**实验报告**

**学院（系）名称：**计算机与通信工程学院

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **姓名** | | 王帆 | | **学号** | | 20152180 | | | **专业** | | 计算机科学与技术 | |
| **班级** | | 2015级1班 | | **实验项目** | | 实验二：栈与队列应用 | | | | | | |
| **课程名称** | | | | 数据结构与算法 | | | | | **课程代码** | | 0668016 | |
| **实验时间** | | | | 2017年 3月 27日 第3节 | | | | | **实验地点** | | 7-215 | |
| 考核标准 | 实验过程  25分 | | 程序运行  20分 | | 回答问题  15分 | | 实验报告  30分 | 特色  功能  5分 | | 考勤违纪情况  5分 | **成绩** |  |
| 成绩栏 |  | |  | |  | |  |  | |  | 其它批改意见:  教师签字： | |
| 考核内容 | 评价在实验课堂中的表现，包括实验态度、编写程序过程等内容等。 | | □功能完善,  □功能不全  □有小错  □无法运行 | | ○正确  ○基本正确  ○有提示  ○无法回答 | | ○完整  ○较完整  ○一般  ○内容极少  ○无报告 | ○有  ○无 | | ○有  ○无 |
|  | | | | | | | | | | | | |
| **一、实验目的**  1、实验设置基本要求：通过实验掌握栈或队列的基本操作的实现，并能灵活运用栈或队列特性，综合运用程序设计、算法分析等知识解决实际问题。  2、实验设置较高要求：理解组成递归算法的基本条件，理解递归算法与相应的非递归算法的区别，理解栈和队列的应用与作用。  **二、实验题目与要求**  1．十进制数与N进制数据的转换  1）问题描述：将从键盘输入的十进制数转换为N（如二进制、八进制、十六进制）进制数据。  2）要求: 利用顺序栈实现数制转换问题  2．表达式语法判读：  1）问题描述：假设表达式中允许包含3种括号：圆括号、方括号和大括号，编写一个算法判断表达式中的括号是否正确配对。  2）要求：输出信息提示表达式中括号是否匹配，属于哪类不匹配（如左括号多？右括号多）。  3. 算术表达式求值算法  1）问题描述：从键盘输入一个算术表达式并输出它的结果  2）要求：算术表达式可包含加、减、乘、除、十进制整数和小括号，利用栈实现  4. 停车场管理问题  1）问题描述：设有一个可以停放n辆汽车的狭长停车场，它只有一个大门可以供车辆进出。车辆按到达停车场的早晚依次从停车场最里面向大门口处停放(最先到达的第一辆车放在停车场的最里面)。如果停车场已放满n辆车，则后来的车辆只能在停车场大门外的便道上等待，一旦停车场内有车走开，则排在便道上的第一辆车就进入停车场。停车场内如有某辆车要开走，在它之后进入停车场的车都必须先退出停车场为它让路，待其开出停车场后，这些车辆再依原来的次序进场。每辆车在离开停车场时，都应根据它在停车场内停留的时间长短交费。如果停留在便道上的车未进停车场就要离去，允许其离去，不收停车费，并且仍然保持在便道上等待的车辆的次序。编写程序模拟该停车场的管理。  2）要求：要求程序输出每辆车到达后的停车位置（停车场或便道上），以及某辆车离开停车场时应缴纳的费用和他在停车场内停留的时间  3）实现提示：以栈模拟停车场，以队列模拟便道，按照从终端读入的车辆“到达”“离开”信息模拟停车场管理  5、迷宫问题：  1）问题描述：设计算法完成迷宫问题求解，输出迷宫中所有路径的求解。  2）要求：使用递归算法和非递归两种方法实现   1. **实验过程与实验结果**   **实验2-01：十进制数与N进制数据的转换**  **数据结构定义：**  #define StackInitSize 100  typedef struct SeqStack{  int data[StackInitSize];//数据  int top;//栈顶元素  }SeqStack;  typedef SeqStack \*StackLink;//顺序栈指针  **算法设计思路简介：**  本实验需要实现以下操作：   1. 顺序栈的初始化、进出栈以及判空栈等基本操作 2. 进制转换   对于一个十进制数，将其转换为N进制数的过程中，首遵循整数部分除以进制取余数的原则，将原数值作划分。通过Conversion方法迭代地获取余数，并进行入栈操作。对于大于十进制的数制，根据10-A对应法则，可最多转换为36进制。通过Num\_in方法将大于9的数字转换为对应的字符，并输出，从而实现进制转换的功能。  C:\Users\Du\AppData\Local\Microsoft\Windows\INetCacheContent.Word\进制转换.png**算法描述：**  **算法的实现和测试结果（参考OJ）**    **实验2-02：表达式语法判读**  **数据结构定义：**  **算法设计思路简介：**  本实验需要实现以下操作：   1. 构建栈式数据结构 2. 将输入的括号依次入栈 3. 判断是否存在匹配的括号，存在则出栈，否则继续入栈，直至入栈结束，给出判断。   **算法描述：**   1. 构建一个栈式的数据结构，参照课本的案例，可以构建一个链式栈，也可以构建顺序栈，并给定最大栈容量。同时实现栈的Pop/Push/判空等基本方法，备用。 2. 利用getchar()方法，将给定的括号依次入栈，此处存在以下几种逻辑 3. 左右括号匹配成功->出栈 4. 未出现左括号即出现右括号->报错结束，格式化输出 5. 依次入栈直至字符串结束，因为缺少右括号导致栈仍非空->报错结束，并输出缺少的右括号   PS:由于OJ平台上的测试用例并未涉及第三点，在设计的过程中我暂时去掉了第三种逻辑情况，实现了AC。但是这是不完全的，正确的情况是可能存在的。  **算法的实现和测试结果（参考OJ）**    **实验2-03：计算器**  **数据结构定义：**  typedef int StackElemType;  typedef struct node{  int data;  struct node\* next;  }node;  typedef node\* LinkStack;  **算法设计思路简介：**  **1.顺序栈的初始化、进出栈以及判空栈等基本操作**   1. **进制转换**   **对于一个十进制数，将其转换为N进制数的过程中，首遵循整数部分除以进制取余数的原则，将原数值作划分。