《数据结构》实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 李爽 | 班级 | 22级软件工程专升本1班 | 学号 | 2206831522 |
| 实验名称 | **实验三栈和队列** | | | | |
| 实验时间 | 2023.4.20 | | 成绩 | |  |
| 1. **实验目的：**   1.掌握栈的顺序存储结构和队列的链接存储结构；  2.验证顺序栈和链队列及其基本操作的实现；  3.验证栈和队列的操作特性。  **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）入栈函数：首先判断当前栈是否已满，若已满则输出上溢错误信息；否则新建一个节点，将待入栈元素存储在节点的数据域中，将该节点插入栈顶并更新栈顶指针 top，最后将栈大小 size 加一。  （2）出栈函数：首先判断传入的参数是否合法，若不合法则输出错误信息。然后判断当前栈是否为空或待出栈的元素个数大于栈中元素个数，若是则输出下溢错误信息。接着，循环 n 次，每次从栈顶取出一个元素，输出该元素的值，删除该节点并释放其内存空间，将栈大小 size 减一。  （3）取栈顶元素函数：首先判断当前栈是否为空，若为空则输出错误信息；否则直接输出栈顶节点的数据域。  （4）入队函数：如果队列已满，则输出提示信息并返回；否则，将元素存储到队尾，然后将队尾指针加 1。  （5）出队函数：首先，如果队列为空，则输出提示信息并返回 -1。否则，从队头开始循环，取出队头元素，队头指针加 1，输出出队元素，并将计数器加 1。当出队元素数量达到 n 或队列已空时，退出循环并输出换行符，然后返回出队元素数量。  （6）取队头元素函数：取队头元素函数的算法比较简单。如果队列为空，则输出提示信息并返回 -1；否则，返回队头元素的值。  **四、实验步骤和内容：**  1.建立一个空栈，并按要求完成链栈的相关操作：  （1）编写入栈函数，随机产生10个100以内的整数，并按输入顺序依次入栈,考虑判断上溢情况；  （2）编写出栈函数，从键盘输入出栈元素个数n（n<10），将n个元素出栈并显示出栈元素，考虑判断下溢情况；  （3）编写取栈顶元素函数，若栈不为空，则取出栈顶元素显示；否则提示栈空。  2. 建立一个空队列，并按要求完成循环队列的相关操作：  （1）编写入队函数，随机产生10个100以内的整数，并按输入循环依次入队；  （2）编写出队函数，从键盘输入出队元素个数n（n<10），将n个元素出队后显示队中数据结果，考虑判断队空情况；  （3）编写取队头元素函数，若队不为空，则取队头元素显示；否则提示队空。  1.代码如下  //用于输入输出  #include <iostream>  // 包含 rand() 函数的头文件  #include <cstdlib>  //命名空间  using namespace std;  // 定义链表节点结构体  struct Node {  // 数据域  int data;  // 指针域，指向下一个节点  Node\* next;  };  // 定义栈类  class Stack {  private:  // 栈顶指针  Node\* top;  // 栈的大小  int size;  public:  // 构造函数，初始化空栈  Stack() {  top = NULL;  size = 0;  }  // 析构函数，释放栈中所有节点的内存  ~Stack() {  while (top != NULL) {  Node\* temp = top;  top = top->next;  delete temp;  }  }  // 入栈函数  void push(int value) {  // 判断上溢  if (size >= 10) {  cout << "错误：栈上溢。" << endl;  return;  }  //定义一个新节点  Node\* newNode = new Node;  newNode->data = value;  newNode->next = top;  //将栈顶元素指向新节点  top = newNode;  //栈大小+1  size++;  }  // 出栈函数  void pop(int n) {  // 参数非法  if (n <= 0) {  cout << "错误：无效的参数。" << endl;  return;  }  // 判断下溢  if (size < n) {  cout << "错误：栈下溢。" << endl;  return;  }  cout << "从栈中弹出 " << n << " 个元素：" << endl;  for (int i = 0; i < n; i++) {  Node\* temp = top;  //逐个显示弹出元素  cout << temp->data << " ";  top = top->next;  delete temp;  //栈大小-1  size--;  }  cout << endl;  }  // 取栈顶元素函数  void getTop() {  // 判断栈空  if (size == 0) {  cout << "错误：栈为空。" << endl;  return;  }  cout << "栈顶元素为：" << top->data << endl;  }    //我添加了一个函数，来打印栈中元素信息  //方便观察栈中元素  void printStack() {  // 判断栈空  if (size == 0) {  cout << "当前栈为空。" << endl;  return;  }  cout << "当前栈中元素为（从栈顶到栈底）：" << endl;  