

**数据结构实验报告（四）**

实验课题：队列的基本操作

学 院：理学院

专 业：数学系

组 员：190110304王君涛

190110312战祥贤

190110318王 鹏

190110302王 琪

190110331陈 立

组 长：王君涛

2020年10月17日

队列的实现及应用

队列的基本操作：

* 队列的初始化
* 返回队列长度
* 入队
* 出队列并返回队列
* 返回队列头元素
* 显示队列内全部元素

具体实例：

* 用顺序循环队列实现进制转换

顺序队列的基本操作具体实现

准备部分：

用Queue实现顺序表并在Queue类内定义：存储空间的基地址（char\* base;//）队头标记（ int front;//类似于头指针）队尾标记（int rear;//类似于尾指针）。类内定义实现函数 初始化（void initQueue();）返回队列长度（int QueueLen()）入队（void EnQueue(char e);）出队（char DeQueue();）返回队列首元素（char GetHead();）显示队列内元素（void ShowQueue()）

注：本程序用到的front与rear为广义的指针标记用于标记队列的首个元素与最后元素记为头指针与尾指针。

class Queue

{

private:

char\* base;//队列初始地址

int front;//类似于头指针

int rear;//类似于尾指针

public:

void InitQueue();//初始化顺序队列

int QueueLen();//返回队列长度

void EnQueue(char e);//入队

char DeQueue();//出队

char GetHead();//取头元素

void ShowQueue();//显示队列元素

};

队列的初始化

* 说明：

首先让队列基地址指向新建的一个队列存储空间（base =char [MAXSIZE]; 初始化队列，将尾指针与头指针归0（front = rear = 0;） 初始化结束。

* 实现函数：

void Queue::InitQueue()//初始化队列，将尾指针与头指针归0，

{

base = new char[MAXSIZE];

front = rear = 0;

1. 返回队列长度

* 说明：

用尾指针与头指针的相对位置计算队列长度（(rear - front + MAXSIZE) % MAXSIZE）

* 实现函数：

int Queue::QueueLen()//返回队列长度

{

return((rear - front + MAXSIZE) % MAXSIZE);

}

1. 入队

* 说明：

首先通过头指针与尾指针下一个位置的标记是否相同（因为这里用到循环队列所以尾指针的下一个位置用(rear + 1) % MAXSIZE计算）判断队列是否已满。（if ((rear + 1) % MAXSIZE==front) {cout << "队列已满，操作失败。。"；）如果队列未满则填充rear“指针指向”的空间（\*(base + rear) = e;） 最后尾指针指向下一个位置（rear = (rear + 1) % MAXSIZE;这里用到循环队列尾指针要用(rear + 1) % MAXSIZE）

* 实现函数：

void Queue::EnQueue(char e)//入队

{

if ((rear + 1) % MAXSIZE == front)//判断是否队满

{

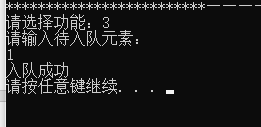
cout << "队满。。。。操作失败" << endl;

return;

}

\*(base + rear) = e;//填充rear“指针指向”的空间

rear = (rear + 1) % MAXSIZE;

}运行结果：

1. 出队列并返回队列顶元素

* 说明：

首先通过比较头指针与尾指针是否相同判断队列内是否有元素（if (rear == front){cout << "队列为空，出队列失败。。" ）如果队列内有元素让头指针指向下一个位置front = (front + 1) % MAXSIZE;返回队列头个元素完成出队

实现函数：

char Queue::DeQueue()//出队，头指针进一

{

if (rear == front)//判断是否队空

{

cout << "队空！操作失败。。。" << endl;

return 0;

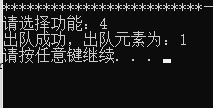
}

char e;

e = \*(base + front);//保存对头元素

front = (front + 1) % MAXSIZE;//头指针前进！！

return e;//返回保存的元素值

}运行结果：

1. 返回队列顶元素

* 说明：

首先通过头指针与尾指针是否相同判断队列是否为空（if (rear == front)）如果非空返回队列顶元素(return \*(base + front);)

* 实现函数：

char Queue::GetHead()//仅返回对头元素

{

if (rear == front)

{

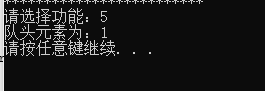
cout << "队空！操作失败。。。" << endl;

return 0;

