C++规定：在嵌套if语句中，if和else按照“就近配对”的原则配对，即相距最近且还没有配对的一对if和else首先配对。

Recursive：递归

C++数组的函数

**//把s2拷贝给s1**

Char \*strcpy(char\*s1,const char \*s2);

//n拷贝

Char \*strncpy(char\*s1,const char \*s2,size\_t n);

//在声明诸如字符数或者数组索引这样的长度变量时用size\_t是好的做法。

**//追加函数**

Char \*strcat(char\*s1,const char\*s2);

Char \*strncat(char\*s1,const char\*s2,size\_t n);

**//返回长度**

Size\_t strlen(const char\*s)

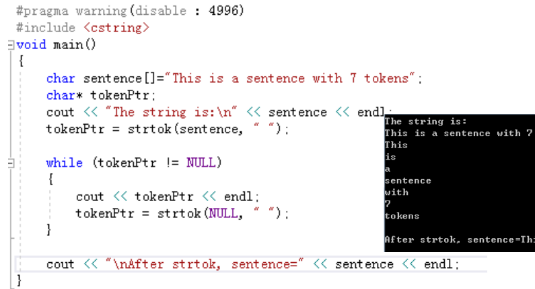
**//比较两字符串长度大小，一样是0，小于是-1**

Int strcmp(const char\*s1,const char\*s2);

Int strncmp(const char\*s1,const char\*s2,size\_t n);

**//分割字符串**

Char \*strtok(char\*s1,const char\*s2);

分隔符字符（如果传入字符串，则传入的字符串中每个字符均为分割符）

**//十进制转十六：**

int a[100] = { 0 }, n,t=0,p=0;

char b[100] = { 0 };

cout << "请输入一个十进制数" << endl;

cin >> n;

do {

a[t++] = n % 16;

n /= 16;

} while (n != 0);//a数组倒序储存十六进制的各位

for(int i=0;i<t;i++)

{

if (a[i] >= 0 && a[i] <= 9) b[p++] = a[i] + '0';//十六进制位为0-9

else b[p++] = a[i] - 10 + 'A';//16进制位为10-15

}

for(int j=t-1;j>=0;j--)

cout << b[j];

**//十六进制转十：**

char b[100];

int a[100], p = 0, sum = 0;

cout << "请输入16进制的数" << endl;

cin >> b;

int k = strlen(b);

for (int i = 0; i < k; i++)

{

if (b[i] >= 'A' && b[i] <= 'F')

a[p++] = b[i] - 'A' + 10;

else if (b[i] >= 'a' && b[i] <= 'f')

a[p++] = b[i] - 'a' + 10;

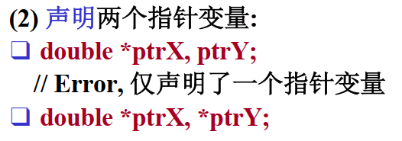
else

a[p++] = b[i] - '0';

}for (int j = 0; j < p; j++)

sum = sum \* 16 + a[j];

cout << "十进制数为" << sum << endl;



让指针=0或NULL均可以

//返回第一个只出现一次的字符

char firstC(const char \* s);

//把字符中的大写字母改为对应小写字母

void change(char \* s);

**//斐波那契数列前40项**

#include<iostream>

#include<iomanip>

Using namespace std;

int main()

{

int f1 = 1,f2 = 1;

for (int i = 1;i <= 20;i++)

{

cout << setw(12) << f1 << setw(12) << f2;

if (i % 2 == 0)

cout << endl;

f1 = f1 + f2;

f2 = f2 + f1;

}}

**//找1-1000各位和为7的素数**

#include<iostream>

#include<cmath>

using namespace std;

int f(int x)

{

int a, i;

a = sqrt(x);

for (i = 2; i <= a; i++)

if (x % i == 0) break;

if (i >= a + 1) return 1;//

else return 0;}

int main()

{

int t;

for (int i = 1; i <= 1000; i++)

{

int m = i, r = 0;//当时写在了循环外面，导致r无法清零一直累加

if (f(i) == 1)

{

do{

t = m % 10; r += t; m /= 10;

} while (m != 0);//求各位之和

if (r == 7) cout << i << '\t';

}}return 0;

}

**//百钱买百鸡**

#include<iostream>

using namespace std;

int main()

{int k;

for(int i=0;i<=20;i++)

