《程序设计艺术与方法》课程实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 实验一 STL 的熟悉与使用 | | | | | | |
| 姓 名 |  | 系院专业 | 计算机与信息学院 | 班 级 |  | 学 号 |  |
| 实验日期 | 2020.11.01 | | 指导教师 | 徐本柱 | | 成 绩 |  |
| **一、实验目的和要求**  1．掌握C++中STL的容器类的使用;  2．掌握C++中STL的算法类的使用. | | | | | | | |
| **二、实验预习内容**  1．预习ICPC讲义，大致了解STL的相关内容。  2．了解STL中一些类 vector list类的使用方法  3．了解泛型算法的使用 | | | | | | | |
| **三、实验项目摘要**  1．练习vector和list的使用  2．练习泛型算法的使用 | | | | | | | |
| **四、实验结果与分析（源程序及相关说明）**   1. vector：   /\*\*  \* @author: Li Haporan  \* @date: 2020.10.20  \* @details: 实验一STL的熟悉与使用  \*/  #include <iostream>  #include <vector>  #include <list>  #include <algorithm>  #include <ctime>  #include <iomanip>  using namespace std;  #define count 10  //define up and down of sort  bool up(int vector1,int vector2)  {  return vector1 < vector2;  }  bool down(int vector1,int vector2)  {  return vector1 > vector2;  }  //main start  int main(void)  {  //define vector and list  vector<int> vector;  list<int> list;  std::vector<int>::iterator iter;  //make random number  srand(time(NULL));  /\* test 1 \*/  cout << "-----------------------------------------" << endl;  for (size\_t i = 0; i < count; i++)  {  //make 10 rand number of int  vector.push\_back( rand() );  }  //before sort output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;  /\* test 2 \*/  cout << "-----------------------------------------" << endl;  int temp = rand();  cout << "The random number insert into the header is： " << temp << endl;  //insert into header  vector.insert(vector.begin(), temp);  // output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;  /\* test 3 \*/  cout << "-----------------------------------------" << endl;  temp = rand();  cout << "number to find is:" << temp << endl;  iter = find(vector.begin(), vector.end(), temp);  if(temp == \*iter)  {  cout << "number is be find" << endl;  }  else  {  cout << "number is not be find" << endl;  vector.insert(vector.end(), temp);  // output vector again  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;  }  /\* test 4 \*/  cout << "-----------------------------------------" << endl;  //sort  sort(vector.begin(), vector.end(), up);  //after sort output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;  /\* test 5 \*/  cout << "-----------------------------------------" << endl;  // delete the end number  vector.pop\_back();  // output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;  /\* test 6 \*/  cout << "-----------------------------------------" << endl;  vector.clear();  if(vector.empty())  {  cout << "vector is empty" << endl;  }    return 0;  }  //main end  2．List  /\*\*  \* @author: Li Haporan  \* @date: 2020.10.20  \* @details: 实验一STL的熟悉与使用  \*/  #include <iostream>  #include <vector>  #include <list>  #include <algorithm>  #include <ctime>  #include <iomanip>  using namespace std;  #define count 10  //define up and down of sort  bool up(int vector1,int vector2)  {  return vector1 < vector2;  }  bool down(int vector1,int vector2)  {  return vector1 > vector2;  }  int main(void)  {  list<int> list;  vector<int> vector;  std::vector<int>::iterator iter;  //make random number  srand(time(NULL));  for (size\_t i = 0; i < count; i++)  {  //make 10 rand number of int  vector.push\_back( rand() );  }  //before sort output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;  cout << "-----------------------------------------" << endl;  //sort  sort(vector.begin(), vector.end(), up);  //after sort output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;  cout << "-----------------------------------------" << endl;  //sort  sort(vector.begin(), vector.end(), down);  //after sort output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;    cout << "-----------------------------------------" << endl;  int min = vector[0];  for (iter = vector.begin(); iter != vector.end(); iter++)  {  if (\*iter < min)  {  min = \*iter;  }  }  cout << "the min number is:" << min << endl;    int max = vector[0];  for (iter = vector.begin(); iter != vector.end(); iter++)  {  if (\*iter > min)  {  max = \*iter;  }  }  cout << "the max number is:" << max << endl;  return 0;  }  **实验结果：**  **实验1结果：**    实验2结果：    **实验心得：**  **较为熟练的掌握了List和vector的** | | | | | | | |

《程序设计艺术与方法》课程实验报告

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 实验二 搜索算法的实现 | | | | | | | |
| 姓 名 | 李浩然 | 系院专业 | | 计算机与信息学院 | 班 级 | 计科19-3 | 学 号 | 2019218211 |
| 实验日期 | 2020.11.01 | | 指导教师 | | 徐本柱 | | 成 绩 |  |
| 1. **实验目的和要求**   1.掌握宽度优先搜索算法。  2.掌握深度优先搜索算法。 | | | | | | | | |
| **二、实验预习内容**  1．预习ICPC讲义中的搜索的内容  2. 了解什么是深度优先搜索和广度优先搜索。 | | | | | | | | |
| **三、实验项目摘要**  1. 将书上的走迷宫代码上机运行并检验结果，并注意体会搜索的思想。  2.八皇后问题：在一个国际象棋棋盘上放八个皇后，使得任何两个皇后之间不相互攻击，求出所有的布棋方法。上机运行并检验结果。  3. 骑士游历问题：在国际棋盘上使一个骑士遍历所有的格子一遍且仅一遍，对于任意给定的顶点，输出一条符合上述要求的路径。  4.倒水问题：给定2 个没有刻度容器，对于任意给定的容积，求出如何只用两个瓶装出L 升  的水，如果可以，输出步骤，如果不可以，请输出No Solution。 | | | | | | | | |
| **四、实验结果与分析（源程序及相关说明）**   1. 八皇后问题   /\*\*  \* @author: Li Haporan  \* @date: 2020.10.20  \* @details: 实验一STL的熟悉与使用  \*/  #include <iostream>  #include <vector>  #include <list>  #include <algorithm>  #include <ctime>  #include <iomanip>  using namespace std;  #define count 10  //define up and down of sort  bool up(int vector1,int vector2)  {  return vector1 < vector2;  }  bool down(int vector1,int vector2)  {  return vector1 > vector2;  }  int main(void)  {  list<int> list;  vector<int> vector;  std::vector<int>::iterator iter;  //make random number  srand(time(NULL));  for (size\_t i = 0; i < count; i++)  {  //make 10 rand number of int  vector.push\_back( rand() );  }  //before sort output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;  cout << "-----------------------------------------" << endl;  //sort  sort(vector.begin(), vector.end(), up);  //after sort output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;  cout << "-----------------------------------------" << endl;  //sort  sort(vector.begin(), vector.end(), down);  //after sort output vector  cout << "vector: ";  for (iter = vector.begin(); iter != vector.end(); iter++)  {  cout << \*iter << std::setw(8);  }  cout << endl;    cout << "-----------------------------------------" << endl;  int min = vector[0];  for (iter = vector.begin(); iter != vector.end(); iter++)  {  if (\*iter < min)  {  min = \*iter;  }  }  cout << "the min number is:" << min << endl;    int max = vector[0];  for (iter = vector.begin(); iter != vector.end(); iter++)  {  if (\*iter > min)  {  max = \*iter;  }  }  cout << "the max number is:" << max << endl;  return 0;  }  2.  /\*\*  \* @author: Li Haporan  \* @date: 2020.10.20  \* @details: 实验二 搜索算法的实现  \*/  #include<iostream>  #include<iomanip>  using namespace std;  //define count to record step  int count = 0;  //initialize  int chessBoard[8][8] ={0};  //check is knight out  bool isKnightOut(int x, int y)  {  if (x >= 0 && x <= 7 && y >= 0 && y <= 7)  return false;  else  return true;  }  //check is visited  bool isVisited(int x, int y)  {  if (chessBoard[x][y] != 0)  return true;  else  return false;  }  void dfs(int x, int y)  {  // traverse is over  if (count == 64)  return;  //if board is unvisited and knight is not out  else if ( !isVisited(x, y) && !isKnightOut(x, y) )  {  count++;  // record the step  chessBoard[x][y] = count;  dfs(x + 2, y + 1);  dfs(x - 2, y - 1);  dfs(x + 2, y - 1);  dfs(x - 2, y + 1);  dfs(x - 1, y - 2);  dfs(x + 1, y - 2);  dfs(x + 1, y + 2);  dfs(x - 1, y + 2);  return;  }  // if knight out, return  else  return;  }  int main()  {  //define start point  int x, y;  cout << "please input the start x, y:";  //check input right  while (1)  {  cin >> x >> y; //输入坐标  if (x > 7 || x < 0 || y> 7 || y < 0)  cout << "the start point is wrong,please inptu again:" << endl;  else  break;  }  // start travel  dfs(x, y);  //print result  for (int i = 0; i < 8; i++)  {  for (int j = 0; j < 8; j++)  {  cout << setw(2) << chessBoard[i][j] << " ";  }  cout << endl;  }  return 0;  }  **实验结果：**  **实验1结果（八皇后问题）：**    **实验2结果（骑士游历问题）：** | | | | | | | | |

《程序设计艺术与方法》课程实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 实验三 计算几何算法实现 | | | | | | |
| 姓 名 | 李浩然 | 系院专业 | 计算机与信息学院 | 班 级 | 计科19-3 | 学 号 | 2019218211 |
| 实验日期 | 2020.11.01 | | 指导教师 | 徐本柱 | | 成 绩 |  |
| **一、实验目的和要求**   * 1. 理解线段的性质、叉积和有向面积。   2. 掌握寻找凸包的算法。   3. 综合运用计算几何和搜索中的知识求解有关问题。 | | | | | | | |
| **二、实验预习内容**  1．预习ICPC讲义中的搜索的内容  理解线段的性质、叉积和有向面积。  掌握寻找凸包的算法。  综合运用计算几何和搜索中的知识求解有关问题。 | | | | | | | |
| **三、实验项目摘要**   * 1. 将讲义第三章第三节中的凸包代码上机运行并检验结果。   2. 完成讲义第三章的课后习题, 上机运行并检验结果。   3. 思考:   判线段相交时, 如果有个线段的端点在另一条线段上, 注意可能与另一条线段上的端点重合, 思考这样的情况怎么办。   * 1. 房间最短路问题:   给顶一个内含阻碍墙的房间, 求解出一条从起点到终点的最最短路径。房间的边界固定在 x=0,x=10,y=0 和 y=10。起点和重点固定在(0,5)和(10,5)。房间里还有 0 到 18 个墙, 每个墙有两个门。输入给定的墙的个数, 每个墙的 x 位置和两个门的 y 坐标区间, 输出最短路的长度。下图是个例子:   |  | | --- | |  | |  |  | | | | | | | | |
| 四、实验结果与分析（源程序及相关说明）  1.  /\*\*  \* @author: Li Haporan  \* @date: 2020.10.28  \* @details: 实验三 计算几何算法实现  \*/  #include<stdio.h>  #include<string.h>  #include<vector>  #include<iostream>  #include<math.h>  #include<algorithm>  using namespace std;  const int maxn = 1100;  const double esp = 1e-10;  // define struct point (x, y)  struct point {  double x, y;  } myPoint[maxn];  //define myline of two point  struct line {  point head, tail;  }myline[maxn];  double dmin(double a, double b) {  return a < b ? a : b;  }  double dmax(double a, double b) {  return a > b ? a : b;  }  //check