1、INT整型变量占用（ ）个字节。

A.4 B.8 C.32 D.128

2、微型计算机的问世是由于（）的出现

1. 中小规模集成电路 B.晶体管电路
2. （超）大规模集成电路 D.电子管电路

3、在磁盘上建立子目录有许多优点，下列描述中不属于建立子目录优点的是（）

A、便于文件管理 B、解决根目录中目录项个数有限问题

C、加快文件查找速度 D、节省磁盘使用空间

4、计算机病毒传染的必要条件是（）

A、在内存中运行病毒程序 B、对磁盘进行读写操作

C、在内存中运行含有病毒的程序 D、复制文件

5、IPV 4地址是由（）位二进制数码表示

A、16 B、32 C、64 D、8

6、TCP/IP协议共有（）层协议

A、3 B、4 C、5 D、6

7、一个高度为h的二叉树最小元素数目是（）

A、2h+1 B、h C、2h-1 D、2h

8、设有一个含有13个元素的Hash表（0~12），Hash函数是：H(key)=key%13，其中%是求余数运算。用线性探查法解决冲突，则对于序列（2,8,31,20,19,18,53,27)，18应放在第几号格中（）

A、9 B、2 C、4 D、0

9、64KB的存储器用十六进制表示，它的最大的地址码是（）

A、10000 B、FFFF C、1FFFF D、EFFFF

10、计算机能直接执行的指令包括两部分，它们是（）

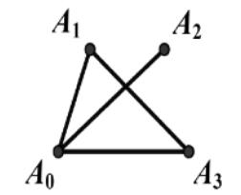
A、源操作数与目标操作数 B、数字与字符

C、ASCII码与汉字代码 D、操作码与操作数

11、二叉树的（ ）第一个访问的节点是根节点

1. 先序遍历 B、中序遍历 C、后序遍历 D、以上都是

12、以 A0 作为起点，对下面的无向图进行深度优先遍历时，遍历顺序不可能是（ ）



1. A0, A1 , A2, A3 B、A0, A1, A3, A2

C、A0, A2, A1, A3 D、A0, A3, A1, A2

13、通常在搜索引擎中，对某个关键词加上双引号表示（ ）

1. 排除关键词，不显示任何包含该关键词的结果
2. 将关键词分解，在搜索结果中必须包含其中的一部分
3. 精确搜索，只显示包含整个关键词的结果
4. 站内搜索，只显示关键词所指向网站的内容

14、在以下各项中，（ ）不是CPU的组成部分

A、控制器 B、运算器 C、寄存器 D、主板

15、在关系数据库中，存放在数据库中的数据的逻辑结构以（ ）为主

A、二叉树 B、多叉树 C、哈希表 D、二维表

16、LAN的含义是（）

A、因特网 B、局域网 C、广域网 D、城域网

17、在下列关于计算机语言的说法中，正确的有（ ）

A、高级语言比汇编语言更高级，是因为它的程序的运行效率更高

B、随着Pascal、C等高级语言的出现，机器语言和汇编语言已经退出了历史舞台

C、高级语言比汇编语言程序更容易从一种计算机上移植到另一种计算机上

D、C是一种面向对象的高级计算机语言

18、一个无法靠自身的控制终止的循环成为“死循环”，例如，在C语言程序中，语句while(1) printf("\*");就是一个死循环，运行时它将无休止地打印\*号。下面关于死循环的说法中，只有（ ）是正确的

1. 不存在一种算法，对任何一个程序及相应的输入数据，都可以判断是否会出现死循环，因而，任何编译系统都不做死循环检查
2. 有些编译系统可以检测出死循环
3. 死循环属于语法错误，既然编译系统能检查各种语法错误，当然也应该能检查出死循环
4. 死循环与多进程中出现的“死锁”差不多，而死锁是可以检测的，因而，死循环也可以检测的

19、在C语言中,表达式23|2^5的值是()

A、18 B、1 C、23 D、32

20、与十进制数1770对应的八进制数是（  ）

A、3350 B、3351 C、3352 D、3540

21、看程序写结果：

#include<stdio.h>

int main()

{

int i, p[5], a, b, c, x, y = 20;

