1. 对于变量定义：int a, b = 0;下列叙述中正确的是( )

a的初始值不确定，b的初始值是0。

1. 在switch语句中必须使用default False

3、switch ( k ) {

case 1: printf("one"); break;

case 2: printf("two"); break;

case 1: printf("one"); break;

default: printf("zero"); break;

} 非法：不能两个一样的case

4、switch ( k%2 ) {

default: printf("zero"); break;

case 1: printf("one");

case 1+1: printf("two");

}合法

5、n = 10;

switch ( k ) {

case n%3: printf("one");

case n%4: printf("two");

default: printf("zero");

}非法：要常量表达式

6、用户标识符就是自定义的变量 字母或下划线开头

break 语句只能用在循环体内和 switch 语句体内 True

7、int k, flag;

if (k = 0) flag = 0;

else flag = 1;

k的值是 0；flag的值是1；

8、输入一个非负整数，输出它的各位数字。例如，输入9837，输出9 8 3 7

int digit, number, pow, t\_number;

scanf ("%d", &number);

t\_number = number;pow = 1;

while (t\_number/10!=0)

{

pow = pow\*10;

t\_number = t\_number / 10;

}

while ( pow >= 1 ) {

digit = number/pow;

number = number%pow;

pow = pow/10;

printf ("%d ", digit);

}

9、输入一个正整数n（1≤n≤10），输出n×n的简易乘法表。当n=3时：

1 2 3

4 6

9

int i, j, n;

scanf("%d", &n);

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

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

printf("%4c", ' '); /\* 处理每行数字前的空格 \*/

}

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

printf("%4d", i\*j);

}

printf("\n");

}

10、按照C语言的规定，在参数传递过程中，既可以将实参的值传递给形参，也可以将形参的值传递给实参，这种参数传递是双向的。 False

11、在C语言中，可以将主调函数放在被调函数的后面，省略函数的声明 Ture

12、在C语言的函数定义中，如果不需要返回结果，就可以省略return语句 Ture

13、自动变量如果没有赋值，其值被自动赋为随机值；静态局部变量如果没有赋值，其存储单元中将是0

14、C语言中函数返回值的类型是由以下（ ）决定的。

函数定义时指定的类型（函数类型）

15、函数未被调用时，系统将不为形参分配内存单元

当形参是变量时，实参可以是变量、常量或表达式

如函数调用时，实参与形参都为变量，则这两个变量不可能占用同一内存空间

16、在C语言程序中，若对函数类型未加显式说明，则函数的隐含类型为（int）。

17、以下程序的运行结果是（3,7）。

int x = 5, y = 6;

void incxy( )

{

x++;

y++;

}

int main(void )

{

int x = 3;

incxy( );

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

return 0;

}

18、输入1234567<回车>，则程序的运行结果是（i = 123, j = 45）

int i,j;

scanf("%3d%2d",&i,&j);

printf("i = %d, j = %d\n",i,j);

19、下列程序的输出结果是（不能确定）

fun(int a, int b, int c)

{ c = a \* b; }

int main(void)

{

int c;

fun(2, 3, c);

printf(“%d\n”, c);

return 0;

}

20、表达式 !x 等价于x != 1。 False（非x）

21、表达式 ~(~2<<1)的值是5。 True（四字节）

22、表达式 (z=0, (x=2)||(z=1),z) 的值是1。 False（或运算中前一个为真则不看后一个）

23、C 语言的运算符只有单目运算符和双目运算符两种。 False

24、变量初始化的含义，就是在定义变量时对变量赋值。 True

25、若变量已正确定义，表达式 (j=3, j++) 的值是\_\_3\_。

26、C语言里，下列选项正确的赋值语句是（ABC）。

A.t += 1; B.n1 = (n2 = (n3 = 0)); C.k = i = j;（不是初始化是正确的） D.a = b + c = 1;

27、在定义数组之后，根据数组中元素的类型及个数，在内存中分配一段连续存储单元用于存放数组中的各个元素。 True

28、数组名表示该数组所分配连续内存空间中第一个单元的地址，即首地址。True

29、数组定义后，数组名的值是一个地址，可以被修改。 False

30、sizeof( )是C语言的一个函数，可以计算参量所占内存的字节数。如sizeof(int)可计算整型所占的内存字节数。 False

31、函数调用语句function((a,b),c)中含有的实参个数为（2）。

32、若下列程序中所有变量已声明和定义，则在函数fun()中所有可用变量为\_x，y\_

void fun(int x)

{

static int y;

……

return;

}

int z;（在函数fun（）引用后再定义则无效）

void main( )

{

int a,b;

fun(a);

}

数组&指针：

1. 在以下描述中，（B）是正确的。

B.static int b[10];定义了静态数组b，且10个数组元素的初值都为0。

C.int fib[45];定义了数组fib，且45个数组元素的值都为0。

Auto类型随机值

D.static int week[7] = {1, 2, 3};定义了静态数组week，并对数组 week 的前3个元素week[0]～week[2]赋初值，week[3]～week[6]值都是不确定的。

2. 字符'\0'的ASCII码值为0。

3. 要通过函数调用来改变主调函数中某个变量的值，可以把指针作为函数的参数。

4. 下述对C语言字符数组的描述中错误的是（C）。

A.字符数组可以存放字符串

B.字符数组中的字符串可以整体输入、输出

C.可以在赋值语句中通过赋值运算符"="对字符数组整体赋值

D.不可以用关系运算符对字符数组中的字符串进行比较 但是比较单个字符是可以的

5. 以下能正确定义数组并正确赋初值的语句是（D）。

A.int N=5, b[N][N];变量不可以作为初始化的下标 但是引用时可以

而且scanf后的变量初始化也可以用 比如int n；scanf（“%d”，&n）；可以int a[n];

B.int a[1][2]={{1}, {3}};

C.int c[2][ ]={{1, 2}, {3, 4}};

D.int d[3][2]={{1, 2}, {34}};

6. 对于如下变量定义及初始化，与m = n等价的表达式是（B）。

int m, n = 5, \*p = &m;

A. m = \*p

B.\*p = n

C. m = &n

D.\*p = m

7. 以下选项中，对基本类型相同的指针变量不能进行运算的运算符是 (A)。

A.+ B.- C.= D.==

A因为两个地址相加无意义也可能越界，所以规定不允许指针相加

B 可以求出两个数据元素储存位置之间的相隔同数据类型的元素个数

D判断两指针所指向的是否是同一数据元素

8. 指针可以用关系运算符进行比较，如 ==、< 和 >

9. 若有以下说明，且0<=i<10，则对数组元素的错误引用是（）。

int a[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, \*p = a, i;

A.\*(a+i)

B.a[p-a+i]

C.p+i 对字符串的引用

D.\*(&a[i])

10.下面程序的输出结果是( )。

#include <stdio.h>

void fun (int \*x, int y)

{

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

\*x=3;

y=4;

}

int main(void)

{

int x = 1, y = 2;

fun(&y, x);

