数据结构实验报告3

学号：2022114467 姓名：林泽宇 专业：软件工程

知识范畴：栈与队列 完成日期：2023年 月 日

实验题目：队列基本操作算法

|  |  |  |  |
| --- | --- | --- | --- |
| 课程目标1（60%） | 课程目标2（40%） | 得分(100分)  期末成绩占比5% | 批阅人签字 |
| 程序代码及测试 | 写作、排版、代码注释等 |  |
|  |  |  |

实验内容及要求：

编写程序，建立容量为n(建议n=8)的循环队列，完成以下程序功能。输入字符#，执行一次出队操作，屏幕上显示出队字符；输入字符@，队列中所有字符依次出队并按出队次序在屏幕上显示各字符；输入其它字符，则输入的字符入队。

要求采用队头/队尾间隔至少一个空闲元素的方法来实现循环队列；空队执行出队操作及队满执行入队操作需显示提示信息。

注意：反复输入字符时，需正确处理好回车键（即输入缓冲区的’\n’字符）的读取问题。

实验目的：掌握循环队列的基本操作算法。

数据结构设计简要描述：

private:

    elemT \*elem;

    int front;            //头指针游标

    int rear;             //尾指针游标

    int size;             //容量

算法设计简要描述：

实现循环队列的创建、销毁、判空、判满、入队和出队的基本操作；

然后通过连续出队实现清空操作。

输入/输出设计简要描述：

读取每行输入的首个字符为ch：

如果ch为#，则出队并输出；如果ch为@，则进行清除操作并按出队次序逐个输出。

编程语言说明：

使用C++编程。循环队列的定义通过类模板实现；动态存储分配采用 C++ 的 new 和 delete 操作符实现；输入输出采用 getchar 与 printf 函数实现。

主要函数说明：

public:

    cirQueue(int initsize = 8);   //构造函数, 初始容量默认为8

    ~cirQueue();                  //析构函数

    void enQueue(const elemT &e); //将元素e入队

    elemT deQueue();              //出队, 返回出队元素

    void Clear();                 //将所有元素依此出队并输出

    bool isEmpty() const;         //判断队列是否为空

    bool isFull() const;          //判断队列是否为满

程序测试简要报告：

测试1:

文本, 信件

描述已自动生成

结论: 入队、出队、判空、清空操作正常。

测试2：

文本, 信件

描述已自动生成

结论: 当一次输入多个字符时, 程序只会读入每行的首个字符, 避免用户发生误输入导致程序出错。

测试3：

文本, 信件

描述已自动生成

结论：程序的队列可以循环存储

源程序代码：

#include <iostream>

using namespace std;

//根据题目要求, 需要实现创建、销毁、出队和入队的基本操作

template <class elemT>

class Queue //原型设计

{

private:

public:

    //创建和销毁由构造函数和析构函数实现

    virtual void enQueue(const elemT &e) = 0; //入队

    virtual elemT deQueue() = 0;              //出队

    virtual void Clear() = 0;                 //清空

    virtual bool isEmpty() const = 0;         //判空

    virtual bool isFull() const = 0;          //判满

};

template <class elemT>

class cirQueue : public Queue<elemT>

{

private:

    elemT \*elem;

    int front;            //头指针游标

    int rear;             //尾指针游标

    int size;             //容量

    void downflow() const //下溢报错

    {

        printf("发生下溢!\n");

        exit(-1);

    }

    void overflow() const //上溢报错

    {

        printf("发生上溢\n");

        exit(-1);

    }

    void badalloc() const //内存分配失败报错

    {

        printf("内存分配失败!\n");

        exit(-1);

    }

public:

    cirQueue(int initsize = 8);   //构造函数, 初始容量默认为8

    ~cirQueue();                  //析构函数

    void enQueue(const elemT &e); //将元素e入队

    elemT deQueue();              //出队, 返回出队元素

    void Clear();                 //将所有元素依此出队并输出

    bool isEmpty() const;         //判断队列是否为空

    bool isFull() const;          //判断队列是否为满

};

int main() // 数据测试

{

    cirQueue<char> q;      //建立字符循环队列

    char ch;               //存储操作符

    string tmp;            // tmp用来处理缓存区的垃圾数据

    while (ch = getchar()) //重复读入输入的首个字符

    {

        switch (ch)

        {

        case '#':

            printf("将元素'%c'出队\n", q.deQueue());

            break;

        case '@':

            printf("执行清空操作:\n");

            q.Clear();

            break;

        default:

            q.enQueue(ch);

            printf("已将输入字符'%c'入队\n", ch);

        }

        getline(cin, tmp); //只读入每行的第一个字符, 将误输入的数据丢到tmp中

        //如果这里使用getchar(), 如果每行垃圾数据不止'\n'的话依旧会出错

    }

}

template <class elemT>

cirQueue<elemT>::cirQueue(int initsize)

{

    // 如果内存分配失败, new只有在nothrow的情况下会返回nullptr, 否则只会抛出异常

    elem = new (nothrow) elemT[initsize];

    if (!elem) //如果不添加nothorw, 这个判断将会是徒劳

        badalloc();

    front = rear = 0;

    size = initsize;

}

template <class elemT>

cirQueue<elemT>::~cirQueue()

{

    front = rear = -1; //将指针置为异常值

    delete elem;

}

template <class elemT>

void cirQueue<elemT>::enQueue(const elemT &e)

{

    if (isFull()) //判满

        overflow();

    elem[rear] = e;

    rear = (rear + 1) % size;

}

template <class elemT>

elemT cirQueue<elemT>::deQueue()

{

    if (isEmpty()) //判空

        downflow();

    rear = (rear - 1 + size) % size;

    return elem[rear];

}

template <class elemT>

void cirQueue<elemT>::Clear()

{

    if (isEmpty())

        return (void)printf("队列目前为空, 请继续输入:\n");

    int cnt = 0;

    while (!isEmpty())

    {

        printf("%5c", deQueue()); //设定位宽为5

        if (++cnt % 5 == 0)       //五个换行

            printf("\n");

    }

    if (cnt % 5)

        printf("\n");

}

template <class elemT>

bool cirQueue<elemT>::isEmpty() const

{

    return front == rear;

}

template <class elemT>

bool cirQueue<elemT>::isFull() const

{

    return front == (rear + 1) % size;

}