# 谷系教程---谷云鹏

这里告诉你最核心的代码！

1：最大公约数

方法一：辗转相除法

核心代码：

int ans(int a, int b) {

int c = b;

while(a % b != 0) {

c = a % b;

a = b;

b = c;

}

return c;

}

方法二：辗转相减法

核心代码：

int ans(int a,int b) {

while(a != b) {

if(a>b) {

a = a - b;

} else {

b = b - a;

}

}

return a;

}

方法三：穷举法

核心代码：

int ans(int a, int b)

{

int c = 0;

for(c = a ; ; c-- )

{

if(a % c == 0 && b % c==0)

break;

}

return c;

}

2: 排序

第一种形式：插入排序

核心代码：

void ans(int \*a, int n) {

int i, j, key;

for(j = 1;j < n;++j) {

key = a[j];

i = j - 1;

while(i >= 0 && key < a[i]) {

a[i+1] = a[i];

i--;

}

a[i+1] = key;

}

}

\*当运行大数量排序时不建议使用

第二种形式：希尔排序（Shell sort）

核心代码：

void ans(int \*a, int n) {

int i, j, r, key;

for(r = n / 2;r >= 1;r /= 2) {

for(i = r;i < n;++i) {

key = a[i];

j = i - r;

while(j >= 0 && key < a[j]) {

a[j+r] = a[j];

j -= r;

}

a[j+r] = key;

}

}

}

\*直接插入排序更高效的改进版

\*也称为“缩小增量排序”

第三种形式：冒泡排序

核心代码：

void ans(int \*a, int n) {

int i, j, t, flag;

for(i = 0;i < n - 1;++i) {

flag = 0;

for(j = 0;j < n - i - 1;++j) {

if(a[j] > a[j+1]) {

t = a[j];

a[j] =a[j+1];

a[j+1] = t;

flag = 1;

}

}

if(flag == 0) {

break;

}

}

}

第四种形式：选择排序

核心代码：

void ans(int \*a, int n) {

int i, j, key, t;

for(i = 0;i < n - 1;++i) {

key = i;

for(j = i + 1;j < n;++j) {

if(a[key] < a[j]) {

key = j;

}

}

if(key != i) {

t = a[key];

a[key] = a[i];

a[i] = t;

}

}

}

第六种形式：快速排序（改进版）

核心代码：

void quickSort(int \*arr, int left, int right){

int i = left, j = right;

int mid = arr[(i+j)/2];

while(i <= j){

while(arr[i] < mid) i ++;

while(arr[j] > mid) j --;

if(i <= j){

int tmp;

tmp = arr[i]; arr[i] = arr[j]; arr[j] = tmp;

i ++; j --;

}

}

if(i < right) quickSort(arr,i, right);

if(left < j) quickSort(arr,left, j);

}

第七种形式：STL（标准模版库）法-----sort函数

核心代码：

#include<iostream>

#include<algorithm> //sort使用时应该加的头文件

using namespace std;

bool cmp1(int a, int b) {

return a > b;

}

bool cmp2(int a, int b) {

return a < b;

}

int main() {

int n, i;

cin>>n;

int \*a = new int[n];

for(i = 0;i < n;++i) {

cin>>a[i];

}

sort(a,a+n,cmp1); //从大到小排序

for(i = 0;i < n;++i) {

cout<<a[i]<<" ";

}

cout<<endl;

sort(a,a+n,cmp2); //从小到大排序

for(i = 0;i < n;++i) {

cout<<a[i]<<" ";

}

delete []a;

return 0;

}

小知识：

输出百分号用两个%%即可

把字符型数字转化为整型数字令'9' - '0'=9即可

最小公倍数 = 两数乘积 / 最大公约数

第八种形式：堆排序

核心代码：

小根堆：

#include<iostream>

#define max 10000000

using namespace std;

int size = 0, heap[max];

void swap(int \*a, int \*b) {

int \*c;

c = a;

a = b;

b = c;

}

void \_push(int e){

heap[++size] = e;

int son = size, father = son / 2;

while(heap[son] < heap[father] && father >= 1){

swap(heap[son],heap[father]);

son = father,father = son / 2;

