Heaps

Max Heap

```
#include<iostream>
using namespace std;
template<class DT>
class MaxHeap
{
public:
     //part1: constructor initializes array of size maxsize
     MaxHeap(int maxsize)
     {
               lastindex=1;
               arr=new DT [maxsize];
               for(int i=0; i<maxsize; i++)
                         arr[i]=NULL;
     }

     //part2: Inserts data into its appropriate position
     //within the Heap
     bool insert(const DT data)
     {
               DT temp=NULL;
               int parent=0;
               int child=lastindex;
               arr[lastindex]=data;
               lastindex++;
               while(child!=1)
               {
                         parent=child/2;
                         if(arr[parent]<arr[child])
                         {
                                   temp=arr[child];
                                   arr[child]=arr[parent];
                                   arr[parent]=temp;
                                   child=parent;
                         }
                         else
                         {
                                   return true;
                         }
               }
               return false;
     }

     //part3: removes the element present in the the root
     //of the Heap and readjusts it to form MaxHeap again


     //part4: prints all the data present in the Heap
     //use the appropriate traversal
     void printContents()
     {
               if(lastindex==1)
                         cout<<"Max Heap is Empty";
               else
                         for(int i=1; i<lastindex; i++)
                         {
                                   cout<<arr[i]<<"\t" ;
                         }
               cout<<endl;
     }
```

```
60.
61.      //part5: destructor, deletes the MaxHeap
62.      ~MaxHeap()
63.      {
64.               delete [] arr;
65.      }
66.
67.
68. private:
69.      DT *arr;
70.      int lastindex;
71. };
72.
73. void main()
74. {
75.      MaxHeap<int> *mxHeap; //creating an object of maxheap
76.      mxHeap=new MaxHeap<int>(40);
77.
78.      //insert following data in the MaxHeap
79.      mxHeap->insert(12);
80.      mxHeap ->insert(43);
81.      mxHeap ->insert(9);
82.      mxHeap ->insert(2);
83.      mxHeap ->insert(14);
84.      mxHeap ->insert(16);
85.      mxHeap ->insert(13);
86.
87.      mxHeap->printContents();
88.
89.      //Carry out 2 deletions from the MaxHeap
90.      int output=0;
91.      output=mxHeap->Delete();
92.
93.      cout<<"Output of first deletion is  "<<output<<endl;
94.      mxHeap->printContents();
95.
96.      output=mxHeap->Delete();
97.
98.      cout<<"Output of second deletion is  "<<output<<endl;
99.      mxHeap->printContents();
100.
101.     output=mxHeap->Delete();
102.
103.     cout<<"Output of third deletion is  "<<output<<endl;
104.     mxHeap->printContents();
105.
106.     output=mxHeap->Delete();
107.
108.     cout<<"Output of fourth deletion is  "<<output<<endl;
109.     mxHeap->printContents();
110.
111.     output=mxHeap->Delete();
112.
113.     cout<<"Output of fifth deletion is  "<<output<<endl;
114.     mxHeap->printContents();
115.
116.
117.     output=mxHeap->Delete();
118.
119.     cout<<"Output of sixth deletion is  "<<output<<endl;
120.     mxHeap->printContents();
121.
122.
123.     output=mxHeap->Delete();
124.
```

```
125.      cout<<"Output of second deletion is  "<<output<<endl;
126.      mxHeap->printContents();
127.
128.
129.      system("pause");
130. }
131.
```

