

专题一参考答案

基础篇

1、【正解】C

【解析】C 中合法关键字的写法应该是：case

2、【正解】×

【解析】scanf（）输入项必须是地址量，正确为 scanf("%d,%f",&a,&b);

3、【正解】A

【解析】合法的标识符由数字，字母和下划线组成，由下划线或字母开头，return 为 C 语言中的关键字。

4、【正解】B

【解析】int 为 C 语言中的关键字，不能作为用户定义的标识符，因为 C 语言区分大小写，所以 STatic 不与关键字 static 重复，可以作为标识符。

5、【正解】A

【解析】字符型变量可以与整型变量相互转化，此题中 c=65，d=66，根据 ASCII 码表可知，c 对应'A',d 对应'B'。

6、【正解】A

【解析】输入数据时的格式必须和 scanf（）函数内的格式对应。

提高篇

1、【正解】C

【解析】class 为 C++中的关键字，for 也为关键字。用户标识符必须以字母或下划线开头。

2、【正解】B

【解析】A 中前后输入项个数不对，C 中输入格式错误，D 中 scanf（）函数输入数据时，不能规定精度。

3、【正解】C

【解析】A 中 scenf、B 中 struet 和 D 中 include 不是关键字，关键字请查看本书 1.2.1

4、【正解】A

【解析】printf（）函数"%md"符号为指定输出宽度，如果数据位数小于 m，则左端补空格，若大于 m，则按实际位数输出。

5、【正解】B

【解析】A 未定义变量类型；C 只给 c 赋值；D 中 b,c 属于未定义变量。

专题二参考答案

基础篇

1、【正解】A

2、【正解】C

【解析】前置自增运算符表示先进行自增运算，再代入表达式，而后置自增运算符则是先代入表达式，再进行自增运算

3、【正解】C

【解析】因为变量 `c` 为 `int` 型，因此赋值运算右边的表达式会自动转换为 `int` 型，`a/b` 转换为 1，结果 1.4，因为 `a` 是 `int` 型，取整后为 1。故选 C

4、【正解】A

【解析】字符型变量在一定条件下可以转换为 `int` 型，所以可以给字符型变量赋整型常量值，而当它以字符型变量输出时，整型常量是其 ASCII 码，输出为 ASCII 码值对应的字符。

