

데이터구조설계



학과 : 컴퓨터정보공학부

학번 : 2017202029

이름 : 전 효 희

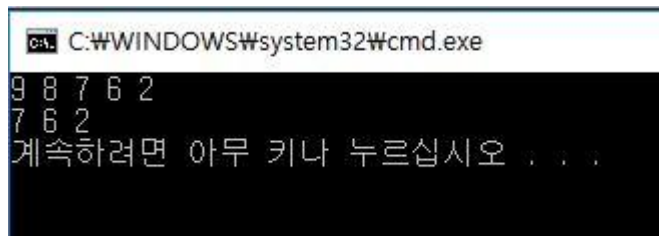
1. Introduction

heap이란 완전 이진 트리로, 부모 노드의 키 값이 자식 노드의 키 값보다 항상 같거나 큰 이진 트리이다.(중복된 값을 허용) heap에는 부모일수록 값이 큰 최대 힙, 부모일수록 값이 작은 최소 heap이 존재한다. heap은 배열에 저장하여 첫번째 인덱스를 제거하고 자식 노드 /2 하여 부모 노드로, 부모 노드 * 2를 하여 자식 노드로 이동하며 데이터를 처리할 수 있게 된다. 이번 과제에서는 이러한 heap 중 최대 heap을 제작하고 그 결과와 데이터 처리를 확인하였다.

프로그램을 제작함에 있어서 매우 많은 오류 및 예외가 발생하게 되고, 이를 일일이 if 문으로 해결해주면 매우 복잡하고 가독성이 떨어지는 코드를 제작하게 된다. 이를 위해 c++에서 제공하는 throw-catch 구문을 사용하면 더 알기 쉽고 안전한 예외처리를 할 수 있게 된다.

2. Result screen

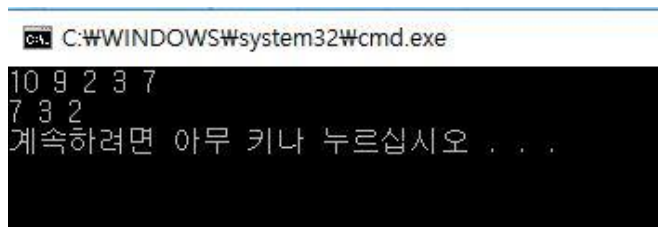
1) 9, 8, 7, 6, 2 순으로 insert한 후 두 번의 pop을 진행



```
C:\WINDOWS\system32\cmd.exe
9 8 7 6 2
7 6 2
계속하려면 아무 키나 누르십시오 . . .
```

- Array의 첫 요소부터 출력한 결과 부모(n/2)가 자식(n)번째 요소보다 항상 큰 값인 것을 볼 수 있다.
- pop하면 root만 pop하여 사라지는 것을 볼 수 있다.

2) 9, 3, 2, 10, 7 순으로 insert한 후, 두 번의 pop 진행



```
C:\WINDOWS\system32\cmd.exe
10 9 2 3 7
7 3 2
계속하려면 아무 키나 누르십시오 . . .
```

- Array의 첫 요소부터 출력한 결과 부모(n/2)가 자식(n)번째 요소보다 항상 큰 값인 것을 볼 수 있다.
- 두 번의 pop을 진행하면 가장 큰 값인 10, 9가 지워지고 이전에 가장 끝에 출력된 7이 가장 먼저 출력되는 것을 볼 수 있다.

3. Code 설명

```

1  #pragma once
2  #pragma warning(disable:4996)
3  #include <iostream>
4  #include <algorithm>
5  using namespace std;
6
7  template <class T>
8  class MaxHeap
9  {
10 private:
11     T* heap; //원소 배열
12     int heapSize; //히프에 있는 원소 수
13     int capacity; //배열 히프의 크기
14
15 public:
16     MaxHeap(int theCapacity = 10);
17     ~MaxHeap() { };
18
19     bool IsEmpty();
20     void ChangeSize1D(T& a, const int oldSize, const int newSize);
21     void Push(const T& e); //최대 히프에 삽입
22     void Pop(); //최대 히프에서의 삭제
23     void Show();
24
25     T* get_heap() { return heap; }
26 };
27

```

- 코드는 max heap을 구현하는 max heap 클래스이다.

```

28 template <class T>
29 inline MaxHeap<T>::MaxHeap(int theCapacity = 10) //constructor
30 {
31     if (theCapacity < 1) throw "Capacity must be >= 1.";
32     capacity = theCapacity;
33     heapSize = 0;
34     heap = new T[capacity + 1]; //heap[0]은 사용되지 않음
35 }
36
37 template <class T>
38 inline bool MaxHeap<T>::IsEmpty()
39 {
40     if (heapSize == 0)
41         return true;
42     else
43         return false;
44 }
45
46 /*Change Size 1D function that using in Push function*/
47 template <class T>
48 inline void MaxHeap<T>::ChangeSize1D(T& a, const int oldSize, const int newSize)
49 {
50     if (newSize < 0) throw "New length must be >= 0";
51
52     T* temp = new T[newSize];
53     int number = min(oldSize, newSize); //oldSize와 newSize 중 작은 것
54     copy(a, a + number, temp);
55     delete[] a; //deallocate old memory
56     a = temp;
57 }
58

```

- Maxheap의 생성자

: 만약 입력된 capacity가 1보다 작으면 예외로 전달하고, 아닐 경우 capacity + 1에 해당하는 메모리를 할당한다.