Min Heap

```
 1. #include<iostream>
 2. using namespace std;
 3. template<class DT>
 4. class MinHeap
 5. {
 6. public:
 7.      //part1: constructor initializes array of size maxsize
 8.      MinHeap(int maxsize)
 9.      {
10.                lastindex=1;
11.                arr=new DT [maxsize];
12.                for(int i=0; i<maxsize; i++)
13.                          arr[i]=NULL;
14.      }
15.
16.      //part2: Inserts data into its appropriate position
17.      //within the Heap
18.      bool insert(const DT data)
19.      {
20.                DT temp=NULL;
21.                int parent=0;
22.                int child=lastindex;
23.                arr[lastindex]=data;
24.                lastindex++;
25.                while(child!=1)
26.                {
27.                          parent=child/2;
28.                          if(arr[parent]<arr[child])
29.                          {
30.                                    temp=arr[child];
31.                                    arr[child]=arr[parent];
32.                                    arr[parent]=temp;
33.                                    child=parent;
34.                          }
35.                          else
36.                          {
37.                                    return true;
38.                          }
39.                }
40.                return false;
41.      }
```

```
42.
43.     //part3: removes the element present in the the root
44.     //of the Heap and readjusts it to form MinHeap again
45.     DT Delete()
46.     {
47.               int parent=1,leftchild=0,rightchild=0;
48.               DT del=arr[parent],temp=NULL;
49.               arr[parent]=arr[lastindex-1];
50.               arr[lastindex--]=NULL;
51.               do
52.               {
53.                         leftchild=2*parent;
54.                         rightchild=2*parent+1;
55.                         if(arr[leftchild]!=NULL && arr[rightchild]!=NULL)
56.                         {
57.                                   if(arr[leftchild]>arr[rightchild])
58.                                   {
59.                                             temp=arr[leftchild];
60.                                             arr[leftchild]=arr[parent];
61.                                             arr[parent]=temp;
62.                                             parent=leftchild;
63.                                   }
64.                                   else
65.                                   {
66.                                             temp=arr[rightchild];
67.                                             arr[rightchild]=arr[parent];
68.                                             arr[parent]=temp;
69.                                             parent=rightchild;
70.                                   }
71.                         }
72.                         else if(arr[leftchild]!=NULL)
73.                         {
74.                                   temp=arr[leftchild];
75.                                   arr[leftchild]=arr[parent];
76.                                   arr[parent]=temp;
77.                                   parent=leftchild;
78.                         }
79.                         else if(arr[rightchild]!=NULL)
80.                         {
81.                                   temp=arr[rightchild];
82.                                   arr[rightchild]=arr[parent];
83.                                   arr[parent]=temp;
84.                                   parent=rightchild;
85.                         }
86.               }while(arr[parent]<arr[leftchild] || arr[parent]<arr[rightchild]);
87.               return del;
88.     }
89.
90.     //part4: prints all the data present in the Heap
91.     //use the appropriate traversal
92.     void printContents()
93.     {
94.               if(lastindex==1)
95.                         cout<<"Min Heap is Empty";
96.               else
97.                         for(int i=1; i<lastindex; i++)
98.                         {
99.                                   cout<<arr[i]<<"\t" ;
100.                        }
101.              cout<<endl;
102.    }
103.
104.    //part5: destructor, deletes the MinHeap
105.    ~MinHeap()
106.    {
```

```
107.                delete [] arr;
108.      }
109.
110.
111. private:
112.      DT *arr;
113.      int lastindex;
114. };
115.
116. void main()
117. {
118.      MinHeap<int> *minheap; //creating an object of MinHeap
119.      minheap=new MinHeap<int>(40);
120.
121.      //insert following data in the MinHeap
122.      minheap->insert(12);
123.      minheap ->insert(43);
124.      minheap ->insert(9);
125.      minheap ->insert(2);
126.      minheap ->insert(14);
127.      minheap ->insert(16);
128.      minheap ->insert(13);
129.
130.      minheap->printContents();
131.
132.      //Carry out 2 deletions from the MinHeap
133.      int output=0;
134.      output=minheap->Delete();
135.
136.      cout<<"Output of first deletion is  "<<output<<endl;
137.      minheap->printContents();
138.
139.      output=minheap->Delete();
140.
141.      cout<<"Output of second deletion is  "<<output<<endl;
142.      minheap->printContents();
143.
144.      output=minheap->Delete();
145.
146.      cout<<"Output of third deletion is  "<<output<<endl;
147.      minheap->printContents();
148.
149.      output=minheap->Delete();
150.
151.      cout<<"Output of fourth deletion is  "<<output<<endl;
152.      minheap->printContents();
153.
154.      output=minheap->Delete();
155.
156.      cout<<"Output of fifth deletion is  "<<output<<endl;
157.      minheap->printContents();
158.
159.
160.      output=minheap->Delete();
161.
162.      cout<<"Output of sixth deletion is  "<<output<<endl;
163.      minheap->printContents();
164.
165.
166.      output=minheap->Delete();
167.
168.      cout<<"Output of second deletion is  "<<output<<endl;
169.      minheap->printContents();
170.
171.
```

```
172.      system("pause");
173. }
174.
```

