加里森的任务

实验报告

目录

[一：需求分析 2](#_Toc85783405)

[二：概要设计 2](#_Toc85783406)

[问题解决思路 2](#_Toc85783407)

[主函数伪代码 3](#_Toc85783408)

[1. 链表类型 3](#_Toc85783409)

[2. Soldiers类定义 3](#_Toc85783410)

[3. 构造函数 4](#_Toc85783411)

[4. 判断是否可以派遣士兵（剩余节点大于1） 4](#_Toc85783412)

[5. 进行相应的数数和删除节点 4](#_Toc85783413)

[6. 返回当前节点数 5](#_Toc85783414)

[7. 摧毁申请空间，便于下一次模拟使用 5](#_Toc85783415)

[三：思路改进 6](#_Toc85783416)

[四：调试分析报告 6](#_Toc85783417)

[五：设计实现的回顾讨论 7](#_Toc85783418)

[六：用户使用说明 8](#_Toc85783419)

[七：测试结果 8](#_Toc85783420)

[八：实验结论 10](#_Toc85783421)

题目：加里森的任务

班级：2020211306 姓名：马天成 学号：2020211376 分工：撰写代码+实验报告

班级：2020211306 姓名：黄洪健 学号：2020211371 分工：撰写代码

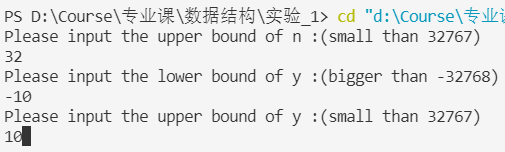
班级：2020211306 姓名：马紫薇 学号：2020211392 分工：撰写实验报告

# 一：需求分析

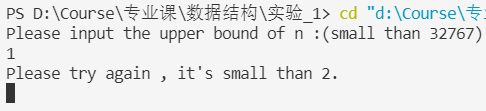
（1）题目的目的为：通过程序的运行，求出当n x y满足何种条件时，加里森能留到最后而不用去执行任务。

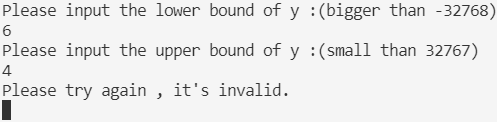
（2)此程序输入输出的形式为: n为大于等于2的正整数，x为大于等于1小于等于n的正整数，y为整数.

输入输出的正确形式为：



输入输出的错误形式为：





(3)为了达到模拟和现实的功能，我们实现了两种程序：

现实：通过用户本人输入的n与y为基础，找出所有大于等于2小于等于n符合条件 的（n,x,y），再进行判断，看其是否满足题目要求。

模拟：通过用户输入的n合法上界，和y的合法上界 下界，找出所有的合法n y x 三元组。

# 二：概要设计

## **问题解决思路**

用单链表模拟人员，并进行编号，然后删除链表，删到最后只剩一个元素，即单链表的 自身next指向自己。通过剩余链表元素和1的相对位置求出正确的x。

## 主函数伪代码

main()

{

输入数据 N , Y\_lower , Y\_upper // N上界，Y下界，Y上界

创建 Soldiers类

while ( 可以派遣士兵 ) {

相关操作

}

算出数值，返回答案

}

为了简要、高效实现主函数，一下有Soldiers类的详细设计：

## 1. 链表类型

typedef struct node

{

int name ;

struct node\* right ;

struct node\* left ;

} list , \*listPtr ;

## 2. Soldiers类定义

class Soldiers

{

public:

Soldiers ( int n ) ;

bool canKill ( void ) ;

void kill ( int y ) ;

int number ( void ) ;

void burden ( void ) ;

private:

listPtr head ;

listPtr cur ;

listPtr prev ;

int peopleSize ;

} ;

## 3. 构造函数

Soldiers::Soldiers ( int n )

{

head = new list ;

head->name = 1 ;

prev = head ;

for ( int i = 2 ; i <= n ; i++ )

{

prev->next = new list ;

prev->next->name = i ;

prev = prev->next ;

}

prev->next = head ;

cur = head ;

peopleSize = n ;

