**《数据结构与算法》实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称** | KMP算法的实现和应用 | | | | |
| **姓名** | 叶鹏 | **学号** | 20020007095 | **日期** | 2022/4/15 |
| **实验内容** | 1. 调通圆周率生成算法，生成尽可能长的圆周率序列。 2. 利用自己实现的KMP算法比对圆周率序列中是否存在某个特殊序列并给出位置序号，比如本课程ID，02003048；自己的生日，YYYYMMDD。 | | | | |
| **实验目的** | 生成函数生成圆周率，并在此基础上实现KMP算法查找自己生日 | | | | |
| **实验步骤** | 1. 圆周率的生成   在数学上圆周率的生成有很多种算法，如  *快速收敛级数:*    *蒙特卡洛方法:*    *贝利-波尔温-普劳夫公式:*    诸如此类，在普通计算机编程中很少用到很高精度的PI，不过还是有办法计算其近似值   * 使用acos()函数   double pi = 2\*acos(0.0);   * 使用asin()函数   double pi = 2\*asin(1.0);   * 使用内建常量**M\_PI（）**(c++20) * 其他数学公式 | | | | |
| **实验步骤** | 用来计算Pi的公式很多，但是基本上都是计算近似值，有精度限制，提高精度需要提高数据量，对于本实验要求在确定数值中寻找字符情况不是很适合，因此我们可以使用别人已经造好的轮子，用先辈们已经计算出的数据来完成我们的实验。   1. 准备Pi数值文件     picture 十亿位的pi数值文件  考虑到可能存在数据溢出的问题(KMP算法的cpp实现中数组长度类型为int)，我们可以先使用Python作为工具提前检测一下所会用到的数据规模    picture Python实现KMP  可以看到我们的课程号(02003048)出现在Pi中第**51124221**处位置，因此确定不会溢出，在C++中使用数据规模为1亿的Pi值即可    picture 在C++中寻找到相同结果   1. 寻找其他可能的数值位置  * 8位生日ID（20011012）      * 国庆日（1001）      * 莫名其妙的数值（114514）      * 长者生日（19260817）      1. 实验中用到的KMP算法源代码  * C++      * Python | | | | |
| **实验总结** | 本次实验非常有趣，在理解KMP算法的基础上完成了一个有趣的小程序，体会到了字符串搜素算法的实际用处，加深了对算法的理解与印象，并通过实际调用感受到其妙处。 | | | | |