5、【正解】D

【解析】八进制数 `'101'` 对应的十进制数是 65，对应于 ASCII 码表的字符 A。

6、【正解】D

【解析】C 语言中两个关系运算符中间需用逻辑运算符连接，排除 A；再判断这是一个与的逻辑，因此排除 B；因为需要的是数字字符，所以排除 C，故选 D。

7、【正解】C

【解析】逗号运算符的运算规则是从左到右依次运算各个表达式，结果为最后一个表达式的值。

8、【正解】C

【解析】A 中 `'\f'`，表示换页，是合法的转义字符。

B 中 `'101'`，表示八进制数 101 代表的字符，`'\x1f'` 表示的是十六进制数 1f 代表的字符是合法的转义字符。

D 中 `'\'`，表示反斜线符 (`\`)，是合法的转义字符。

9、【正解】D

【解析】表达式先计算 `x*x=25`，再计算 `x-=25`，`x=-30`，最后计算 `x+=x`，结果为 -60。

提高篇

1、【正解】C

【解析】% 运算的操作对象为整数，赋值运算左边为变量，右边为表达式。

2、【正解】D

【解析】宏定义中（这个会在后面函数章节讲述）`d` 为常量，常量不能进行自增操作。

3、【正解】A

【解析】阶数必须为 1~3 位无符号整型常量，不能为空，且 `E/e` 前面的部分不可完全省去，因此排除 B,C,D。

4、【正解】C

【解析】根据条件运算符的规则，因为不满足 `10>20`，所以执行 `(60, 70)`，由于逗号运算的值为最后一个表达式的值，所以最后输出为 70。

5、【正解】C

【解析】当与运算左边的表达式已经为 0，则右边的表达式不运算，因此 `n` 依然是 2。

6、【正解】D

【解析】当或运算左边的表达式已经为 1，则右边的表达式不运算，因此 `b` 的值依然为 1。

7、【正解】A

【解析】B 中余数运算的左右操作数必须是整型，而 `x` 是双精度型数；C 中数值不能赋给等式；D 中应该为 `y=(double)i`

专题三参考答案

基础篇

1、【正解】D

【解析】D 相当于 $(c > 2 \&\& c \leq 6) \vee \neg (c == 3 \&\& c == 5)$ 。

2、【正解】B

【解析】此 if 语句用于求出 a,b,c 中的最小值。

3、【正解】A

4、【正解】C

【解析】表达式 $(! a == 1)$ 等于 0。

5、【正解】6

6、【正解】7

提高篇

1、【正解】B

【解析】语句 $\text{if}(a==1 \&\& b++==2)$ 中表达式的值为 1，条件成立，同时， $b++$ 等于 3；在语句 $\text{if}(b!=2 \vee c--!=3)$ 中， $b==3$ ，所以或语句“ \vee ”左边的值为 1，由逻辑表达式的短路特性可得，“ \vee ”右边的语句不再执行，则 c 的值不变。

2、【正解】C

【解析】语句还可以表示为：**if(a=1)**

{ b=1; } //属于是单分支的 if 语句。

c=2;

else d=3; //在此处的 **else** 无相应的 if 语句与之对应，用法错误。

printf("%d,%d,%d,%d\n",a,b,c,d);

3、【正解】A

【解析】B 选项中 $\text{switch}((\text{int})x);$ 语句中不应该有最后的分号。C 和 D 选项中 case 后面只能是整型常量、字符常量或符号常量组成的表达式，不能是变量。

4、【正解】A

5、【正解】20

【解析】程序还可以表示为：

main()

{ int a=0,b=1,c=0,d=20;

if(a) d=d-10;

else if(!b)

{ if(!c) d=15;

else d=25; } //通过 if 语句的判断，可以发现并不执行 if 的嵌套语句，d 仍然为 20 不变

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

}

6、【正解】3

【解析】 $a > b > c$ 即为 $(a > b) > c$ ，即为 $1 > c$ ；表达式的值为 0；表达式 $c-1 \geq d$ 即为 $(c-1) \geq d$ ，值为 1。

专题四参考答案

基础篇

1、【正解】A

2、【正解】D

【解析】每次循环 a 值加 1, b 值为 0; 最后一次时 a=4; 进行条件的判断, a<4 的值为 0, 则不需要再进行右边的++b 的运算判断, 则 a=4, b=0;

3、【正解】A

【解析】总执行次数 4*5=20。

4、【正解】D

【解析】每次循环结束后 y, x 的值分别为: 第一次: y=2, x=2
第二次: y=4, x=6
第三次: y=6, x=12
第四次: y=8, x=20

5、【正解】D

【解析】最后一次循环 m=n=5。

6、【正解】B

【解析】关系运算符优先级高于逻辑运算符, 但逻辑运算符中的“!”属于单目运算符, 单目运算符的运算优先级高于双目运算符。

7、【正解】D

【解析】switch 中的表达式先进行 i 值的判断, 决定执行 case 11:i++; 语句, 然后自增, 由于分支语句 case 11:i++; 后无 break 语句, 所以继续向下执行, 此次共有三次自增操作, i 的值为 14。

8、【正解】j=5, i=3

【解析】程序也可表示为:

```
for(i=0; i<5; i++)
{
    for(j=1; j<10; j++)
    {
        if(j==5)
            break; } //每次 j==5 时才会执行下面的语句, 所以 j=5;
        if(i<2)
            continue;
        if(i>2)
            break;
        printf("j=%d", j);
    }
    printf("i=%d\n", i);
}
```