}

}

void \_pop(){

swap(heap[1],heap[size]);

heap[size--]=0;

int father=1,son=2;

while(son<=size){

if(son<size && heap[son] > heap[son+1]) son++;

if(heap[father] > heap[son]){

swap(heap[father],heap[son]);

father=son,son=father\*2;

}else break;

}

}

int \_top(){

return heap[1];

}

int main() {

int n;

cin>>n;

while(n--) {

int a;

cin>>a;

\_push(a);

}

while(size) {

cout<<\_top()<<" ";

\_pop();

}

cout<<endl;

return 0;

}

小知识：

可以用gets()函数输入空格

第九种形式：计数排序

核心代码：

void ans(int \*a, int n) {

int arr[101];

int i, j;

for(i = 0;i <= 100;++i) {

arr[i] = 0;

}

for(i = 0;i < n;++i) {

int t = a[i];

arr[t]++; //统计一个数出现的次数

}

for(i = 0;i <= 100;++i) {

if(arr[i]) {

while(arr[i]--) {

cout<<i<<" ";

}

}

}

}

\*对于小区域数字排序，例如0到100之间的排序，是最好的排序

小知识：

#include<iostream>

#include<algorithm>

using namespace std;

int main() {

int a, i;

int number[5] = {1, 2, 9, 10, 10};

reverse(number,number+5); // 数组转置

for(i = 0;i < 5;++i) {

cout<<number[i]<<" ";

}

// cin>>a>>i;

// cout<< max(a,i); (或者min(a,i))

// int \*c;

// c = min\_element(number,number+5); //输出数组中最小值

// c = max\_element(number,number+5); //输出最大值

// cout<<\*c;

}

第十种形式：基数排序----LSD

核心代码：

int findmaxbit(int \*a, int n) { //找出最高位位数 例如：1278 位数为：4

int i, max = a[0], d = 0;

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

if(max < a[i])

max = a[i];

while(max) {

d++;

max /= 10;

}

return d;

}

int ans(int \*a, int n) {

int num[10] = {0};

int time, i , j, k, key = 1;

int \*\*b = new int\* [10]; //

for(i = 0;i < 10; i++) //动态申请二维数组

b[i]=new int[n]; //

for(i = 0;i < 10;++i)

for(j = 0;j < n;++j)

b[i][j] = 0;

for(time = 0;time < findmaxbit(a,n);++time) {

for(i = 0;i < n;++i) {

k = (a[i] / key) % 10;

b[k][num[k]++] = a[i]; // 按照基数，将数据分到不同的桶中

}

key \*= 10; //把基数向高位移动一位

k = 0;

for( i = 0;i < 10;i++){

if( num[i] != 0){

for( j=0;j<num[i];j++){

a[k]=b[i][j]; //将桶中的数据再次倒回a数组中

k++ ;

}

num[i] = 0;

}

}

}

for( i = 0; i < 10; i++) //

delete[] b[i]; //删除动态二维数组

delete[] b; //

}

\*负数情况下另行考虑 \*也称为LSD排序

第十一种形式：桶排序

核心代码：

int maxbit(int \*a, int n) { //找最大位数

int max = a[0], d = 1, i = n - 1;

while(i--)

if(max < a[i])

max = a[i];

while((max /= 10))

d++;

return d;

}

void back(int \*a, int \*\*b) { //将桶中元素倒回a数组中去

int i, j, k = 0;

for(i = 0;i < 10;++i)

for(j = 1;j <= b[i][0];++j)

a[k++] = b[i][j];

}

void give(int \*a, int \*\*b, int n, int maxbit) { //分配到桶

int div = 10, i, j;

for(i = 1;i < maxbit;i++)

div \*= 10; //除数

for(j = 0;j < n;j++) {

int nu = (a[j] % div - a[j] % (div / 10)) / (div / 10);

//nu为相应的(div/10)位的值,如当div=10时,求的是个位数

int num = ++b[nu][0];

b[nu][num] = a[j];

