# 2021～2022年春季学期

**序号：**

# 《程序设计专题》课程

## 实验报告

# 实验一 简单计算问题

# 专业班级： 计22-1

# 学生姓名： 陈志虎

# 学生学号： 22101020124

# 实验题目1：十进制到R进制

## 实验数据储存结构设计

String类.

## 实验设计思想

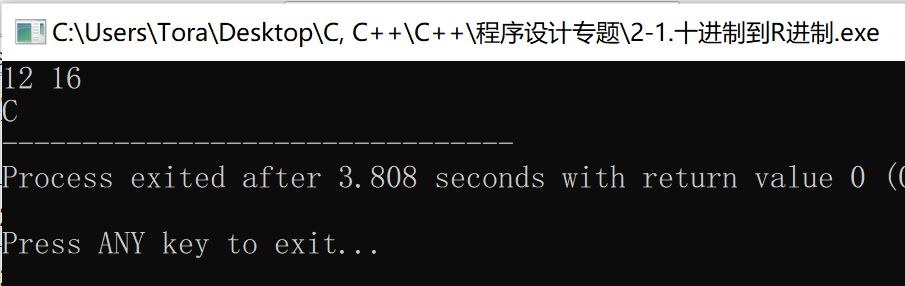
通过不断相除，实现将十进制整数 n 转换为任意进制 r 的功能。

## 实验源程序

#include **<**iostream**>**#include **<**string**>**#include **<**algorithm**>  
using** **namespace** **std;  
void** D\_to\_any**(int** n**,** **int** r**)** **{**    string result **=** **"";**    **while** **(**n **>** 0**)** **{**        **if** **(**n **%** r **>=** 10**)** **{**            result **+=** 'A' **-** 10 **+** n **%** r**;**        **}** **else** **{**            result **+=** '0' **+** n **%** r**;**        **}**        n **/=** r**;**    **};**    reverse**(**result**.**begin**(),** result**.**end**());**    cout **<<** result**;  
}  
int** main**()** **{**    **int** n**,** r**;**    cin **>>** n **>>** r**;**    D\_to\_any**(**n**,** r**);**    **return** 0**;  
}**

## 实验结果及问题分析

### 1．运行结果图



### 2．程序分析

通过不断地对 n 进行除以 r 的操作，得到 n 在 r 进制下的每一位数字，并将其转换为相应的字符，最后将这些字符拼接在一起，形成最终的 r 进制表示。其中如果 n % r 的值大于等于 10，则将其转换为相应的大写字母。最后将字符串翻转后输出即可。程序使用了 string 类型存储结果，以方便反复拼接字符串。

# 实验题目2：R进制到十进制

## 实验数据储存结构设计

无。

## 实验设计思想

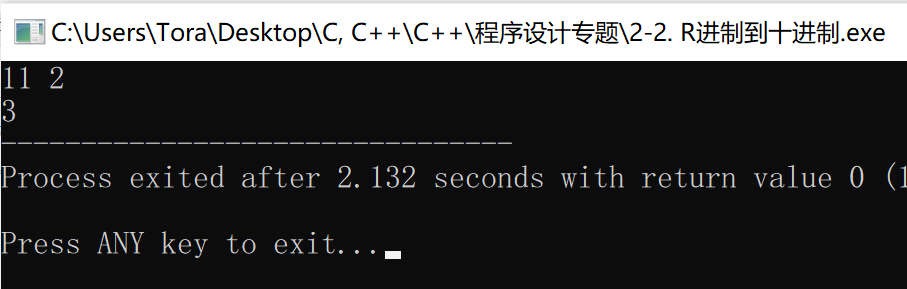
采用逐位转换的思路，实现将 r 进制下的整数 a 转换为十进制整数的功能.

## 实验源程序

#include **<**iostream**>  
using** **namespace** **std;  
void** R\_to\_D**(int** a**,** **int** r**)** **{**    **int** r\_moto **=** r**;**    **int** result**{**0**};**    **while(**a **>** 0**){**        **int** each **=** a **%** 10**;**        result **+=** each **\*** **(**r**/**r\_moto**);**        a **/=** 10**;**        r **\*=** r\_moto**;**    **}**    cout **<<** result**;  
}  
int** main**()  
{**    **int** a**,** r**;**    cin **>>** a **>>** r**;**      
    R\_to\_D**(**a**,** r**);**    **return** 0**;  
}**

## 实验结果及问题分析

### 1．运行结果图



### 2．程序分析

每次取出 a 的最低位 each，根据权值公式将其转换为十进制下的值，并将结果累加到 result 中。在每一轮循环中，将 r 值不断乘以进制数 r\_moto，以便计算下一位数字的权值。最终，程序输出累加得到的十进制整数。