{for(int j=0;j<=33;j++)

{k=100-i-j;

if(k%3==0&&(5\*i+3\*j+k/3)==100) cout<<i<<" "<<j<<" "<<k<<" ";

}

}

return 0;

}

**//计算字符串中单词个数**

#include<iostream>

#include<cstring>

using namespace std;

int wasd(char a[])

{

int b,num=1;

b = strlen(a);

for (int i = 1; i < b; i++)

if (a[i] == ' ' && a[i - 1] != ' ') num++;

return num;

}

int main()

{

char a[100];

int b;

cout << "请输入一串单词" << endl;

cin.getline(a,100);

b=wasd(a);

cout << "单词的个数为" << b;

return 0;

}

**//输入到某数停止输入**

#include <iostream>

using namespace std;

int main()

{

int b,i=0;

int a[10] = { 0 };

while (1)

{

cin >> b;

if (b != 10) a[i++] = b;//此处设该值为10

else break;

}

cout << "共输入"<<i<<"个数"<<endl;

for (int j = 0; j < i; j++)

cout << a[j] << '\t';

return 0;

}

**//递归求阶乘**

int f(int x)

{if(x==1) return 1;

else return x\*f(x-1);

}

**//递推求阶乘**

Int w(int n)

{

Int q=1;

For(int i=1;i<=n;i++)

Q\*=I;

Return q;

}

**//打印正三角**

int main()

{

int n;

cout << "请输入行数" << endl;

cin >> n;

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

{

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

cout << " ";//n-i个空格

cout << "\*";//1个\*

for (int z = 1; z <= i - 1; z++)

cout << " \* ";//i-1个\*

cout << endl;

}

return 0;

}//正三角；jz换一下就是反三角

**//十进制转二进制递归**

#include<iostream>

#include<cmath>

using namespace std;

void ma(int x)

{

if (x == 0) cout<<0;

else { ma(x / 2); cout << x % 2; }

}

int main()

{

int x;

cin >> x;

ma(x);

return 0;

}

**//八皇后**

#include<stdio.h>

#include<windows.h>

#define N 8 //可以根据N来修改棋盘的格数

int count = 0;//设置一个计数器

int chess[N][N] = {0};//用于存放棋盘的二维数组

void print()//打印函数

{

int i = 0;

int j = 0;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

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

{

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

{

printf("%d ",chess[i][j]);

}

printf("\n");

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

//判断是否会互吃

//关键条件

//返回1 表示存在互吃

//返回0 表示正常

int check(int i, int j)//i = 7,j = 4

{

if(i == 0)

return 0;//表示正常

int k = 0;

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

{

if(chess[k][j] == 1)//(0,4)(1,4)...

return 1;

}

for( int s = 0,k = j+1; k<N ;k++)

{

//(7,4)(6,5),(5,6),(4,7)

if(chess[i-s-1][k] == 1)//(0,11),(1,10),(2,9),(3,8),(4,7)

return 1;

s++;

}

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

{

if(chess[i-k-1][j-k-1] == 1)//(6,3)(5,2)(4,1)(3,0)

return 1;

}

for(k = 0; k<N ; k++)

{

if(chess[k][j]==1)

return 1;

}

return 0;

}

//判断棋盘上是否有一行存在没有皇后的情况

//返回0 ，表示棋盘正常（每一行都有皇后）

//返回1 ，表示棋盘有错

int check\_all()

{

int i = 0;

int j = 0;

int flag = 0;

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

{

flag = 0;

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

{

if(chess[i][j]==1)

{

flag = 1;

break;

}

}

if(flag == 0)

return 1;//有错

}

return 0;

}

//检查某一行是否存在皇后

//返回0 表示存在

//返回1 表示没皇后

int check\_line(int line)

{

if(line==0)

return 0;

int k = 0;

int s = 0;

int flag = 1;

for(s = 0; s<line-1 ; s++)

{

flag = 1;

for(k = 0; k<N ;k++)

{

if(chess[s][k]==1)

flag = 0;

}

if(flag==1)

return 1;

}

return 0;

}

//递归的主要算法

void queen(int i,int j)