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

return 0;

}

A.1234 B.1221 C.2131 D.2113

11. 下面程序段的运行结果是（）。

char s[ ] = "language", \*p = s;

while( \*p++ != 'u') {

printf("%c", \*p – 'a' + 'A');

}

A.LANGUAGE

B.ANGU

C.LANGU

D.LANG

12. 写出下列程序的执行结果 4 6

#include <stdio.h>

int main(void)

{

int a[10], b[10], \*pa, \*pb, i;

pa = a; pb = b;

for( i = 0; i < 3; i++, pa++, pb++){

\*pa = i; \*pb = 2\*i;

}

pa = &a[0];

pb = &b[0];

for ( i = 0; i < 3; i++, pa++, pb++){

\*pa = \*pa + i; \*pb = \*pb + i;

}//结束时指向a[3]的地址

printf("%d %d", \*--pa, \*--pb); /\* 输出的数字之间有一个空格 \*/

return 0;

}

13. 在字符串 str 中找出最大的字符，将在该字符前的所有字符往后顺序移动一位，再把最大字符放在字符串的第一个位置上。如"knowledge"变成"wknoledge"。

#include <stdio.h>

int main()

{

char max, str[80], \*p, \*q；

p = str;

gets(p);

max = \*p;

p++;

while ( \*p != '\0' ){

if ( max < \*p ){

max = \*p;

q=p; //记录最大字符及之后的字符串

}

p++;

}

p = q;//p的首地址是最大字符

while (p!=str){

\*p = \*(p-1);//向左移动一位

p--;

}

\*p = max;//此时的p是首字符

puts(p);

return 0;

}

14. 对于如下说明，语法和语义都正确的赋值是\_\_\_\_\_。

int c, \*s, a[]={1, 3, 5};

1. c=\*s； B. s[0]=a[0]; C.s=&a[1]; D.c=a;

A、语法上正确（可能引起编译警告）。若非静态变量或全局变量，否则s没有被初始化，是随机值，\*s的值无法确定，即没有确定的语义；若是静态变量或全局变量，s被初始化为0，对地址0的解引用操作没有确定的语义。

B、错误。理由同上，对随机地址的操作不具有确定的语义（s[0]等价于\*s）。另外对随机地址单元的写操作（赋值）引起的后果不可预料。

C、正确，将a[1]的地址赋值给指针s。

D、语法错误，c（int）和a（const int\*）的类型不匹配，会引起编译错误。

15. 字符串常量在内存中的存放位置由系统自动安排。字符串常量实质上是一个指向该字符串首字符的指针常量。T

16.‘abc\0’不是字符串

17. 判断回文：先消除输入字符串s的前后空格，再判断其是否为“回文”（即字符串正读和倒读都是一样的），若是则输出YES，否则输出NO。

#include <stdio.h>

#include <string.h>

int main(void)

{

char ch, s[80], \*p, \*q;

int i, j, n;

gets(s);

p = s;

while ( \*p == ' '){ p++; }

n = strlen(s);

q = s+n-1;

while ( \*q == ' '){ q--; }

while ( p<q && \*p == \*q){

p++;q--;

}

if ( p<q ){

printf("NO\n");

}else{

printf("YES\n");

}

return 0;

}

18. 函数strcmp从头至尾顺序地将其对应字符比较，遇到两个字符不等时，两个字符相减得到一个int型值 T

19. char s1[80],s2[80];s1=s2;这样赋值是错误的。T

20. 语句int \*p;和printf("%d",\*p);中的\*p含义相同。F

int \*p这里星号只是说明 变量 p 是一个指针。

printf(“%d”, \*p)这里的 \*p 是指 指针变量 p 所指向内存地址的内容；

21. 假设scanf语句执行时输入ABCDE<回车>，能使puts(s)语句正确输出ABCDE字符串的程序段是\_\_。

A.char s[5]={"ABCDE"}; puts(s); A、B都装不下‘\0’

B.char s[5]={'A', 'B', 'C', 'D', 'E'}; puts(s);

C.char \*s; scanf("%s", s); puts(s);

D.char \*s; s="ABCDE"; puts(s);