# 实验题目3：电话号码

## 实验数据储存结构设计

C风格字符数组。

## 实验设计思想

对于每一个输入的电话号码，程序将其转换为数字串，并在所有号码中搜索相同数字串的个数，并输出重复次数不为 0 的号码。

## 实验源程序

#include **<**iostream**>**#include **<**string**.**h**>**#include **<**algorithm**>  
using** **namespace** **std;  
int** main**()  
{**    **int** n**;**    cin **>>** n**;**    *//* *输入所有号码*    **char** numbers**[**10001**][**8**]{** 0 **};**    **for** **(int** i **=** 0**;** i **<** n**;** i**++)** **{**        **for(int** j **=** 0**;** j **<** 7**;**j**++)** **{**            **do{**                cin **>>** numbers**[**i**][**j**];**            **}while(**numbers**[**i**][**j**]** **==** '-'**);** *//* *遇‘-’跳过*              
            *//* *trans*            **if** **(**numbers**[**i**][**j**]** **>=** 'A' **&&** numbers**[**i**][**j**]** **<=** 'Y'**)** **{**                numbers**[**i**][**j**]** **=** **"2223334445556667777888999"[**numbers**[**i**][**j**]** **-** 'A'**];**            **}**        **}**    **}**    *//* *标记重复的号码*    **int** memo**[**10001**][**2**]{**0**};** *//* *第一位为重复号码的下标，第二位为重复次数*    **int** DoCounted**[**10001**]{**0**};** *//* *标记此号是否已经比较过*    **int** end **=** 0**;**    **for** **(int** i **=** 0**;** i **<** n**;** i**++)** **{**        **if(**DoCounted**[**i**]** **==** 1**)** **continue;**          
        **bool** IfSame **=** 0**;**        **for** **(int** j **=** i **+** 1**;** j **<** n**;** j**++)** **{**            **if** **(**strcmp**(**numbers**[**i**],** numbers**[**j**])** **==** 0**)** **{**                IfSame **=** 1**;**                DoCounted**[**j**]** **=** 1**;**                memo**[**end**][**0**]** **=** i**;**                memo**[**end**][**1**]** **+=** 1**;**            **}**        **}**        end **+=** IfSame**;**    **}**      
    *//* *Sort*    **for(int** i**=**0**;**i **<** end **-** 1**;**i**++){**        **for** **(int** j **=** 0**;** j **<** end **-** i **-** 1**;** j**++)** **{**            **if(**strcmp**(**numbers**[**memo**[**j**][**0**]],** numbers**[**memo**[**j**+**1**][**0**]])** **>** 0**){**                **int** temp **=** memo**[**j**][**0**];**                memo**[**j**][**0**]** **=** memo**[**j**+**1**][**0**];**                memo**[**j**+**1**][**0**]** **=** temp**;**                  
                **int** temp\_ **=** memo**[**j**][**1**];**                memo**[**j**][**1**]** **=** memo**[**j**+**1**][**1**];**                memo**[**j**+**1**][**1**]** **=** temp\_**;**                  
            **}**        **}**    **}**      
    *//* *输出重复的号码以及重复次数*    **if(**end**){**        **for** **(int** i **=** 0**;** i **<** end**;** i**++)** **{**            **for(int** j **=** 0**;**j **<** 7**;**j**++){**                **if(**j **==** 3**)** cout **<<** '-'**;**                cout **<<** numbers**[**memo**[**i**][**0**]][**j**];**            **}**            cout **<<** ' ' **<<** memo**[**i**][**1**]** **+** 1 **<<** '\n'**;**        **}**    **}** **else**   
        cout **<<** **"No duplicates.**\n**";**    **return** 0**;  
}**

## 实验结果及问题分析

### 运行结果图



### 2．程序分析

在输入数据的处理上，利用了字符串**"2223334445556667777888999"**来简洁地表示每个字母对应的数字，使得输入时就把号码转换为数字形式。在比较电话号码时，使用 memo 数组存储重复次数不为 0 的电话号码的下标以及对应的重复次数，并DoCounted 数组标记已经比较过的电话号码，避免重复比较。最后在输出时，按照电话号码从小到大排序，避免对所有的号码排序，提高了程序的效率。