{

//符合，置一，进入下一行

if(check\_line(i)==1)//若该行有皇后，返回

return ;

if((i==(N-1)))//若此时是最后一行

{

if(check(i,j)==0)//当最后一行的皇后可以放下（表示可以成功放置）

{

chess[i][j] = 1;//将该位置1，表示皇后

print();//打印

count++;//计数器+1

}

}

if(check(i,j)==0)//当可以放皇后时

{

chess[i][j] = 1;//放入

//print();

//Sleep(1000);

queen(i+1,0);//进行下一行的皇后放置

}

if(j==N)//如果j等于列数，表明越界，返回

return ;

chess[i][j] = 0;//将该位置0

//print();

//Sleep(500);

//不符合，置零，右诺

queen(i,j+1);//将该行皇后右移一位

}

int main(void)

{

queen(0,0);

printf("\ncount = %d\n",count);

return 0;

}

**//二维数组求转置且用指针输出**

void transpose(int\* a, int m, int n)//m是新的行，n是新的列

{

cout << \*(a + 5\*n + m)<<"\t";

}

int main()

{

int arr[3][5] ;

int\* p = &arr[0][0];

for (int i = 0;i < 3;i++)

{

for (int j = 0;j < 5;j++)

{

cin >> arr[i][j];

}

}

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

{

for (int j = 0;j < 3;j++)

{

transpose(p, i, j);

}

cout << endl;

}

system("pause");

return 0;

}

**//合并两串c数组，并把里面的字母全部改成大写**

void f(char\* s)

{

while (\*s != '\0')

{

if (\*s >= 'a' && \*s <= 'z')

\*s -= 32;

if (\*s == '!' && \*(s + 1) != NULL)

\*s = ',';

s++;

}

}

int main()

{

char a1[50], a2[50];

cout << "Please intput two strings:" << endl;

cin.getline(a1, 50);

cin.getline(a2, 50);

f(a1);

f(a2);

cout << a1 << "," << a2;

return 0;

}

**//冒泡排序**

int main()

{

int array[10];

for (int i = 1;i <= 10;i++)

{

int num;

cout << "input number:";

cin >> num;

array[i - 1] = num;

}

for (int i = 0;i < 9;i++)

{

for(int j=0;j<9-i;j++)

{

if (array[j] >= array[j+1])

{ int temp = array[j];

array[j] = array[j+1];

array[j+1] = temp;

}

}

}

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

{

cout << array[i] << " ";

}

system("pause");

return 0;

}

**//7.32 find the minimum value in an array**

int recursiveMinimum(int array[], int size)

{

static int min = array[size-1];

if (array[size-2] < min)

{

min = array[size-2];

}

if (size <=2)

return min;

return recursiveMinimum(array, size-1);

}

int main()

{

int array[10];

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

{

cout << "输入数据"<<i<<":" ;

cin >> array[i];

}

cout << "数组中最小元素为:" << recursiveMinimum(array, 10) << endl;

system("pause");

return 0;

}

**//7.31 print a string backward**

void stringReverse(string a,int b)

{

cout << a[b];

b--;

if (b >= 0)

{

stringReverse(a, b);

}

}

int main()

{

string a;

cout << "请输入一个字符串：";

cin.getline(a, "\n");

int b = a.size();

cout << "逆序打印字符串：";

stringReverse(a, b);

cout << endl;

system("pause");

return 0;

}

**//7.27**

int main()

{

array<int, 1000>era;

for (int i = 0;i < 1000;i++)

{

era[i] = 1;

}

for (int i = 2;i < 1000;i++)

{

if (era[i]==1)

{

for (int j = i+1;j < 1000;j++)

{

if (j % i == 0)

{

era[j] = 0;

}

}

}

}

for (int i = 2;i < 1000;i++)

{

if (era[i])

{

cout << i << endl;

}

}

system("pause");

return 0;

}

**//7.21 sales summary**

int main()

{

int sales[4][5] = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 };//行是售货员，列是产品

int c = 0;

cout << "ps.a为售货员编号，b为产品编号，c为当天销售总额,输入c=-1来结束循环" << endl;

for(;c!=-1; ){

for (int i = 0;i <= 3;i++)

{

for (int j = 0;j <= 4;j++)

{

cout << "a)" << i << " b)" << j << " c)";

cin >> c;

if(c>=0)

sales[i][j] += c;

if (c == -1)break;

}

if (c == -1)break;

}

}

for (int j = 0;j <= 3;j++)

{

cout << "第" << j+1 << "个产品的业绩：";

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

{

cout << " "<<sales[j][i]<<" ;";