//用b中第一列的元素来存每行元素个数

}

}

void sort(int \*a, int \*\*b, int n) { //桶排序

int bit = maxbit(a,n);

int i, j, k, m, num = 0;

for(i = 1;i <= bit;i++)

{

give(a,b,n,i);

back(a,b);

if(i != bit) {

for(j = 0;j < 10;++j)

for(k = 0;k < n + 1;++k)

b[j][k] = 0;

}

}

}

\*动态定义二维数组

第十二种形式：归并排序（也称合并排序）

核心代码：

void merge(int \*a, int left, int mid, int right) {

int i = left, j = mid + 1, n = 0, length = right - left;

while(i <= mid && j <= right) {

if(a[i] > a[j]) {

temp[n++] = a[i++];

} else {

temp[n++] = a[j++];

}

}

if(i > mid) {

while(j <= right) {

temp[n++] = a[j++];

}

} else {

while(i <= mid) {

temp[n++] = a[i++];

}

}

for(int k = 0;k <= length;++k) {

a[left+k] = temp[k];

}

}

void mergesort(int \*a, int left, int right) {

if(left < right) {

int mid = (left + right) / 2;

mergesort(a,left,mid);

mergesort(a,mid+1,right);

merge(a,left,mid,right);

}

}

3 不用第三个变量交换两个数

核心代码：

一：

void ans(int &a, int &b) {

a = a + b;

b = a - b;

a = a - b;

}

二：

void ans(int &a, int &b) { //用位运算时最好是用于整型数据

a = a ^ b;

b = a ^ b;

a = a ^ b;

}

三：

//更简单的做法是

scanf("%d,%d"&a,&b);

printf("%d,%d",b, a);

//那么你想到了什么？有时思维更重要.

4 输出最大值，最小值

#include<iostream>

#include<queue>

#include<vector>

#include<algorithm>

using namespace std;

int main() {

//priority\_queue <int,vector<int>,less<int> >p;//最大值输出

priority\_queue <int,vector<int>,greater<int> >p;//最小值输出

int n, i;

cin>>n;

int \*a = new int [n] ;

for(i =0;i < n;++i) {

cin>>a[i];

p.push(a[i]);

}

cout<<p.top()<<endl; //输出最值

delete []a;

return 0;

}

\*可以用来排序

5 判断质数

核心代码：

int ans(long long int x) {

for (long long int i = 2; i \* i <= x; i++) {

if (x % i == 0) {

return 0;

}

}

return x != 1;

}

6 判断回文数

核心代码：

int ans(int num) {

int t;

int sum = 0;

t = num;

while(num)

{

sum = sum \* 10 + num % 10;

num /= 10;

}

if(t == sum) {

return true;

} else {

return false;

}

}

7动态开辟数组

c语言:

#include<stdlib.h> //加上头文件

类型 \*p;

p = calloc(number,sizeof(类型)); (值为零)

p = malloc(number \* sizeof(类型)); (值随机)

free(p);

C++:

类型 \*p = new int;

类型 \*p = new int (初值);

delete p;

类型 \*p = new 类型 [大小];

delete []p;

类型 \*\*a = new 类型\* [number1];

for(i = 0;i < number1;++i) {

a[i] = new 类型 [number2];

}

for(i = 0;i < number1;++i) {

delete [] a[i];

}

delete []a;

8 十进制转换为任意进制

#include<iostream>

using namespace std;

#define N 2

int main(){

int a[100];

int i = 0, j, n;

cin>>n;

while(n != 0) {

a[i] = n % N;

i++;

n /= N;

}

for(j = i - 1;j >= 0;--j) {

cout<<a[j];

}

return 0;

}

小知识：（计算你的程序代码运算时间）

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main() {

int sum = 0;

clock\_t begin, end;

double cost;

begin = clock();

/\* 程序代码 \*/

end = clock();

cost = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("%lf seconds\n", cost);

system("pause");

return 0;

}

小知识：（简易API函数）

#include <iostream>

#include <Windows.h>

using namespace std;

int main()

{

cout << "Hello C++" << endl;