# 实验题目4­：字符串判等

## 实验数据储存结构设计

String类。

## 实验设计思想

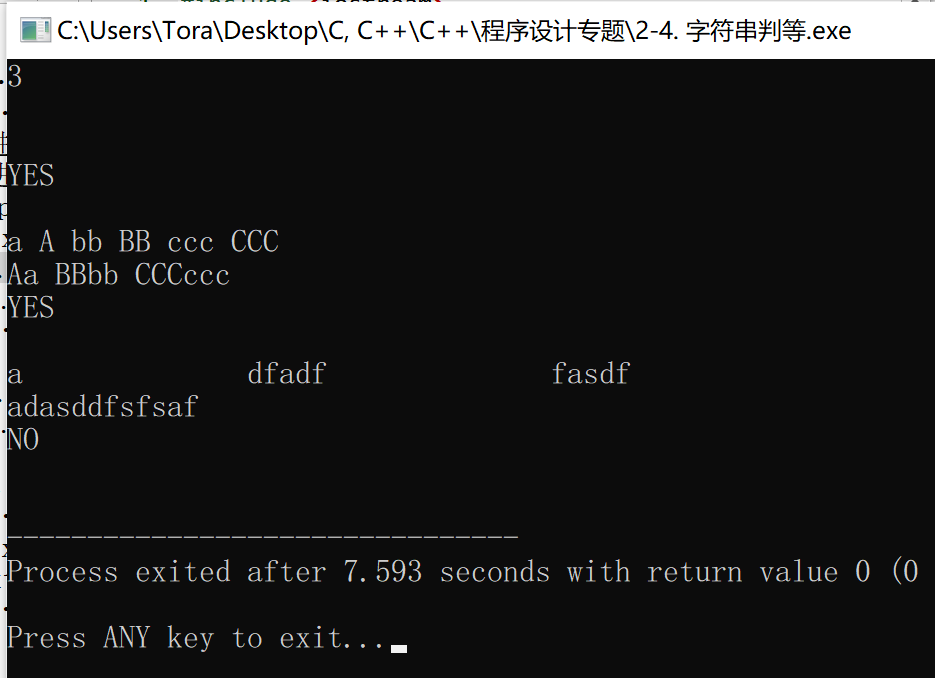
输入多组两个字符串，但是不存储空格，再把大写字符统一转换成小写，然后判断是否相等，输出结果。

## 实验源程序

#include **<**iostream**>  
using** **namespace** **std;  
void** getl**(**string**&** A**)  
{**    **while** **(**1**)** **{**        **char** temp **=** getchar**();**        **if** **(**temp **==** ' '**)** **continue;**        **else** **if** **(**temp **==** '\n'**)** **break;**        **else** **if** **(**temp **>=** 'A' **&&** temp **<=** 'Z'**)** **{**            temp **=** temp **-** 'A' **+** 'a'**;**        **}**        A **+=** temp**;**    **}  
  
  
}  
int** main**()  
{**    **int** n**;**    cin **>>** n**;**    getchar**();**    **for** **(int** i **=** 0**;** i **<** n**;** i**++)** **{**        string A **=** **"",** B **=** **"";**        getl**(**A**);**        getl**(**B**);**        cout **<<** **(**A **==** B **?** **"YES"** **:** **"NO")** **<<** '\n'**;**        getchar**();**    **}**    **return** 0**;  
}**

## 实验结果及问题分析

### 运行结果图



### 程序分析

在 getl() 函数中，使用 getchar() 读取输入的字符，过滤掉空格，并将大写字母转换为小写字母，最终将字符存储到字符串 A 中。

程序主函数中，首先读入需要比较的字符串的数量 n，然后通过调用 getl() 函数分别读入两个字符串，并将它们存储到字符串变量 A 和 B 中。最后使用 == 运算符比较两个字符串是否相同，如果相同则输出 "YES"，否则输出 "NO"。

由于 getchar() 读取字符时并不会读取换行符，所以使用 getchar() 函数将剩余的字符读取并丢弃，以便下一次输入时能够正确读取数据。

# 实验题目5：All in All

## 实验数据储存结构设计

C风格字符数组。

## 实验设计思想

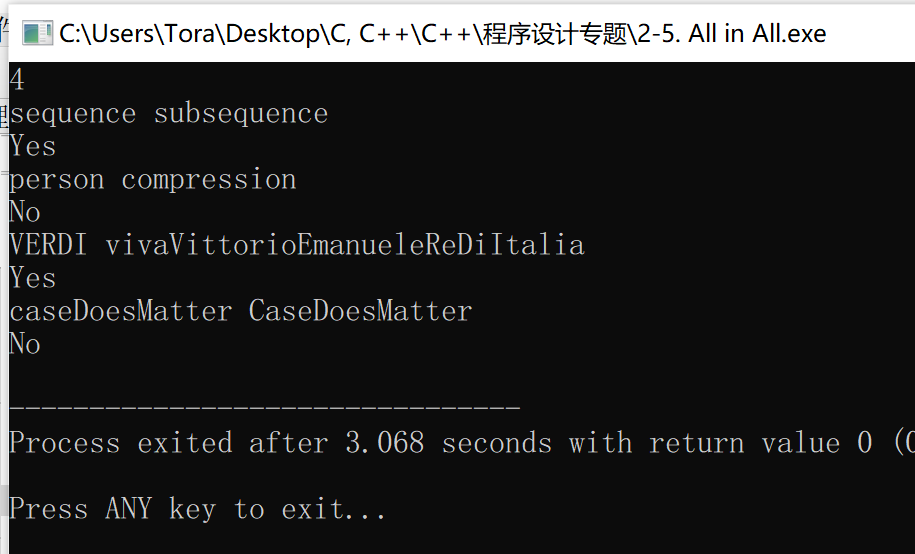
采用指针移动的方式逐个比较两字符串中的各个字符，判断前一字符串是否被后一字符串包含。