通过Conversion方法迭代地获取余数，并进行入栈操作。对于大于十进制的数制，根据10-A对应法则，可最多转换为36进制。通过Num\_in方法将大于9的数字转换为对应的字符，并输出，从而实现进制转换的功能。**  **算法描述：可以用自然语言、伪代码或流程图等方式**  **算法的实现和测试结果（参考OJ）**    **实验2-04：停车场管理问题**  **数据结构定义：**  **算法设计思路简介：**  **算法描述：可以用自然语言、伪代码或流程图等方式**  **算法的实现和测试结果（参考OJ）**    **实验2-05：迷宫问题**  **数据结构定义：**  #define MaxSize 25  typedef struct PosType {  int x;  int y;  }PosType;  typedef struct LNode{  struct PosType data;  struct LNode \*next;  }LNode;  **算法设计思路简介：**  **算法描述：可以用自然语言、伪代码或流程图等方式**  **算法的实现和测试结果（参考OJ）**     1. **收获与体会**   通过本次实验，我加深了对栈和队列操作的理解。对于栈和队列基本的出入栈/队操作有了更进一步的掌握。本次实验中最大的难点在于对实际问题与数据结构的相互组合，例如在算术表达式求值实验中，如何将表达式符号和出入栈操作相结合。同样，本次实验充分利用了TJUT OJ平台，便于发现一些较为隐蔽的边缘数据，也有助于修补BUG。总之，在今后的实验中，我将继续通过更多的方式对这些弱点与问题进行强化和修正，在实践中成长。   1. **源代码清单**   **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Copyright 1997-2017 Duke.Wang\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**  **//实验2-01：进制转换**  **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**  #include <iostream>  #include <cstdlib>  #define StackInitSize 100  using namespace std;  **//定义顺序栈**  typedef struct SeqStack{  int data[StackInitSize];  int top;  }SeqStack;  typedef SeqStack \*StackLink;  StackLink InitStack(){  StackLink s;  s=(StackLink)malloc(sizeof(SeqStack));  if(s!=NULL){  s->top=-1;  return s;  }  else{  exit(0);  }  }  int IsEmpty(StackLink s){  return (s->top==-1)?1:0;  }  void Push(StackLink s, int x){  if(s->top==StackInitSize){  exit(0);  }  else{  s->top++;  s->data[s->top]=x;  }  return;  }  int Pop(StackLink s){  int temp;  if(IsEmpty(s)){  exit(0);  }  else{  temp=s->data[s->top];  s->top--;  return temp;  }  }  void num\_in(int x){  x+=55;  cout<<char(x);  }  void conversion\_num(int N, int r){  StackLink s;  int x;  s=InitStack();  while(N){  Push(s,N%r);  N=N/r;  }  while(!IsEmpty(s)){  x=Pop(s);  if(x<=9&&x>=0) cout<<x;  else num\_in(x);  }  cout<<endl;  }  int main(){  int N,r;  cin>>N>>r;  conversion\_num(N,r);  return 0;  }  **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**  **//实验2-03：计算器**  **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**  #include <iostream>  #include <cstdlib>  using namespace std;  **//定义链式栈**  typedef int StackElemType;  typedef struct node{  int data;  struct node\* next;  }node;  typedef node\* LinkStack;  int priority[8][8]={3,3,1,1,1,1,3,3,  3,3,1,1,1,1,3,3,  3,3,3,3,3,1,3,3,  3,3,3,3,3,1,3,3,  3,3,3,3,3,1,3,3,  1,1,1,1,1,1,2,2,  3,3,3,3,3,0,3,3,  1,1,1,1,1,1,0,2};  int IsEmpty(LinkStack S);  **//1.初始化置空栈**  LinkStack InitStack(LinkStack S){  LinkStack top;  top=NULL;  return top;  }  **//2.进栈**  LinkStack Push(LinkStack S, int x){  LinkStack p;  p=(LinkStack)malloc(sizeof(node));  if(p){  p->data=x;  p->next=S;  S=p;  }  else exit(0);  return S;  }  **//3.出栈**  LinkStack Pop(LinkStack S, int \*x){  LinkStack temp;  if(!IsEmpty(S)){  temp=S;  \*x=S->data;  S=S->next;  free(temp);  return S;  }  else return NULL;  }  **//4.