//弹出一个对话框函数，第一个参数为副窗口一般写NULL，第二个参数是消息的内容，第三个参数 //是标题，第四个参数是按钮（常量）

MessageBox(NULL, TEXT("Hellow C++"), TEXT("标题"), MB\_OK); //MB\_OK MB\_YESNO

system("pause");

return 0;

}

//system就是调用从程序中调用系统命令（和shell命令）。

// system("pause")就是从程序里调用“pause”命令；

// 而“pause”这个系统命令的功能很简单，就是在命令行上输出一行类似于“Press any key //to exit”的字，等待用户按一个键，然后返回。

9 键盘控制移动

#include<string.h>

#include<iostream>

#include<conio.h>

using namespace std;

int main() {

char a[10][10], c, \*p = (char \*)a; //注意(char \*)不能省略

int i, x = 0, y = 0, k;

memset(a,'.',sizeof(char) \* 100);

a[0][0] = 1;

for(i = 0;i < 100;++i) {

if(i % 10 == 0 && i != 0)

cout<<endl;

cout<<\*(p + i) <<" ";

}

while(c = getch()) {

switch(c) {

case 'w':

if(x - 1 >= 0 && x - 1 < 10) {

a[x][y] = '.'; a[--x][y] = 1;

} break;

case 's':

if(x + 1 >= 0 && x + 1 < 10) {

a[x][y] = '.'; a[++x][y] = 1;

} break;

case 'a':

if(y - 1 >= 0 && y - 1 < 10) {

a[x][y] = '.'; a[x][--y] = 1;

} break;

case 'd':

if(y + 1 >= 0 && y + 1 < 10) {

a[x][y] = '.'; a[x][++y] = 1;

} break;

}

system("cls");

for(i = 0;i < 100;++i) {

if(i % 10 == 0 && i != 0)

cout<<endl;

cout<<\*(p + i) <<" ";

}

}

return 0;

}

\*你可以使代码变得更加优美

10 广度优先搜索（宽度优先搜索）---BFS

#include<iostream>

#include<string.h>

#include<queue>

using namespace std;

struct location{

int x, y, step;

};

int main() {

location start, now, vn, end;

queue<location>q;

int i, j, n, visit[4][4], dir[][2] = {{0,1}, {1,0}, {-1,0}, {0,-1}};

int map[4][4] = {

0, 1, 0, 0,

0, 1, 1, 0,

0, 0, 0, 0,

0, 1, 1, 1,

};

memset(visit,false,sizeof(visit));

start.x = 0; start.y = 0; start.step =1;

end.x = 3; end.y = 3;

q.push(start);

visit[start.x ][start.y ] = true;

while(!q.empty()) {

now = q.front();

q.pop();

for(i = 0;i < 4;++i) {

vn.x = now.x + dir[i][0];

vn.y = now.y + dir[i][1];

vn.step = now.step;

if(vn.x == end.x && vn.y == end.y) {

cout<<"you can go through. And that costs you "<<vn.step<<" steps"; return 0;

}

if(vn.x < 4 && vn.x >= 0 && vn.y < 4 && vn.y >= 0 && !visit[vn.x][vn.y] && map[vn.x ][vn.y ] == 0) {

vn.step += 1;

q.push(vn); visit[vn.x][vn.y] = true;

}

}

}

cout<<"you can't win!"; return 0;

}

11 文件输入

#include<iosteram>

#include<stdlib.h>

using namespace std;

int main(){

FILE \*fp;

char ch;

char file[10];

cin>>file;

if((fp = fopen(file,"w")) == NULL) { //"w"表示打开文件（.txt,.doc等等），如果文件不存在，则新建文件

cout<<"Data error";

exit(0);

}

ch = getchar(); //吃掉你敲的回车

while(ch != '#') {

fputc(ch,fp);

putchar(ch);

ch = getchar();

}

fclose(fp);

return 0;

}

#include <iostream>

using namespace std;

int main() {

FILE \*fp = NULL;

fp = fopen("text.txt", "w");

fprintf(fp, "This is testing for fprintf...\n");

fputs("This is testing for fputs...\n", fp);

fclose(fp);

}

12 文件输出

#include <iostream>

using namespace std;

int main()