## 实验源程序

#include **<**iostream**>  
using** **namespace** **std;  
int** main**()  
{**    **int** n**;**    scanf**("%d",** **&**n**);**    **for(int** i**=**0**;**i**<**n**;**i**++){**        **char** a**[**100001**]{**0**},** b**[**100001**]{**0**};**        scanf**("%s %s",** a**,** b**);**          
        **int** pa **=** 0**,** pb **=** 0**;**        **while(**1**){**            **if** **(**a**[**pa**]** **==** b**[**pb**])** **{**                pa**++;**                pb**++;**            **}** **else** **{**                pb**++;**            **}**              
            **if** **(**b**[**pb**]** **==** 0 **&&** a**[**pa**]** **!=** 0**)** **{**                printf**("No**\n**");**                **break;**            **}** **else** **if** **(**b**[**pb**]** **==** 0 **&&** a**[**pa**]** **==** 0**)** **{**                printf**("Yes**\n**");**                **break;**            **}**        **}**    **}**    **return** 0**;  
}**

## 实验结果及问题分析

### 1．运行结果图



### 2．程序分析

首先输入一个正整数n，表示有n组测试数据。对于每组测试数据，输入两个字符串a和b。

从字符串的第一个字符开始比较，若相同则两个指针同时往后移动一位，若不同则只移动b的指针。如果比较结束后a的指针不在字符串结尾而b的指针已经到达字符串结尾，说明b不是a的子串，输出No。

如果比较结束后a的指针也已经到达字符串结尾，说明b是a的子串，输出Yes。

# 实验题目6：不吉利日期

## 实验数据储存结构设计

无.

## 实验设计思想

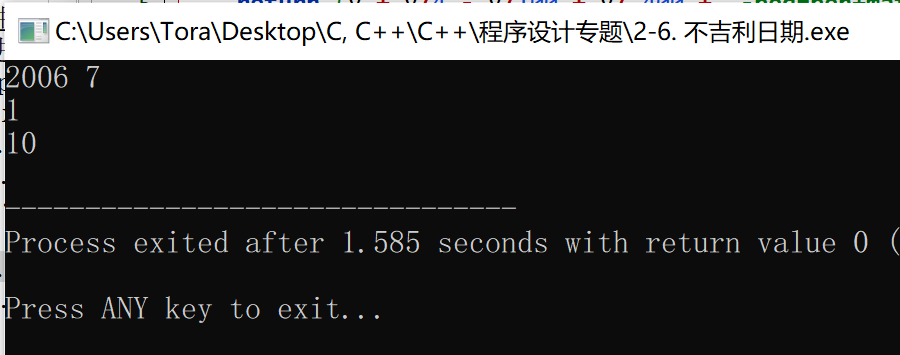
对于每个输入的年份，利用坂本智彦算法计算其每个月13号是星期几，若是星期五，输出这个月份。

## 实验源程序

#include **<**iostream**>  
using** **namespace** **std;  
int** dow**(int** y**,** **int** m**,** **int** d**)** **{**   
    y **-=** m**<**3**;**   
    **return** **(**y **+** y**/**4 **-** y**/**100 **+** y**/** 400 **+** **"-bed=pen+mat."[**m**]** **+** d**)** **%** 7**;  
}  
int** main**()  
{**    **int** year**;**    cin **>>** year**;**      
    **for(int** i**=**1**;**i**<=**12**;**i**++){**        **if** **(**dow**(**year**,** i**,** 13**)** **==** 5**)**            cout **<<** i **<<** '\n'**;**    **}**    **return** 0**;  
}**

## 实验结果及问题分析

### 1．运行结果图



### 2．程序分析

函数dow(y, m, d)返回日期(y, m, d)是星期几，其中y, m, d分别表示年、月、日。该函数中首先对月份进行了处理，即将1、2月视为上一年的13、14月，以便处理涉及到2月份的情况。然后利用蔡勒公式计算星期几，并将结果映射为"bed=pen+mat."字符串中相应位置上的字符，得到星期几的表示。主函数读入一个年份，然后对1-12月中的每个月调用dow函数，判断13日是否是星期五。如果是，就输出月份。