这部分表明只有当 i=2 时才会输出 j 的值, 同时 i++ 等于 3。

9、【正解】B

【解析】第一次循环结束: x=3, y=2; 第二次循环结束: x=2, y=4; 第三次循环结束: x=1, y=6; 第四次循环结束: x=0, y=8; 不再满足条件, 循环结束。

提高篇

1、【正解】D

【解析】for 语句中判断是否继续进行循环的部分是赋值语句 k=1; 也就是说, 无论 k 为何值, 均可继续循环。

2、【正解】C

【解析】A 中的表达式可以是任意的合法数值，不限于逻辑表达式；B 选项中“12”不是合法的符号，编译时会报错。D 中 break 是终止循环，而不是终止程序的执行。

3、【正解】D

【解析】#include<stdio.h>

```
void main()
{   int n=1;
    while(n--);           //当 n 等于 0 时终止循环，然后 n 自减为-1
    printf("n=%d\n",n);
}
```

4、【正解】C

【解析】for(i=0;i<4;i++,i++)

for(k=1;k<3;k++); //表明这个嵌套语句是一个空语句

```
printf("*");
```

循环的结果就是执行最后一个 C 语句，即输出“*”。

5、【正解】D

【解析】int x=8;

```
for(;x>0;x--)
```

```
{   if(x%3)
```

```
{   printf("%d",x--); //先输出 x 的值，再进行自减
```

```
    continue;           //结束本次循环，接着进行是否执行下一次循环的判断
```

```
}
```

```
printf("%d",--x);
```

```
}
```

6、【正解】采用穷举法，因 100 钱最多可买 20 只公鸡，母鸡最多可买 33 只，所以，可令 x 的取值 0 到 20，y 的取值 0 到 33，z=100-x-y,此时只要判断 x、y、z 是否满足 $5x+3y+z/3=100$ 即可。

```
#include<stdio.h>
```

```
void main()
```

```
{   int x,y,z;
```

```
    for(x=0;x<=20;x++)           //x 从 0 到 20 遍历
```

```
        { for(y=0;y<=33;y++)           //y 从 0 到 33 遍历
```

```
            {z=100-x-y;           //根据 100 只鸡的条件得到 z
```

```
            //满足方程，输出一组 x、y、z 的解
```

```
            if(x*5+y*3+z/3.0==100)
```

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

```
            }
```

```
        }
```

```
}
```

7、【正解】#include<stdio.h>

```
void main()
```

```
{   int x,y;
```

```
    do{
```

```
        system("cls");           //清屏
```

```
printf("输入 1 个数字: ");
//如果 scanf()正常执行, 其值等于输入的数字的个数: 1, 则 y 为 1; 反之 y 为 0
y=scanf("%d",&x);
if(y==1)
    break;    //y 为 1, 表示 x 得到了正确的输入值
else
    scanf("%c",&x); //y 为 0, 处理输入流中仍旧保留的未处理数据
}while(1);
printf("x=%d",x);
}
```

8、【正解】

```
#include<stdio.h>
#include<math.h>
void main()
{ float a,b;
  char op;
  while(1)                                //while (1) 表示无限循环, 参见 4.1.1-2
  { scanf("%f%c%f",&a,&op,&b);
    if(op!= '+'&&op!= '-'&&op!= '*'&&op!= '/') //不是加减乘除运算, 退出
      break;
    switch(op)                            //按运算符进行相应运算
    { case '+':printf("%f+%f=%f",a,b,a+b);
      break;
      case '-':printf("%f-%f=%f",a,b,a-b);
      break;
      case '*':printf("%f*%f=%f",a,b,a*b);
      break;
      case '/':if(fabs(b)<1e-6)
        printf("除法错");
      else
        printf("%f/%f=%f",a,b,a/b);
      break;
    }
  }
}
```

专题五参考答案

基础篇

1、【正解】A

【解析】字符串的长度是不包含'\0'在内的实际字符的长度

2、【正解】B

【解析】A 中 gets 函数的形参只能有一个; C 中 scanf 函数的形参必须是地址值, 而 a, b 已经是首地址了, 不

需要"&"符; D 中 gets 函数使用格式错误。

3、【正解】B

【解析】

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
    int n[3][3],i,j;
```

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

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