{

FILE \*fp = NULL;

char k, b;

fp = fopen("text.txt", "r");

k = fgetc(fp); //清除空格

while((b = fgetc(fp)) != EOF) {

putchar(b);

}

fclose(fp);

}

#include <iostream>

#include<stdlib.h>

using namespace std;

int main()

{

FILE \*fp = NULL;

char k, b;

char buff[500];

if((fp = fopen("text.txt","r")) == NULL) {

cout<<"Data error";

exit(0);

}

k = fgetc(fp); //第一个 清除空格

fgets(buff, 255, fp);

cout<<buff;

fclose(fp);

}

13 输入数据到磁盘并且输出

第一步 建立stu.doc文件用来存数据

第二步：输入学生数据到磁盘

#include<iostream>

#include<stdlib.h>

#define SIZE 3

using namespace std;

struct Student\_type{

char name[10];

int number;

int age;

char addr[15];

}stud[SIZE];

void save() {

FILE \*fp;

int i;

if((fp = fopen("stu.dat","wb")) == NULL) {

cout<<"data error!";

exit(0);

}

for(i = 0;i < SIZE;++i) {

if(fwrite(&stud[i],sizeof(struct Student\_type),1,fp) != 1) {

cout<<"file write error!";

fclose(fp);

exit(0);

}

}

}

int main() {

int i;

cout<<"Please enter the date of students:"<<endl;

for(i = 0;i < SIZE; ++i) {

cin>>stud[i].name >>stud[i].number >>stud[i].age >>stud[i].addr ;

}

save();

return 0;

}

第三步：从磁盘输出学生数据

#include<iostream>

#include<stdlib.h>

#include<stdio.h>

#define SIZE 3

using namespace std;

struct Student\_type{

char name[10];

int number;

int age;

char addr[15];

}stud[SIZE];

int main() {

FILE \*fp;

int i;

if((fp = fopen("stu.dat","rb")) == NULL) {

cout<<"data error!";

exit(0);

}

for(i = 0;i < SIZE;++i) {

fread(&stud[i],sizeof(struct Student\_type),1,fp);

printf("%-7s %4d %4d %-15s\n",stud[i].name, stud[i].number , stud[i].age, stud[i].addr);

}

fclose(fp);

return 0;

}

文件输入：

形式1

int main(){

FILE \*fp;

char ch;

char file[10];

scanf("%s",file);

if((fp = fopen(file,"w")) == NULL) {

printf("Data error");

exit(0);

}

ch = getchar();

while(ch != '#') {

fputc(ch,fp);

putchar(ch);

ch = getchar();

}

fclose(fp);

return 0;

}

形式2

int main()

{

FILE \*fp = NULL;

fp = fopen("text.txt", "w");

fprintf(fp, "This is testing for fprintf...\n");

fputs("This is testing for fputs...\n", fp);

fclose(fp);

}

文件输出：

形式1

int main()

{

FILE \*fp = NULL;

char k, b;

fp = fopen("text.txt", "r");

k = fgetc(fp); //清除空格

while((b = fgetc(fp)) != EOF) {

putchar(b);

}

fclose(fp);

}

形式2

int main()

{

FILE \*fp = NULL;

char k, b;

char buff[500];

if((fp = fopen("text.txt","r")) == NULL) {

printf("Data error");

exit(0);

}

k = fgetc(fp); //清除空格

fgets(buff, 255, fp);

printf("%s", buff);

fclose(fp);

}

线性筛

#include<iostream>

#define MAXN 10000005

using namespace std;

int prime[MAXN];

bool pp[MAXN];

int vis[MAXN];

int main() {

int a = 0;

for(int i = 2;i <= MAXN;i++) {

if(!vis[i]) prime[a++] = i,pp[i] = 1;

for(int j = 0;j < a && i \* prime[j] <= MAXN ;j++) {

vis[i \* prime[j]] = i;

if(i % prime[j] == 0) break;

}

}

int i = 2;

while(i <= MAXN) {

if(pp[i]) {

cout<<i<<endl;

}

i++;

}

return 0;

}

字符串连接函数：

n=sprintf(k,"qwer123%dhellow world",test);

sprintf(k,"qwer123hellow world");