MAX Heap Header

```
 1. #ifndef MAXHEAP_H
 2. #define MAXHEAP_H
 3. using namespace std;
 4. template<class DT>
 5. class MaxHeap
 6. {
 7. public:
 8.       //part1: constructor initializes array of size maxsize
 9.       MaxHeap(int maxsize);
10.
11.       //part2: Inserts data into its appropriate position
12.       //within the Heap
13.       bool insert(const DT data);
14.
15.       //part3: removes the element present in the the root
16.       //of the Heap and readjusts it to form MaxHeap again
17.       DT Delete();
18.
19.       //part4: prints all the data present in the Heap
20.       //use the appropriate traversal
21.       void printContents();
22.
23.       //part5: destructor, deletes the MaxHeap
24.       ~MaxHeap();
25.
26. private:
27.       DT *arr;
28.       int size;
29. };
30. #endif
31.
```

MIN Heap Header

```
 1. #ifndef MINHEAP_H
 2. #define MINHEAP_H
 3. using namespace std;
 4. template<class DT>
 5. class MinHeap
 6. {
 7. public:
 8.       //part1: constructor initializes array of size maxsize
 9.       MinHeap(int maxsize);
10.
11.       //part2: Inserts data into its appropriate position
12.       //within the Heap
13.       bool insert(const DT data);
14.
15.       //part3: removes the element present in the the root
16.       //of the Heap and readjusts it to form MaxHeap again
```

```
17.       DT Delete();
18.
19.       //part4: prints all the data present in the Heap
20.       //use the appropriate traversal
21.       void printContents();
22.
23.       //part5: destructor, deletes the MaxHeap
24.       ~MinHeap();
25.
26. private:
27.       DT *arr;
28.       int size;
29. };
30. #endif
31.
```

Max Heap CPP

```
 1. #include <iostream>
 2. #include "MaxHeap.h"
 3. using namespace std;
 4.
 5. template<class DT>
 6. MaxHeap<DT>::MaxHeap(int maxsize)
 7. {
 8.       size = maxsize;
 9.       arr = new DT[maxsize];
10.       for (int i = 0; i < maxsize; i++)
11.       {
12.                 arr[i] = 0;
13.       }
14. }
15.
16. template<class DT>
17. bool MaxHeap<DT>::insert(const DT data)
18. {
19.       int count = 0;
20.       for (int i = 0; i < size; i++)
21.       {
22.                 if (arr[i] != 0)
23.                 {
24.                           count++;
25.                 }
26.       }
27.
28.       if (count > 39)
29.       {
30.                 return false;
31.       }
32.
33.       int index = count;
34.       if (data > arr[(index - 1) / 2])
35.       {
36.                 while (data > arr[(index - 1) / 2])
37.                 {
38.                           arr[index] = arr[(index - 1) / 2];
39.                           arr[(index - 1) / 2] = data;
40.                           index = (index - 1) / 2;
41.                 }
42.       }
```

```
43.      else
44.      {
45.               arr[index] = data;
46.      }
47.      return true;
48. }
49.
50. template<class DT>
51. void MaxHeap<DT>::printContents()
52. {
53.      int count = 0;
54.      for (int i = 0; i < size; i++)
55.      {
56.               if (arr[i] != 0)
57.               {
58.                        count++;
59.               }
60.      }
61.      for (int i = 0; i < count; i++)
62.      {
63.               cout << arr[i] << endl;
64.      }
65. }
66.
67. template<class DT>
68. DT MaxHeap<DT>::Delete()
69. {
70.      int last = 0;
71.      int temp;
72.      int root = arr[0];
73.
74.      for (int i = 0; i < size; i++)
75.      {
76.               if (arr[i] != NULL)
77.               {
78.                        last++;
79.               }
80.      }
81.
82.      arr[0] = arr[last - 1];
83.      arr[last - 1] = NULL;
84.
85.      for (int i = 0; i < size; i++)
86.      {
87.               if (arr[i] != NULL)
88.               {
89.                        int l = 2 * i + 1;
90.                        int r = 2 * i + 2;
91.
92.                        if (arr[l] != NULL)
93.                        {
94.                                 if (arr[i] < arr[l])
95.                                 {
96.                                          temp = arr[l];
97.                                          arr[l] = arr[i];
98.                                          arr[i] = temp;
99.                                 }
100.                       }
101.                       if (arr[r] != NULL)
102.                       {
103.                                if (arr[i] < arr[r])
104.                                {
105.                                         temp = arr[r];
106.                                         arr[r] = arr[i];
107.                                         arr[i] = temp;
```

```
                                        }
                                }
                        }
        }

        return root;
}

template<class DT>
MaxHeap<DT>::~MaxHeap()
{
        delete arr;
}