读取栈顶元素**  int GetTop(LinkStack S){  if(IsEmpty(S))  return 0;  else return S->data;  }  **//5.判断栈空满**  int IsEmpty(LinkStack S){  return S?0:1;  }  **//6.获取优先级**  int sub(int c){  switch(c){  case '+':  return 0;  case '-':  return 1;  case '\*':  return 2;  case '/':  return 3;  case '%':  return 4;  case '(':  return 5;  case ')':  return 6;  case '\n':  return 7;  }  }  int main(){  int x1,x2,i,j,op,temp;  char ch;  LinkStack OPTR=NULL;  LinkStack OPND=NULL;  OPTR=Push(OPTR,'\n');  ch=getchar();  while(!(ch=='\n'&&GetTop(OPTR)=='\n')){  if(ch=='+'||ch=='-'||ch=='\*'||ch=='/'||ch=='%'||ch=='('||ch==')'||ch=='\n'){  i=sub(GetTop(OPTR));  j=sub(ch);  if(priority[i][j]==3){  OPTR=Pop(OPTR,&op);  OPND=Pop(OPND,&x2);  OPND=Pop(OPND,&x1);  switch(op){  case '+':  OPND=Push(OPND,x1+x2);  break;  case '-':  OPND=Push(OPND,x1-x2);  break;  case '\*':  OPND=Push(OPND,x1\*x2);  break;  case '/':  OPND=Push(OPND,x1/x2);  break;  case '%':  OPND=Push(OPND,x1%x2);  }  continue;  }  if(priority[i][j]==1)  OPTR=Push(OPTR,ch);  if(priority[i][j]==2)  OPTR=Pop(OPTR,&temp);  }  else  OPND=Push(OPND,ch-48);  ch=getchar();  }  cout<<GetTop(OPND);  return 0;  }  **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///实验2-04：停车场管理**  **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**#include <iostream>  #include <cstdlib>  #define MaxSize 100  using namespace std;  typedef struct SNode{  int time[MaxSize];  int length;  int top;  }SNode;  typedef struct QNode{  int time[MaxSize];  int front;  int rear;  }QNode;  typedef SNode \*SeqStack;  typedef QNode \*SeqQueue;  SeqStack \_S;  SeqQueue \_Q;  **//1.初始化顺序栈**  SeqStack Ini\_Stack(int num){  SeqStack S = (SeqStack)malloc(sizeof(SNode));  if(S){  S->length = num;  S->top = -1;  }  else exit(0);  return S;  }  **//2.初始化顺序队列**  SeqQueue Ini\_Queue(){  SeqQueue Q = (SeqQueue)malloc(sizeof(QNode));  if(Q){  Q->front = 0;  Q->rear = -1;  }  else exit(0);  return Q;  }  **//3.判断栈是否空/满**  int Is\_Stack\_Full(SeqStack S){  return S->top>=S->length-1?1:0;  }  int Is\_Stack\_Empty(SeqStack S){  return S->top<=-1?1:0;  }  **//4.判断队列是否为空**  int Is\_Queue\_Empty(SeqQueue Q){  return Q->rear<0?1:0;  }  **//5.判断特定车辆是否在停车场内**  int Is\_Car\_In(SeqStack S,int n){  return S->top>=(S->length-n)?1:0;  }  **//6.车辆操作**  SeqQueue Insert\_Queue(SeqQueue Q,int t){  Q->time[++Q->rear]=t;  return Q;  }  SeqStack Insert\_Stack(SeqStack S,int t){  S->time[++S->top] = t;  return S;  }  SeqStack Insert\_From\_Queue(SeqStack S,SeqQueue Q){  S->time[++S->top] = Q->front;  for(int i=0; i<Q->rear; i++){  Q->time[i] = Q->time[i+1];  }  return S;  }  **//7.停车**  void Pause(SeqStack S,SeqQueue Q,int t){  if(!Is\_Stack\_Full(S))  S = Insert\_Stack(S,t);  else  Q = Insert\_Queue(Q,t);  }  **//8.