简介：计算机考研之家搜集的华为C语言经典面试题，来试试你的C语言水平吧。每道题都附有详细解答和讲解，很有参考价值的C语言面试题。

怎么判断链表中是否有环？

bool CircleInList(Link\* pHead)

{

if(pHead = = NULL || pHead->next = = NULL)//无节点或只有一个节点并且无自环

return (false);

if(pHead->next = = pHead)//自环

return (true);

Link \*pTemp1 = pHead;//step 1

Link \*pTemp = pHead->next;//step 2

while(pTemp != pTemp1 && pTemp != NULL && pTemp->next != NULL)

{

pTemp1 = pTemp1->next;

pTemp = pTemp->next->next;

}

if(pTemp = = pTemp1)

return (true);

return (false);

}

两个字符串，s,t;把t字符串插入到s字符串中，s字符串有足够的空间存放t字符串

void insert(char \*s, char \*t, int i)

{

memcpy(&s[strlen(t)+i],&s[i],strlen(s)-i);

memcpy(&s[i],t,strlen(t));

s[strlen(s)+strlen(t)]='\0';

}

1。编写一个 C 函数，该函数在一个字符串中找到可能的最长的子字符串，且该字符串是由同一字符组成的。

char \* search(char \*cpSource, char ch)

{

char \*cpTemp=NULL, \*cpDest=NULL;

int iTemp, iCount=0;

while(\*cpSource)

{

if(\*cpSource == ch)

{

iTemp = 0;

cpTemp = cpSource;

while(\*cpSource == ch)

++iTemp, ++cpSource;

if(iTemp > iCount)

iCount = iTemp, cpDest = cpTemp;

if(!\*cpSource)

break;

}

++cpSource;

}

return cpDest;

}

2。请编写一个 C 函数，该函数在给定的内存区域搜索给定的字符，并返回该字符所在位置索引值。

int search(char \*cpSource, int n, char ch)

{

int i;

for(i=0; ireturn i;

}

一个单向链表，不知道头节点,一个指针指向其中的一个节点，问如何删除这个指针指向的节点？

将这个指针指向的next节点值copy到本节点，将next指向next->next,并随后删除原next指向的节点。

#include

void foo(int m, int n)

{

printf("m=%d, n=%d\n", m, n);

}

int main()

{

int b = 3;

foo(b+=3, ++b);

printf("b=%d\n", b);

return 0;

}

输出：m=7,n=4,b=7(VC6.0)

这种方式和编译器中得函数调用关系相关即先后入栈顺序。不过不同

编译器得处理不同。也是因为C标准中对这种方式说明为未定义，所以

各个编译器厂商都有自己得理解，所以最后产生得结果完全不同。

因为这样，所以遇见这种函数，我们首先要考虑我们得编译器会如何处理

这样得函数，其次看函数得调用方式，不同得调用方式，可能产生不同得

结果。最后是看编译器优化。

2.写一函数，实现删除字符串str1中含有的字符串str2.

第二个就是利用一个KMP匹配算法找到str2然后删除（用链表实现的话，便捷于数组）

/\*雅虎笔试题(字符串操作)

给定字符串A和B,输出A和B中的最大公共子串。

比如A="aocdfe" B="pmcdfa" 则输出"cdf"

\*/

//Author: azhen

#include

#include

#include

char \*commanstring(char shortstring[], char longstring[])

{

int i, j;

char \*substring=malloc(256);

if(strstr(longstring, shortstring)!=NULL) //如果……，那么返回shortstring

return shortstring;

for(i=strlen(shortstring)-1;i>0; i--) //否则，开始循环计算

{

for(j=0; j<=strlen(shortstring)-i; j++){

memcpy(substring, &shortstring[j], i);

substring[i]='\0';

if(strstr(longstring, substring)!=NULL)

return substring;

}

}

return NULL;

}

main()

{

char \*str1=malloc(256);

char \*str2=malloc(256);

char \*comman=NULL;

gets(str1);

gets(str2);

if(strlen(str1)>strlen(str2)) //将短的字符串放前面

comman=commanstring(str2, str1);

else

comman=commanstring(str1, str2);

printf("the longest comman string is: %s\n", comman);