对结构体进行排序:

关键是写出cmp函数 用sort来进行排序

例如：

struct node {

int h;

int s;

}d[5001];

int cmp(node a, node b) {

if(a.s < b.s ) { //这里注意 先比较哪个 就优先按照这个的来也就是第一关键字

return true;

} else if((a.s == b.s ) && (a.h < b.h)) {

return true;

} else {

return false;

}

}

sort(d,d+n,cmp);

全排列：

#include <iostream>

#include <algorithm>

using namespace std;

int main(){

int n;

cin>>n;

int a[1000];

for(int i = 0;i < n;++i) {

cin>>a[i];

}

sort(a,a+n); //不加这个sort的话会漏掉很多排列 因为next\_permutation(a,a+n)是按照字典排列的

do {

cout<<a[0]<<" ";

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

cout<<a[i]<<" ";

}

cout<<endl;

}while(next\_permutation(a,a+n));

return 0;

}

小知识： scanf(“%s”,a) 输出空格就结束 意思是 a就是一个单词

栈：（手动）

核心代码：

#include<iostream>

#include<stdlib.h>

using namespace std;

class \_stack {

private:

int \*top;

int \*base;

int stacksize;

public:

int init();

int \_push(int e);

int \_pop();

int \_top();

int \_empty();

int getlen();

int destory();

};

int \_stack::init() {

base = (int\*)malloc(sizeof(int));

if(!base) {

exit(0);

}

top = base;

stacksize = 1;

return 1;

}

int \_stack::\_push(int e) {

int \*p;

if(top - base >= stacksize) {

p = (int\*)realloc(base,(stacksize + 1) \* sizeof(int));

if(!p) {

exit(0);

}

base = p;

top = base + stacksize;

stacksize++;

}

\*top = e;

top++;

return 1;

}

int \_stack::\_pop() {

if(top == base) {

return 0;

}

--top;

}

int \_stack::\_top() {

if(base == top) {

return 0;

}

int e;

e = \*(top - 1);

return e;

}

int \_stack::\_empty() {

if(base == top) {

return 1;

} else {

return 0;

}

}

int \_stack::getlen() {

return (top - base);

}

int \_stack::destory() {

if(!stacksize) {

exit(0);

}

free(base);

return 1;

}

int main() {

int n;

int num;

cin>>n;

\_stack q;

q.init() ;

while(n--) {

cin>>num;

q.\_push(num);

}

while(!q.\_empty() ) {

cout<<q.\_top() <<" ";

q.\_pop() ;

}

q.destory() ;

return 0;

}

队列 （手动）：

核心代码：

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node {

int data;

struct node\* next;

};

class linkqueue {

private:

node \*front;

node \*rear;

public:

int init();

int \_push(int e);

int \_pop();

int \_top();

int \_empty();

int destory();

};

int linkqueue::init() {

front = (node\*)malloc(sizeof(node));

if(!front) {

exit(0);

}

front -> next = NULL;

rear = front;

return 1;

}

int linkqueue::\_push(int e) {

node \*p;

p = (node\*)malloc(sizeof(node));

if(!p) {

exit(0);

}

p -> data = e;

p -> next = NULL;

rear -> next = p;

rear = p;

return 1;

}

int linkqueue::\_pop() {

node \*p;

if(front == rear) {

return 0;

}

p = front -> next;

front -> next = p -> next;

if(p == rear) {

rear = front;

}

free(p);

return 1;

}

int linkqueue::\_top() {

int e;

if(front == rear) {

return 0;

}

e = front -> next -> data;

return e;

}

int linkqueue::\_empty() {

if(front == rear) {

return 1;

} else {

return 0;

}

}

int linkqueue::destory() {

while(front) {

rear = front -> next;

free(front);

front = rear;

}

}

int main() {

int n;

cin>>n;

int num;

linkqueue q;

q.init() ;

while(n--) {

cin>>num;

q.\_push(num);

}

while(!q.\_empty() ) {

cout<<q.\_top() <<" ";

q.\_pop() ;

}

q.destory() ;

return 0;

}