```
    n[i][j]=i+j; //
```

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

```
        for(j=0;j<2;j++)
```

```
    n[i+1][j+i]+=n[i][j]; //注意此处为 j+i
```

```
    printf("%d\n",n[i][j]);
```

```
}
```

仔细画图计算一下就能得出答案了哦

0	1	2
1	2	3
2	3	4

第一次循环: $i=0, j=0, n[1][0]=n[1][0]+n[0][0]=1$

第二次循环: $i=0, j=1, n[1][1]=n[1][1]+n[0][1]=3;$

第三次循环: $i=1, j=0, n[2][1]=n[2][1]+n[1][0]=3+1=4$

第四次循环: $i=1, j=1, n[2][2]=n[2][2]+n[1][1]=4+3=7$

4、【正解】D

【解析】二维数组初始化时, 可省略第一维大小, 但第二维大小不能省略;

5、【正解】合法

【解析】因为每一个字符常量都有 ASCII 码与之对应, $a['A']$ 就相当于 $a[65]$ 。

6、【正解】10

【解析】putchar() 是输出单个字符的函数, 由题意, 每次循环输出一个 #, 再结合程序循环可知, 共输出了 10 个 #。

7、【正解】D

【解析】int 型变量占用内存为 4 个字节, 而定义的数组有 10 个 int 型元素, 固应当分配 40 个字节的内存空间

8、【正解】B

【解析】数组的一般形式为 $a[\text{第一维}][\text{第二维}]$, 因为第二维为 4, 数组元素数量为 12, 所以第一维只能为 3。

9、【正解】D

【解析】将其拆分开来看: $\backslash t \backslash b \backslash 012 \ 3 \backslash n$, 因此长度为 5。

10、【正解】A

【解析】程序执行过程为: 在 $a[i]$ 不等于 $\backslash 0$ 情况下, 判断当前元素是否为空格, 若为空格将下一个元素以及之后的所有字符复制到 a 数组中。当 $i=3$ 时, if 条件成立, a 中元素为 "are you!", 因为执行下一轮循环时 i 的值为 4, 所以数组 $a[4]$ 后面没有空格, 最后输出的 a 为 "are you!"。

提高篇

1、【正解】*&*&#%

2、【正解】B

【解析】'\0'是字符串终止符，strlen 函数返回值是不包含'\0'在内的实际字符的长度。

3、【正解】5 20

【解析】'\0'是字符串终止符，strlen 函数返回值是不包含'\0'在内的实际字符的长度，所以字符串长度为 5；sizeof 函数是计算所占内存的字节数，因为 st 为字符数组，每个字符所占字节为 1，所以，整个数组所分配的内存空间为 20 个字节。

4、【正解】

```
#include <stdio.h>

void main()
{
    char ch;
    int count1=0,count2=0;
    ch=getchar();
    while(ch!= '*')
    {
        if(ch<='9'&&ch>='0') count1++;
        if( (ch<='z'&&ch>='a') ||(ch>='A'&&ch<='Z')) count2++;

        ch=getchar();
    }
    printf("%d  %d",count1,count2);
}
```

5、【正解】

```
#include <stdio.h>
#define N 5
void main()
{
    int i,j,sum,a[N][N];
    sum=0;
    for(i=0;i<N;i++)
    for(j=0;j<N;j++)
    scanf("%d",&a[i][j]);
    for(i=0;i<N;i++)
    for(j=0;j<N;j++)
    if(i==j)
    sum+=a[i][j];
    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            if(i+j==5)
    sum+=a[i][j];
    printf("%d",sum);
}
```

6、【正解】