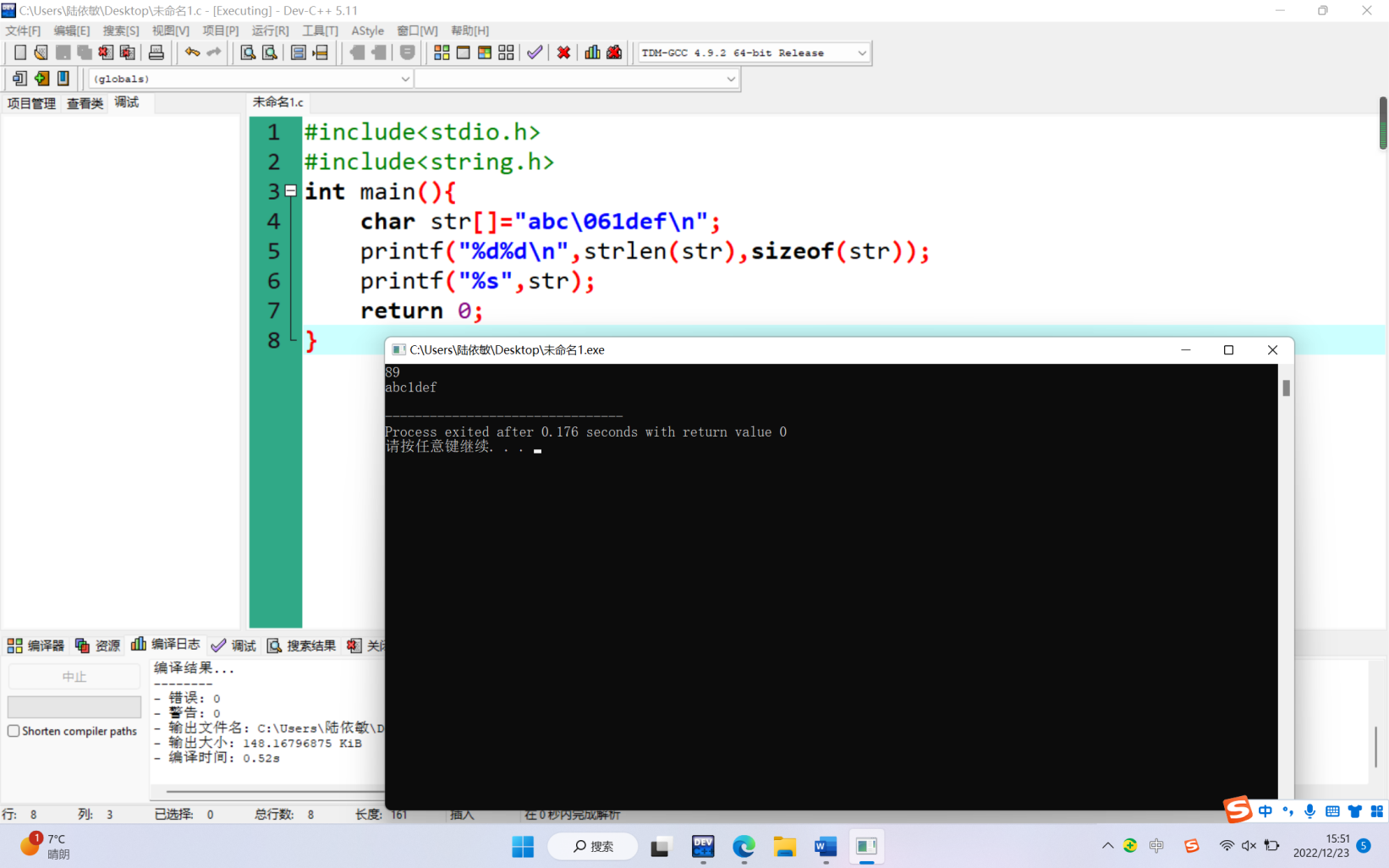
22. 指向数组元素的指针只可指向数组的首元素。F

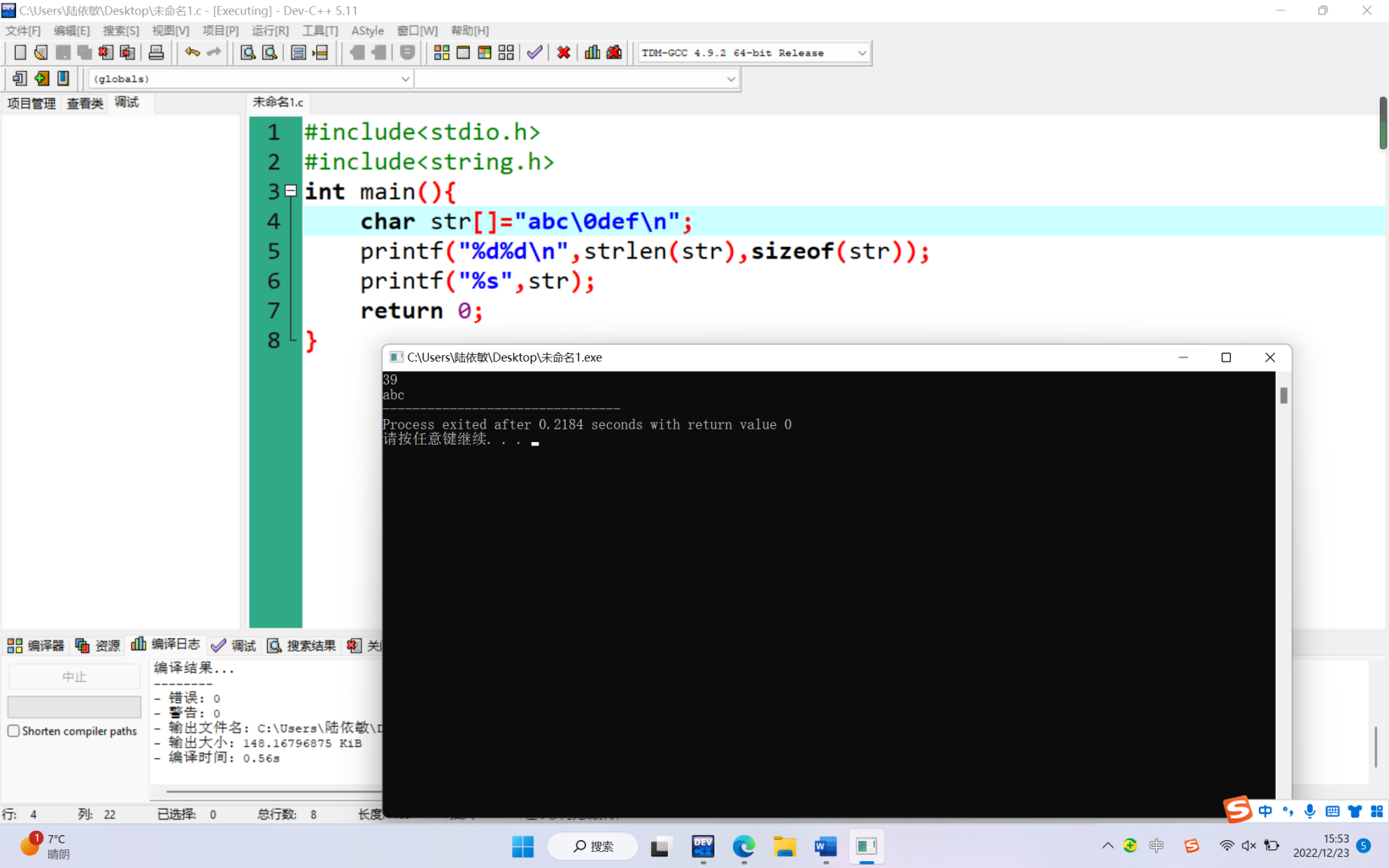
！23. 对于定义 char str[] = "abc\000def\n"（注：其中0为数字零），求字符串str的长度len和数组str的大小size，len和size分别是：3 9

sizeof会包括结尾的‘\0’的；\000计算大小时是转义字符，计算长度时\000直接结束

对于定义 char str[] = "abc\045def\n" strlen和size为8 9

计算strlen并非遇到\0就结束 实际上以转义字符保存





24. 下列程序段中哪一个完全正确？

A.int \*p; scanf("%d", &p); A、B未定义指针的地址

B.int \*p; scanf("%d", p);

C.int k, \*p=&k; scanf("%d", p);

D.int k, \*p; \*p= &k; scanf("%d", p); 应该为p=&k

25. 下列赋值语句或初始化语句中，\_\_\_\_是错误的。

A.char s[ ]="hello";

B.char s[10]; s="hello";

C.char \*p="hello";

D.char \*p; p="hello";

26. 运行结果bcdBCD

char s[20]= "abcd" ;

char \*sp = s ;

puts(strcat(sp+1, "ABCD"+1)) ;

27. 写出下面程序的运行结果。sum=25

#include <stdio.h>

int main()

{

int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

int \*p = a, sum = 0;

for ( ; p < a + 10; p++)

{

sum += \*p++;

}

printf("sum=%d", sum);

return 0;

}

28. 在主函数中输入一个字符串和一个字符，调用match函数，如果该字符在字符串中，就从该字符首次出现的位置开始输出字符串中的字符。如果未找到，输出“Not Found”。

int main(void )

{

char ch, str[80], \*p = NULL;

scanf("%s", str);

getchar();

while((ch = getchar())!='\n')

{

if((p = match(str, ch)) != NULL)

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

else

printf("Not Found\n");

}

return 0;

}

char match(char \*s,char ch) /\* 字符定位函数定义：match函数\*/

{

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

if (\*s == ch) return s;

else s++;

}

return NULL;

