《数据结构》实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 马昆 | 班级 | 软件工程专升本1班 | 学号 | 2206831544 |
| 实验名称 | **实验四 字符串和数组** | | | | |
| 实验时间 | 2023年4月15日 | | 成绩 | |  |
| 1. **实验目的：**   1.掌握串的顺序存储结构；  2.验证顺序串及其基本操作的实现；  3.掌握串的操作特点,掌握数组的寻址方式。  **二、实验工具：**  软件平台：Windows7或以上版本，Visual Studio 2019  **三、实验原理：**  1、算法描述：  （1）求字符串长度函数：函数接收一个常量字符指针，并返回字符串的长度。先定义一个计数器变量，然后定义一个while循环，循环条件是当前的常量字符指针所指向的字符不是空字符，指针判断时使用后自加让常量字符指针指向下一个字符，循环体内计数器自增一。最后在循环体外返回计数器变量的值。  （2）字符串拼接函数：函数接收两个常量字符指针，用于接收两个字符串，并返回一个常量字符指针，指向拼接后的字符串地址。先分别获取两个字符串的长度，并根据计算出的长度使用malloc函数开辟出新字符串所需的空间。使用两个for循环分别将第一个字符串与第二个字符串中的字符插入到新开辟的空间中。最后在新开辟空间的末尾加上字符串结束符，然后返回新开辟空间的指针。  （3）比较大小函数：函数接收两个常量字符指针，用于接收两个字符串，返回int类型，表示两个字符串的比较结果，小于0表示左边小于右边，大于0表示左边大于右边，等于0表示两字符串相等。分别获取两个字符串的长度，然后进行比较，如果长度不一样就直接返回第一个字符串长度减第二个字符串长度的差值。如果长度一样则使用一个while循环，如果两个字符串的字符指针所指向的字符相等就一直往下指，直到到达字符串的末尾或者是碰到了不一样的字符，最后返回最后那次比较的字符的ASCII码差值。  （4）奇数阶幻方函数：参数接收一个阶数，一个列数为29的二维数组，无返回值。判断阶数不是1-30的奇数就抛出异常。如果结束满足条件，则先定义两个整型，分别表示接下来循环时插入元素的当前行、当前列，其中当前行应该从0开始，当前列应该定义成方阵的最中间。定义一个for循环，从1开始，到阶数的平方结束，循环控制变量就能表示每次插入的元素的值，循环体内首先将插入元素的值赋值给二维数组对应的当前元素，然后计算下一次插入元素的行和列，其中行向上移动一行、列向右移动一列，并存放在两个临时变量中，如果预测的下一个元素已经被填充那么就只更新当前行向下移动一行，如果预测的下一个元素没被填充，那么就将计算出的下一个位置的行和列的临时变量的值赋值给当前行和列的变量。  **四、实验步骤和内容：**  1.定义任意两个字符串，完成以下操作：  （1）编写求字符串长度函数，显示该字符串长度；  （2）编写字符串拼接函数，将该字符串与键盘输入的另一字符串拼接，并显示拼接结果；  （3）编写比较大小函数，比较两字符串大小，并显示比较结果。  2.实现奇数阶幻方。在一个由若干个排列整齐的数组成的正方形中，图中任意一横行、一纵行的几个数之和都相等，具有这种性质的图表，称为“幻方”。按目前填写幻方的方法，是把幻方分成了三类，即奇数阶幻方、双偶阶幻方、单偶阶幻方。奇数阶幻方的方法比较简单，这里只考查奇数阶幻方的程序。输入30以内的任一奇数n，自动输出n\*n阶幻方。如：  输入示例：  3 5 7  输出示例：  6 1 8 15 8 1 24 17 28 19 10 1 48 39 30  7 5 3 16 14 7 5 23 29 27 18 9 7 47 38  2 9 4 22 20 13 6 4 37 35 26 17 8 6 46  3 21 19 12 10 45 36 34 25 16 14 5  9 2 25 18 11 4 44 42 33 24 15 13  12 3 43 41 32 23 21  20 11 2 49 40 31 22  表格 1 源代码及运行结果   |  | | --- | | **MyString.h**  #pragma once  /\*\*  \* 求字符串长度函数  \* param str 需要计算的字符串  \* return 字符串的长度  \*/  int GetLen(const char\* str);  /\*\*  \* 字符串拼接函数  \* param legt 需要拼接的前半部分字符串  \* param right 需要拼接的后半部分字符串  \* return 返回拼接好的字符串  \*/  const char\* Splice(const char\* left, const char\* right);  /\*\*  \* 比较大小函数  \* param left 需要比较的第一个字符串  \* param right 需要比较的第二个字符串  \* return 小于0表示左边小于右边，大于0表示左边大于右边，等于0表示两字符串相等  \*/  int Compare(const char\* left, const char\* right); | | **MyString.cpp**  #include <iostream>  #include <stdlib.h>  using namespace std;  /\*\*  \* 求字符串长度函数  \* param str 需要计算的字符串  \* return 字符串的长度  \*/  int GetLen(const char\* str) {  int cnt = 0;    while (\*(str++) != '\0') {  cnt++;  }  return cnt;  }  /\*\*  \* 字符串拼接函数  \* param legt 需要拼接的前半部分字符串  \* param right 需要拼接的后半部分字符串  \* return 返回拼接好的字符串  \*/  const char\* Splice(const char\* left, const char\* right) {  // 分别获取两个字符串的长度  int left\_size = GetLen(left), right\_size = GetLen(right);  int length = left\_size + right\_size; // 计算拼接后字符串的长度  char\* str = (char\*)malloc((length + 1) \* sizeof(char)); // 定义一串连续的空间存放拼接后的数组，并且计算最后的空字符位    int i = 0;  // 将第一个字符串加入到新开辟的空间中  for (; i < left\_size; i++) {  str[i] = left[i];  }  // 将第二个字符串加入到新开辟的空间中  for (; i < length; i++) {  str[i] = right[i - left\_size];  }  // 设置字符串结束符  str[i] = '\0';  return str;  }  /\*\*  \* 比较大小函数  \* param left 需要比较的第一个字符串  \* param right 需要比较的第二个字符串  \* return 小于0表示左边小于右边，大于0表示左边大于右边，等于0表示两字符串相等  \*/  int Compare(const char\* left, const char\* right) {  // 分别获取两个字符串的长度  int left\_len = GetLen(left), right\_len = GetLen(right);    // 长度不一样就直接返回相差的长度  if (left\_len != right\_len) return left\_len - right\_len;  // 长度不一样就返回ASCII码的差值  // 一直循环到两个字符串的某个字符不相等或者循环到头  while (\*(left++) == \*(right++) && \*left != '\0') {}  // 返回ASCII码的差值  return \*(--left) - \*(--right);  } | | **Main.cpp**  #include <iostream>  #include "MyString.h"  using namespace std;  /\*\*  \* 奇数阶幻方函数  \* 构造方法：  \* 1. 将数字 1 放置在第一行的中间位置，即第一行的正中间列。  \* 2. 从数字 2 开始，依次填充到 n\*n。  \* 3. 如果当前位置在第一行，则下一个位置是当前位置的正上方。  \* 4. 如果当前位置在最后一列，则下一个位置是当前位置的正右方。  \* 5. 如果当前位置在第一行且在最后一列，则下一个位置是当前位置的正下方。  \* 6. 如果当前位置已经被填充，则下一个位置是当前位置的正下方。  \* 7. 如果下一个位置超出边界，则将其移到相应的边界上。  \* 8. 重复步骤 3-7，直到填充完所有的数字。  \*/  void MagicSquareOfOddOrder(int n, int magic\_square[][29]) {  // 如果不是1-30的奇数就抛出异常  if (!(n % 2) || (n < 1 && n > 29)) throw "odd number & range from 1 to 30";  int row = 0; // 当前行  int col = n / 2; // 当前列  // 从1到n\*n依次填充幻方矩阵  for (int num = 1; num <= n \* n; num++) {  magic\_square[row][col] = num; // 填充当前位置  // 计算下一个位置的行和列  int next\_row = (row - 1 + n) % n; // 上移一行  int next\_col = (col + 1) % n; // 右移一列  // 如果下一个位置已经被填充，向下移一行  if (magic\_square[next\_row][next\_col] != 0) {  row = (row + 1) % n;  }  else {  row = next\_row;  col = next\_col;  }  }  }  void PrintMagicSquare(int arr[][29], int n) {  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  cout << arr[i][j] << "\t";  }  cout << endl;  }  }  int main() {  char first[100], second[100];  cout << "输入两个字符串：" << endl;  cin >> first >> second;  cout << "拼接后的结果：" << Splice(first, second) << endl;  cout << "比较后的结果：" << Compare(first, second) << endl;  int n = 0;  cout << "输入奇数阶幻方的阶数：" << endl;  cin >> n;  int magic\_square[29][29] = { 0 }; // 定义并初始化  MagicSquareOfOddOrder(n, magic\_square);  cout << "某一" << n << "阶奇数阶幻方如下：" << endl;  PrintMagicSquare(magic\_square, n);  return 0;  } | | **运行结果**  **案例1**    **案例2**    **案例3**    **案例4** |   **五、实验总结：**  本次实验完成并掌握了字符串的三个算法，自己实现起来还是没什么压力，但是在奇数阶幻方卡了很久，因为之前没有了解过这个东西，不清楚它的规律，没法总结出一个算法来，然后就到网上去查了一下它的规律，然后根据这个规律完成了代码，其中还出现了一些小插曲，就是使用malloc函数动态创建二维数组出现了很多问题，最后就采取了参数传数组的方式来解决，但是代码耦合性很强，知道了奇数阶幻方的算法，但是还是不太明白为什么要这样做。  **六、教师评语：** | | | | | |