《数据结构》实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 马昆 | 班级 | 软件工程专升本1班 | 学号 | 2206831544 |
| 实验名称 | **实验三栈和队列** | | | | |
| 实验时间 | 2023年4月15日 | | 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：如果栈顶指针指向栈的最大长度-1，就直接抛异常，表示栈已满。如果栈没满就将栈顶指针指向下一位并将栈中这个位置的元素的值设置为入栈的元素的值。  （2）出栈函数：根据栈顶指针，将栈顶元素的数据的值返回出去，并将栈顶指针往回移动一位，可以使用自减运算符，减少变量的创建。  （3）取栈顶元素函数：直接根据栈顶指针，将栈顶元素的值返回出去。  （4）入队函数：因为使用了循环队列，所以当队尾指针+1的和与栈总长度取模的值与头指针相等，就说明栈满了，就抛出异常。没有满就将队尾指针更新为当前队尾指针+1的和与栈总长度取模的值，然后再将此位置的元素赋值为入队元素的值。并将队列对象的length成员变量加1。  （5）出队函数：如果头指针与尾指针相同，说明队列是空的，就直接抛出异常。如果不是空的，就将成员变量length减1，并将头指针重新赋值为当前头指针+1的和与队列总长度取模的值。最后返回新头指针所指向的元素的值。  （6）取队头元素函数：如果头指针与尾指针相同，说明队列是空的，就直接抛出异常。如果不为空，就返回头指针+1的和与队列总长度取模的指针所指向的元素的值。  **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；   |  | | --- | | // 入栈  template <typename DataType>  void SeqStack<DataType>::Push(DataType x) {  if (this->top == StackSize - 1) throw "栈溢出";  this->data[++top] = x;  } | | #include "SeqStack.h"  #include <iostream>  #include <random>  using namespace std;  int main() {  default\_random\_engine engine; // 创建随机数引擎  engine.seed(time(0)); // 将当前时间的时间戳设置为随机数种子  try {  SeqStack<> stack; // 创建顺序栈  // 设置10个0~100的随机数入栈  for (int i = 0; i < 10; i++) {  stack.Push(engine() % 101); // 随机数入栈  }  while (!stack.Empty()) {  cout << stack.Pop() << "弹出了" << endl;  }  }  catch (const char \* msg) {  cout << "出错了：" << msg << endl;  }  return 0;  } | | 正常情况  设置栈大小为2 |   （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；   |  | | --- | | // 出栈  template <typename DataType>  DataType SeqStack<DataType>::Pop() {  if (this->top == -1) throw "栈下溢";  return this->data[this->top--];  } | | #include "SeqStack.h"  #include <iostream>  #include <random>  using namespace std;  int main() {  default\_random\_engine engine; // 创建随机数引擎  engine.seed(time(0)); // 将当前时间的时间戳设置为随机数种子  try {  SeqStack<> stack; // 创建顺序栈  // 设置10个0~100的随机数入栈  for (int i = 0; i < 10; i++) {  stack.Push(engine() % 101); // 随机数入栈  }  int user\_in = 0;  cout << "请输入要弹出的个数：";  cin >> user\_in;  for (int i = 0; i < user\_in; i++) {  cout << i + 1 << "、" << stack.Pop() << "弹出了" << endl;  }  }  catch (const char \* msg) {  cout << "出错了：" << msg << endl;  }  return 0;  } | | 正常情况  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。   |  | | --- | | // 取栈顶  template <typename DataType>  DataType SeqStack<DataType>::GetTop() {  if (this->top == -1) throw "栈下溢";  return this->data[this->top];  } | | #include "SeqStack.h"  #include <iostream>  #include <random>  using namespace std;  int main() {  default\_random\_engine engine; // 创建随机数引擎  engine.seed(time(0)); // 将当前时间的时间戳设置为随机数种子  try {  SeqStack<> stack; // 创建顺序栈  // 设置10个0~100的随机数入栈  for (int i = 0; i < 10; i++) {  stack.Push(engine() % 101); // 随机数入栈  }  if (stack.Empty()) {  cout << "栈空";  }  else {  cout << "栈顶元素为：" << stack.GetTop();  }  }  catch (const char \* msg) {  cout << "出错了：" << msg << endl;  }  return 0;  } | |  |   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；   |  | | --- | | // 入队  template <typename DataType>  void CirQueue<DataType>::EnQueue(DataType x) {  if ((this->end + 1) % QueueSize == this->start) throw "满";  this->data[this->end = ((this->end + 1) % QueueSize)] = x;  this->length++;  } | | #include "CirQueue.h"  #include <iostream>  #include <random>  using namespace std;  int main() {  default\_random\_engine engine; // 创建随机数引擎  engine.seed(time(0)); // 将当前时间的时间戳设置为随机数种子  try {  CirQueue<> queue; // 创建循环队列  // 设置10个0~100的随机数入队  for (int i = 0; i < 10; i++) {  queue.EnQueue(engine() % 101); // 随机数入队  }  while (!queue.Empty()) {  cout << queue.DeQueue() << "出队了\n";  }  }  catch (const char \* msg) {  cout << "出错了：" << msg << endl;  }  return 0;  } | |  |   （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；   |  | | --- | | // 出队  template <typename DataType>  DataType CirQueue<DataType>::DeQueue() {  if (this->start == this->end) throw "空队";  this->length--;  return this->data[this->start = (this->start + 1) % QueueSize];  } | | #include "CirQueue.h"  #include <iostream>  #include <random>  using namespace std;  int main() {  default\_random\_engine engine; // 创建随机数引擎  engine.seed(time(0)); // 将当前时间的时间戳设置为随机数种子  try {  CirQueue<> queue; // 创建循环队列  // 设置10个0~100的随机数入队  for (int i = 0; i < 10; i++) {  queue.EnQueue(engine() % 101); // 随机数入队  }  int user\_in = 0;  cout << "请输入要出队的数量：";  cin >> user\_in;  for (int i = 0; i < user\_in; i++) {  cout << i + 1 << "、" << queue.DeQueue() << "出队了\n";  }  }  catch (const char \* msg) {  cout << "出错了：" << msg << endl;  }  return 0;  } | | 正常情况  错误情况 |   （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。   |  | | --- | | // 取队头  template <typename DataType>  DataType CirQueue<DataType>::GetHead() {  if (this->start == this->end) throw "空队";  return this->data[(this->start + 1) % QueueSize];  } | | #include "CirQueue.h"  #include <iostream>  #include <random>  using namespace std;  int main() {  default\_random\_engine engine; // 创建随机数引擎  engine.seed(time(0)); // 将当前时间的时间戳设置为随机数种子  try {  CirQueue<> queue; // 创建循环队列  // 设置10个0~100的随机数入队  for (int i = 0; i < 10; i++) {  queue.EnQueue(engine() % 101); // 随机数入队  }  if (queue.Empty()) {  cout << "队空";  }  else {  cout << "队头元素为：" << queue.GetHead();  }  }  catch (const char \* msg) {  cout << "出错了：" << msg << endl;  }  return 0;  } | |  |   **五、实验总结：**  **本次实验完成了顺序栈与循环队列的编写，由于之前作业4就已经写过基于这两个数据结构的应用题，就直接拿过来用了，只需要更改一下main函数就行了，然后由于需要处理溢出的情况，就在顺序栈与循环队列中可能出现的错误情况抛出异常，然后在main函数中捕获异常，提高程序的健壮性。**  **六、教师评语：** | | | | | |