# 实验题目7：循环数

## 实验数据储存结构设计

String类。

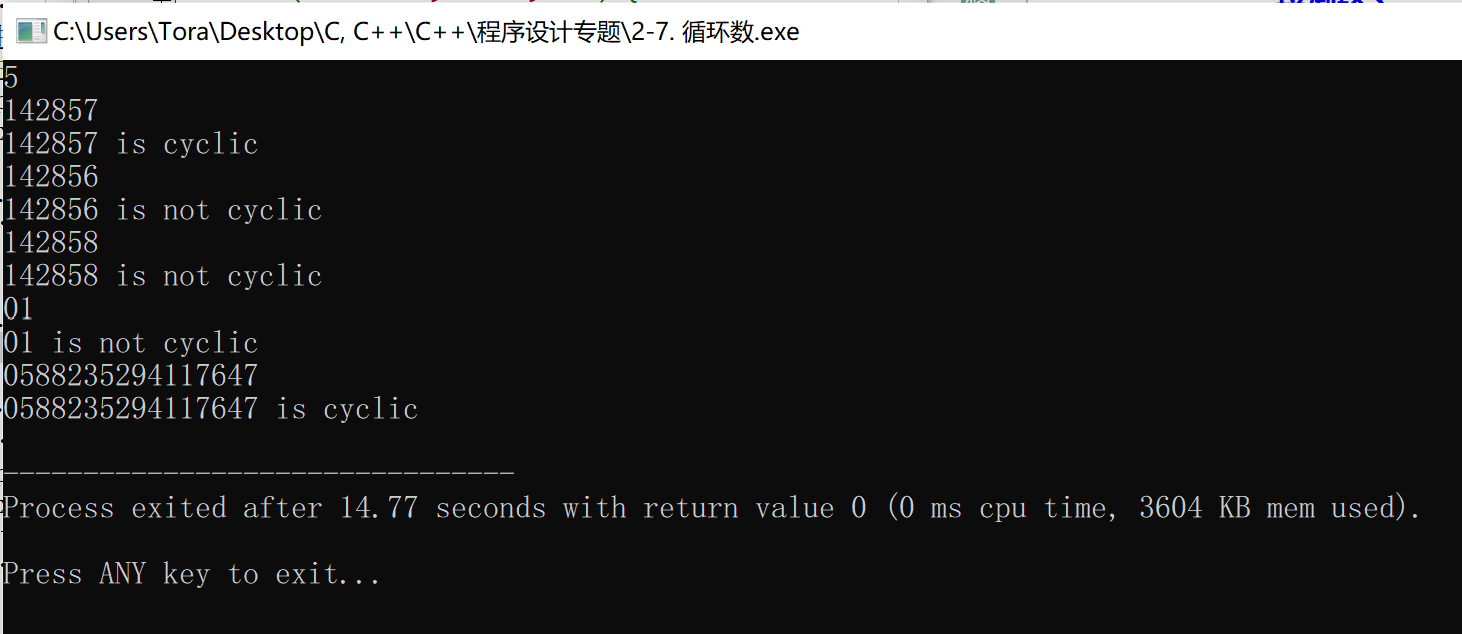
## 实验设计思想

通过比较A数字乘以比其位数低的数形成的数是否包含在A数字写两遍形成的新数中，判断出A是否是循环数.

## 实验源程序

## 实验结果及问题分析

### 1．运行结果图



### 2．程序分析

输入整数个数n，并循环n次，对每个数进行判断。将输入的整数转成字符串s\_num，并计算它的位数size。将字符串s\_num转回整数形式i\_num，用来依次乘以2、3、4……，再依次计算i\_num、2i\_num、3i\_num……的结果，并将结果转回字符串s\_new\_num，判断它是否为循环数。

然后判断s\_new\_num是否可以由s\_num的数字重新排列组成，每个数字只出现一次，如果不能，则不是循环数。

如果2-5步中的每个数字都是循环数，则s\_num是循环数，输出 "s\_num is cyclic"，否则输出 "s\_num is not cyclic"。

# 实验题目8：计算2的N次方

## 实验数据储存结构设计

String数组.

## 实验设计思想

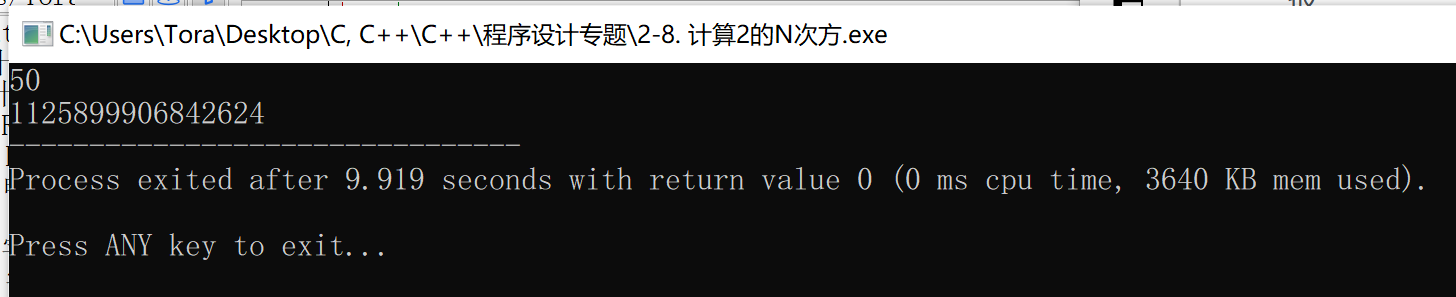
利用Python计算出题目范围内2的所有幂，结果存储在string数组中，查表输出结果。