Node\* temp = top;  while (temp != NULL) {  cout << temp->data << " ";  temp = temp->next;  }  cout << endl;  }  };  int main() {  Stack s;  // 入栈  for (int i = 0; i < 10; i++) {  int value = rand() % 100;  s.push(value);  }    s.printStack();  // 出栈  int n;  cout << "请输入您想从栈中弹出的元素个数（小于10）：";  cin >> n;  s.pop(n);    s.printStack();  // 取栈顶元素  s.getTop();  return 0;  }    2.代码如下  //用于输入输出  #include <iostream>  //用于生成随机数  #include <cstdlib>  //用于获取当前时间  #include <ctime>  //命名空间  using namespace std;  // 队列最大容量  const int MAX\_SIZE = 10;  class Queue {  private:  // 用于存储队列元素的数组  int arr[MAX\_SIZE];  // 队头和队尾指针  int front, rear;  public:  Queue() {  // 初始化队头指针  front = 0;  // 初始化队尾指针  rear = 0;  }  //队列是否为空  bool isEmpty() {  // 队列为空的条件是队头和队尾指针相等  return front == rear;  }  //队列是否已满  bool isFull() {  // 队列已满的条件是队尾指针加 1 等于队头指针  return (rear + 1) % MAX\_SIZE == front;  }  //入队函数  void enqueue(int value) {  // 如果队列已满，输出提示信息并返回  if (isFull()) {  cout << "队列已满。" << endl;  return;  }  // 将元素存储到队尾  arr[rear] = value;  // 队尾指针加 1  rear = (rear + 1) % MAX\_SIZE;  }  //出队函数  int dequeue(int n) {  // 如果队列为空，输出提示信息并返回 -1  if (isEmpty()) {  cout << "队列为空。" << endl;  return -1;  }  // 记录出队元素的数量  int count = 0;  // 只要队列非空且出队元素数量不足 n，就一直循环  while (!isEmpty() && count < n) {  // 取出队头元素  int value = arr[front];  // 队头指针加 1  front = (front + 1) % MAX\_SIZE;  // 输出出队元素  cout << value << " ";  // 计数器加 1  count++;  }  // 输出换行符  cout << endl;  // 返回出队元素数量  return count;  }  //返回对头元素  int peek() {  // 如果队列为空，输出提示信息并返回 -1  if (isEmpty()) {  cout << "队列为空。" << endl;  return -1;  }  // 返回队头元素  return arr[front];  }  };  int main() {  //创建一个队列对象  Queue q;  // 初始化随机数生成器  srand(time(NULL));    // 入队10个随机数  for (int i = 0; i < 10; i++) {  int value = rand() % 100;  q.enqueue(value);  }    // 从队列中出队n个元素  int n;  cout << "请输入需要出队的元素数量：";  cin >> n;  int count = q.dequeue(n);  cout << "已出队 " << count << " 个元素。" << endl;    // 获取队头元素  int front = q.peek();  if (front != -1) {  cout << "队头元素为：" << front << endl;  }    return 0;  }    **五、实验总结：**  在本次实验中，我们使用C++语言编写了栈和队列的数据结构，并验证了它们的基本操作及其特性。  首先，我们掌握了栈的顺序存储结构和队列的链接存储结构。栈的顺序存储结构使用数组实现，可以使用顶部指针来实现入栈和出栈操作。队列的链接存储结构使用链表实现，可以使用头指针和尾指针来实现入队和出队操作。  接着，我们验证了顺序栈和链队列的实现，包括入栈、出栈、入队、出队等基本操作。我们通过编写代码，并进行多组测试，验证了它们的正确性和可靠性。  最后，我们验证了栈和队列的操作特性。栈具有“后进先出”的特性，也就是说，最后入栈的元素会最先出栈。队列具有“先进先出”的特性，也就是说，最先入队的元素会最先出队。我们通过编写代码，并进行多组测试，验证了它们的特性。  总之，本次实验让我们更深入地了解了栈和队列的数据结构及其基本操作，也让我们更加熟悉了C++编程语言。通过实验，我们进一步掌握了数据结构和算法的知识，为我们以后的学习和实践奠定了坚实的基础。  **六、教师评语：** | | | | | |