实验二

|  |  |  |
| --- | --- | --- |
| 姓名： | | 学号： |
| 班级： | | 专业：人工智能 |
| 实验名称： | 栈和队列的基本操作 | |
| 实验目的： | 1、掌握栈和队列的定义；  2、掌握栈的基本操作的基本操作，如建立、查找、插入和删除等以及栈的应用 | |
| 实验要求： | | |
| (1) 程序要添加适当的注释，程序的书写要采用缩进格式。  (2) 程序要具在一定的健壮性，即当输入数据非法时，程序也能适当地做出反应，如插入删除时指定的位置不对等等。  (3) 程序要做到界面友好，在程序运行时用户可以根据相应的提示信息进行操作。  （4）每项任务给出代码及运行结果截图 | | |
| 实验内容： | | |
| 1. 定义一个顺序栈，完成栈的初始化，入栈、出栈、取栈顶元素操作，并在main函数中调用： 2. 输入一个非负十进制整数，打印输出等值的八进制数（用栈实现） 3. 定义一个循环队列，实现循环队列的初始化、入队、出队、取队头元素基本操作。并在main函数中调用 | | |
| 实验代码及运行结果截图 | | |
| 任务1：  #include<iostream>  #include<stdlib.h>  using namespace std;//标准格式化输出  #define MAX\_SIZE 100  #define Status int  #define OVERFLOW -1  #define OK 0  #define ERROR -2  #define EXTENSION\_SIZE 10  typedef int s\_Elemtype;  typedef struct { //定义栈的结构体  s\_Elemtype\* base;//定义基指针  s\_Elemtype\* top; //定义顶指针（注：按照分配顺序，top指针在base指针后一位）  int stacksize;//定义栈的大小  }Sqstack;//重命名为Sqstack  Status Init\_Stack(Sqstack& s) {//初始化栈  s.base = new s\_Elemtype[MAX\_SIZE];//使用base指针和new方法拓展一个MAX\_SIZE\*sizeof(s\_Elemtype)的空间  //s.base = (s\_Elemtype\*)malloc(MAX\_SIZE \* sizeof(s\_Elemtype));是一样的  if (!s.base) exit(OVERFLOW);//初始化失败，返回-1  s.top = s.base;//将base指针的位置赋给top指针  s.stacksize = MAX\_SIZE;//定义stacksize的目的：记录栈的大小  return OK;//返回0  }  Status Stack\_Push(Sqstack& s, s\_Elemtype e) {//栈元素的插入  if (s.top - s.base == s.stacksize) { // 栈已满，进行栈的扩展  s\_Elemtype\* newBase = (s\_Elemtype\*)realloc(s.base, (s.stacksize + EXTENSION\_SIZE) \* sizeof(s\_Elemtype));//使用realloc函数重新分配一个size+1的位置  if (!newBase) {//如果realloc函数分配失败（内存太小或者其他的什么东西）  return ERROR;  }  s.base = newBase;//base指针到新的base域？  s.top = s.base + s.stacksize;//top要再base的基础上加上目前栈的大小  s.stacksize += 1;//此时栈的大小加1（目的是给新元素腾地方）  }  \*s.top++ = e;//将e元素的值赋值给刚刚开辟的空间中  return OK;//返回正确  }  Status Stack\_Pop(Sqstack& s, s\_Elemtype& e) {//栈元素的弹出  if (s.top == s.base) return ERROR;//如果没有元素，返回错误  e = \*--s.top;//将删除的元素赋值给e，因为是引用，原e直接对应赋值  return OK;//返回成功  }  Status Stack\_DetTop(Sqstack& s, s\_Elemtype& e) {//栈顶元素的弹出  if (s.top != s.base) {//如果栈内存在元素才执行此操作  e = \*(s.top - 1);//将栈顶元素的值赋给e  return OK;//返回完成  }  return ERROR;//否则未完成操作  }  int main() {  Sqstack stack1, stack2;//新建两个栈，目的是后面调换的时候用，避免栈1元素的丢失  Init\_Stack(stack1);//初始化栈1  Init\_Stack(stack2);//初始化栈2  cout << "--------------------------------------" << endl;  cout << "This is the homework from Zhouxing in AI2302" << endl;  cout << "-------------------------" << endl;  cout << "Insert the number of elements you want to push into the stack:" << endl;  int num1; cin >> num1;//定义num1，然后插入num1  cout << "-------------------------" << endl;  cout << "Insert the elements you want to push into the stack:" << endl;  for (int i = 1; i <= num1; i++) {//插入num1个元素  int ele1; cin >> ele1;  Stack\_Push(stack1, ele1);  }  cout << "-------------------------" << endl;  cout << "The elements of the stack are as follows: " << endl;//输出栈1的所有元素（此时会发现都是倒着输出的）  int cnt = 0;  s\_Elemtype topElement;//删除栈顶元素  while (stack1.top != stack1.base) {//将栈1的所有元素输出，同时压入栈2  Stack\_Pop(stack1, topElement);  Stack\_Push(stack2, topElement);  cnt++;  cout << topElement << " ";//输出栈1的元素  }  for (int j = 0; j < cnt; j++) {//再将刚刚栈2压入的元素导入回栈1，此时栈1相当于没有变化  Stack\_Pop(stack2, topElement);  Stack\_Push(stack1, topElement);  }  cout << "" << endl;  cout << "-------------------------" << endl;  cout << "The top element of the stack is: "<<endl;//输出栈顶元素  if (Stack\_DetTop(stack1, topElement) == OK) {//看返回值如果是OK对应的值就输出栈顶元素  cout << topElement << endl;  }  else {//否则栈就是空的  cout << "Stack is empty!" << endl;  }  cout << "-------------------------" << endl;  return 0;//结束程序  } | | |
| 实验2  #include<iostream>  #include<string>  #include<cstdlib> // For malloc and realloc  #include<cstdio>  using namespace std;  #define OK 1  #define ERROR -1  #define INIT\_SIZE 100  #define EXTENSION\_SIZE 10  typedef int Status;  typedef int s\_Elemtype;  typedef struct {  s\_Elemtype\* base; // 栈的基址  s\_Elemtype\* top; // 栈顶指针  int stacksize; // 当前栈的大小  } Sqstack;  Status STACK\_INIT(Sqstack& s) {  s.base = (s\_Elemtype\*)malloc(sizeof(s\_Elemtype) \* INIT\_SIZE); // 为栈分配初始内存  if (!s.base) return ERROR; // 内存分配失败  s.top = s.base; // 栈顶指针指向栈底  s.stacksize = INIT\_SIZE; // 设置初始栈大小  return OK;  }  Status STACK\_PUSH(Sqstack& s, s\_Elemtype e) {  if (s.top - s.base == s.stacksize) { // 栈满时需要扩展内存  s\_Elemtype\* newbase = (s\_Elemtype\*)realloc(s.base, (s.stacksize + EXTENSION\_SIZE) \* sizeof(s\_Elemtype)); // 扩展内存  if (!newbase) return ERROR; // 扩展失败  s.base = newbase; // 更新栈基址  s.top = s.base + s.stacksize; // 更新栈顶指针  s.stacksize += EXTENSION\_SIZE; // 更新栈大小  }  \*s.top++ = e; // 将元素入栈  return OK;  }  Status STACK\_POP(Sqstack& s, s\_Elemtype& e) {  if (s.top == s.base) return ERROR; // 栈空时无法出栈  e = \*--s.top; // 将栈顶元素出栈  return OK;  }  int main() {  cout << "--------------------------------------" << endl;  cout << "This is the homework from Zhouxing in AI2302" << endl;  cout << "--------------------------------------" << endl;  cout << "Ok, now you can insert the numbers you want to trans" << endl;  Sqstack stack1;  STACK\_INIT(stack1); // 初始化栈  int x, y;  cin >> x;  cout << "The number required is " << x << endl;  // 将十进制数转换为八进制数并入栈  while (x != 0) {  y = x % 8;  STACK\_PUSH(stack1, y); // 将余数入栈  x /= 8; // 更新商作为下一轮计算的被除数  }  int e, a[100], cnt = 0;  cout << "--------------------------------------" << endl;  cout << "Well, the % process is as follows" << endl;  // 依次出栈并输出各位八进制数字  while (STACK\_POP(stack1, e) != ERROR) {  cout << e << endl;  a[cnt] = e; // 将出栈的元素保存到数组中  cnt++; // 统计出栈元素个数  }  cout << "--------------------------------------" << endl;  cout << "The final answer is " << endl;  // 逆序输出数组中的八进制数字，得到最终结果  for (int j =0; j < cnt ; j++) {  cout << a[j];  }  return 0;  } | | |