- IsEmpty: heap안에 내용이 없는지를 확인한다.

- ChangeSize1D: old size와 new size를 전달받아 newsize를 새 메모리 크기로 할당하고, 두 size중 더 작은 값을 더한 만큼을 기존 배열에 더해 temp에 저장한후 기존 배열을 삭제한다. 후에 바꾼 temp를 기존 배열로 교체한다.

```
59  /*Push function*/
60  template <class T>
61  inline void MaxHeap<T>::Push(const T& e)
62  {
63      if (heapSize == capacity) {
64          ChangeSize1D(heap, capacity, 2 * capacity);
65          capacity *= 2; //change capacity into double capacity
66      }
67
68      int currentNode = ++heapSize;
69      while (currentNode != 1 && heap[currentNode / 2] < e)
70      {
71          heap[currentNode] = heap[currentNode / 2]; //move parent down
72          currentNode /= 2; //move to parent
73      }
74      heap[currentNode] = e;
75  }
76
```

- Push

: 만약 heap가 capacity만큼 가득 찬 경우 그 사이즈를 2배로 바꿔 insert될 자리를 만들어 준다. 이후 밑부터 부모 노드와 비교 해가면서 만약 부모가 더 작은 값이면 위치를 교환해가면서 한 레벨 씩 위로 올라간다. 부모 노드 보다 작은 값이 되는 자리에 새 노드를 insert한다.

```

77  /*Pop function*/
78  template <class T>
79  inline void MaxHeap<T>::Pop()
80  {
81      if (IsEmpty()) throw "Heap is empty. Cannot delete.";
82      heap[1].~T(); //최대 원소 삭제
83
84      T lastE = heap[heapSize--];
85
86      int currentNode = 1; //root
87      int child = 2; //currentNode의 자식
88      while (child <= heapSize)
89      {
90          //child를 currentNode의 큰 자식으로 결정
91          if (child < heapSize && heap[child] < heap[child + 1]) child++;
92
93          if (lastE >= heap[child]) break; //we can put lastE in currentNode
94
95          //we cannot put lastE in currentNode
96          heap[currentNode] = heap[child]; //move child up
97          currentNode = child;
98          child *= 2; //move down a level
99      }
100     heap[currentNode] = lastE;
101 }
102
103
104 template <class T>
105 inline void MaxHeap<T>::Show()
106 {
107     for (int i = 1; i <= heapSize; i++)
108     {
109         cout << heap[i] << " ";
110     }
111     cout << endl;
112 }

```

- POP

: 루트를 삭제한 후, 그 자리에 힙의 가장 마지막 노드(last E)를 가져온다. 이 값을 자식 노드와 비교하여 자식 노드가 크면 자리를 교환한다. Last E가 자식 노드보다 큰 값을 가지면 반복을 멈춘다.

- Show

: heap 배열에 저장된 첫 요소(내용이 들어있는)부터 끝 요소까지 출력한다.

4. 예외처리(try, throw, catch)

1) 예외처리

: 예외는 프로그램이 실행 중, 예상치 못한 오류가 발생하여 실행중인 프로그램이 중지하는 것이다. 이런 상황을 개별로 처리하도록 프로그래밍 해주는 것을 예외 처리라고 한다.

2) try, throw, catch

① try, throw, catch

try: 예외가 발생할 수 있는 실행할 코드 블록

throw: try에서 발생한 예외에 대한 오류 정보를 전달하여 catch로 전달

catch: 발생한 예외에 대하여 어떻게 처리할 지 담은 코드블록(try 아래에 1개 이상 존재해야 한다.)

② 예외처리 과정

- 프로그램이 try문을 만나고 try 내 코드를 실행한다.
- 예외가 발생하면, try에서 가장 가까운 catch의 조건부터 탐색한다. 해당 예외에 해당하는 catch를 찾지 못하면 바로 다음 바깥쪽의 try문 다음에 있는 catch절을 차례로 검사한다.
- 적절한 catch를 찾게 되면, throw문에서 예외객체를 매개변수로 catch에 전달한다.
- Catch가 예외처리에 해당하는 코드를 실행한다.
- 예외처리가 끝나면 마지막 catch까지 뛰어넘어 다음 코드로 이동한다.

③ 예외처리 예시

```
#include <iostream>
using namespace std;
int main(void) {
    int a, b;

    try {
        cin >> a;
        cin >> b;

        if (b == 0){
            throw b;
        }

        cout << "a : " << a << " , b : " << b << endl;
        cout << "a/b : " << a / b << endl;

    }
    catch (int expn) {
        cout << "[error]" << endl;
        cout << "The denominator can not be zero." << expn << endl;
    }

    cout << "Program End" << endl;
    return 0;
}
```

```
5 0
[error]
The denominator can not be zero.0
Program End
계속하려면 아무 키나 누르십시오 . . .
```

- 프로그램이 사용자에게 값을 입력 받기 시작하면서 try를 시작한다.
- 두 번째 입력 값이 0이 되면 throw가 발생한다.
- 발생한 throw가 try문 바로 아래 catch로 이동한다.
- throw에서 준 인자 b를 받아 catch문을 실행한다.