}

11.写一个函数比较两个字符串str1和str2的大小，若相等返回0，若str1大于

str2返回1，若str1小于str2返回－1

int strcmp ( const char \* src,const char \* dst)

{

int ret = 0 ;

while( ! (ret = \*(unsigned char \*)src - \*(unsigned char \*)dst) && \*dst)

{

++src;

++dst;

}

if ( ret < 0 )

ret = -1 ;

else if ( ret > 0 )

ret = 1 ;

return( ret );

}

3,求1000！的未尾有几个0（用素数相乘的方法来做，如72=2\*2\*2\*3\*3）;

求出1->1000里,能被5整除的数的个数n1,能被25整除的数的个数n2,能被125整除的数的个数n3,

能被625整除的数的个数n4.

1000!末尾的零的个数=n1+n2+n3+n4;

#include

#define NUM 1000

int find5(int num){

int ret=0;

while(num%5==0){

num/=5;

ret++;

}

return ret;

}

int main(){

int result=0;

int i;

for(i=5;i<=NUM;i+=5)

{

result+=find5(i);

}

printf(" the total zero number is %d\n",result);

return 0;

}

1. 有双向循环链表结点定义为：

struct node

{ int data;

struct node \*front,\*next;

};

有两个双向循环链表A，B，知道其头指针为：pHeadA,pHeadB，请写一函数将两链表中data值相同的结点删除

BOOL DeteleNode(Node \*pHeader, DataType Value)

{

if (pHeader == NULL) return;

BOOL bRet = FALSE;

Node \*pNode = pHead;

while (pNode != NULL)

{

if (pNode->data == Value)

{

if (pNode->front == NULL)

{

pHeader = pNode->next;

pHeader->front = NULL;

}

else

{

if (pNode->next != NULL)

{

pNode->next->front = pNode->front;

}

pNode->front->next = pNode->next;

}

Node \*pNextNode = pNode->next;

delete pNode;

pNode = pNextNode;

bRet = TRUE;

//不要break或return, 删除所有

}

else

{

pNode = pNode->next;

}

}

return bRet;

}

void DE(Node \*pHeadA, Node \*pHeadB)

{

if (pHeadA == NULL || pHeadB == NULL)

{

return;

}

Node \*pNode = pHeadA;

while (pNode != NULL)

{

if (DeteleNode(pHeadB, pNode->data))

{

if (pNode->front == NULL)

{

pHeadA = pNode->next;

pHeadA->front = NULL;

}

else

{

pNode->front->next = pNode->next;

if (pNode->next != NULL)

{

pNode->next->front = pNode->front;

}

}

Node \*pNextNode = pNode->next;

delete pNode;

pNode = pNextNode;

}

else

{

pNode = pNode->next;

}

}

}

2. 编程实现：找出两个字符串中最大公共子字符串,如"abccade","dgcadde"的最大子串为"cad"

int GetCommon(char \*s1, char \*s2, char \*\*r1, char \*\*r2)

{

int len1 = strlen(s1);

int len2 = strlen(s2);

int maxlen = 0;

for(int i = 0; i < len1; i++)

{

for(int j = 0; j < len2; j++)

{

if(s1[i] == s2[j])

{

int as = i, bs = j, count = 1;

while(as + 1 < len1 && bs + 1 < len2 && s1[++as] == s2[++bs])

count++;

if(count > maxlen)

{

maxlen = count;

\*r1 = s1 + i;

\*r2 = s2 + j;

}

}

}

}

3. 编程实现：把十进制数(long型)分别以二进制和十六进制形式输出，不能使用printf系列库函数

char\* test3(long num) {

char\* buffer = (char\*)malloc(11);

buffer[0] = '0';

buffer[1] = 'x';

buffer[10] = '\0';

char\* temp = buffer + 2;

for (int i=0; i < 8; i++) {

temp[i] = (char)(num<<4\*i>>28);

temp[i] = temp[i] >= 0 ? temp[i] : temp[i] + 16;

temp[i] = temp[i] < 10 ? temp[i] + 48 : temp[i] + 55;