}

29. 输出结果Happy

# include <stdio.h>

void main(){

int j,n;

char \*pa,\*s;

pa="Happy";

for(s=pa,n=0;\*s!='\0';++s,++n);

for(s=pa,j=0;j<n;j++){

putchar(\*s++);

}

}

30. 输出结果 035

#include<stdio.h>

void main(){

int m;

char ss[6]="12345",\*s;

s=&ss;

for(m=10;m!=0;m--);

printf("%d",m);

while(\*s++) putchar(\*++s);

}

1. char s[10] = "abcd"; printf("%d\n", sizeof(s));

输出10

结构&文件：

1.结构是变量的集合，可以按照对同类型变量的操作方法单独使用其成员变量。T

2.在定义嵌套的结构类型时，必须先定义成员的结构类型，再定义主结构类型。T

3.在对结构变量操作时，.被称为成员或者分量运算符。->被称为指向运算符。

4.设有如下定义，则错误的输入语句是（ ）。

struct s{

char name[10];

int age;

char sex;

} std[3], \*p = std;

A. scanf("%d", &(\*p).age);

B. scanf("%d", p->&age);

C. scanf("%c", &std[0].sex);

D. scanf("%c", &(p->sex));

E. scanf("%s",&std.name);

5. C语言中结构类型变量在程序执行期间所有成员一直驻留在内存中

6. 对于以下结构定义，++p->str中的++加在（A）。(\*p)->str++中的++加在（D）。

struct {

int len;

char \*str;

} \*p;

A.指针str上

B.指针p上

C.str指向的内容上

D.语法错误

7. 以下程序的输出结果是（51,60,21 ）。

struct stu{

int x;

int \*y;

} \*p;

int dt[4] = {10, 20, 30, 40};

struct stu a[4] = {50, &dt[0], 60, &dt[1], 70, &dt[2], 80, &dt[3]};

int main( )

{

p=a;

printf("%d,", ++p->x); 地址->成员

printf("%d,", (++p)->x);

printf("%d", ++(\*p->y)); //因为成员y是地址,所以要得到数据则要用\*

return 0;

}

期末真题复习

1.写出下列程序段的输出结果。 1，0.5

float x1, x2;

x1=3/2;

x2=x1/2;

printf("%d, %.1f", (int)x1, x2) 不加int则是1.0

2.表达式 (7<<1>>2^2 ) 的值是 1

3.输入12345#后，写出下列程序的输出结果。135

void main(){

char c;

for(c=getchar(); getchar()!=’#’; c=getchar())

putchar(c);

}

4.语言源程序经过编译连接最后生成的文件,其后缀名 是\_.exe\_\_

5.对于以下的变量定义，表达式\_\_\_B\_\_\_不符合C语言语法。

struct node {

int len;

char \*pk;

} x = {2, "right"}, \*p = &x;

A、p->pk B、\*p.pk C、\*p->px D、\*x->px

6.printf("%o\n", 010<<1^1); 输出21(八位二进制

7.下列程序段的输出是\_\_5，25\_\_\_\_\_。

#define f(a, b, x) a\*x+b

printf("%d, %d\n", f(3,2,1), f(6, 5, f(3, 2, 1))

8.typedef定义一个文件指针FP：typedef FILE \*FP

9.对于下面变量定义，赋值使用方法不正确的是\_\_B\_\_\_。

struct Student {

long num;

char name[20];

} st1, st2={101,”Tom”}, \*p=&st1;

1. st1=st2 B．p->name=st2.name C．p->num=st2.num D．\*p=st2

例如char a[10] = "123"; /\*正确，在定义的时候初始化\*/

char a[10]; a = "123"; /\*错误，不能用“=”直接为字符数组赋值\*/

strcpy(a, "123"); /\*正确

不能直接字符串赋值字符串 但可以整个struct 赋值

1. 结构和变量定义如下，则对于“\*p->str++”表达式语句，下面叙述正确的是

struct {

int no; char \*str;

} a={1,”abc”}, \*p=&a;

1. \* 作用在指针p上
2. ++ 作用在指针str上
3. ++ 作用在str所指的内容上
4. 表达式语句有错

\*(（p->str）++)

10.1表达式-k++等价于\_\_\_\_\_\_。A．-(k++) B．(-k)++ C．-(k+1) D．-(++k)

后缀运算比前缀运算的优先级高

1. 用typedef 定义新类型名PNODE，使得PNODE表示含有10个struct Node结构指针型元素的数组 为 typedef struct Node\* PNODE[10]
2. 下面代码段输出结果是\_\_-1\_\_\_\_\_.

int s=-1; printf(“%d”,(unsigned int)s)

因为是％d 所以当成有符号

若要改为无符号％u

1. 字符'0'不等于'\0' 0等于'\0' 计算字符个数包括‘\0’ 计算字符串长度不包括计算sizeof包括
2. 下列程序的输出是\_a-c56c-i\_.

#include <stdio.h>

char s1[80]="abc56cdefghi\0jklm";

int main(void){

int j=0, k=0, c;

char ch, bf=0, s2[80];

while(ch=s1[j++]) {

if(ch - bf != 1) {

s2[k++] = ch; c=0;

}

else if(s1[j] - ch != 1) {

if (c) s2[k++] = '-';

s2[k++] = ch;

}else{

c++; bf = ch;

}

}

s2[k]='\0';

printf("%s", s2);

}

1. 程序运行时输出结果为\_\_4#2#1\_\_\_\_\_.

#include <stdio.h>

int findStr (char \*str, char \*substr){

int n=0;

char \*p , \*r;

while ( \*str ){

p=str;r=substr;

while(\*r){

if(\*r==\*p) {r++;p++;}

else break;

}

if(\*r=='\0') n++;

str++;

}

return n;

}

void main(){

char s1[80]="AAscA\\101\101scA\nAscA00AscA\0AscAs",

char s2[10]="Asc";

printf("%d#",findStr(s1,s2));

printf("%d#", findStr(s1+10,s2));