}

return \*(base + front);

}

运行结果：

1. 显示队列内全部元素

* 说明

首先用求长度函数（QueueLen()）判断队列内是否有元素（ n = QueueLen(); if (n == 0)）当队列内有元素时通过for循环（ for (int i = 0; i < n; i++)）（ cout << \*(base + front + i) << ' ';）实现队列内所有元素的展示

实现函数

void Queue::ShowQueue()

{

int n;

n = QueueLen();

if (n == 0)

{

cout << "空队。。。" << endl;

return;

}

for (int i = 0; i < n; i++)

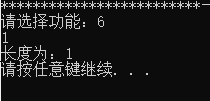
{

cout << \*(base + front + i) << ' ';

}

cout << "\n长度为：" << n << endl;

}

运行结果：

主函数

void showMenu()

{

cout << "\*\*\*\*\*\*\*\*操作菜单\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*功能1：初始化循环队列\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*功能2：返回队列长度 \*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*功能3：入队\* \*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*功能4：出队\* \*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*功能5：返回队头元素\* \*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*功能6：显示循环队列中的元素 \*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*功能0：退出并结束程序\* \*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*———————\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\* 1901103班2组 \*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*———————\*\*\*\*\*\*\*\*\*\*" << endl;

}

int main()

{

Queue Q;//创建一个队列对象

int Choice = 0;

Q.InitQueue();

cout << "初始化循环队列成功。。。。" << endl;

while (true)

{

system("cls");//清屏

showMenu();//显示菜单

cout << "请选择功能：";

while (true)

{

cin >> Choice;

if (Choice >= 0 && Choice <= 6)

break;

else

cout << "输入错误，请重新输入" << endl;

}

switch (Choice)

{

case 1://功能1：初始化循环队列

Q.InitQueue();

cout << "初始化顺序栈成功。。" << endl;

system("pause");

break;

case 2://功能2：返回队列长度

int e;

e = Q.QueueLen();

cout << "队列长度为：" << e << endl;

system("pause");

break;

case 3://功能3：入队

char e3;

cout << "请输入待入队元素：" << endl;

cin >> e3;

Q.EnQueue(e3);

system("pause");

break;

case 4://功能4：出队

char e4;

e4 = Q.DeQueue();

if (e4)

cout << "出队成功，出队元素为：" << e4 << endl;

system("pause");

break;

case 5://功能5：返回队头元素

char e5;

e5 = Q.GetHead();

cout << "队头元素为：" << e5 << endl;

system("pause");

break;

case 6://功能6：显示循环队列中的元素

Q.ShowQueue();

system("pause");

break;

case 7://功能7：

system("pause");

break;

case 0://功能8：退出系统

cout << "欢迎下次使用！" << endl;

system("pause");

return 0;

break;

}

}

return 0;

}

实例

（用顺序队列实现舞伴配对）

* 总体说明：

首先定义节点存储数据类型 （姓名性别）然后利用上述顺序队列的基本操作实现人员导入到一个队列然后根据性别分别导入到两个队中男女配对返回配对列表及第一个未配对人姓名。

具体实现：

1. 导入人员

* 说明

从键盘输入人员信息利用入队函数实现入队

* 实现函数

void EnPerson(Queue& Q)//导入人员

{

string name;

char sex;

char f;

while (true)//构造循环，可以一直导入

{

cout << "是否导入成员？（y/n）" << endl;

cin >> f;

if (f == 'n')

break;

cout << "请输入姓名+空格+性别(m/f)：";//m是男人，f是女人

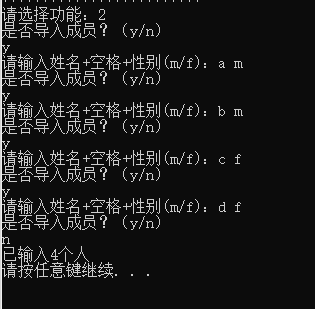
cin >> name;

cin >> sex;

Person P = newPerson(name, sex);

Q.EnQueue(P);

}

cout << "已输入" << Q.QueueLen() << "个人" << endl;}运行结果：

1. 根据性别分队存储

* 说明

利用出队函数依次按序返回队内人员信息并通过if语句判断该人员性别，将男女分开存储存储时用到入队操作为入队函数

* 实现函数

void DePerson(Queue& Q, Queue& Qm, Queue& Qf)//从队列中导出成员，并根据性别保存到对应队列

{

int n = Q.QueueLen();

while (n)