is cross  bool cross(const line& line1, const line& line2)  {  // save point  point a = line1.head;  point b = line1.tail;  point c = line2.head;  point d = line2.tail;  if (dmin(a.x, b.x) > dmax(c.x, d.x) ||  dmin(a.y, b.y) > dmax(c.y, d.y) ||  dmin(c.x, d.x) > dmax(a.x, b.x) ||  dmin(c.y, d.y) > dmax(a.y, b.y))  return 0;  double h, i, j, k;  h = (b.x - a.x) \* (c.y - a.y) - (b.y - a.y) \* (c.x - a.x);  i = (b.x - a.x) \* (d.y - a.y) - (b.y - a.y) \* (d.x - a.x);  j = (d.x - c.x) \* (a.y - c.y) - (d.y - c.y) \* (a.x - c.x);  k = (d.x - c.x) \* (b.y - c.y) - (d.y - c.y) \* (b.x - c.x);  return (h \* i <= esp && j \* k <= esp);  }  int main() {  int n = 0;  int cas = 1;  while (scanf("%d", &n) != EOF)  {  if (n == 0) break;  if (n < 3)  {  printf("Figure %d: Impossible\n", cas++);  continue;  }  if (cas != 1)  puts("");  // get input point x, y  for (int i = 0; i < n; i++)  {  cin >> myPoint[i].x >> myPoint[i].y;  }    // get the line of every side two point  for (int i = 0; i < n; i++)  {  point u, v;  if (i == (n - 1))  {  u.x = myPoint[i].x;  u.y = myPoint[i].y;  v.x = myPoint[0].x;  v.y = myPoint[0].y;  }  else  {  u.x = myPoint[i].x;  u.y = myPoint[i].y;  v.x = myPoint[i + 1].x;  v.y = myPoint[i + 1].y;  }  myline[i].head = u;  myline[i].tail = v;  }  bool flag = false;  bool tmp = false;  for (int i = 0; i < n; i++)  {  for (int j = i + 2; j < n; j++) {  if (i == 0 && (j == (n - 1)))  continue;  tmp = cross(myline[i], myline[j]);  if (tmp) {  flag = true;  break;  }  }  if (flag)  break;  }  if (flag)  {  printf("Figure %d: Impossible\n", cas++);  }  // get the area  else  {  double area = 0;  double x1, x2, y1, y2, x0, y0;  x0 = x1 = myPoint[0].x;  y0 = y1 = myPoint[0].y;  for (int i = 1; i <= n; i++)  {  if (i < n)  {  x2 = myPoint[i].x;  y2 = myPoint[i].y;  }  else  {  x2 = x0;  y2 = y0;  }  area += (y1 + y2) \* (x2 - x1) \* 0.5;  x1 = x2;  y1 = y2;  }  if (area < 0)  area \*= -1;  printf("Figure %d: %.2f\n", cas++, area);  }  }  return 0;  }  2.  /\*\*  \* @author: Li Haporan  \* @date: 2020.10.28  \* @details: 实验三 计算几何算法实现  \*/  #include<stdio.h>  #include<string.h>  #include<math.h>  #include<algorithm>  #define eps 10e-8  #define inf 1<<29  using namespace std;  struct point  {  double x, y;  };  int n;  double xx[22], yy[22][5];  point p[90];  double adj[90][90];  int psize;  double ff(double x)  {  return x \* x;  }  double dis(point p1, point p2)  {  return sqrt(ff(p1.x - p2.x) + ff(p1.y - p2.y));  }  double cross(double x1, double y1, double x2, double y2, double x3, double y3)  {  return (x2 - x1) \* (y3 - y1) - (y2 - y1) \* (x3 - x1);  }  bool judge(point p1, point p2)  {  if (p1.x >= p2.x)return 0;  int i = 0;  while (xx[i] <= p1.x && i < n)i++;  while (xx[i] < p2.x && i < n)  {  if (cross(p1.x, p1.y, p2.x, p2.y, xx[i], 0) \* cross(p1.x, p1.y, p2.x, p2.y, xx[i], yy[i][1]) < 0  || cross(p1.x, p1.y, p2.x, p2.y, xx[i], yy[i][2]) \* cross(p1.x, p1.y, p2.x, p2.y, xx[i], yy[i][3]) < 0  || cross(p1.x, p1.y, p2.x, p2.y, xx[i], yy[i][4]) \* cross(p1.x, p1.y, p2.x, p2.y, xx[i], 10) < 0)  return 0;  i++;  }  return 1;  }  void floyd()  {  int