}

return buffer;

}

输入N, 打印 N\*N 矩阵

比如 N = 3，打印：

1 2 3

8 9 4

7 6 5

N = 4，打印：

1 2 3 4

12 13 14 5

11 16 15 6

10 9 8 7

解答：

1 #define N 15

int s[N][N];

void main()

{

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

int a = 1;

for( ; k < (N+1)/2; k++ )

{

while( j < N-k ) s[i][j++] = a++; i++; j--;

while( i < N-k ) s[i++][j] = a++; i--; j--;

while( j > k-1 ) s[i][j--] = a++; i--; j++;

while( i > k ) s[i--][j] = a++; i++; j++;

}

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

{

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

cout << s[i][j] << '\t';

cout << endl;

}

}

2 define MAX\_N 100

int matrix[MAX\_N][MAX\_N];

/\*

\*（x,y）：第一个元素的坐标

\* start：第一个元素的值

\* n：矩阵的大小

\*/

void SetMatrix(int x, int y, int start, int n) {

int i, j;

if (n <= 0) //递归结束条件

return;

if (n == 1) { //矩阵大小为1时

matrix[x][y] = start;

return;

}

for (i = x; i < x + n-1; i++) //矩阵上部

matrix[y][i] = start++;

for (j = y; j < y + n-1; j++) //右部

matrix[j][x+n-1] = start++;

for (i = x+n-1; i > x; i--) //底部

matrix[y+n-1][i] = start++;

for (j = y+n-1; j > y; j--) //左部

matrix[j][x] = start++;

SetMatrix(x+1, y+1, start, n-2); //递归

}

void main() {

int i, j;

int n;

scanf("%d", &n);

SetMatrix(0, 0, 1, n);

//打印螺旋矩阵

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

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

printf("%4d", matrix[i][j]);

printf("\n");

}

}

斐波拉契数列递归实现的方法如下：

int Funct( int n )

{

if(n==0) return 1;

if(n==1) return 1;

retrurn Funct(n-1) + Funct(n-2);

}

请问，如何不使用递归，来实现上述函数？

请教各位高手！

解答：int Funct( int n ) // n 为非负整数

{

int a=0;

int b=1;

int c;

if(n==0) c=1;

else if(n==1) c=1;

else for(int i=2;i<=n;i++) //应该n从2开始算起

{

c=a+b;

a=b;

b=c;

}

return c;

}

解答：

现在大多数系统都是将低字位放在前面，而结构体中位域的申明一般是先声明高位。

100 的二进制是 001 100 100

低位在前 高位在后

001----s3

100----s2

100----s1

所以结果应该是 1

如果先申明的在低位则：

001----s1

100----s2

100----s3

结果是 4

1、原题跟little-endian，big-endian没有关系

2、原题跟位域的存储空间分配有关，到底是从低字节分配还是从高字节分配，从Dev C++和VC7.1上看，都是从低字节开始分配，并且连续分配，中间不空，不像谭的书那样会留空位

3、原题跟编译器有关，编译器在未用堆栈空间的默认值分配上有所不同，Dev C++未用空间分配为

01110111b，VC7.1下为11001100b,所以在Dev C++下的结果为5，在VC7.1下为1。

注：PC一般采用little-endian，即高高低低，但在网络传输上，一般采用big-endian，即高低低高，华为是做网络的，所以可能考虑big-endian模式，这样输出结果可能为4

判断一个字符串是不是回文

int IsReverseStr(char \*aStr)

{

int i,j;

int found=1;

if(aStr==NULL)

return -1;

j=strlen(aStr);

for(i=0;iif(\*(aStr+i)!=\*(aStr+j-i-1))

{

found=0;

break;

}

return found;

}

Josephu 问题为：设编号为1，2，… n的n个人围坐一圈，约定编号为k（1<=k<=n）的人从1开始报数，数到m 的那个人出列，它的下一位又从1开始报数，数到m的那个人又出列，依次类推，直到所有人出列为止，由此产生一个出队编号的序列。

