1. 实现n个元素的全排列

void permutation(T list[], int k, int m)

{

if (k == m)

{

copy(list, list + m + 1, ostream\_iterator<T>(cout, "")); //将当前list排序

cout << endl;

}

else{

for (int i = k; i <= m; i++)

{

swap(list[i], list[k]); //将下标为i的元素交换到k位置，类似从list[k:m]中剔除操作

permutation(list, k + 1, m);

swap(list[i], list[k]);

}

}

}

int main(int argc, char\* argv[])

{

char arr[3] = { 'a', 'b', 'c' };

cout << "排序结果如下：" << endl;

permutation(arr, 0, 2);

return 0;

}

1. longlong型数据操作

表示unsigned long long：

printf("num = %llu\n", num);

表示long long int：

printf("num = %lld\n", num);

3.接收

7

5 -3 6 5 -5 -1 6

-6 1 4 -2 0 -2 0

这种格式的方法

**scanf**("%d", &n);

**for** (**int** i=0; i<n; i++) **scanf**("%d", &a[i]);

**for** (**int** i=0; i<n; i++) **scanf**("%d", &b[i]);

用getchar()可以获取一个字符就可以读走换行符。

如果读取

3,5

则用scanf("%d,%d",&N,&M);以什么间隔scanf就写什么

此时缓冲区还有一个\n如果不读入则会影响后面的输入

1. C语言字符串分割

#include<string.h>

分解字符串为一组字符串。s为要分解的字符串，delim为分隔符字符串。

strtok()用来将字符串分割成一个个片段。参数s指向欲分割的字符串，参数delim则为分割字符串中包含的所有字符。当strtok()在参数s的字符串中发现参数delim中包涵的分割字符时,则会将该字符改为\0 字符。在**第一次调**用时，strtok()必需给予参数s字符串，往后的调用则将参数s设置**成NULL**，则**函数保存的指针SAVE\_PTR在下一次调用中将作为起始位置**。每次调用成功则返回指向被分割出片段的指针s开头开始的一个个被分割的串。当没有被分割的串时则返回NULL。所有delim中包含的字符都会被滤掉，并将被滤掉的地方设为一处分割的节点。

例子：

#include <stdio.h>

#include <conio.h>

int main(void)

{

char input[16]="abc,d,yuwen";

char \*p;

p=strtok(input,",");//此时input里面只有abc三个字符了

while(p!=NULL)

{

printf("%s\n",p);

p=strtok(NULL,",");

}

return 0;

}

输出：

abc

d

yuwen

4**.while (x=0)与while (x==0)两条语句的循环条件有何区别**

x=0是赋值，可以拆成两句，int x,x=0,while（x），x为零，条件为假，while结束。  
x==0是判断语句，当x为零时，条件为真，while继续。

**5.n++和++n的区别**

无论前置还是后值，++ 都会使其操作数的值增 1。不同的是，++ 前置时，自增表达式（如 ++n）的值等于其操作数自增后的值；++ 后置时，自增表达式（如 n++）的值等于其操作数自增前的值。例如：知int n = 5, post = 1, pre = 1;pre = ++n + pre;    // 运算结束后 pre 为 7，n 为 6n = 5;post = n++ + post;  // 运算结束后 post 为 6，n 为 6上例中，++n 的值等于 n 自增后的值，道即 6，所以 pre 最后得到的值为 7。n++ 的值等于 n 自增前的值，即 5，所以 post 最后得到的值为 6。

1. **scanf会返回接收正确值的个数**
2. **时间复杂度就是语句执行次数的最高次**

空间复杂度指的来是运行时**临时占**用的存储空间大小.， 主要是看运行程序时临时变量的数目和数据规模之间的关系。举例：无论数据规模多大，我都定义100个变量，这就叫做源数据规模和临时变量数目无关。就是说空间复杂度zhidao是O(1).int i;  
for(i=0;i<n;i++)  
i++;  
时间复杂度是O(n),空间复杂度是O(1);  
int i;  
int j;  
int k;  
for(i=0; i<n; i++)  
for(j=0; j<i; j++)  
for(k=0; k<j; k++)  
;  
时间复杂度是O(n^3),空间复杂度还是O(1)；