## 实验源程序

#include **<**iostream**>  
using** **namespace** **std;  
int** main**()  
{**    string rusults**[**101**]{**        **"1",**   
        **"2",**   
        **"4",**   
        **"8",**   
        **"16",**   
        **"32",**   
        **"64",**   
        **"128",**   
        **"256",**   
        **"512",**   
        **"1024",**   
        **"2048",**   
        **"4096",**   
        **"8192",**   
        **"16384",**   
        **"32768",**   
        **"65536",**   
        **"131072",**   
        **"262144",**   
        **"524288",**   
        **"1048576",**   
        **"2097152",**   
        **"4194304",**   
        **"8388608",**   
        **"16777216",**   
        **"33554432",**   
        **"67108864",**   
        **"134217728",**   
        **"268435456",**   
        **"536870912",**   
        **"1073741824",**   
        **"2147483648",**   
        **"4294967296",**   
        **"8589934592",**   
        **"17179869184",**   
        **"34359738368",**   
        **"68719476736",**   
        **"137438953472",**   
        **"274877906944",**   
        **"549755813888",**   
        **"1099511627776",**   
        **"2199023255552",**   
        **"4398046511104",**   
        **"8796093022208",**   
        **"17592186044416",**   
        **"35184372088832",**   
        **"70368744177664",**   
        **"140737488355328",**   
        **"281474976710656",**   
        **"562949953421312",**   
        **"1125899906842624",**   
        **"2251799813685248",**   
        **"4503599627370496",**   
        **"9007199254740992",**   
        **"18014398509481984",**   
        **"36028797018963968",**   
        **"72057594037927936",**   
        **"144115188075855872",**   
        **"288230376151711744",**   
        **"576460752303423488",**   
        **"1152921504606846976",**   
        **"2305843009213693952",**   
        **"4611686018427387904",**   
        **"9223372036854775808",**   
        **"18446744073709551616",**   
        **"36893488147419103232",**   
        **"73786976294838206464",**   
        **"147573952589676412928",**   
        **"295147905179352825856",**   
        **"590295810358705651712",**   
        **"1180591620717411303424",**   
        **"2361183241434822606848",**   
        **"4722366482869645213696",**   
        **"9444732965739290427392",**   
        **"18889465931478580854784",**   
        **"37778931862957161709568",**   
        **"75557863725914323419136",**   
        **"151115727451828646838272",**   
        **"302231454903657293676544",**   
        **"604462909807314587353088",**   
        **"1208925819614629174706176",**   
        **"2417851639229258349412352",**   
        **"4835703278458516698824704",**   
        **"9671406556917033397649408",**   
        **"19342813113834066795298816",**   
        **"38685626227668133590597632",**   
        **"77371252455336267181195264",**   
        **"154742504910672534362390528",**   
        **"309485009821345068724781056",**   
        **"618970019642690137449562112",**   
        **"1237940039285380274899124224",**   
        **"2475880078570760549798248448",**   
        **"4951760157141521099596496896",**   
        **"9903520314283042199192993792",**   
        **"19807040628566084398385987584",**   
        **"39614081257132168796771975168",**   
        **"79228162514264337593543950336",**   
        **"158456325028528675187087900672",**   
        **"316912650057057350374175801344",**   
        **"633825300114114700748351602688",**        **"1267650600228229401496703205376"**    **};**      
    **int** n**;**    cin **>>** n**;**    cout **<<** rusults**[**n**];**    **return** 0**;  
}**

## 实验结果及问题分析

### 1．运行结果图



### 程序分析

采用了空间换时间的策略，极短地缩短了计算用时，时间复杂度为O(1).

# 实验题目9：拼装模型

## 实验数据储存结构设计

优先队列（priority\_queue）。

## 实验设计思想

使用贪心算法来解决。思想是每次选择局部最优解，最终得到全局最优解。

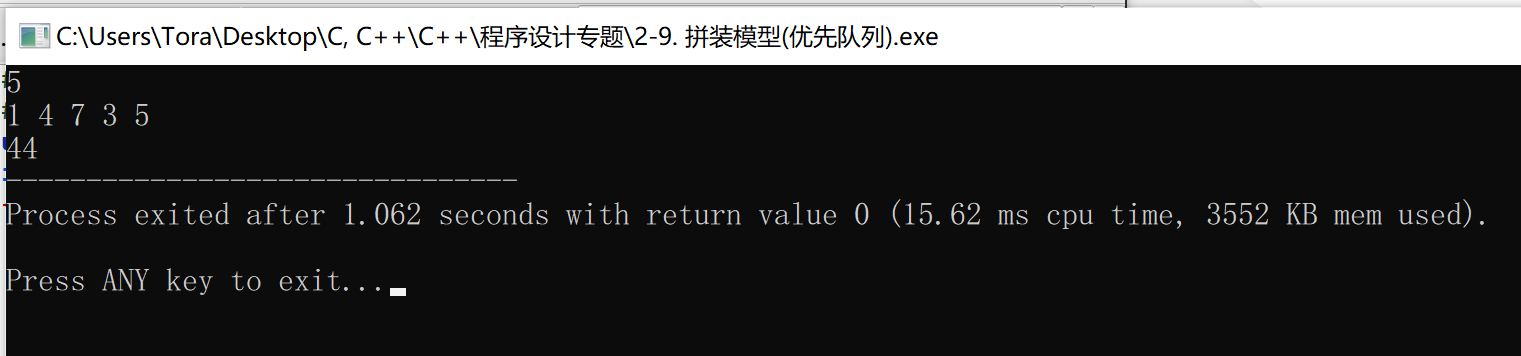
使用一个队列来记录当前所有零件的复杂程度，每次从队列中选取两个最小的零件进行拼接，并将新的零件的复杂程度加入队列中。重复这个过程，直到队列中只剩下一个零件。输出最终的拼装时间。