数组实现：

#include

#include

int Josephu(int n, int m)

{

int flag, i, j = 0;

int \*arr = (int \*)malloc(n \* sizeof(int));

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

arr[i] = 1;

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

{

flag = 0;

while (flag < m)

{

if (j == n)

j = 0;

if (arr[j])

++flag;

++j;

}

arr[j - 1] = 0;

printf("第%4d个出局的人是：%4d号\n", i, j);

}

free(arr);

return j;

}

int main()

{

int n, m;

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

printf("最后胜利的是%d号！\n", Josephu(n, m));

system("pause");

return 0;

}

链表实现：

#include

#include

typedef struct Node

{

int index;

struct Node \*next;

}JosephuNode;

int Josephu(int n, int m)

{

int i, j;

JosephuNode \*head, \*tail;

head = tail = (JosephuNode \*)malloc(sizeof(JosephuNode));

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

{

tail->index = i;

tail->next = (JosephuNode \*)malloc(sizeof(JosephuNode));

tail = tail->next;

}

tail->index = i;

tail->next = head;

for (i = 1; tail != head; ++i)

{

for (j = 1; j < m; ++j)

{

tail = head;

head = head->next;

}

tail->next = head->next;

printf("第%4d个出局的人是：%4d号\n", i, head->index);

free(head);

head = tail->next;

}

i = head->index;

free(head);

return i;

}

int main()

{

int n, m;

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

printf("最后胜利的是%d号！\n", Josephu(n, m));

system("pause");

return 0;

}

已知strcpy函数的原型是：

char \* strcpy(char \* strDest,const char \* strSrc);

1.不调用库函数，实现strcpy函数。

2.解释为什么要返回char \*。

解说：

1.strcpy的实现代码

char \* strcpy(char \* strDest,const char \* strSrc)

{

if ((strDest==NULL)||(strSrc==NULL)) file://[/1]

throw "Invalid argument(s)"; //[2]

char \* strDestCopy=strDest; file://[/3]

while ((\*strDest++=\*strSrc++)!='\0'); file://[/4]

return strDestCopy;

}

错误的做法：

[1]

(A)不检查指针的有效性，说明答题者不注重代码的健壮性。

(B) 检查指针的有效性时使用((!strDest)||(!strSrc))或(!(strDest&&strSrc))，说明答题者对C语言中类型的隐式转换没有深刻认识。在本例中char \*转换为bool即是类型隐式转换，这种功能虽然灵活，但更多的是导致出错概率增大和维护成本升高。所以C++专门增加了bool、true、false 三个关键字以提供更安全的条件表达式。

(C)检查指针的有效性时使用((strDest==0)||(strSrc==0))，说明答题者不知道使用常量的好处。直接使用字面常量（如本例中的0）会减少程序的可维护性。0虽然简单，但程序中可能出现很多处对指针的检查，万一出现笔误，编译器不能发现，生成的程序内含逻辑错误，很难排除。而使用NULL代替0，如果出现拼写错误，编译器就会检查出来。

[2]

(A)return new string("Invalid argument(s)");，说明答题者根本不知道返回值的用途，并且他对内存泄漏也没有警惕心。从函数中返回函数体内分配的内存是十分危险的做法，他把释放内存的义务抛给不知情的调用者，绝大多数情况下，调用者不会释放内存，这导致内存泄漏。

(B)return 0;，说明答题者没有掌握异常机制。调用者有可能忘记检查返回值，调用者还可能无法检查返回值（见后面的链式表达式）。妄想让返回值肩负返回正确值和异常值的双重功能，其结果往往是两种功能都失效。应该以抛出异常来代替返回值，这样可以减轻调用者的负担、使错误不会被忽略、增强程序的可维护性。

[3]

(A)忘记保存原始的strDest值，说明答题者逻辑思维不严密。

[4]

(A)循环写成while (\*strDest++=\*strSrc++);，同[1](B)。

(B)循环写成while (\*strSrc!='\0') \*strDest++=\*strSrc++;，说明答题者对边界条件的检查不力。循环体结束后，strDest字符串的末尾没有正确地加上'\0'。