

队列实验报告

课程名称：数据结构与算法

学 院： 计算机与信息安全学院

专业班级：19003603

学 号：1900301517

姓 名：陆洪业

报告日期：2021年 01 月 10 日

目录

**[1. 利用队列实现密码加密解密实验目的](#_Toc1831915961_WPSOffice_Level1)** **[1](#_Toc1831915961_WPSOffice_Level1)**

[1.1. 选用储存结构](#_Toc647397264_WPSOffice_Level2) [1](#_Toc647397264_WPSOffice_Level2)

[1.2. 实验代码](#_Toc1130979933_WPSOffice_Level2) [1](#_Toc1130979933_WPSOffice_Level2)

[1.3. 系统测试与结果](#_Toc1068613576_WPSOffice_Level2) [7](#_Toc1068613576_WPSOffice_Level2)

[1.4. 实验小结](#_Toc1598252537_WPSOffice_Level2) [8](#_Toc1598252537_WPSOffice_Level2)

# 利用队列实现密码加密解密实验目的

实现队列基本操作，并使用队列基本操作解决密码加密和解密问题。

## 选用储存结构

链表

## 实验代码

typedef char DataType;

//队列数据元素结构

typedef struct node

{

DataType info;

struct node \*next;

}QueueData;

typedef struct queueRecord{

QueueData \*front, \*rear ;

}LINKQUEUE;

typedef struct queueRecord \*PLinkQueue;

PLinkQueue createEmptyQueue\_link( )

{

//创建一个空队列，实质：生成一个LINKQUEUE类型的结点，并给front 和 rear 成员赋值

//请在此处填写代码，完成相应功能

/\*-------begin----------\*/

PLinkQueue Q = new LINKQUEUE;

if(NULL == Q) {

cout << "Out of Space\n!" << endl;

return NULL;

}

Q->front = Q->rear = NULL;

return Q;

/\*-------end----------\*/

}

int isEmptyQueue\_link(PLinkQueue queue)

{ //判定队列是否为空，实质： 看队列的front指针是否为空，若为空，则队列为空

//请在此处填写代码，完成相应功能

/\*-------begin----------\*/

if (NULL == queue || NULL == queue->front)

return 1;

else

return 0;

/\*-------end----------\*/

}

void enQueue\_link(DataType x, PLinkQueue queue)

{

//将数据元素x插入队尾。实质：生成一个struct node类型的结点，并给相应成员赋值后插入队尾

//请在此处填写代码，完成相应功能

/\*-------begin----------\*/

QueueData \* p = new QueueData;

if(NULL == p) {

cout << "Out of Space\n!" << endl;

return;

}

p->info = x;

p->next = NULL;

if (isEmptyQueue\_link(queue)) {

queue->front = queue->rear = p;

} else {

queue->rear->next = p;

queue->rear = p;

}

/\*-------end----------\*/

}

void enQueue\_link(QueueData \*p,PLinkQueue Q)

{

//本函数的功能是将 struct node 类型的结点插入队尾，与enQueue\_link(DataType x, PLinkQueue queue)的区别是，不需要生成结点

//请在此处填写代码，完成相应功能

/\*-------begin----------\*/

if (isEmptyQueue\_link(Q)) {

Q->front = Q->rear = p;

} else {

Q->rear->next = p;

Q->rear = p;

}

/\*-------end----------\*/

}

DataType deQueue\_link(PLinkQueue Q)

{

//出队，实质： 取出Q队列的队首结点，返回该结点的数据元素，并将该结点使用enQueue\_link(QueueData \*p,PLinkQueue Q)插入队尾

//本函数为针对本实验特殊设计，可实现秘钥的循环使用

//请在此处填写代码，完成相应功能

/\*-------begin----------\*/

// 出队

if (isEmptyQueue\_link(Q))

return 0;

QueueData \* p = Q->front;

Q->front = Q->front->next;

if (NULL == Q->front)

Q->rear = NULL;

// 入队

enQueue\_link(p, Q);

return p->info;

/\*-------end----------\*/

}

void decrypt(char src[], char key[], char dest[])

{

/\* 参数： src 放的是源文 ； key 放的是秘钥 ； dest 放加密后的密文

加密函数： 步骤提示

1. 初始化一个空队列Q

2. 将key数组的元素逐个插入队列Q

3. 逐个访问src数组的源文，并加密， 使用deQueue\_link(Q)函数可得到队首秘钥，并使秘钥循环使用

\*/

int i;

// 创建队列

PLinkQueue Q = createEmptyQueue\_link();

// 将密文加入队列

for (i = 0; key[i] != '\0'; ++i) {

enQueue\_link(key[i], Q);

}

// 读取源文并加密

for (i = 0; src[i] != '\0'; ++i) {

dest[i] = src[i] + deQueue\_link(Q) - '0';

}

dest[i] = '\0';

}

int main(void)

{

char src[80] , key[20], dest[80] ;

cin>>src ;

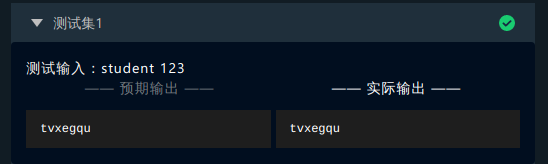
cin>>key;

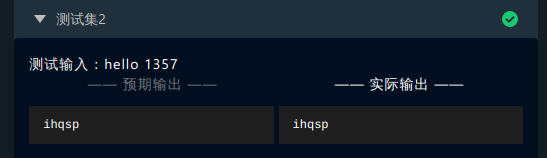
decrypt(src , key , dest);

cout<<dest;

}

## 系统测试与结果





上述测试及结果证明了系统测试通过,各项操作函数能正常执行,并达到了实验要求的目标,实现了使用队列进行密码加密的要求.

## 实验小结

通过本次实验, 我学会了循环队列的使用, 并且能使用它设计一些简单的加密程序.