项目说明文档

算法设计与分析

——assignment1

作 者 姓 名: 沈星宇

学　　号: 1951576

指 导 教 师: 罗烨

学院、 专业: 软件学院 软件工程

同济大学

Tongji University

一、 设计

（1） 数据结构设计

```
class Node {
private:
    int _num;
    Node* _left;
    Node* _right;
public:
    Node(int num,Node* left,Node* right):_num(num),_left(left),_right(right){}
    int getValue() { ... }
    void setValue(int num) { ... }
    Node* getLeftChild() { ... }
    void setLeftChild(Node* left) { ... }
    Node* getRightChild() { ... }
    void setRightChild(Node* right) { ... }
};
```

```
class Number {
private:
    int n;
    int k;
    Node* root;
    long result;
    int resultNum;
public:
    Number(int intN, int intK) { ... }
    // ...
    bool rangeCheck(int num) { ... }
    // ...
    int changeCharToInt(char c) { ... }
    // ...
    void creatTree(Node* p,int depth) { ... }
    // ...
    void travelsal(Node* p,int depth,std::ofstream& out) { ... }
    // ...
    void travelsal(Node* p) { ... }
    void reset(Node* p) { ... }
    bool run() { ... }
};
```

本题的解题思路是将数字的每一位作为二叉树的结点存储。

因为第 x 位(1 < x <= n)是根据第 x-1 位加减 k 生成的，因此只有两种可能，使用树结构来存储最为合适。

本题用 ofstream 类实现运行程序时能够自动在同一目录下将结果存入 result.txt 文件当中。

（2） 程序流程设计

首位从 1-9 进行遍历，作为二叉树的根节点。第 x 位(2 < x <= n)根据第 x-1 位加减 k 所得，合法的数值(介于 1-9 之间的正整数)挂入二叉树中，其中较小的节点挂在左

子树，较大的节点挂在右子树。而二叉树的深度就是给定的 n。

之后用深度优先搜索遍历二叉树，采用中序遍历，就能将输出按照从小到大的顺序给出。

二、 实现

1、Number::createTree(Node* p, int depth)的实现

```
void creatTree(Node* p,int depth) {
    if (depth >= n || p == nullptr) {
        return;
    }else{
        Node* left = (Node*)malloc(sizeof(Node));
        Node* right = (Node*)malloc(sizeof(Node));
        left->setValue(p->getValue() - k);
        right->setValue(p->getValue() + k);
        left->setLeftChild(nullptr);
        left->setRightChild(nullptr);
        right->setLeftChild(nullptr);
        right->setRightChild(nullptr);
        //-----------------------------------
        if (rangeCheck(left->getValue())) {
            p->setLeftChild(left);
            creatTree(left, depth + 1);
        }else{
            free(left);
            left = nullptr;
        }
        //-----------------------------------
        if (rangeCheck(right->getValue())) {
            p->setRightChild(right);
            creatTree(right, depth + 1);
        }else{
            free(right);
            right = nullptr;
        }
    }
    return;
```

为当前节点的子女分配内存，子女节点的数值合法时将会被保留，并进入更深一层的递归构造；而数值非法时就会被释放。

2、Number::travelsal(Node* p, int depth, std::ofstream& out)

```
void travelsal(Node* p,int depth,std::ofstream& out) {
    if (p == nullptr) {
        return;
    }else{
        result *= 10;
        result += p->getValue();
        if (depth == n) {
            resultNum++;
            if (resultNum != 1) {
                std::cout << ',';
                out << ',';
            }
            std::cout << result;
            out << result;
            return;
        }
        Node* pLeft = p->getLeftChild();
        Node* pRight = p->getRightChild();
        if (pLeft != nullptr) {
            travelsal(pLeft, depth + 1, out);
            result /= 10;
        }
        if (pRight != nullptr) {
            travelsal(pRight, depth + 1, out);
            result /= 10;
        }
        return;
    }
}
```

用深度优先搜索遍历，result 用于存储当前数字的序列用于输出。

3、Number::run()的主要实现

```
if (n == 1) {
    std::cout << "[]" << std::endl;
    out << "[]" << std::endl;
}
else {
    std::cout << '[';
    out << '[';
    for (int rootNum = 1; rootNum <= 9; rootNum++) {
        root = (Node*)malloc(sizeof(Node));
        root->setLeftChild(nullptr);
        root->setRightChild(nullptr);
        root->setValue(rootNum);
        /*cause the number cannot start with zero
        * 因为数字不能以0开头 */
        if (k == 0) {
            /*if (k == 0) then it would not be a binary tree
            *如果 (k == 0) 则就不会是一棵二叉树，可以直接输出*/
            for (int i = 0; i < n; i++) {
                std::cout << rootNum;
                out << rootNum;
            }
            if (rootNum != 9) {
                std::cout << ',';
                out << ',';
            }
        }
        else {
            creatTree(root , 1);
            travelsal(root, 1, out);
        }
        reset(root);
    }
    std::cout << ']' << std::endl;
    out << ']' << std::endl;
}
```

对本题给出的边界条件进行了判断，生成根节点 root，之后生成二叉树，遍历二叉树，输出结果，释放内存。

三、 测试

包含了作业要求当中给出的 Function Test、Boundary Test、Performance Test 共六个样例的测试截图。