}

## 4. 判断是否可以派遣士兵（剩余节点大于1）

bool Soldiers::canKill ( void )

{

if ( peopleSize == 1 ) {

//std::cout << "Just one people left." << std::endl ;

return false ;

}

return true ;

}

## 5. 进行相应的数数和删除节点

void Soldiers::kill ( int y )

{

while ( --y )

prev = prev->next ;

cur = prev->next ;

prev->next = cur->next ;

delete cur ;

cur = prev->next ;

peopleSize-- ;

}

## 6. 返回当前节点数

int Soldiers::number ( void )

{

if ( peopleSize > 1 )

return -1 ;

return cur->name ;

}

## 7. 摧毁申请空间，便于下一次模拟使用

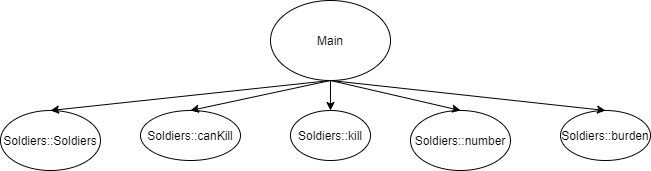
void Soldiers::burden ( void )

{

delete cur ;

}

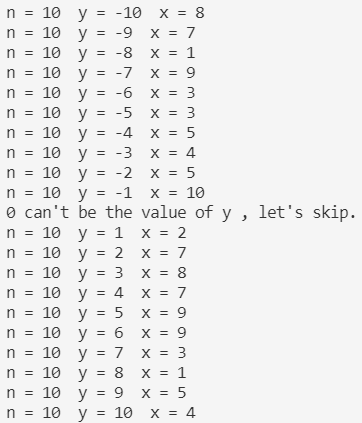
函数调用关系图如下：



# 三：思路改进

对于相应顺序的数数，假如 y < 0 ，那么编号顺序和数数顺序相反，那么，我们就需要双向链表来实现两个方向的数数。此时，我们设计了 - - Program\_2++，来实现双向操作。