开车**  SeqQueue Exit\_queue(SeqQueue Q,int n) {  if(!Is\_Queue\_Empty(Q)) {  for(int i=n-1; i<Q->rear; i++){  Q->time[i] = Q->time[i+1];  }  Q->rear--;  cout<<"0"<<endl;  return Q;  }  else return Q;  }  SeqStack Exit\_Stack(SeqStack S,int n,int t) {  int cost;  n = S->length - n;  cost = t - S->time[n];  for(n; n<S->top; n++) {  S->time[n] = S->time[n+1];  }  S->top--;  cout<<cost<<endl;  return S;  }  void Exit(SeqStack S,SeqQueue Q,int n,int t) {  if(Is\_Car\_In(S,n)) {  S = Exit\_Stack(S,n,t);  if(!Is\_Queue\_Empty(Q)) {  S = Insert\_From\_Queue(S,Q);  }  }  else{  n = n - S->length;  Q = Exit\_queue(Q,n);  }  }  int main(){  int n,Q,type,time;  cin>>n>>Q;  \_S = Ini\_Stack(n);  \_Q = Ini\_Queue();  for(int i=0;i<Q;i++){  cin>>type;  switch(type){  case 1:{  cin>>time;  Pause(\_S,\_Q,time);  break;  }  case 2:{  cin>>n>>time;  Exit(\_S,\_Q,n,time);  break;  }  default:break;  }  }  return 0;  }  **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///实验2-05：迷宫问题解决**  **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**#include <iostream>  #include <cstdlib>  using namespace std;  #define MaxSize 25  typedef struct PosType {  int x;  int y;  }PosType;  typedef struct LNode{  struct PosType data;  struct LNode \*next;  }LNode;  typedef LNode \*LinkList;  typedef int MazeType[MaxSize][MaxSize];  PosType start, End;  MazeType Q;  int row, col;  int flag = 0;  LinkList L;  void Init\_List() {  L = (LinkList)malloc(sizeof(LNode));  if(L) L->next = NULL;  else exit(0);  }  void Insert\_List(PosType cur){  LinkList s = (LinkList)malloc(sizeof(LNode));  if(s){  s->data = cur;  s->next = L->next;  L->next = s;  }  else exit(0);  }  void Show\_List(){  LinkList s = L->next;  int t = 1;  while(s){  if(t){  cout<<"("<<s->data.x-1<<","<<s->data.y-1<<")";  t = 0;  }  else cout<<"->("<<s->data.x-1<<","<<s->data.y-1<<")";  s = s->next;  }  }  void Show(){  if(flag){  cout<<"YES"<<endl;  Show\_List();  }  else cout<<"NO";  }  void Try(PosType cur,int curstep){  PosType next;  PosType direc[4] = {{0,1},{1,0},{0,-1},{-1,0}};  for(int i=0; i<4; i++){  next.x = cur.x + direc[i].x;  next.y = cur.y + direc[i].y;  if(Q[next.x][next.y] == 1){  Q[next.x][next.y] = ++curstep;  if(!(next.x==End.x&&next.y==End.y)){  Try(next,curstep);  if(flag){  Insert\_List(cur);  }  }  else{  Insert\_List(next);  Insert\_List(cur);  flag = 1;  }  Q[next.x][next.y] = 1;  curstep--;  }  }  }  int main(){  Init\_List();  int i,j;  cin>>row>>col;  for(i=0;i==0||i==row+1;i+=row) {  for(j=0; j<=col; j++){  Q[i][j] = -1;  }  }  for(j=0;j==!0||j==col+1;j+=col) {  for(i=0; i<=row; i++){  Q[i][j] = -1;  }  }  for(i=1; i<=row; i++){  for(j=1; j<=col; j++){  cin>>Q[i][j];  if(Q[i][j]==0){  start.x = i;  start.y = j;  }  if(Q[i][j]==2){  End.x = i;  End.x = i;  End.x = i;  End.y = j;  Q[i][j] = 1;  }  }  }  Try(start,2);  Show();  return 0;  } | | | | | | | | | | | | |