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

scanf( "%d", &p[i] );

a = (p[0] + p[1]) + (p[2] + p[3] + p[4]) / 7;

b = p[0] + p[1] / ( (p[2] + p[3]) / p[4]);

c = p[0] \* p[1] / p[2];

x = a + b - p[(p[3] + 3) % 4];

if ( x > 10 )

y += (b \* 100 - a) / (p[p[4] % 3] \* 5);

else

y += 20 + (b \* 100 - c) / (p[p[4] % 3] \* 5);

printf( "%d,%d\n", x, y );

return(0);

}

{注：本例中，给定的输入数据可以避免分母为0或数组元素下表越界。} 输入：6 6 5 5 3，输出（）

A、15，46 B、46，15 C、16，45 D、45,16

1. 看程序写结果

#include<stdio.h>

void fun( int \*a, int \*b )

{

int \*k;

k = a; a = b; b = k;

}

int main()

{

int a = 3, b = 6, \*x = &a, \*y = &b;

fun( x, y );

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

}

A、6,3 B、3,6 C、3,3 D、6,6

1. 看程序写结果

#include <iostream>

using namespace std;

int main()

{

int a[6] = {1, 2, 3, 4, 5, 6};

int pi = 0;

int pj = 5;

int t, i;

while (pi < pj)

{

t = a[pi];

a[pi] = a[pj];

a[pj] = t;

pi++;

pj--;

}

for (i = 0; i < 6; i++)

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

cout << endl;

return 0;

}

输出（）

A、1,2,3,4,5,6 B、1,2,3,4,5 C、5,4,3,2,1 D、6,5,4,3,2,1

1. 看程序写结果

#include <iostream>

using namespace std;

int main()

{

char a[100][100], b[100][100];

string c[100];

string tmp;

int n, i = 0, j = 0, k = 0, total\_len[100], length[100][3];

cin >> n;

getline(cin, tmp);

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

{

getline(cin, c[i]);

total\_len[i] = c[i].size();

}

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

{

j = 0;

while (c[i][j] != ':')

{

a[i][k] = c[i][j];

k = k + 1;

j++;

}

length[i][1] = k - 1;

a[i][k] = 0;

k = 0;

for (j = j + 1; j < total\_len[i]; j++)

{

b[i][k] = c[i][j];

k = k + 1;

}

length[i][2] = k - 1;

b[i][k] = 0;

k = 0;

}

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

{

if (length[i][1] >= length[i][2])

cout << "NO,";

else

{

k = 0;

for (j = 0; j < length[i][2]; j++)

{

if (a[i][k] == b[i][j])

k = k + 1;

if (k > length[i][1])

break;

}

if (j == length[i][2])

cout << "NO,";

else

cout << "YES,";

}

}

cout << endl;

return 0;

}

输入：

3

AB:ACDEbFBkBD

AR:ACDBrT

SARS:Severe Atypical Respiratory Syndrome

输出：\_\_\_\_\_\_\_\_\_ （注：输入各行前后均无空格）

1. YES,NO,NO B、YES,NO,YES

C、NO,NO,YES D、NO,YES,YES

25、看程序写结果

#include <iostream>

using namespace std;

int lps(string seq, int i, int j)

{

int len1, len2;

if (i == j)

return 1;

if (i > j)

return 0;

if (seq[i] == seq[j])

return lps(seq, i + 1, j - 1) + 2;

len1 = lps(seq, i, j - 1);

len2 = lps(seq, i + 1, j);

if (len1 > len2)

return len1;

return len2;

}

int main()

{

string seq = "acmerandacm";

int n = seq.size();

cout << lps(seq, 0, n - 1) << endl;

return 0;

}

输出（）

A、3 B、4 C、5 D、6

26、补全程序

（格雷码，Gray Code)格雷码是对十进制数的一种二进制编码。编码顺序与相应的十进制数的大小不一致。其特点是：对于两个相邻的十进制数，对应的两个格雷码只有一个二进制位不同。另外，最大数与最小数之间也仅有一个二进制位不同，以4 位二进制数为例，编码如下：

