# 顺序存储二叉树、堆实验

## 1 小根堆 minHeap 的实现

新建 minHeap.h 文件，输入如下代码：

```
#ifndef MINHEAP_H
#define MINHEAP_H

#include<vector>
#include<iostream>
using namespace std;
template<class T>
class minHeap{
private:
    vector<T> m_vec;
    void percolate_down(int h);   //向下调整堆结构
    void percolate_up();          //向上调整堆结构
public:
    minHeap(){};
    minHeap(vector<T> &vt);
    T top(){return m_vec[0];}     //返回堆顶元素
    void insert(T item);          //插入元素
    void delete_min();            //删除堆顶元素
    unsigned long size(){return m_vec.size();}
};

#endif
```

编辑 main 函数：

```
#include <iostream>
#include "minHeap.h"
using namespace std;
int main() {
    minHeap<int> a;
    return 0;
}
```

编译运行。
🥳 如果一切顺利，继续下面的操作。

# 插入元素

在 minHeap.h 文件中添加以下代码：

```
template<class T>
void minHeap<T>::percolate_up() {
    int c = m_vec.size()-1;
    int p = (c-1)/2;
    T tmp = m_vec[c];
    while(c>0){
        if(m_vec[p] <= tmp){
            break;
        }else{
            m_vec[c] = m_vec[p];
            c=p;
            p=(c-1)/2;
        }
    }
    m_vec[c] = tmp;
}

template<class T>
void minHeap<T>::insert(T item) {
    m_vec.push_back(item);
    percolate_up();
}
```

修改 main函数：

```
int main() {
    minHeap<int> a;
    a.insert(5);
    cout<<a.top()<<endl;
    a.insert(10);
    cout<<a.top()<<endl;
    a.insert(1);
    cout<<a.top()<<endl;
    return 0;
}
```

编译运行，应有如下输出：

```
5
5
1
```

🥳 如果一切顺利，继续下面的操作。

# 删除堆顶元素

在 minHeap.h 文件中添加以下代码：

```
template<class T>
void minHeap<T>::percolate_down(int h){
    int p = h;
    int c = 2*p +1;
    T tmp = m_vec[p];
    while(c <m_vec.size()){
        if(c+1<m_vec.size() && m_vec[c+1] < m_vec[c]){
            ++c;
        }
        if(tmp <= m_vec[c]){
            break;
        }else{
            m_vec[p] = m_vec[c];
            p = c;
            c = 2*p+1;
        }
    }
    m_vec[p] = tmp;
}

template<class T>
void minHeap<T>::delete_min(){
    if(m_vec.empty()){
        cout<<"堆为空"<<endl;
        exit(1);
    }
    m_vec[0] = m_vec.back();
    m_vec.pop_back();
    percolate_down(0);
}
```

修改 main函数：

```cpp
int main() {
    minHeap<int> a;
    a.insert(5);
    cout<<a.top()<<endl;
    a.insert(10);
    cout<<a.top()<<endl;
    a.insert(1);
    cout<<a.top()<<endl;

    a.delete_min();
    cout<<a.top()<<endl;
    a.delete_min();
    cout<<a.top()<<endl;
    return 0;
}
```

编译运行，应有如下输出：

```
5
5
1
5
10
```

🥳 如果一切顺利，继续下面的操作。

## 从数组建堆

在 minHeap.h 文件中添加以下代码：

```cpp
template<class T>
minHeap<T>::minHeap(vector<T> &vt){
    for (int i=0; i<vt.size(); ++i){
        m_vec.push_back(vt[i]);
    }

    for(int i= m_vec.size()/2 - 1; i>=0; --i){
        percolate_down(i);
    }
}
```

更改 main 函数：

```cpp
int main() {
    vector<int> v = {5,2,6,7,1,10};
    minHeap<int> a(v);

    a.delete_min();
    cout<<a.top()<<endl;
    a.delete_min();
    cout<<a.top()<<endl;
    a.delete_min();
    cout<<a.top()<<endl;
    a.delete_min();
    cout<<a.top()<<endl;
    return 0;
}
```

🥳 如果以上操作都一切顺利，你已掌握带小根堆的基本实现方法。

# 2 小根堆挑战

根据堆的性质，实现堆排序函数。
并对 heap_exp.txt 文件中的数据进行排序。

heap_exp.txt 文件的第一行为元素个数，之后的每一行都为一个浮点数，共20000个元素。
heap_exp.txt 的前几行如下：

```
200000
4842.9024
2890.1323
8359.5612
3827.4987
3488.7261
6242.1600
5350.4973
……
```

同时实现冒泡排序算法，对比冒泡排序算法与堆排序算法在这个数据集上的效率差异。

可以使用 clock 函数对操作进行计时：

```cpp
#include <iostream>
#include <ctime>  //使用clock函数需要此头文件
using namespace std;
int main() {
    auto t = clock();

    ...

    t = clock() - t;
    cout<<(float)t/CLOCKS_PER_SEC;
    return 0;
}
```

以上代码将打印第5-9行之间操作的耗时。