}

cout << endl;

}

system("pause");

return 0;

}

**//计算pi的值**

double b = 1, a = 1, pai = 0, c = 1;

for (int n = 1;n <= 1000;n++)

{

b = 4 / c;

pai += b;

a += 2;

if (n % 2)

c = a \* (-1);

else

c = a;

cout <<"取前"<<n<<"项时Π的值为" << pai << endl;

}

**//打印菱形**

int r = 1;

cout << "请输入菱形的行数（1-19内的奇数）" << endl;

cin >> r;

for (int a = 1;a <= (r+1)/2;a++)

{

for (int b = 1;b <= (r + 1) / 2 - a;b++)

cout << " ";

for (int c = 1;c <= (2 \* a - 1);c++)

cout << "\*";

cout << endl;

}

for (int a = 1;a <= (r - 1) / 2;a++)

{

for (int b = 1;b <= a;b++)

cout << " ";

for (int c = (r-2\*a);c >=1;c--)

cout << "\*";

cout << endl;

}

**//算n次方的函数**

int power(int a, int b)

{

if (b == 1)return a;

else return a \* power(a, b - 1);

}

int main()

{

int base = 1, exponent = 1;

cout << "请输入a的b次方：";

cin >> base>>exponent;

cout << power(base, exponent);

cout << endl;

system("pause");

return 0;

}

template<class T>

保留小数点后几位

//第一种写法

cout<<setiosflags(ios::fixed)<<setprecision(2);

//第二种写法

cout.setf(ios::fixed);

cout<<setprecision(2);

//第三种写法

cout<<fixed<<setprecision(2)***;***

如果是保留几位数：cout << setprecision(n);

//二进制转十进制

int main()

{

int num = 1;

cout << "请输入一个二进制值:";

cin >> num;

int c = 1;

int b = 1;

int a = 1;

int e = 1;

int plus = 0;

while (num/b!=0)

{

a = (num / b) % 10;

b = b \* 10;

plus = plus + a \* e;

e = e \* 2;

c++;

}

cout << "这个十进制数为" << plus << endl;

system("pause");

return 0;

}

辗转相除(计算两个整数

a,b 的最大公约数)：

int Gcd\_2(int a,int b)

{if (a<=0 || b<=0)return 0;

int temp;

while (b > 0) {temp = a % b;

a = b; b = temp;}

return a;}

template<class Type> //冒泡排序

void BubbleSort(Type\* array,int start,int end){

for(int i = 0;i < end - start;i++)

for(int j = 0;j < end - start;j++){

if(array[j] > array[j + 1])

swap(array[j],array[j + 1]);

}

}

template<class Type> //选择排序

void SelectionSort(Type\* array,int start,int end){

for(int i = 0;i < end - start;i++)

for(int j = i + 1;j < end - start + 1;j++){

if(array[i] > array[j])

swap(array[i],array[j]);

}

}

template<class Type> //插入排序

void InsertionSort(Type\* array,int start,int end){

for(int i = 1;i < end - start + 1;i++)

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

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

int temp = array[i];

for(int k = i;k > j;k--)

array[k] = array[k - 1];

array[j] = temp; //此时array[j]是最大的了 break; } }

}

template<class Type> 快速排序

int Partition(Type \*a,int start,int end){

Type x = a[start];

int i = start;

for(int j = start + 1;j <= end;j++){

if(a[j] <= x){

i++;

swap(a[i],a[j]);

}

}

swap(a[i],a[start]); //把基准的值放在中间，则左边都小于他，右边都大于他 return i;

}

template<class Type>

void QuickSort(Type \*array,int start,int end){

if(start < end){

int q = Partition(array,start,end); //分割成两部分

QuickSort(array,start,q - 1);

QuickSort(array,q + 1,end); } }

template<class Type> //希尔排序

void ShellPass(Type\* array,int start,int end,int d){

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

for(int j = start + i + d;j < end - start + 1;j += d){

for(int k = start + i;k < j;k += d){

if(array[k] > array[j]){

Type temp = array[j];

for(int l = j;l > k;l -= d)

array[l] = array[l - d];

array[k] = temp;

} } } } }

template<class Type> 希尔排序

void ShellSort(Type\* array,int start,int end){

int d = 10;

while(d > 0){

d = (d + 1) / 2;

ShellPass(array,start,end,d);

if(d == 1)

break; } }

string str="abc";

char \*p=str.data();

1、如果要将string转换为char\*，可以使用string提供的函数c\_str() ，或是函数data()，data除了返回字符串内容外，不附加结束符'\0'，而c\_str()返回一个以‘\0’结尾的字符数组。

2、const char \*c\_str();

c\_str()函数返回一个指向正规C字符串的指针,内容与本string串相同.