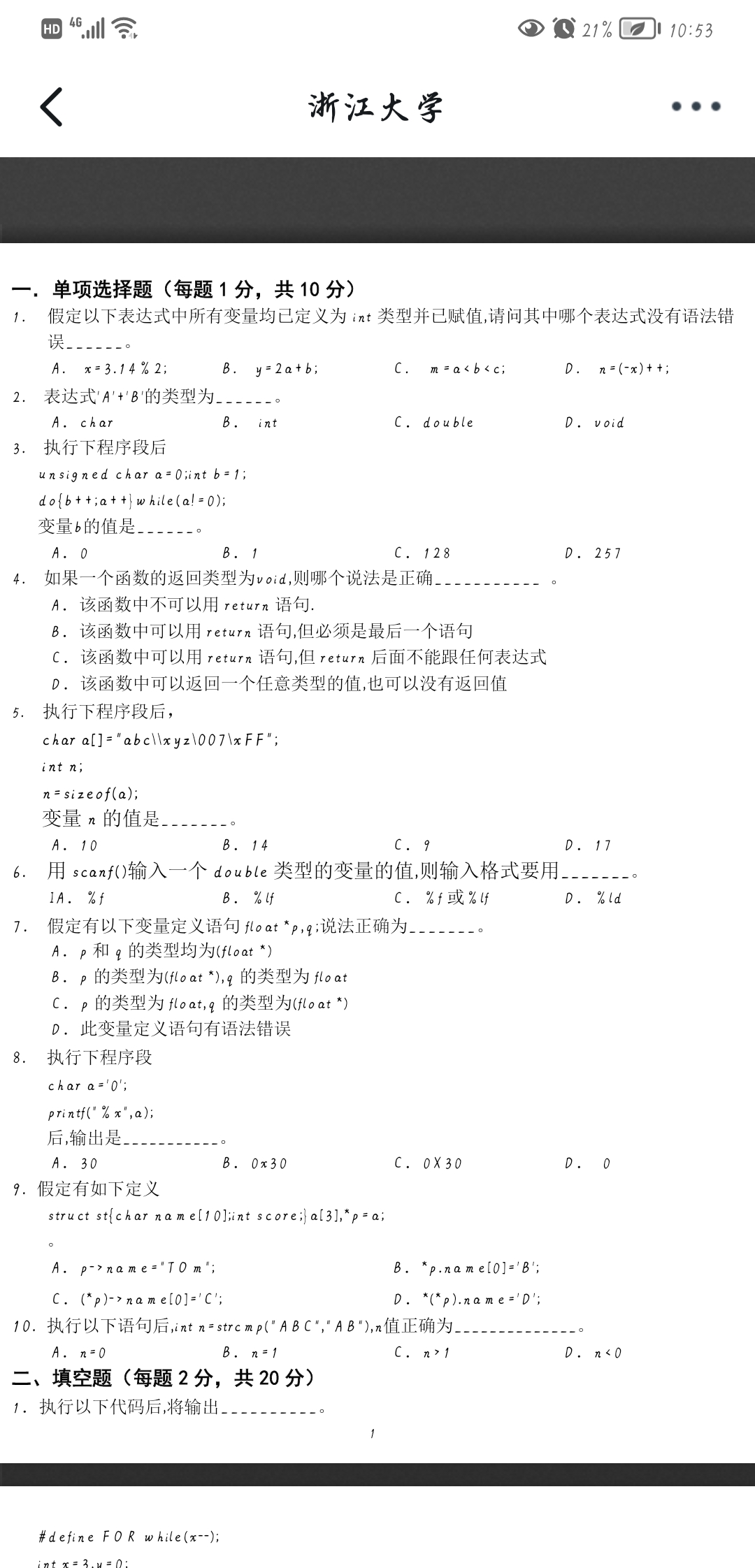
printf("%d#\n", findStr(s1+25,s2));

}

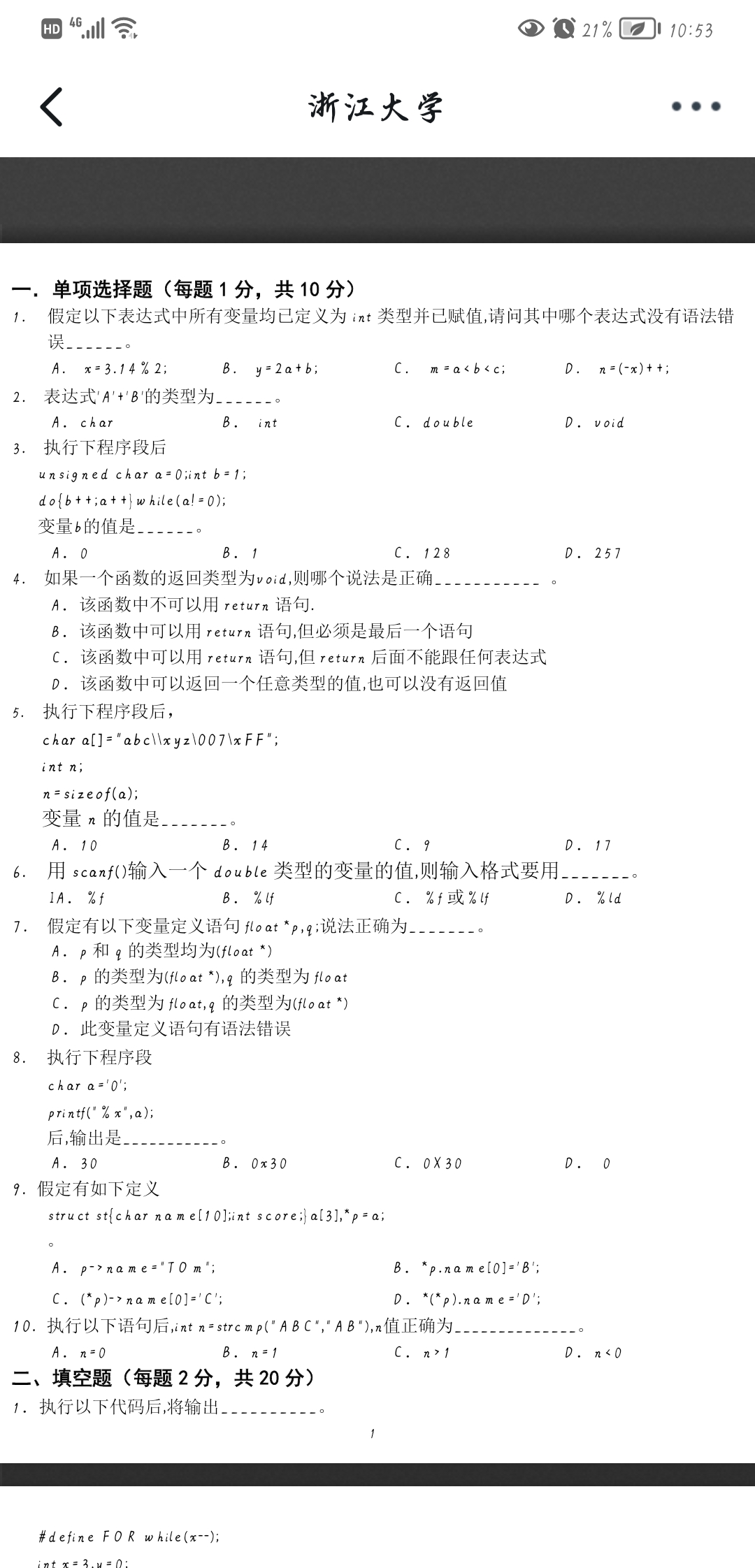
15.整型变量，表达式 (x=10.0/4.0) 的值是\_\_\_D\_\_ 。A．2.5 B．2.0 C．3 D．2