| 实验3  #include<iostream>  #include<stdlib.h>  #define MAX\_SIZE 100  #define OK 1  #define ERROR -1  using namespace std;  typedef int QElemtype;  typedef int Status;  typedef struct { //定义循环队列  QElemtype\* base;//定义一个基指针  int front; //头指针  int rear; //尾指针  }SqQueue;  /\*在循环队列中，基指针的作用是指向队列的起始位置，即队列的底层数组的首地址。  具体来说，基指针指向存储队列元素的数组的首地址，它用于帮助确定队列中元素的实际位置。通过基指针和下标的组合，可以访问到队列中的各个元素。  循环队列中的数据元素在数组中是连续存储的，从基指针开始，通过不断增加下标，可以循环地遍历数组，实现队列的循环性质。\*/  Status Init\_Queue(SqQueue& Q) {//初始化循环队列  Q.base = new QElemtype[MAX\_SIZE];//通过基指针为循环队列静态分配空间  //Q.base = (QElemtype\*)malloc(MAX\_SIZE \* sizeof(QElemtype));  if (!Q.base) exit(OVERFLOW);//如果分配失败了  Q.front = Q.rear = 0;//表示队列的初始索引  return OK;//完成  }  int QueueLength(SqQueue Q) {//返回队列的长度  return (Q.rear - Q.front + MAX\_SIZE) % MAX\_SIZE;//队尾的序列减去队首的索引（加上MAX取余是循环操作）  }  Status In\_Queue(SqQueue& Q, QElemtype e) {//插入元素  if ((Q.rear + 1) % MAX\_SIZE == Q.front) return ERROR;//如果循环队列没有位置了，返回错误  Q.base[Q.rear] = e;//将e的数值通过队尾的索引添加到队列中  Q.rear = (Q.rear + 1) % MAX\_SIZE;//队尾指针往后走一位  return OK;//完成  }  Status DeQueue(SqQueue& Q, QElemtype& e) {//删除元素  if (Q.front == Q.rear) return ERROR;;//如果循环队列没有元素，返回错误  e = Q.base[Q.front];//将队首指针的元素赋给e  Q.front = (Q.front + 1) % MAX\_SIZE;//队首指针向后一位  return OK;//完成  }  QElemtype GetHead(SqQueue& Q) {//取出队列的队首元素  if (Q.front != Q.rear)//如果队里面有元素  return Q.base[Q.front];//返回队首元素  else {  exit(ERROR);//否则退出  }  }  int main() {  SqQueue q1;  Init\_Queue(q1);  cout << "This is the homework from Zhouxing in AI2302" << endl;  cout << "--------------------------------------" << endl;  cout << "-------------------------" << endl;  cout << "Insert the number of elements you want to push into the queue:" << endl;  int num1; cin >> num1;  cout << "-------------------------" << endl;  cout << "Insert the elements you want to push into the queue:" << endl;  for (int i = 1; i <= num1; i++) {  int ele1; cin >> ele1;  In\_Queue(q1, ele1);  }  cout << "-------------------------" << endl;  cout << "The elements you inserted are as follows:" << endl;  // 输出插入的元素  int front = q1.front;  int rear = q1.rear;  while (front != rear) {  cout << q1.base[front] << " ";  front = (front + 1) % MAX\_SIZE;  }  cout << endl;  cout << "-------------------------" << endl;  cout << "The top element of the queue is :" << endl;  cout << GetHead(q1) << endl;  cout << "-------------------------" << endl;  cout << "Insert the number of elements you want to pop from the queue:" << endl;  int num2; cin >> num2;  cout << "-------------------------" << endl;  cout << "The elements you popped from the queue are as follows:" << endl;  for (int i = 1; i <= num2+1; i++) {  int ele2;  if (DeQueue(q1, ele2) == OK) {  cout << ele2 << endl;  }  }  return 0;  } | | |
| 实验结论： | | |
| 成功完成所有任务并且对循环队列，栈，十进制转八进制有了更深的了解 | | |