实习题目1：约瑟夫环

1. 需求分析：

1.输入的形式及范围：

使用键盘在屏幕上输入，数据类型为数字。

1. 输出的形式：

以数字形式输出到屏幕上。

1. 程序需要达到的功能：

允许用户输入自己的数据，也可用默认数据使用

用户输入完数据后，程序能够按照约瑟夫环的要求进行计算，并且能够正确输出数据。操作界面要简单易懂，有适当的提示语言，要能预防用户错误输入的情况。

1. 测试数据：

默认的测试数据为：3 1 7 2 4 8 4

结果应该为：6 1 4 7 2 3 5

1. 概要设计：
2. 数据结构：使用链表进行数据处理。在C++中编一个链表类，并在类中添加需要使用的方法。
3. 节点：包含一个指针 一个值 一个序号 一个int量表示是否在队中
4. 大致流程：用户输入完数据后，或者使用默认数据，将数据添加入链表。起初为单向链表，将尾节点的link指针连接至头得到循环链表。使用类中的count函数让各个节点报数，出列的节点in\_queue量置0，并将其值传递给count函数，count函数用next函数知道下一个在队中的节点，从该点开始报数,如此重复知道所有的节点出列。

4.解决方案伪代码：

Node{ 值 value;序号 number;int in\_queue;Node\* link}

Node \*pt = 循环链表的第一个节点 \*head

Int num = 初始值 initial

Node \*read

循环（1）{

read = Count（num，pt）//count 返回一个指针

read 的in\_queue置0 出列

num = read的值

如果（都出列）{break;}

pt = next（read）下一个在队中的节点

}

1. count的实现

Count（num,\*pt）{

Node \*read = pt

For(int i = 0;i<num;i++,read = read->link){

要是read->in\_queue = 0，那num++多数

一次

}

}

1. 详细设计：（完整的代码可看源程序）
2. 节点：

typedef struct NODE{ //-----------------in\_queue说明节点是否在队中

TYPE value;

NODE \*link;

int in\_queue;

int num;

}node;

1. 数据结构：（列出了需要使用的设计）

class linkedlist{ //--------------------------------------------链表类

private:

node \*head;

node \*end;

node \*count\_pt;

int outline;

int len;

int circle;

public:

linkedlist(){

head = NULL;

end = NULL;

len = 0;

circle = 0;

count\_pt = NULL;

outline = 0;

}

node\* count(node\*pt,int n){ //从某个节点开始报数的函数

if(circle){

node \*read = pt;

int i =n;

while(i-1){

if(read->in\_queue==1){

i--;

}

read=next(read);

}

return read;

}

}

void incircle(){ //将一个链表连接成循环链表

end->link = head;

circle = 1;

}

node\* next(node\*pt){ //寻找这个节点后第一个位于队中的节点

node\* read=pt->link;

while(1){

if(read->in\_queue==1){

break;

}

read = read->link;

}

return read;

}

};

1. 解决方法：

void joseph\_problem(int n){

int num = n;

node \*read = head;

while(1){

read = count(read,num); //报数

if(read->in\_queue==1){

cout<<read->num<<' ';

read->in\_queue = 0; //让其出列

++outline;

if(outline == len){ //是否全部出列

break;

}

num = read->value;

read = next(read); //找到下一个在队中节点

}

}

}

四、调试分析：

1.方法分析：

使用的解决方法不删除并保留节点的空间，只是计数时跳过了 in\_queue = 0的节点。比起删除节点的方法，这种方法在有很多多余的计算，在数据少，处理量小的时候缺点并不明显，并且编程简单，不用改动内存，可以保留原来的内存地址。但是一旦计算量大，这种方法就要做许多无用的计数，其缺点也显露出来。综合看来，使用删除节点的方法要比该方法要更好一些。

2.编程体会与心得：

当时编程时原定的方法是删除节点。但是突然又想到了这个方法，感觉好像编起来比较方便，不改动内存和节点也不容易出错。这种方法确实简单一些，本身链表类已经做好了，只要在计数上下一点功夫就弄好了，但是算法本身不先进，想必处理大数据的时候会很慢。所以以后再解决问题时，不能只贪图程序员自己的方便，要为程序的性能考虑。

1. 用户手册

程序有明显的提示语言，按照提示语言操作即可

默认的测试纸为：

初始值20

数组：6 1 4 7 2 3 5

1. 测试结果：

