 Node를 사용하여 Node자체를 바꾸는 알고리즘이 아닙니다.

1. Main

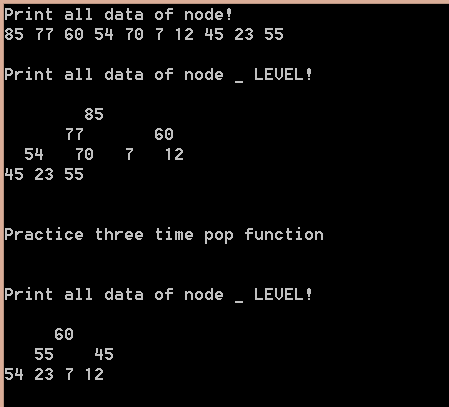






구현한 함수를 가시적으로 나타내기 위해 구현하 Main함수입니다. 최대한 보기 좋기 하기 위하여 for문을 통하여 데이터를 출력하며 space를 같이 넣어주면서 트리의 형태처럼 출력 될 수 있도록 하였습니다.

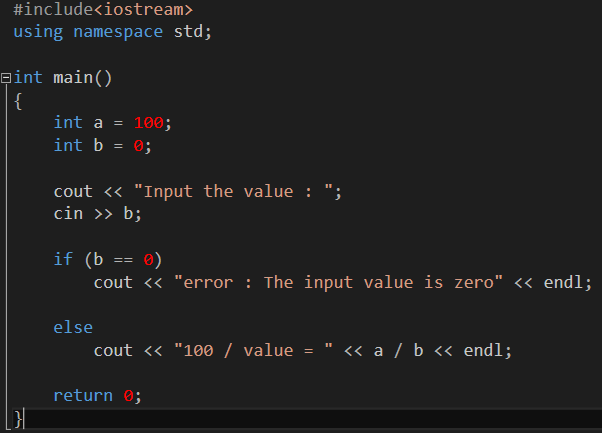
1. 결과 화면



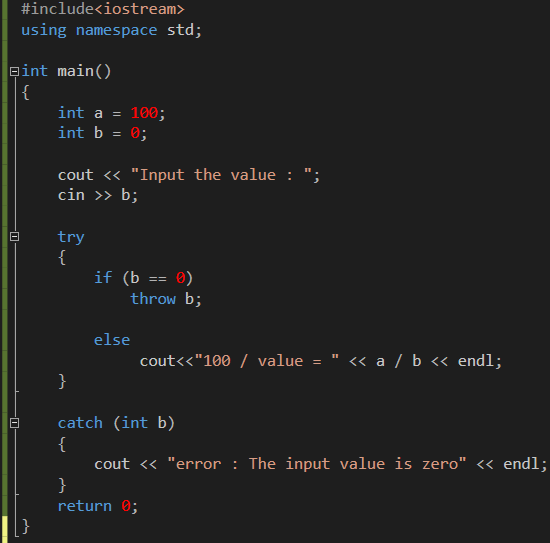
가장 처음 부분은 배열에 저장된 data들을 순차적으로 출력한 결과입니다. 다음은 레벨별로 출력한 것입니다. 트리의 형태로 출력하여 이해하기 쉽습니다. 다음은 Pop함수를 세번 실행한 후 다시 레벨별로 트리를 출력한 결과입니다. 3개의 Node가 지워진 것을 확인 할 수 있습니다.

1. Try, Throw, Catch

프로그래밍을 하면서 알맞은 명령어에 정상적으로 동작하는 것은 가장 기본적인 것입니다. 하지만 정말 좋은 프로그램이라 함은 예외처리를 깔끔하게 해서 구현한 프로그램이 가장 좋은 프로그램이라고 생각됩니다. 이러한 프로그램의 예외처리를 도와주는 명령어가 이번 주제에서 다루게 될 Try, Throw, Catch입니다. 간단한 소스코드의 예로 다음 명령어를 실행하여 보겠습니다.



해당 그림의 소스코드는 평소에 많이 쓰이는 if와 else를 통하여 구현한 프로그램에 대한 예외 처리입니다. 이번에 주제에서 다룬 함수를 통하여 해당 프로그램을 구현하면 아래와 같습니다.



해당 그림은 이번 주제에서 다루는 함수를 사용하여 예외처리를 한 소스 코드입니다. 위와 아래의 소스코드는 같은 동작을 실행하며 같은 소스코드입니다. 프로그램의 작성자가 필수로 사용하는 것이 아닌 편의에 의한 선택적으로 사용할 수 있는 함수입니다. 조금 더 나아가 throw의 특징으로는 던질 수 있는 값의 형식이 int형인지 char형인지에 따라 catch가 잡는 장소가 다르며 해당하는 함수도 다를 수 있습니다. 이에 유의하여 프로그래머는 프로그램을 작성할 때 알맞은 함수를 사용하면 효율적으로 프로그램을 작성할 수 있다고 생각합니다.