```
#include<stdio.h>
```

```
void main()
{
    int a[3][2],b[2][3],c[3][3];
    int i,j,k;
    for(i=0;i<3;i++)
    {
        for(j=0;j<2;j++)
            scanf("%d",&a[i][j]);
    }
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
            scanf("%d",&b[i][j]);
    }
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            c[i][j]=0;
            for(k=0;k<2;k++)
                c[i][j]+=a[i][k]*b[k][j];
        }
    }
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
            printf("%d ",c[i][j]);
        printf("\n");
    }
}
```

7、【正解】

```
#include<stdio.h>
#include<string.h>
void main()
{
    char m_str[100];
    char a_str[100];
    char b_str[100];
    char new_str[100]="\0";
    char *ptr;
    printf("请输入母串:\n");
    gets(m_str);
    printf("请输入 A 子串:\n");
```

```

gets(a_str);
printf("请输入 B 子串:\n");
gets(b_str);

while((ptr=strstr(m_str,a_str))!=NULL)
{
    *ptr='\0';
    strcat(new_str,m_str);
    strcat(new_str,b_str);
    strcpy(m_str,ptr+strlen(a_str));
}
strcat(new_str,m_str);
printf("新串为:\n");
puts(new_str);
}

```

8. 【正解】

```

#include<stdio.h>
main()
{
    int a[100];
    int i,n,p=0,q;
    printf("input number:");
    scanf("%d",&n);
    q=n;
    for(i=0;i<n;i++) a[i]=i+1;
    for(i=0;;i++)
    {
        if(i==n) i=0; //当 i++一直到 n 时，肯定有一些没有被选到，比如我们输入 8,第一轮是 3,6 被赋值 0，当 i=8 时，
            继续下一轮
        if(a[i]!=0) p++;//我们下面定义的是当循环到三时，就赋值 0，所以这边等 0 的不考虑在内
        else continue;
        if(p%3==0)//这个就是从 0 一直加，到三的倍数就赋值为 0，从而达到我们的目的
        {a[i]=0;q--;} //上面 q=n;表明 q==n,只有一个为 0 就减一，为下面做铺垫
        if(q==1) break; //当剩下最后一个就输出
    }
    for(i=0;i<n;i++)
    if(a[i]!=0)
    printf("spare: %d\n\n",a[i]);
}

```

专题六参考答案

1、【正解】B

【解析】指针的正确定义形式为：类型名 *指针名。如果要赋值，仅能将某个变量的地址值（而不能为一个整数，或其它任何非地址类型的数据）即&x 赋给指针变量，故选 B。

2、【正解】D

【解析】通过字符数组名或者指针变量可以输出一个字符串。

3、【正解】D

【解析】用 printf（）函数输出一个字符时，输出项是一个值，而 A、B、C 的输出项都是一个地址值，故选 D

4、【正解】D

【解析】由题意，需要找到与 s[1]的地址等价的表示方法，A 中 s++是非法的，数组名是常量不能自增自减运算；B 中取的地址是 p 的地址；C 中并不是一个地址值。

5、【正解】C

【解析】strcpy（s，p+=6）是将 p+6 之后的字符串替换 s 字符串，所以最后输出的字符串应该是"xuexiyun"。

6、【正解】C

【解析】一个字符常量占用的存储单元为一个字节，如果 p 指向字符'O'，那么，总共跳过了 4 个字符，所以 p 指向'O'时候的值为 0x2FFF FFFE。

7、【正解】B

【解析】因为 p 是一级指针，**p 是不合法的，&*p 指的是 p 的值，是 a 的地址值，排除 A、C 选项；pp 是一个二级指针，*pp 指的是 p 的值，是一个地址值，故选 B。

8、【正解】B

【解析】**s 定义了一个二级指针，因此*s 应为一个字符串，**s 为一个字符。

9、【正解】D

10、【正解】2 1 2 1 1

【解析】此程序的功能是统计并输出字符'a','b','c','d','e'在字符串中的个数。

11、【正解】4

【解析】#include<stdio.h>