16.对变量定义为int a[2]={1，3}，\*p=&a[0]+1；则\*p=3

17.Typedef定义一个十个元素的整型数组类型arr，再用arr定义一个十个元素的整型数组a：typedef int arr[10]；arr a；

1. C

（注意D）

1. B
2. D
3. C
4. D

A改变

了地址

B少括号

C多括号

1. C
2. 定义unsigned long int a;且变量a已赋值,试写出判断a的二进制最高位即最左边那位与1相等的表达式 (a>>31)==1
3. 假定运行以下程序时输 入3+4\*5-3/2=则程序的输出结果是\_\_\_19（倒着算）#include <stdio.h>

int calc(void){

int x;char p;

scanf("%d %c",&x,&p);

if( p=='=') return x;

switch(p){

case '+':return x+calc();

case '-':return x-calc();

case '\*':return x\*calc();

case '/':return x/calc();

}

}

void main(){

printf("%d",calc());

}

17.不正确的赋值或赋初值的方式是\_\_\_\_\_\_。

A．char str[ ]="string";

B．char str[10]; str="string";

C．char \*p="string";

D．char \*p; p="string”;

1. 从键盘输入一个字符ch，输出该字符在文本文件input.txt的每一行中出现的次数。 ( 必须调用第1题定义的函数 ff 统计ch的出现次数 )。例如：如果文件input.txt中存放了下列数据：

every

121

I am a student.

运行程序，并输入'e'后，输出

2

0

1

#include <stdio.h>

#include <stdlib.h>

void main(){

char ch,c,s[80];

int k;

FILE \*fp;

if((fp=fopen("input.txt","r"))==NULL) {printf("error\n"); exit(0);}

ch=getchar();k=0;

while(1){

c=fgetc(fp);

if(c=='\n'||c==EOF){

s[k]='\0';k=0;printf("%d\n",ff(s,ch));

if(c==EOF) break;

}

else s[k++]=c;

}

fclose(fp);

}

19.

输入3 9 8 6 5 3 7 1 2 4 2 1 2 3 4<回车>后，下列程序的输出结果是\_\_\_\_a[0][0]=9#a[1][2]=7#a[1][1]=4\_\_\_\_。

# include <stdio.h>

void main(){

int flag,i,j,k,col,n,ri,a[6][6];

for(ri=1;ri<=2;ri++){

scanf(“%d”,&n);

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

for(j=0;j<n;j++) scanf(“%d”,&a[i][j]);

}

flag=0;

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

col=0;

for(j=0;j<n;j++) if(a[i][col]<a[i][j])col=j;

for(k=0;k<n;k++)if(a[i][col]<a[k][col])break;

}

if(k>=n){printf(“a[%d][%d]=%d#”,i,col,a[i][col]); flag=1;}

}

if(!flag)printf(“NO#”);

}

20.

输入30位学生的姓名和成绩，输出成绩最高的学生的姓名及其成绩。

#include <stdio.h>

struct student{ char name[10]; int score;};

void main(){

int k; struct student s[30],\*ptr;

struct student \*f(struct student \*stud,int n);

for(k=0;k<30;k++) scanf(“%s%d”,s[k].name,&s[k].score);

ptr=f(s,30);

printf(“%s %d\n”,ptr->name,ptr->score) ;

}

struct student \*f(struct student \*stud,int n){

struct student \*pmax,\*p;

for(p=stud;p<stud+n;p++){

pmax=stud;

if(p->score>pmax->score) pmax=p;

}

retrun pmax;

1. 有函数原型为void f(int, int \*)，主函数中有变量定义：int a=2, \*p=&a; 则下列函数调用正确的是\_\_\_\_.
2. f(a, &p). B．f(\*p, &a). C．f(a, \*p). D．f(\*p, a)

22.表达式-10>>2+3的值为\_\_\_\_\_\_.（假设是算术移位）

虽然后缀运算符先于前缀运算符 但是单目运算符更加先于双目

整数用补码存放-10原码10001010，补码就是11110110

右移五个并补上符号位的数，即11111111，变回原码10000001，即-1

1. 下列程序段执行后，z的值是 \_\_4\_\_\_\_.

static struct { int x, y[3];} a[3] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}}, \*p=a+3;

int z;z=((int \*)(p-2))[ -1]

24.运行以下程序后，将输出\_\_\_10#20#457#124#31#\_\_\_\_\_

#include <stdio.h>

int g=123;

int \* f(int a, int b){

static int g;a/=2; b/=2; g += a+b;

return &g;

}

int main(){

int a=10, b=20, \*p=&g;（存的是全局变量的g的地址

int g=456, y;

(\*p)++; g++;

(\*f(a, b))++;

y = \*f(a,b);printf("%d#%d#%d#%d#%d#", a,b,g,\*p,y);

}

25输入一个英文句子保存到字符数组a中(句子中各个单词用一个或多个空格分隔,第一个单词前面可以有空格,最后一个单词后面也可以有空格),接下去对句子中构成一个单词的各个字符进行逆向变换,逆向后的单词重新存放到数组a中原先的位置,最后输出新的句子.

例如,输入句子:pots & pans I saw a reed将输出stop & snap I was a deer

注意:只要是非空格字符均构成一个单词.例如这里的&也算一个单词;

#include <stdio.h>;

void main(){

char a[81],c; int i,j,n; gets(a);

i=0;

while(a[i]!=' \0') {

while(a[i]==' ') i++;

if(a[i]==‘\0’) break;

n=0;

while(a[i+n]!=‘\0’&&a[i+n]!= ‘ ’) n++;

for(j=0;j<n/2;j++){

c=a[i+j]; a[i+j]= a[i+n-1-j] ; a[i+n-1-j]=c ;

}

puts(a);

i+=c;

}

}

26以下程序段的输出是\_\_16#80\_\_\_.

int i; char s[80]={"apple\0grape\0pear\0coco\0"};

for(i=0; i<2; i++) s[strlen(s)] = '\n';

printf("%d#%d\n",strlen(s),sizeof(s) )

**C大程理论**

1、定义带参数的宏“#define JH(a,b,t) t = a; a = b; b = t”，对两个参数a、b的值进行交换，下列表述中正确的是（）。

A.不定义参数a和b将导致编译错误

B.不定义参数a、b、t将导致编译错误

C.不定义参数t将导致运行错误

D.不需要定义参数a、b、t类型

D

2、执行下面程序，正确的输出是（）

int x = 5, y = 7;

void swap ( )

{

int z ;

z = x ; x = y ; y = z ;

}

int main(void)

{

int x = 3, y = 8;

swap ( ) ;

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

return 0 ;

}

3,8

3、下面说法中正确的是（）。

A.若全局变量仅在单个C文件中访问，则可以将这个变量修改为静态全局变量，以降低模块间的耦合度

B.若全局变量仅由单个函数访问，则可以将这个变量改为该函数的静态局部变量，以降低模块间的耦合度

C.设计和使用访问动态全局变量、静态全局变量、静态局部变量的函数时，需要考虑变量生命周期问题

D.静态全局变量使用过多，可那会导致动态存储区（堆栈）溢出

A

解析C选项生命周期都是函数运行期间，其中静态局部变量是函数退出时存在但不能使用

4、函数递归调用自身时，每次调用都会得到一个与以前的变量集合不同的新的变量集合。

T

1. 递归的执行速度并不快，但递归代码比较紧凑，并且比相应的非递归代码更易于编写与理解。递归并不节省存储器的开销，因为递归调用过程中必须在某个地方维护一个存储处理值的栈。