```

Min Heap CPP

```
#include <iostream>
#include "MinHeap.h"
using namespace std;

template<class DT>
MinHeap<DT>::MinHeap(int maxsize)
{
        size = maxsize;
        arr = new DT[maxsize];
        for (int i = 0; i < maxsize; i++)
        {
                arr[i] = 0;
        }
}

template<class DT>
bool MinHeap<DT>::insert(const DT data)
{
        int count = 0;
        for (int i = 0; i < size; i++)
        {
                if (arr[i] != 0)
                {
                        count++;
                }
        }
        if(count > 39)
        {
                return false;
        }
        if (data < arr[(count - 1) / 2])
        {
                while (data < arr[(count - 1) / 2])
                {
                        arr[count] = arr[(count - 1) / 2];
                        arr[(count - 1) / 2] = data;
                        count = (count - 1) / 2;
                }
        }
        else
        {
                arr[count] = data;
        }

```

```
44.      return true;
45. }
46.
47. template<class DT>
48. void MinHeap<DT>::printContents()
49. {
50.      int count = 0;
51.      for (int i = 0; i < size; i++)
52.      {
53.               if (arr[i] != NULL)
54.               {
55.                        count++;
56.               }
57.      }
58.      for (int i = 0; i < count; i++)
59.      {
60.               cout << arr[i] << endl;
61.      }
62. }
63.
64. template<class DT>
65. DT MinHeap<DT>::Delete()
66. {
67.      int last = 0;
68.      int temp;
69.      int root = arr[0];
70.
71.      for (int i = 0; i < size; i++)
72.      {
73.               if (arr[i] != NULL)
74.               {
75.                        last++;
76.               }
77.      }
78.
79.      arr[0] = arr[last - 1];
80.      arr[last - 1] = NULL;
81.
82.      for (int i = 0; i < size; i++)
83.      {
84.               if (arr[i] != NULL)
85.               {
86.                        int l = 2 * i + 1;
87.                        int r = 2 * i + 2;
88.
89.                        if (arr[l] != NULL)
90.                        {
91.                                 if (arr[i] > arr[l])
92.                                 {
93.                                          temp = arr[l];
94.                                          arr[l] = arr[i];
95.                                          arr[i] = temp;
96.                                 }
97.                        }
98.                        if (arr[r] != NULL)
99.                        {
100.                                if (arr[i] > arr[r])
101.                                {
102.                                         temp = arr[r];
103.                                         arr[r] = arr[i];
104.                                         arr[i] = temp;
105.                                }
106.                       }
107.              }
108.     }
```

```
109.
110.      return root;
111. }
112.
113. template<class DT>
114. MinHeap<DT>::~MinHeap()
115. {
116.      delete arr;
117. }
118.
```

Main

```
#include "MaxHeap.h"
 #include "MaxHeap.cpp"
 using namespace std;
 int main()
 {
          MaxHeap<int> *mxHeap; //creating an object of maxheap
          mxHeap = new MaxHeap<int>(40);

          //insert following data in the MaxHeap
          mxHeap->insert(12);
          mxHeap->insert(43);
          mxHeap->insert(9);
          mxHeap->insert(2);
          mxHeap->insert(14);
          mxHeap->insert(16);
          mxHeap->insert(13);
          mxHeap->insert(12);

          mxHeap->printContents();

          //Carry out 2 deletions from the MaxHeap
          int output;
          output = mxHeap->Delete();

          cout << "Output of first deletion is  " << output << endl;
          mxHeap->printContents();

          output = mxHeap->Delete();

          cout << "Output of second deletion is  " << output << endl;
          mxHeap->printContents();

          system("pause");
          return 0;
 }

 //#include<iostream>
 //#include "MinHeap.h"
 //#include"MinHeap.cpp"
 //using namespace std;
 //int main()
 //{
 //       MinHeap<int> *mnHeap; //creating an object of MinHeap
 //       mnHeap = new MinHeap<int>(40);
 //
 //       //insert following data in the MinHeap
 //       mnHeap->insert(12);
 //       mnHeap->insert(43);
```

```
//       mnHeap->insert(9);
//       mnHeap->insert(2);
//       mnHeap->insert(14);
//       mnHeap->insert(16);
//       mnHeap->insert(13);
//       mnHeap->insert(12);
//
//       mnHeap->printContents();
//
//       //Carry out 2 deletions from the MinHeap
//       int output;
//       output = mnHeap->Delete();
//
//       cout << "Output of first deletion is  " << output << endl;
//       mnHeap->printContents();
//
//       output = mnHeap->Delete();
//
//       cout << "Output of second deletion is  " << output << endl;
//       mnHeap->printContents();
//
//       system("pause");
//       return 0;
//}
```