1. 课后作业请在规定时间之前提交到ftp相应文件夹，地址为ftp://public.sjtu.edu.cn，账号：x8h8n8，密码：123456，每次作业都会建文件夹，代表第几次，**千万**不要放在其它的作业文件夹。本次作业提交**截止时间**为4月25日18：00，过期不侯。
2. 本次作业出现多文件，提交方式如下：.h，.cpp放在题号命名的文件夹下，各个题目分不同的文件夹，最后整个压缩为一个以学号姓名命名的压缩文件
3. 做作业有问题可以与其他同学讨论，但请勿复制他人程序，也可以发送邮件至你所对应的助教邮箱或我的邮箱进行讨论。各位同学所对应助教的联系方式见ftp

1 输入一个字符串(长度不超过30)以及Z字形的行数，将它按照z字形重排后输出。例如：

当输入为:

3 acm.sjtu.edu.cn

将输入按照3行的Z字形排列：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a |  | s |  | . |  | . |
| c | . | j | u | e | u | c |
| m |  | t |  | d |  | n |

将它按行输出：

as..c.jueucmtdn

当输入为:

4 acm.sjtu.edu.cn

将输入按照4行的Z字形排列：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a |  |  | t |  |  | . |
| c |  | j | u |  | u | c |
| m | s |  | . | d |  | n |
| . |  |  | e |  |  |  |

将它按行输出：

at.cjuucms.dn.e

2 输入若干个字符串(每个字符串的长度不超过30个字符，字符串总数不超过30)，和一个英文字符ch。 要求删除每个字符串中的字符ch(区分大小写)，得到新的字符串，然后将新的字符串按照字典逆序排序后输出

3利用结构体表示一个复数，编写实现复数的加法，乘法，输入和输出的函数

4 利用链表实现多项式的输入与加法

5设计一个学生成绩管理程序，学生信息包括学号、姓名、语文成绩、数学成绩和英语成绩。程序运行时出现如下菜单：

欢迎使用成绩管理系统

1. 添加学生信息
2. 修改学生信息
3. 显示全部学生信息
4. 按学号查询学生信息
5. 按姓名查询学生信息
6. 按学号升序排序
7. 按总分降序排序
8. 退出

本题需要提供word文件，内容为功能模块划分、模块间调用关系、函数之间调用关系、关键变量的说明

6 设计一个长整形数处理库，可以处理任意的不超过100位的正整数，其功能包括任意正整数的输入，加法，输出，在此基础上，编写一个main.cpp演示这些功能

7设计一个程序用于向后推算指定日期经过n天后的具体日期。输入为长度为8的字符串str和一个正整数n，str前四位表示年份，后四位表示月和日。当推算出的年份大于4位数时，输出"out of limitation!"，否则输出8位的具体日期。

8利用类表示一个复数，编程实现复数的加法和乘法，写一个main函数演示各项功能

9 定义时间类，能够提供和设置由时、分、秒组成的时间

阅读理解

上机运行以下程序，思考一下为什么是这样的结果（不用提交）

#include <iostream>

#include <typeinfo>

using namespace std;

int main()

{

char\* proverb="An apple a day keeps the doctor away";

cout << proverb << '\t' << sizeof (proverb) << '\t'<< typeid(proverb).name() << '\n' ;

cout << proverb+1 << '\t' << sizeof (proverb+1) << '\t'<< typeid(proverb+1).name() << '\n' ;

cout << proverb[0] << '\t' << sizeof (proverb[0]) << '\t'<< typeid(proverb[0]).name() << '\n' ;

cout << \*proverb << '\t' << sizeof( \*proverb) << '\t'<< typeid(\*proverb).name() << '\n' ;

cout << proverb[1] << '\t' << sizeof( proverb[1]) << '\t'<< typeid(proverb[1]).name() << '\n' ;

cout << proverb[0]+1 << '\t' << sizeof( proverb[0]+1) << '\t'<< typeid(proverb[0]+1).name() << '\n' ;

return 1;

}

int main()

{

char\* proverb[]={"An apple a day keeps doctor away","Two apples a day are better !"};

cout << proverb << '\t' << sizeof( proverb) << '\t'<< typeid(proverb).name() << '\n' ;

cout << proverb[0] << '\t' << sizeof (proverb[0]) << '\t'<< typeid(proverb[0]).name() << '\n' ;

cout << \*proverb << '\t' << sizeof( \*proverb) << '\t'<< typeid(\*proverb).name() << '\n' ;

cout << \*proverb[1] << '\t' << sizeof( \*proverb[1]) << '\t'<< typeid(\*proverb[1]).name() << '\n' ;

cout << (\*proverb)[0] << '\t' << sizeof ((\*proverb)[0]) << '\t'<< typeid((\*proverb)[0]).name() << '\n' ;

cout << (\*proverb)[1] << '\t' << sizeof ((\*proverb)[1]) << '\t'<< typeid((\*proverb)[1]).name() << '\n' ;

cout << \*\*proverb << '\t' << sizeof( \*\*proverb) << '\t'<< typeid(\*\*proverb).name() << '\n' ;

cout << proverb[1] << '\t' << sizeof (proverb[1]) << '\t' << typeid(proverb[1]).name() << '\n' ;

cout << \*proverb[1] << '\t' << sizeof (\*proverb[1]) << '\t' << typeid(\*proverb[1]).name() << '\n' ;

cout << proverb << '\t' << sizeof (proverb) << '\t'<< typeid(proverb).name() << endl;

cout << proverb+1 << '\t' << sizeof (proverb+1) << '\t'<< typeid(proverb+1).name() <<endl;

cout << \*(proverb+1) << '\t' << sizeof (\*(proverb+1)) << '\t'<< typeid(\*(proverb+1)).name() <<endl;

cout << proverb[0]+1 << '\t' << sizeof( proverb[0]+1) << '\t'<< typeid(proverb[0]+1).name() << '\n' ;

cout << \*(proverb[0]+1) << '\t' << sizeof (\*(proverb[0]+1) ) << '\t' << typeid(\*(proverb[0]+1)).name() << '\n' ;

cout << proverb[0][1] << '\t' << sizeof( proverb[0][1]) << '\t' << typeid(proverb[0][1]).name() << '\n' ;

cout << (\*proverb)[1] << '\t' << sizeof ((\*proverb)[1] ) << '\t' << typeid((\*proverb)[1] ).name() << '\n' ;

return 1;

}