i, j, k;  for (k = 0; k < psize; k++)  for (i = 0; i < psize; i++)  for (j = i + 1; j < psize; j++)  if (i != k && j != k)  adj[i][j] = min(adj[i][j], adj[i][k] + adj[k][j]);  printf("最短距离为：%.2f\n", adj[0][psize - 1]);  }  int main()  {  int i, j;  printf("输入墙的个数：\n");  scanf("%d", &n);  printf("分别输入墙的x坐标和门的y坐标\n");  p[0].x = 0; p[0].y = 5; psize = 1;  for (i = 0; i < n; i++)  {  scanf("%lf", &xx[i]);  for (j = 1; j <= 4; j++)  {  p[psize].x = xx[i];  scanf("%lf", &p[psize].y);  yy[i][j] = p[psize].y;  psize++;  }  }  p[psize].x = 10; p[psize].y = 5;  psize++;  for (i = 0; i < psize; i++)  for (j = 0; j < psize; j++)  adj[i][j] = inf;  for (i = 0; i < psize; i++)  for (j = i + 1; j < psize; j++)  if (judge(p[i], p[j]))  adj[i][j] = dis(p[i], p[j]);  floyd();  return 0;  } | | | | | | | |

《程序设计艺术与方法》课程实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 实验四 动态规划算法的实现 | | | | | | |
| 姓 名 | 李浩然 | 系院专业 | 计算机与信息学院 | 班 级 | 计科19-3 | 学 号 | 2019218211 |
| 实验日期 | 2020.11.01 | | 指导教师 | 徐本柱 | | 成 绩 |  |
| **一、实验目的和要求**   * 1. 理解动态规划的基本思想、动态规划算法的基本步骤。   2. 掌握动态规划算法实际步骤 | | | | | | | |
| **二、实验预习内容**  1．预习ICPC讲义中的搜索的内容 | | | | | | | |
| 1. **实验项目摘要** 2. 求两个字符串的最长公共子序列。   X 的一个子序列是相应于 X 下标序列{1, 2, … , m}的一个子序列, 求解两个序列的所有子序列中长度最大的, 例如输入: pear, peach 输出: pea。   1. 给定两个字符串 a 和 b, 现将串 a 通过变换变为串 b, 可用的操作为, 删除串 a 中的一个字符; 在串 a 的某个位置插入一个元素; 将串 a 中的某个字母换为另一个字母。对于任意的串 a 和串 b, 输出最少多少次能够将串变为串 b。   思考: 输出变换的步骤。   1. 输入一个矩阵, 计算所有的子矩阵中和的最大值。例如, 输入   0 -2 -7 0  9 2 -6 2  -4 1 -4 1  - 1 8 0 -2  输出为: 15  思考: 当矩阵很大时, 比如 100\*100 的矩阵, 你的程序还能够很快的得出结果吗, 如果不能, 请思考如何用动态规划的思想解决。 | | | | | | | |
| **四、实验结果与分析（源程序及相关说明）**  1.  /\*\*  \* @author: Li Haporan  \* @date: 2020.10.28  \* @details: 实验四 动态规划算法的实现  \*/  #include <iostream>  #include <string>  #include <algorithm>  using namespace std;  int minDist(string A, string B)  {  int aLen = A.length();  int bLen = B.length();  int D[bLen + 1][aLen + 1];  for (int j = 0; j <= aLen; j++) {  //初始化第一行  D[0][j] = j;  }  for (int i = 0; i <= bLen; i++) {  //初始化第一列  D[i][0] = i;  }  for (int i = 1; i <= bLen; i++) {  for (int j = 1; j <= aLen; j++) {  //初始化表格  D[i][j] = min(min(D[i - 1][j] + 1, D[i][j - 1] + 1), (A[j - 1] == B[i - 1] ? D[i - 1][j - 1] : D[i - 1][j - 1] + 1));  }  }  return D[bLen][aLen];  }  int main()  {  string a, b;  cout << "请输入字符串A :" << endl;  cin >> a;  cout << "\n请输入字字符串B :" << endl;  cin >> b;  cout << "\n最少次数："<<minDist(a, b)<<endl;  return 0;  }  2.  /\*\*  \* @author: Li Haporan  \* @date: 2020.10.28  \* @details: 实验四 动态规划算法的实现  \*/  #include <iostream>  #include <cstring>  using namespace std;  int main()  {  int a, b;  cout << "请输入矩阵的行数和列数:" << endl;  cin >> a >> b;  int d[a][b];  cout << "请输入矩阵:" << endl;  for(int i = 0; i < a; i++){  for(int j = 0; j < b; j++){  cin >> d[i][j];  }  }  int sum, tmp;  int c[b];  sum = -999;  for(int i = 0; i < a; i++){  memset(c, 0 ,sizeof(c));  for(int j = i; j < a; j++){  tmp = 0;  for(int k = 0; k < b; k++){  c[k] += d[j][k];  if(tmp >= 0)  tmp += c[k];  else  tmp = c[k];  if(sum < tmp)  sum = tmp;  }  }  }  cout << "矩阵最大的子矩阵和为:" << sum;  }  **实验结果：**  **实验1结果：**    **实验2结果：** | | | | | | | |