1. 项目1-1：编程实现以下实验项目

约瑟夫生死游戏

每30个乘客同乘一艘船，因为严重超载，加上风高浪大，危险万分，因此船长告诉乘客，只有将全船一半乘客投入海中，其余人才能幸免于难。无奈，大家只得同意这种办法，并议定30个人围成一圈，由第1个人数起，依次报数，数到第9人，便把他投入大海中，然后再从他的下一个人数起，数到第9人，再将他扔到大海中，如此循环地进行，直到剩下15个乘客为止。问哪些位置是将被扔下大海的位置。

**项目1-1实验报告**

1. **实验内容简介**

n个人围成一圈，由第一个人数起（设第i个人的序号为i），数到第x个人，便把他剔除，然后再从他的下一个人数起，，数到第x个人，再把他剔除，依次继续，直到剩下n/2（或y）个人,输出所有被剔除的人的序号。本项目n=30，x=9，y=15。

1. **程序设计思路**
2. **模拟**

按照题意去做，用一个visit数组记录下已经出队了的人，visit初始全部赋值为false。

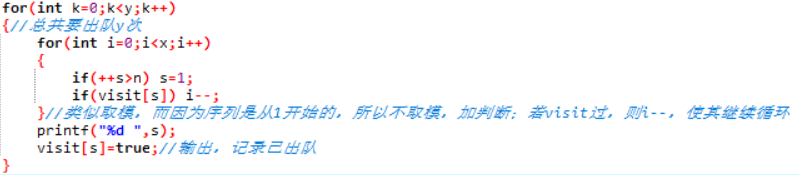
1. **链表**

构建一个链表，使整个游戏在链表中运行，包含删除节点、移动节点等功能，尽量在节点被删除时不必移动很多节点。输入n的值来构建一个具体的链表，对删除了节点后的链表进行重连，使循环能够继续下去，输入间隔值x使间隔确定，输入y来中止循环，最终输出所有被删除节点的序号。

1. **算法分析与设计**
2. **模拟**

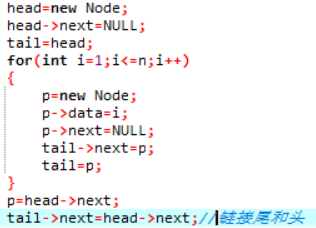
循环y次，每次循环里将一个原本为false的visit元素修改为true，最终输出visit元素为true的下标。

具体代码为：

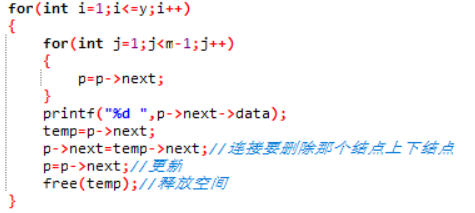


1. **链表**

构建一个循环链表，读入完数据后要将头和尾相连，代码如下：



数到x时删除节点，重新连接前后结点，不然的话它就会变成野指针，代码如下：



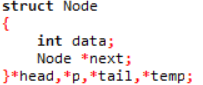
1. **数据结构设计**

**1. 模拟**

一个visit数组：

**2. 链表**

一个结构体Node：



1. **系统实现**

两种方法的流程图的思路是一致的，但实现方法不同，如下所示：

开始

输入n,m,y

创建数组（或链表）

计数

赋值true（删除节点）

（连接链表）

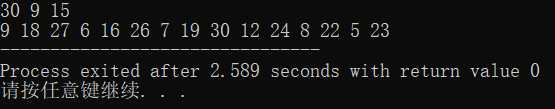
还剩y个节点？

 结束，输出

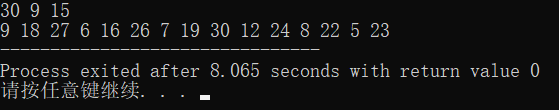
是

否

1. **测试结果**
2. **模拟**



1. **链表**



**附件：源代码**

1. **模拟**

#include<cstdio>

using namespace std;

int main()

{

int n,x,y,s=0;

scanf("%d%d%d",&n,&x,&y);//入读

bool visit[200]={0};//visit赋初始值

for(int k=0;k<y;k++)

{//总共要出队y次

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

{

if(++s>n) s=1;

if(visit[s]) i--;

}

printf("%d ",s);

visit[s]=true;//输出，记录已出队

}

return 0;

}

1. **链表**

#include<iostream>

#include<cstdio>

#include<cstdlib>//用free（）要用这个库

using namespace std;

int m,n,y;

struct Node

{

int data;

Node \*next;

}\*head,\*p,\*tail,\*temp;

int main()

{

scanf("%d%d%d",&n,&m,&y);

head=new Node;

head->next=NULL;

tail=head;

for(int i=1;i<=n;i++)

{

p=new Node;

p->data=i;

p->next=NULL;

tail->next=p;

tail=p;

}

p=head->next;

tail->next=head->next;//链接尾和头

for(int i=1;i<=y;i++)

{

for(int j=1;j<m-1;j++)

{

p=p->next;

}

printf("%d ",p->next->data);

temp=p->next;

p->next=temp->next;//连接要删除那个结点上下结点

p=p->next;//更新

free(temp);//释放空间

}

return 0;

}