1. C语言fork()函数

一个进程调用fork（）函数后，系统先给新的进程分配资源，例如存储数据和代码的空间。然后把原来的进程的所有值都复制到新的新进程中，只有少数值与原来的进程的值不同。相当于克隆了一个自己。

fork调用的一个奇妙之处就是它仅仅被调用一次，却能够返回两次，它可能有三种不同的返回 值：  
    1）在父进程中，fork返回新创建子进程的**进程ID**；  
    2）在子进程中，fork返回0；  
    3）如果出现错误，fork返回一个负值；

通过判断**返回值**来确定当前是父还是子进程。

1. C语言指针详解：**变量先与[]结合,因为其优先级比\*高**

**判断变量含义方法：从变量开始，按照先后顺序进行解析，指针指向的内容是（数组，int等等）数组的内容是。。。**

**说出含义的时候先判断类型，可以按照从低到高的顺序来解释：**

比如int (\*p)[3] p是一个指针，然后开始从低到高读，指向int型大小为三的数组指针。

int p; //这是一个普通的整型变量    
 int \*p; //首先从P 处开始,先与\*结合,所以说明P 是一个指针,然后再与int 结合,说明指针 所指向的内容的类型为int 型.所以P是一个返回整型数据的指针    
 int p[3]; //首先从P 处开始,先与[]结合,说明P 是一个数组,然后与int 结合,说明数组里的 元素是整型的,所以P 是一个由整型数据组成的数组    
 int \*p[3]; //首先从P 处开始,先与[]结合,因为其优先级比\*高,所以P 是一个数组,然后再与\* 结合,说明数组里的元素是指针类型,然后再与int 结合,说明指针所指向的内容的类型是整型 的,所以P 是一个由返回整型数据的指针所组成的数组    
 int (\*p)[3]; //首先从P 处开始,先与\*结合,说明P 是一个指针然后再与[]结合(与"()"这步可 以忽略,只是为了改变优先级),说明指针所指向的内容是一个数组,然后再与int 结合,说明数 组里的元素是整型的.所以P 是一个指向由整型数据组成的数组的指针    
 int \*\*p; //首先从P 开始,先与\*结合,说是P 是一个指针,然后再与\*结合,说明指针所指向的元 素是指针,然后再与int 结合,说明该指针所指向的元素是整型数据.由于二级指针以及更高级 的指针极少用在复杂的类型中,所以后面更复杂的类型我们就不考虑多级指针了,最多只考虑一 级指针.    
 int p(int); //从P 处起,先与()结合,说明P 是一个函数,然后进入()里分析,说明该函数有一 个整型变量的参数,然后再与外面的int 结合,说明函数的返回值是一个整型数据    
 Int (\*p)(int); //从P 处开始,先与指针结合,说明P 是一个指针,然后与()结合,说明指针指向 的是一个函数,然后再与()里的int 结合,说明函数有一个int 型的参数,再与最外层的int 结 合,说明函数的返回类型是整型,所以P 是一个指向有一个整型参数且返回类型为整型的函数的 指针    
 **int \*(\*p(int))[3]; //可以先跳过,不看这个类型,过于复杂从P 开始,先与()结合,说明P 是 一个函 数,然后进入()里面,与int 结合,说明函数有一个整型变量参数,然后再与外面的\*结 合,说明函数返回的是一个指针,,然后到最外面一层,先与[]结合,说明返回的指针指向的 是 一个数组,然后再与\*结合,说明数组里的元素是指针,然后再与int 结合,说明指针指向的 内 容是整型数据.所以P 是一个参数为一个整数据且返回一个指向由整型指针变量组成的数 组 的指针变量的函数.**

**指针的访问**

一个指针ptrold 加(减)一个整数n 后，结果是一个新的指针ptrnew，ptrnew 的类型和ptrold 的类型相同，ptrnew 所指向的类型和ptrold所指向的类型也相同。ptrnew 的值将比ptrold 的值增加(减少)了n 乘sizeof(ptrold 所指向的类型)个字节。就是说，ptrnew 所指向的内存区将比ptrold 所指向的内存区向高(低)地址方向移动了n 乘sizeof(**ptrold 所指向的类型比如int \*p p指向的就是int ， int （\*p）[5] 这里p指向的就是int[5]**)个字节。指针和指针进行加减：两个指针不能进行加法运算，这是非法操作，因为进行加法后，得到的结果指向一个不知所向的地方，而且毫无意义。两个指针可以进行减法操作，但必须类型相同，一般用在数组方面，不多说了。