表格 1 源代码及运行结果

|  |
| --- |
| SeqTree.h  #pragma once  const int max\_length = 100; // 最大长度  const int ignore\_num = 0x7FFFFFFF; // 忽略数据  // 顺序  class SeqTree {  public:  SeqTree(); // 构造函数  ~SeqTree(); // 析构函数  void Insert(int index, int data); // 插入结点  int Desert(int index); // 删除结点  int ParentsNum(); // 双亲结点个数  int LeavesNum(); // 叶结点个数  private:  int datas[max\_length]; // 数据  }; |
| SeqTree.cpp  #include "SeqTree.h"  #include <cmath>  #include <iostream>  using namespace std;  // 构造函数  SeqTree::SeqTree() {  // 将数据默认为忽略值，表示此位置没有数据  for (int i = 0; i < max\_length; i++) {  datas[i] = ignore\_num;  }  }  SeqTree::~SeqTree() {  }  // 双亲结点个数  int SeqTree::ParentsNum() {  int cnt = 0; // 计数器  // 最后一层不用查，一定不是双亲结点  int i = 0;  for (; i < pow(2, floor(log(max\_length) / log(2))) - 1; i++) {  if (this->datas[i] == ignore\_num) continue; // 如果是空的就跳过  if (this->datas[i \* 2 + 1] != ignore\_num || this->datas[i \* 2 + 2] != ignore\_num) cnt++; // 如果有孩子就是双亲结点  }  return cnt;  }  // 叶结点个数  int SeqTree::LeavesNum() {  int cnt = 0; // 计数器  // 最后一层不用查，一定是叶子结点  int i = 0;  for (; i < pow(2, floor(log(max\_length) / log(2))) - 1; i++) {  if (this->datas[i] == ignore\_num) continue; // 如果是空的就跳过  if (this->datas[i \* 2 + 1] == ignore\_num && this->datas[i \* 2 + 2] == ignore\_num) cnt++; // 如果没有孩子就是叶结点  }  // 遍历最后一层，如果有数据就是叶结点  for (; i < max\_length; i++) {  if (this->datas[i] != ignore\_num) cnt++;  }  return cnt;  }  // 插入结点  void SeqTree::Insert(int index, int data) {  this->datas[index] = data;  }  // 删除结点  int SeqTree::Desert(int index) {  int result = this->datas[index];  this->datas[index] = ignore\_num;  return result;  } |
| Main.cpp  #include "SeqTree.h"  #include <iostream>  #include <string>  using namespace std;  int main() {  SeqTree tree;    int cnt = 0;  string user\_in;  cout << "#表示此位置为空，$结束输入" << endl;  while (1) {  cout << "请输入第" << cnt << "个结点的数据：";  cin >> user\_in;  // 判断输入  if (!user\_in.compare("$")) {  cout << "----------------结束输入-------------------" << endl;  cout << "双亲结点个数为：" << tree.ParentsNum() << "个" << endl;  cout << "叶子结点个数为：" << tree.LeavesNum() << "个" << endl;  break;  }  else if (!user\_in.compare("#")) {  // 什么都不做  }  else {  tree.Insert(cnt, stoi(user\_in));  }    cnt++; // 计数器自加  }  return 0;  } |
| 案例1 圈出来的代表存在的结点，没圈表示是空结点，为了方便存的数据就是结点标号      案例2，这是完全二叉树，没画图    案例3圈出来的代表存在的结点，没圈表示是空结点，为了方便存的数据就是结点标号 |