这是为了与c语言兼容,在c语言中没有string类型,故必须通过string类对象的成员函数c\_str()把string 对象转换成c中的字符串样式.

注意：一定要使用strcpy()函数 等来操作方法c\_str()返回的指针

比如：最好不要这样:

char\* c;

string s="1234";

c = s.c\_str(); //c最后指向的内容是垃圾,因为s对象被析构,其内容被处理

应该这样用：

char c[20];

string s="1234";

strcpy(c,s.c\_str());

这样才不会出错,c\_str()返回的是一个临时指针,不能对其进行操作

再举个例子

c\_str() 以 char\* 形式传回 string 内含字符串

如果一个函数要求char\*参数,可以使用c\_str()方法：

string s = "Hello World!";

printf("%s",s.c\_str()); //输出 "Hello World!"

char \*转换成string

可以直接赋值。

string s;

char \*p = "adghrtyh";

s = p;

a) string s; //生成一个空字符串s

b) string s(str) //拷贝构造函数 生成str的复制品

c) string s(str,stridx) //将字符串str内“始于位置stridx”的部分当作字符串的初值

d) string s(str,stridx,strlen) //将字符串str内“始于stridx且长度顶多strlen”的部分作为字符串的初值

e) string s(cstr) //将C字符串作为s的初值

f) string s(chars,chars\_len) //将C字符串前chars\_len个字符作为字符串s的初值。

g) string s(num,c) //生成一个字符串，包含num个c字符

h) string s(beg,end) //以区间beg;end(不包含end)内的字符作为字符串s的初值

i) s.~string() //销毁所有字符，释放内存

深拷贝浅拷贝区别

简单的来说就是，在有指针的情况下，浅拷贝只是增加了一个指针指向已经存在的内存，而深拷贝就是增加一个指针并且申请一个新的内存，使这个增加的指针指向这个新的内存，采用深拷贝的情况下，释放内存的时候就不会出现在浅拷贝时重复释放同一内存的错误！

我列举一个例子来说吧：

你正在编写C++程序中有时用到，操作符的重载。最能体现深层拷贝与浅层拷贝的，就是‘=’的重载。

看下面一个简单的程序：

class string

{

char \*m\_str;

public:

string(char \*s）

{

m\_str=s;

}

string()

{};

String & operator=(const string s)

{

m\_str=s.m\_str;

return \*this

}

};

int main()

{

string s1("abc"),s2;

s2=s1;

cout<<s2.m\_str;

}

上面的 =重载其是就是实现了浅拷贝原因。是由于对象之中含有指针数据类型.s1,s2恰好指向同一各内存。所以是浅拷贝。而你如果修改一下原来的程序：

string&operator=(const string&s)

{

if(strlen(m\_str)!=strlen(s.m\_str))

m\_str=new char[strlen(s.m\_str)+1];

if(\*this!=s)

strcopy(m\_str,s.m\_str);

return \*this;

}

这样你就实现了深拷贝，原因是你为被赋值对象申请了一个新的内存所以就是深拷贝。

//类封装

#include <iostream> #include <string>

using namespace std;

/\*\*

\* 定义类：Student

\* 数据成员：m\_strName

\* 无参构造函数：Student()

\* 有参构造函数：Student(string \_name)

\* 拷贝构造函数：Student(const Student& stu)

\* 析构函数：~Student()

\* 数据成员函数：setName(string \_name)、getName()

\*/

class Student

{

public:

Student()

{

}

Student(string \_name)

{

}

Student(const Student& stu)

{

}

~Student()

{

}

void setName(string \_name)

{

m\_strName=\_name;

}

string getName()

{

return m\_strName;

}

private:

string m\_strName;

};

int main(void)

{

// 通过new方式实例化对象\*stu

Student \*stu = new Student();

// 更改对象的数据成员为“慕课网”

stu->setName("慕课网");

// 打印对象的数据成员

cout<<stu->getName()<<endl;

return 0;

}