## 实验源程序

#include **<**iostream**>**#include **<**queue**>  
using** **namespace** **std;  
int** main**()  
{**    priority\_queue **<** **int,** vector**<int>,** greater**<int>** **>** model**;**      
    **int** n**;**    cin **>>** n**;**      
    **for(int** i**=**0**;**i**<**n**;**i**++){**        **int** temp**;**        cin **>>** temp**;**        model**.**push**(**temp**);**    **}**      
    **long** **long** time**{**0**};**    **while(**model**.**size**()** **>** 1**){**        **int** a **=** model**.**top**();**        model**.**pop**();**        **int** b **=** model**.**top**();**        model**.**pop**();**          
        model**.**push**(**a**+**b**);**        time **+=** a**+**b**;**    **}**      
    cout **<<** time**;**      
    **return** 0**;  
}**

## 实验结果及问题分析

### 1．运行结果图



### 2．程序分析

使用STL中的priority\_queue（优先队列）来记录当前所有零件的复杂程度，队列的队首元素为所有元素中最小的元素。

代码首先输入零件数目n，然后循环n次，每次输入一个零件的复杂程度，将它加入优先队列中。

接下来的循环中，每次从优先队列中取出队首的两个元素，它们的复杂程度就是当前所有零件中复杂程度最小的两个。将这两个元素拼接在一起得到一个新的零件，将新的零件的复杂程度加入优先队列中。同时，将这两个零件的复杂程度之和加入time中，最终得到的time就是最短的拼装时间。

# 实验题目10：摘花生

## 实验数据储存结构设计

结构体，优先队列。

## 实验设计思想

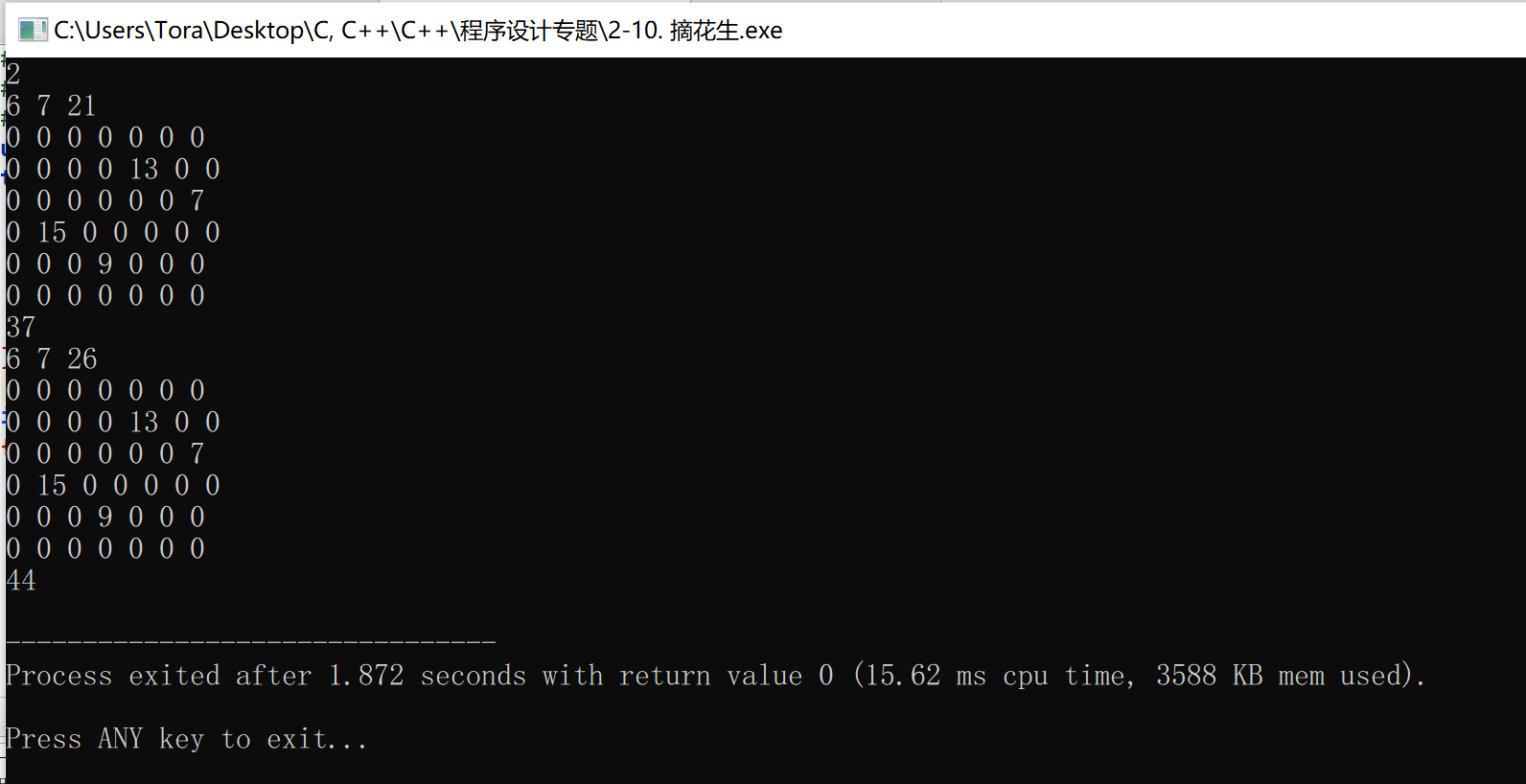
将输入的所有花生存储到一个优先队列中，按照花生数量排序，每次取出数量最多的花生作为当前位置。从当前位置到下一个位置需要一定时间，因此将时间加上这段路程所需的时间，同时将花生数量加入收获总量中。如果加上这段时间后时间超过规定的时间限制，则将此处采摘的花生数量从收获总量中减去，结束循环，输出最终的收获总量。

