- **代码实践**

Leetcode

617. 合并二叉树

```
class Solution {
public:
    TreeNode* mergeTrees(TreeNode* t1, TreeNode* t2) {
        if(!t1) return t2;
        if(!t2) return t1;
        t1->val+=t2->val;
        t1->left=mergeTrees(t1->left,t2->left);
        t1->right=mergeTrees(t1->right,t2->right);
        return t1;
    }
};
```

28. 实现 strStr()

```
class Solution {
public:
    int strStr(string haystack, string needle) {
        if(needle.size()==0) return 0;
        int next[needle.size()];
        next[0]=-1;
        int l=needle.size()-1,i=-1,j=0;
        while(j<l)
        {
            if(i==-1||needle[i]==needle[j])
            {
                i++;
                j++;
                next[j]=i;
            }
            else
                i=next[i];
        }
        i=0;j=0;
        int l1=haystack.size();
        int l2=needle.size();
        while(i<l1&&j<l2)
        {
            if(j==-1||haystack[i]==needle[j])
            {
                i++;
                j++;
            }
```

```
            else
                j=next[j];
        }
        if(j==needle.size()) return i-j;
        return -1;
    }
};
```

- **计算机基础知识整理**

PCB：为了便于系统描述和管理进程，在操作系统的核心为每个进程专门定义了一个数据结构，**进程控制块 PCB**。PCB 是进程的唯一标志。

程序顺序执行的特征：

（1）顺序性：每一操作必须在下一操作开始之前结束

（2）封闭性：程序运行时独占全机资源，资源的状态（除初始状态外）只有本程序才能改变，程序一旦执行，其结果不受外界影响

（3）可再现性：程序执行环境和初始条件相同，重复执行时，结果相同

程序并发执行的特征：

（1）间断性：程序并发运行时，共享系统资源，为完成同一任务相互合作，会形成相互制约关系，导致并发程序具有“执行-暂停-执行”这种间断性的活动规律

（2）失去封闭性：程序并发执行时，资源状态由多个程序改变，某程序执行时，会受到其他程序影响，失去封闭性

（3）不可再现性：失去封闭性，导致失去可再现性

- **开源软件特训营总结**

通过 Git 将 9 月 22 日作业文件添加到了 GitHub 仓库中，熟悉了相关 git 操作。

之前通过 Git 上传文件时，总会发生错误，如下

```
zy@LAPTOP-6KBDVIUD MINGW32 ~/Desktop/OSproject-miniprogram2020 (master)
$ git push origin master
To https://github.com/zyy1225/OSproject-miniprogram2020
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/zyy1225/OSproject-miniprogram2020'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

之前解决问题的方法都是通过重新 clone 文件，这次查了一下，是因为本地仓库的文件和远程仓库不一样，所以要先用命令 git pull -f origin master 将远程仓库的文件拉到本地，再次 push，成功。

```
zy@LAPTOP-6KBDVIUD MINGW32 ~/Desktop/OSproject-miniprogram2020 (master)
$ git push origin master
Enumerating objects: 17, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 72.16 KiB | 18.04 MiB/s, done.
Total 9 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 4 local objects.
To https://github.com/zyy1225/OSproject-miniprogram2020
   f17b2cc..ecfb683  master -> master
```