**河南开封科技传媒学院实验报告**

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

**20 -20 学年第 学期**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名： | 刘东 | | 学号： | 2136101045 | 专业： | 软件工程 | |
| 所在学院： | | 理工学院 | | | 指导老师： | | 袁帅 |
| 实验时间： | | 2021/2/22 | | | 实验地点： | | 6503 |
| 课程名称： | | 数据结构 | | | | | |
| 实验题目： | | 数据结构第一次实验 | | | | | |
| 实验目的：  一：  3，三者时间复杂度除递归都是O(n) 因为像调用函数什么的浪费了时间，指针要开辟空间，递归的是占用的空间大。  4，没有，不断的递归，函数嵌套层数过多最后导致栈内存溢出。  二:  第二种，第一种x计算了n！次，an每次增加一次，第一种都得从新一个个相乘，第二种仅仅乘了n次。  三:  n(n+1)(n+2)/6 n(n+1)(n+3)/6 | | | | | | | |
| 实验内容：   1. 实现一个程序，使得传入一个正整数为N的参数之后，能顺序打印从1到N的全部正整数：   （1）.用三种方法实现上述程序，如循环、递归等；  （2）.实现后分别记录三种方法中N为1千、1万、10万时的用时和出现的结果；  （3）.计算三种方法的时间复杂度，并解释为什么三个程序的用时不同；  （4）.如果你采用了递归算法，当N为10万时，出现你想要的效果了吗？如果没有出现，请尝试解释为什么？   1. （思考题）写程序计算给定多项式在给定点x处的值，请尝试实现并评价以下两种方法，哪种效率更高，为什么？   （1）    （2）    3.（思考题）  （1）求划线部分的执行次数为  for (int i = 1; i <= n; i++) {undefined  for (int j = 1; j <= i; j++) {undefined  for (int k = 1; k <= j; k++) {undefined  x=x+y;  }  }  }  （2）求划线部分的执行次数为  for (int i = 1; i <= n; i++) {undefined  for (int j = 1; j <= i; j++) {undefined  for (int k = 0; k <= j; k++) {undefined  x=x+y;  }  }  } | | | | | | | |
| 实验代码：  第一次实验：  #include<iostream>  #include<stdio.h>  #include <windows.h>  using namespace std;  void test1()  {        }  typedef struct Node{  int data;  struct Node\*next;  } linkList;  void test01(int N) {  cout << "循环遍历" << endl;  for (int i = 1; i <= N; i++) {  cout << i << " ";  }  cout << endl;  }  void test02(int N, int t) {  if (t > N) return;  cout << t << " ";  test02(N, t + 1);  }  linkList\* createlinkList(int n) {  linkList \*head = new Node;  head->data = 0;  head->next = NULL;  linkList \*r = new Node;  r = head;  for (int i = 1; i <= n; i++) {  linkList \*p = new Node;  p->data = i;  r->next = p;  r = p;  }  r->next = NULL;  return head;  }  void test03(int N) {  cout << "链表遍历" << endl;  linkList \*head = createlinkList(N);  linkList \*p = new Node;  p = head->next;  while (p != NULL) {  cout << p->data << " ";  p = p->next;  }  delete p;  delete head;  }  int main() {  DWORD star\_time, end\_time;    int N;  cin >> N;  star\_time = GetTickCount();  //循环  test01(N);  end\_time = GetTickCount();  auto t1 = (end\_time - star\_time);      star\_time = GetTickCount();  cout << "递归遍历" << endl;  //递归  test02(N, 1);  cout << endl;  end\_time = GetTickCount();  auto t2 = (end\_time - star\_time);      star\_time = GetTickCount();  //链表  test03(N);  cout << endl;  end\_time = GetTickCount();  auto t3 = (end\_time - star\_time);  cout << "循环运行时间为：" << t1 << "ms." << endl;  cout << "递归运行时间为：" << t2 << "ms." << endl;  cout << "链表运行时间为：" << t3 << "ms." << endl;  } | | | | | | | |
| 实验结果截图及实验心得：    10000 | | | | | | | |