T

6、以下描述错误的是( )。

A.函数调用可以出现在执行语句中

B.函数调用可以出现在一个表达式中

C.函数调用可以作为一个函数的实参

D.函数调用可以作为一个函数的形参

D

7、以下函数调用中，实参的个数为（ ）。

func((exp1,exp2),(exp3,exp4,exp5));

2

8、以下正确的叙述是（ ）。

A.宏替换不占用运行时间，只占用编译时间

B.在程序的一行中可以出现多个有效的预处理命令行

C.使用带参数宏定义时，参数的类型应与宏定义时一致

D.宏定义不能出现在函数内部

A

9、在一个C源程序文件中，若要定义一个只允许本源文件中所有函数使用的全局变量，则该变量需要使用的存储类别是。

Static

10、C语言的编译预处理功能包括（）。

A.文件包含

B.宏定义

C.条件编译

ABC

11、下列程序的输出结果为

#include <stdio.h>

int fib(int g)

{

switch(g){

case 0: return 0;

case 1:

case 2: return 2;

}

printf("g=%d,", g); /\* 中间没有空格 \*/

return fib(g-1) + fib(g-2);

}

int main(void)

{

int k;

k = fib(4);

printf("k=%d\n", k); /\* 中间没有空格 \*/

return 0;

}

g=4,g=3,k=6

1. 如果函数的返回类型是指针，则可以返回函数内部任意变量的地址。

F

1. 如果函数的返回类型是指针，则可以返回0。

T

1. 语句 int \*p[5]; 定义了一个指针数组p，用于指向一个有5个元素的数组。

F

int a[10];  [数组](https://so.csdn.net/so/search?q=%E6%95%B0%E7%BB%84&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_42544051/article/details/_blank)

int \*p[5];  [指针数组](https://so.csdn.net/so/search?q=%E6%8C%87%E9%92%88%E6%95%B0%E7%BB%84&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_42544051/article/details/_blank)-----定义了指针数组 p，数组含5个元素，每个元素都是int 的指针  
int (\*p)[5];数组指针-----可以看成是\*（P[5]）早期程序设计未来简化书写就简写成（\*p）[5]------>这样就理解了只有一个指针p指向有四个元素的数组

1. 定义一个指针数组并初始化赋值若干个字符串常量，则指针数组并不存放这些字符串，而仅仅指向各个字符串。

T

1. 对于 int 型的二级指针变量，它既可以指向同类型的一级指针变量，也可以指向同类型的普通（非指针）变量。

F

17、如果有函数char \*func(char \*p, char ch)，则下面说法错误的是（ ）。

A.函数返回一个字符指针

B.可以通过语句return NULL;返回函数结果

C.可以通过语句return -1;返回函数结果

D.可以通过语句return p;返回函数结果

C

18、对于以下程序，能够正确表示二维数组 t 的元素地址的表达式是（ ）。

int main(void)

{

int k, t[3][2], \*pt[3];

for ( k = 0; k < 3; k++) {

pt[k] = t[k];

}

return 0;

}

A.&t[3][2]

B.\*pt[0]

C.\*(pt+1)

D.&pt[2]

C

解析

A溢出B取出t[0][0]的内容D已经是地址不用再取地址 C\*(pt+1)就是pt[1]==&t[1][0]

pt是指向pt数组的指针，通过pt+1，就是指向数组pt里面第二个元素的指针，第二个元素即pt[1]=t[1]就是指向数组t[1][0]的指针，这样\*(pt+1)就是取的是pt数组第一个元素的内容，保存的是t[1][0]的地址

19、下面程序的运行结果是（）。

#include<stdio.h>

int main(void)

{

int x[5] = { 2, 4, 6, 8, 10 }, \*p, \*\*pp;

p = x;

pp = &p;

printf("%d ", \*(p++)); /\* 数字后有一个空格 \*/

printf("%d\n", \*\*pp);

return 0;

}

2 4

20、对于以下变量定义，正确的赋值是（）。

int \*p[3], a[3];

A.p = a

B.\*p = a[0]

C.p = &a[0]

D.p[0] = &a[0]

D

解析：int \*p[3]  是一个指针数组，p是数组名，数组元素类型是 int \*。即表示一个一维数组中存放3个指针变量，分别是p[0]、p[1]、p[2]

int a[3] 是一个数组，a是数组名，数组元素类型是 int 。即表示一个一维数组中存放3个int类型的元素，分别是a[0]、a[1]、a[2]

不能给一个数组名（p）赋值，所以A和C ×。

\*p等价于\*(p+0)等价于p[0]，而p[0]中存放的是指针变量（int \*），a[0]是int型。B×

21、以下正确的说明语句是( )。

A.int \*b[ ] = {1, 3, 5, 7, 9};

B.int a[5], \*num[5] = {&a[0], &a[1], &a[2], &a[3], &a[4]};

C.int a[ ] = {1, 3, 5, 7, 9};

int \*num[5] = {a[0], a[1], a[2], a[3], a[4]};

D.int a[3][4], (\*num)[4];

num[1] = &a[1][3];

B(注意D num=&a就指向整个数组a（虽然在值上也是首元素的地址），num一次移动一个整体四个元素

22、设有以下程序段，若 k 为 int 型变量且 0≤k＜4，则对字符串的不正确引用是（ ）。

char str[4][10] = {"first", "secone", "third", "fourh"}, \*strp[4];

int i;

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

strp[i] = str[i];

}

A.strp

B.str[k]

C.strp[k]

D.\*strp

A

23、若有定义 double \*p;，使p指向一个双精度型的动态存储单元的语句为（）。

A.p = malloc(sizeof(double));

B.p = (double \*)malloc(sizeof(\*p));

C.p = malloc(8);

D.p = (double \*)malloc(sizeof(double));

BD

24、执行以下程序，第1行输出？最后1行输出？

#include<stdio.h>

#include<string.h>

int main(void)

{

int i;

const char \*color[2] = {"blue", "green"};

for(i = 0; i < 2; i++) {

printf("%s %c\n", color[i], \*color[i]); /\* 输出数据之间间隔一个空格 \*/

}

for(i = 0; i < 2; i++) {

printf("%s %c\n", color[i]+1, \*color[i]+1); /\* 输出数据之间间隔一个空格 \*/

}

return 0;

}

blue b； reen h

执行以下程序，第2行输出？第3行输出？

#include<stdio.h>

int main(void)

{

int i, flag = 0;

char ch;

const char \*color[2] = {"blue", "green"};

const char \*\*pc;

pc = color;

for(i = 0; i < 2; i++) {

printf("%s %c\n", \*(pc+i), \*\*(pc+i)); /\* 输出数据之间间隔一个空格 \*/

}

for(i = 0; i < 2; i++) {

printf("%s %c\n", \*(pc+i)+1, \*\*(pc+i)+1); /\* 输出数据之间间隔一个空格 \*/

}

return 0;