就像开始检验输入时的那样，我们可以有很好的输出：



这样的话我们就可以拓展这个问题，让思考变得更加全面。

这也是我们程序员在设计软件时要面对的客户各种奇怪的输入而要提升的能力。

# 四：调试分析报告

调试过程中遇到的问题及解决：

1. 在链表实现和对C++ 引用传递和按地址传递不是很熟练，出现一些错误，在后续设计过程中的相关问题上会有一些疑惑。

2. 在对终止条件的判断时，黄洪健同学选择当加里森0号元素删除后，跳出break；马天

成同学选择在只剩一个人的时候，即：Soldiers.peopleSize == 1时跳出。

3由于选择了双向环链表，所以产生了关于y>0与y<0的讨论，而最初完成版本中并没有考

虑到y < 0这一情况，但在随后的检查与讨论中解决了这一问题。

4在链表使用时由于对于Ptr != NULL这一条件的忽略，导致非法调用结构体成员，在检查过

程中讨论发现了此问题，及时增加。

5在对于删除双向链表的节点的left与right时不够清楚，造成一些删除节点的错误。

# 五：设计实现的回顾讨论

在本次实验中，链表需要多次构造并且释放。在本次实验中实现了双重循环与多组数据计算。

马天成同学的代码：

Soldiers::Soldiers() （生成链表）的时间复杂度为O(n),空间复杂度为O(n)。

Soldiers::kill() （删除元素）的时间复杂度为O(n),空间复杂度为O(1)。

该程序的时间复杂度为O(n^2\*(y上界-y下界))，

特殊的：y上界为n，y下界为-n，该程序的时间复杂度为O(n^3)。

该程序的空间复杂度为O(n)。

黄洪健同学的代码：

由于本代码主要用于测试各种功能，因此只有一个main函数。

main() 的时间复杂度为O(n\*(y上界-y下界)^2),

特殊的：当y上界为n，y下界为-n时，该程序的时间复杂度为O(n^3)。

该程序的空间复杂度为O(n)。

# 六：用户使用说明

打开程序，首先会出现如下界面：



用户输入合法的n后将会继续显示：



用户只需要按要求输入即可，如未按要求输入，请重新输入

当输入不合法时，会出现：

n不合法：



Y\_lower > Y\_upper：

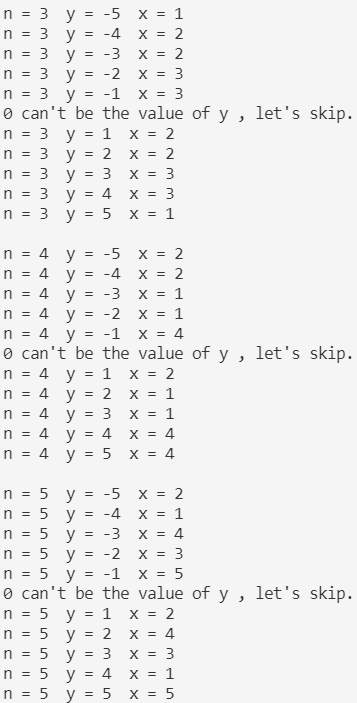
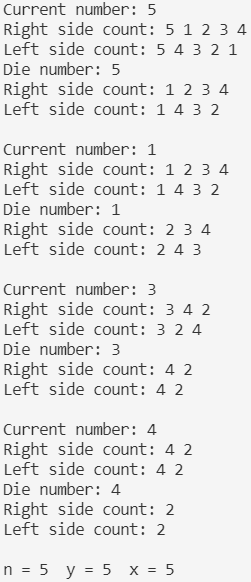


用户可根据提示，重新输入相应数值并回车即可

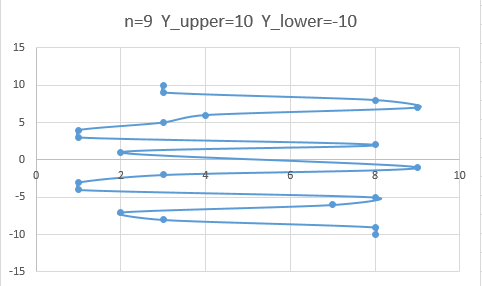
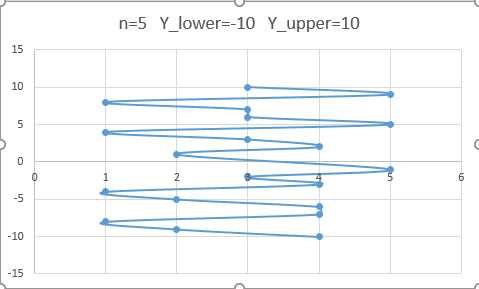
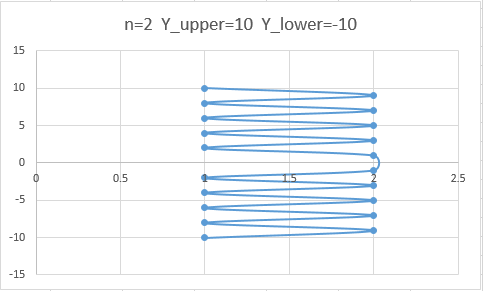
# 七：测试结果

以N等于5，Y上界等于5，Y下界等于-5为例，程序输出如下图：

只输出 n , x , y: 输出过程：

其中，运行结果导入Excel后绘画图表，典型如下：



# 八：实验结论

在经过大量数据测试以及画图后，我们发现以下结论：

前提条件：我们定义每组mod n为一个循环。

如：1 2 3 4 5 6 ( mod 6 ) ; 7 8 9 10 11 12 ( mod 6 ) 为两个循环。

1. 在每个循环中，如果我们将派遣顺序以及相应的x和y画出路径图，我们发 现每个循环内部线状是无序的，但是每个循环都有相似性；每两个相邻的循环大体运动方向一样，落点会有一些差别。

2. （1）所述的差别在所剩人数大小不同，mod n 相同的情况下差别的相对差值相同；但是对于自身规模来说，所剩人数越小，其分布越离散，相似性越小。

EX: 从社会主义核心价值观出发，这并不是一个好题目，因为它具有一些消极因素。我们也希望人与人之间不要相互猜忌，相互利用，团结起来，共同发展。