{

n--;

Person P;

P = Q.DeQueue();

if (P.sex == 'm')//男的

Qm.EnQueue(P);//保存到男性队列

else if (P.sex == 'f')//女的

Qf.EnQueue(P);//保存到女性队列

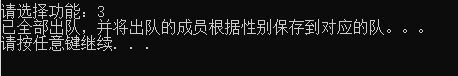
else

cout << "性别有误，删除此人。。" << endl;

}

cout << "已全部出队，并将出队的成员根据性别保存到对应的队。。。" << endl;

}

运行结果：

1. 男女配对返回第一个未配对人信息

* 说明

取男女队列的最小长度为L将两个队列中的L个元素配对这里用到出队函数。将两个队列中较长的队列的头一个元素出队该素未配对。

* 实现函数

void PaPerson(Queue& Qm, Queue& Qf)

{

int L;

Qm.QueueLen() < Qf.QueueLen() ? L = Qm.QueueLen() : L = Qf.QueueLen();//取男女队列中最小的队列长度

while (L)//配对

{

Person Pm, Pf;//Pm是男人中间变量，Pf是女人中间变量

Pm = Qm.DeQueue();

Pf = Qf.DeQueue();

cout << Pm.name << "与" << Pf.name << "配对成功" << endl;

L--;

}

if (Qm.QueueLen() < Qf.QueueLen())//如果男生的人数比女生少

{

Person P;

P = Qf.GetHead();//返回队头女生

cout << P.name << "未配对" << endl;

}

else if (Qm.QueueLen() > Qf.QueueLen())//如果男生人数比女生多

{

Person P;

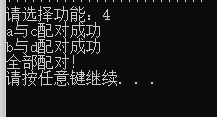
P = Qm.GetHead();//返回队头男生

cout << P.name << "未配对" << endl;

}

else

cout << "全部配对！" << endl;

}运行结果：

主函数：

void showMenu()

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*操作菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*功能1：初始化循环队列 \*" << endl;

cout << "\*功能2：导入全部人 \*" << endl;

cout << "\*功能3：出队并根据性别保存到对应队 \*" << endl;

cout << "\*功能4：男女配对返回配对列表及第一个未配对人姓名\*\*\*\*\*" << endl;

cout << "\*功能0：退出并结束程序 \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*———————\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\* 1901103班2组 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*———————\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

int main()

{

Queue Q,Qm,Qf;//创建一个队列对象

int Choice = 0;

Q.InitQueue();

Qm.InitQueue();

Qf.InitQueue();

cout << "初始化循环队列成功。。。。" << endl;

while (true)

{

system("cls");//清屏

showMenu();//显示菜单

cout << "请选择功能：";

while (true)

{

cin >> Choice;

if (Choice >= 0 && Choice <= 4)

break;

else

cout << "输入错误，请重新输入" << endl;

}

switch (Choice)

{

case 1://功能1：初始化循环队列

Q.InitQueue();

cout << "初始化顺序栈成功。。" << endl;

system("pause");

break;

case 2://功能2：导入全部人

EnPerson(Q);

system("pause");

break;

case 3://功能3：出队并根据性别保存到对应队

DePerson(Q, Qm, Qf);

system("pause");

break;

case 4://功能4：男女配对返回配对列表及第一个未配对人姓名

PaPerson(Qm, Qf);

system("pause");

break;

case 0://功能5：退出系统

cout << "欢迎下次使用！" << endl;

system("pause");

return 0;

break;

}

}

return 0;

}

总结：

顺序队列通过头指针与尾指针可实现队列长度判断以及队列空队列满判断，通过尾指针的移动进行入队操作通过头指针的移动实现队列出队列操作。要注意队列的操作要符合先进先出的原则。

与链表队列比较 顺序表队列的优点是物理地址相邻便于直观理解顺序表可以通过计算地址来定位需要的元素操作方便，链表只能从头到尾来找需要的元素.。缺点是顺序表存储位置是连续的,一般顺序表的长度有限。而链表是通过指针来定位下一个元素的位置,所以没有长度的限制,随时可以增加。