## 实验源程序

#include **<**iostream**>**#include **<**queue**>**#include **<**cmath**>  
using** **namespace** **std;  
typedef** **struct** node **{**    **int** num**;**    **int** where**[**2**];**    **friend** **bool** **operator** **<** **(const** node a**,** **const** node b**)** **{**        **return** a**.**num **<** b**.**num**;**    **}  
}**node**;  
  
int** main**()  
{**    **int** n1**;**    cin **>>** n1**;**      
    **for(int** i**=**0**;**i**<**n1**;**i**++){**        priority\_queue **<**node**>** peanut**;**        **int** m**,**n**,**k**;**        cin **>>** m **>>** n **>>** k**;**          
        *//* *输入花生的个数，地点,将其加入队列*        **for(int** r**=**1**;**r**<=**m**;**r**++){**            **for(int** c**=**1**;**c**<=**n**;**c**++){**                  
                **int** temp**;**                cin **>>** temp**;**                **if** **(**temp **==** 0**)** **continue;**                **else** **{**                    node tmp**;**                    tmp**.**num **=** temp**;**                    tmp**.**where**[**0**]** **=** r**;**                    tmp**.**where**[**1**]** **=** c**;**                    peanut**.**push**(**tmp**);**                **}**            **}**        **}**          
        *//* *第一颗“花生”设为路边*        node road **{**999**,** **{**0**,** peanut**.**top**().**where**[**1**]}};**        peanut**.**push**(**road**);**          
        **int** ans **=** 0**,** time **=** 0**;**        **while(**1**){**            node now **=** peanut**.**top**();**            peanut**.**pop**();**            **if** **(**peanut**.**size**()** **==** 0**)** **break;** *//* *如果花生全采完了*            node nxt **=** peanut**.**top**();**              
            time **+=** abs**(**now**.**where**[**0**]** **-** nxt**.**where**[**0**])** **+** abs**(**now**.**where**[**1**]** **-** nxt**.**where**[**1**])** **+** 1**;**            ans **+=** nxt**.**num**;**              
            **if(**time **+** nxt**.**where**[**0**]** **>** k**)** **{**                ans **-=** nxt**.**num**;**                **break;**            **}**        **}**        cout **<<** ans **<<** '\n'**;**    **}**    **return** 0**;  
}**

## 实验结果及问题分析

### 1．运行结果图



### 2．程序分析

定义了一个名为node的结构体，用于存储花生的数量和位置信息。其中，friend bool operator < 用于实现结构体node的小于号运算符重载，从而使得该结构体可以被优先队列priority\_queue所使用；

程序主函数中，首先读入了一个整数n1，代表有n1组数据需要处理；

对于每一组数据，首先读入三个整数m、n和k，分别代表花生的数量、花生田的长和宽、采摘时间限制；

接下来，程序通过两重循环读入花生的信息，将其存储到优先队列peanut中，其中不包括花生数量为0的位置；

将第一颗“花生”设为路边（999代表花生数量不可能有这么多），将其加入优先队列peanut中；

初始化ans为0，time为0，表示当前未采摘任何花生，时间为0；

进入循环，每次取出队列中的两个元素，计算它们之间的距离，并将时间和收获量相应地累加到ans和time中；

如果时间超出了限制，则将当前位置的花生数量从ans中减去，并跳出循环；

循环结束后，输出ans。

………………………………………………………………………………………

# 本次实验总结体会

学到了优先队列的使用，在很多需要排序的场景下，用优先队列来写程序往往会方便很多。

学到了计算日期对应星期的坂本智彦算法，能够方便、迅速地解决某些日期问题。

借鉴坂本智彦把许多if语句改写为字符串+下标的做法，在“电话号码”一题中用一段字符串替代了一大块if…else…语句串，使得程序更为简洁、易读。