}

green g； lue c

25、第1行输出？最后1行输出？

#include<stdio.h>

int main(void)

{

int a, b, t;

int \*pa = &a, \*pb = &b, \*pt;

int \*\*ppa = &pa, \*\*ppb = &pb, \*\*ppt;

a = 10, b = 20;

printf("%d %d %d %d %d %d\n", a, b, \*pa, \*pb, \*\*ppa, \*\*ppb);

ppt = ppb; ppb = ppa; ppa = ppt;

printf("%d %d %d %d %d %d\n", a, b, \*pa, \*pb, \*\*ppa, \*\*ppb);

pt = pb; pb = pa; pa = pt;

printf("%d %d %d %d %d %d\n", a, b, \*pa, \*pb, \*\*ppa, \*\*ppb);

t = b; b = a; a = t;

printf("%d %d %d %d %d %d\n", a, b, \*pa, \*pb, \*\*ppa, \*\*ppb);

return 0;

}

10 20 10 20 10 20； 20 10 20 10 20 10

1. 写出以下程序段的运行结果。

const char \*st[] = {"Hello", "world", "!"}, \*\*p = st;

p++;

printf("%s-%c\n", \*p, \*\*p);

(\*p)++;

printf("%s-%c-%c\n", \*p, \*\*p, (\*\*p)+1);

第1行输出？第2行输出？

world-w ； orld-o-p

27、一个单词表存放了五个表示颜色的英文单词，输入一个字母，在单词表中查找并输出所有以此字母开头的单词，若没有找到，输出Not Found。要求用指针数组实现。

/\* 单词索引（用指针数组实现）\*/

#include<stdio.h>

int main(void)

{

int i, flag = 0;

char ch;

const char \*p[5]= {"red", "blue", "yellow", "green", "black" };

ch = getchar();

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

if(\*p[i]==ch){

flag = 1;

puts(p[i]);

}

}

if(flag == 0) {

printf("Not Found\n");

}

return 0;

}

或者

int main(void)

{

int i, flag = 0;

char ch;

const char \*color[5] = {"red", "blue", "yellow", "green", "black" };

const char \*\*pc;

pc = color;

ch = getchar();

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

if(\*\*(pc + i) == ch){

flag = 1;

puts( \*(pc + i) );

}

}

if(flag == 0) {

printf("Not Found\n");

}

return 0;

}

1. 输入10个成绩，查找最高分并输出

#include <stdio.h>

int \*GetMax(int score[ ], int n);

int main(void)

{

int i, score[10], \*p;

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

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

}

p = GetMax(score,10)；

printf("%d\n", \*p);

return 0;

}

int \*GetMax(int score[ ], int n)

{

int i, max, pos = 0;

max = score[0] ;

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

if(score[i] > max){

max = score[i];

pos = i ;

}

}

return score+pos;

}

28、以下程序可以逐行输出由language数组元素所指向的5个字符串。

#include<stdio.h>

int main(void)

{

const char \*language[] = {"BASIC", "FORTRAN", "PROLOG", "JAVA", "C++" } ;

const char \*\*q;

int k;

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

q=&language[k];

printf ("%s\n", \*q);

}

return 0;

}

29、在单向链表中，头指针中存放的是头结点的内容。

用链表代替数组进行数据操作时，查询更加方便。

F 头指针存放地址 查询不便 但是更改方便

1. 单向链表中的每个结点都需要动态分配内存空间。

T

31、以下程序段的功能是：输入一行字符，按输入的逆序建立一个链表。

char c;

struct node{

char info;

struct node \*link;

} \*top, \*p;

top = NULL;

while ( (c = getchar() ) != '\n'){

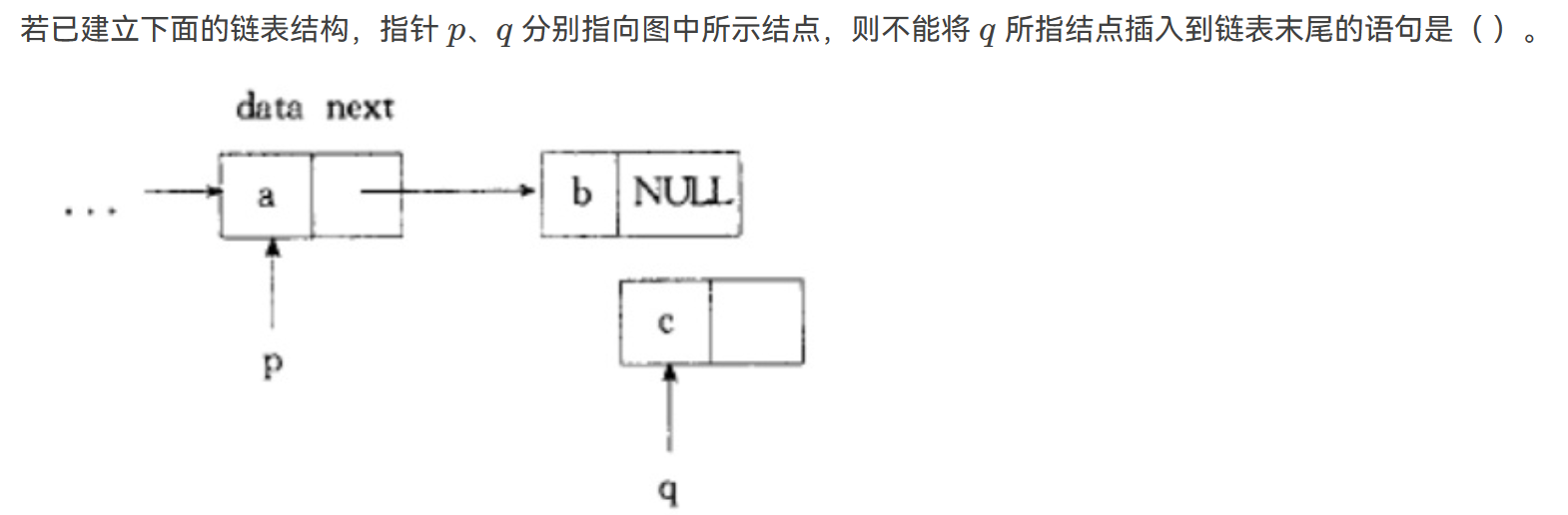
p = (struct node\*) malloc(sizeof(struct node));

p->info = c;

p->link = top ;

top = p;

}

32、下面的链表结构指针 p、q 分别指向图中结点，不能将 q 结点插入到链表尾的语句是

A.q->next = NULL;

p = p->next;

p->next = q;

B.p = p->next;

q->next = p->next;

p->next = q;

C.p = p->next;

q->next = p;

p->next = q;

D.p = (\*p).next;

(\*q).next = (\*p).next;

(\*p).next = q;

C