十进制数格雷码十进制数格雷码

0 0000 8 1100

1 0001 9 1101

2 0011 10 1111

3 0010 11 1110

4 0110 12 1010

5 0111 13 1011

6 0101 14 1001

7 0100 15 1000

如果把每个二进制的位看作一个开关，则将一个数变为相邻的另一个数，只须改动一个开关。因此，格雷码广泛用于信号处理、数-模转换等领域。

下面程序的任务是：由键盘输入二进制数的位数n (n<16)，再输入一个十进制数m(0≤m<2n)，然后输出对应于m 的格雷码（共n 位，用数组gr[]存放）。

为了将程序补充完整，你必须认真分析上表的规律，特别是对格雷码固定的某一位，从哪个十进制数起，由0 变为1，或由1 变为0。

|  |
| --- |
| #include <stdio.h>  main()  {  int bound = 1, m, n, i, j, b, p, gr[15];  printf( "input n,m\n" );  scanf( "%d%d", &n, &m );  for ( i = 1; i <= n; i++ )  bound = ①;  if ( m < 0 || m >= bound )  {  printf( "Data error!\n" );  ②;  }  b = 1;  for ( i = 1; i <= n; i++ )  {  p = 0; b = b \* 2;  for ( ③; j <= m; j++ )  if ( ④ )  p = 1 - p;  gr[i] = p;  }  for ( i = n; ⑤ )  printf( "%1d", gr[i] ); /\* 在"%1d" 中出现的是数字1，不是字母l \*/  printf( "\n" );  } |

1. （）
2. bound++ B、bound\*2 C、bound\*3 D、bound/2

2、（）

A、break B、exit(1) C、return D、continue

3、（）

A、j=0 B、i=0 C、j=1 D、i=1

4、（）

A、(j%b-(b/2))=0 B、(j/b-(b/2))=0 C、(j%b-(b%2))=0 D、(j/b-(b%2))=0

5、（）

A、<1 B、<2 C、<=1 D、<=2

27、补全程序

（两元序列）试求一个整数序列中，最长的仅包含两个不同整数的连续子序列。如有多 个子序列并列最长，输出任意一个即可。例如，序列“1 1 2 3 2 3 2 3 3 1 1 1 3 1”中， 有两段满足条件的最长子序列，长度均为 7，分别用下划线和上划线标出。

#include <stdio.h>

int main()

{

const int SIZE = 100;

int n, i, j, a[SIZE], cur1, cur2, count1, count2,

ans\_length, ans\_start, ans\_end; //cur1, cur2 分别表示当前子序列中的两个不同整数 //count1, count2 分别表示 cur1, cur2 在当前子序列中出现的次数

scanf("%d", &n);

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

scanf("%d", &a[i]);

i = 1;

j = 1; //i, j 分别表示当前子序列的首尾，并保证其中至多有两个不同整数

while ((j <= n) && (a[j] == a[i]))

j++;

cur1 = a[i];

cur2 = a[j]; count1 = (1) ; //（3 分）

count2 = 1;

ans\_length = j - i + 1;

while (j < n) {

j++;

if (a[j] == cur1)

count1++;

else if (a[j] == cur2)

count2++;

else { if (a[j - 1] == (2) ) { //（3 分）

while (count2 > 0) {

if (a[i] == cur1)

count1--;

else

count2--;

i++;

}

cur2 = a[j];

count2 = 1;

}

else {

while (count1 > 0) {

if (a[i] == cur1) (3) ; //（2 分）

else (4) ; //（2 分）

i++;

} (5) ; //（3 分）

count1 = 1;

}

}

if (ans\_length < j - i + 1) {

ans\_length = j - i + 1;

ans\_start = i;

ans\_end = j;

}

}

for (i = ans\_start; i <= ans\_end; i++)

printf("%d ", a[i]);

return 0;

}

1、（）

A、j+1 B、j-1 C、j+2 D、j-2

2、（）

A、cur1 B、cur2 C、count1 D、count2

3、（）

A、count1++ B、count2++ C、count1-- D、count2--

4、（）

A、count1++ B、count2++ C、count1-- D、count2--

5、（）

A、cur1=a[j] B、cur2=a[j] C、cur1=a[i] D、cur2=a[j]