```
void main()
{
    int a[]={1,2,3},b[]={4,2,5};//初始化数组
    int *p,*q;    //定义指针
    int i,x;
    p=a;q=b;
    for(i=0;i<3;i++) //共三次循环
        if(*(p+i)==*(q+i))//当数组 a, b 中有相同元素时，令 x=*(p+i) *2
            x=*(p+i)*2;
    printf("%d",x);
}
```

12、【正解】1 2 3

4 5 6 7

【解析】#include<stdio.h>

```
void main()
```

```

{
int a[]={1,2,3,-1},b[]={4,5,6,7,-1};
int *p[]={a,b},**pp=p,i;
for(i=0;i<2; i++)
{
while(**pp>=0)
{
printf("%3d",**pp);//因为-1<0,所以第一次 for 的循环输出数组 a 的前三个元素第二次输出 b 数
组的前四个元素

(*pp)++;
}
printf("\n");
pp++;//跳转到 b 数组
}
}

```

专题七参考答案

基础篇

1、【正解】全局变量 extern

【解析】外部变量的类型为 extern

2、【正解】C

【解析】auto 属性变量被存储在动态存储区内

register 属性变量被存放在 CPU 的通用存储器内

static 和 extern 属性变量被存放在静态存储区内

3、【正解】1 10 1

【解析】在函数 f() 中的 a 是局部动态变量，在函数 f() 运行结束后会被释放，也就是当第二次运行函数 f() 时，变量 a 会被重新定义。而主函数中的 a 与函数 f() 中的 a 是两个不同的变量，因此主函数中的 printf() 函数输出的值为 10，而不是 1，在 f() 函数中输出的值为 1。

提高篇

1、【正解】sum=153

【解析】首先，确定 y 是 f() 函数内的静态变量，只能被初始赋值一次；主函数内共循环五次，sum 每次循环都加上 f(i)。

2、【正解】64

【解析】首先，确定 x 是 fun() 函数内的静态变量，只能被初始赋值一次，每次运行函数时，x 的值乘 2；主函数内共循环 3 次，每次循环 s 乘以 fun()。

3、【正解】main:x=2,y=1,a=1

func:x=2,y=5,a=3

main:x=2,y=1,a=3

func:x=3,y=7,a=5

main:x=2,y=1,a=3

【解析】

```
#include<stdio.h>
int a=1;           //首先定义 a 为全局变量
void func()
{
    static int x=1; //定义 x 为函数 func 中的局部静态变量，只能初始化一次
    int y=2;        //定义 y 为函数 func 中的局部动态变量，每次函数执行完都会被释放。
    x=x+1;
    a=a+2;
    y=y+a;
    printf("func:x=%d,y=%d,a=%d\n",x,y,a); //输出经过运算后的 x, y, a 的值
}
void main()
{
    static int x=2; //定义主函数中的局部静态变量 x，与 func 函数中的 x 不是同一个变量。
    int y;          //定义主函数中的局部动态变量 y，与 func 函数中的 y 不是同一个变量。
    y=a;
    printf("main:x=%d,y=%d,a=%d\n",x,y,a); //输出为 main:x=2,y=1,a=1
    func(); //输出 func:x=2,y=5,a=3
    printf("main:x=%d,y=%d,a=%d\n",x,y,a); //主函数中 x 的值没变，全局变量 a 通过 func 函数改变了。输出
    main:x=2,y=1,a=3
    func();           //func 函数中的局部静态变量 x，与全局变量 a 改变，
    但 func 函数内局部动态变量 y 不改变。输出 func:x=3,y=7,a=5

    {
        int a; //定义另一个局部动态变量 a，与全局变量 a 不是同一个变量
        a=x+y;
        printf("main:x=%d,y=%d,a=%d\n",x,y,a); //输出的 a 为主函数的 x, y 之和
    }
}
```

专题八参考答案

基础篇

1、【正解】D

【解析】C 语言中参数也可以通过地址传送，这样能够实现参数的双向传递。

2、【正解】C

【解析】函数名也相当于是标识符，因此不能以数字开头，排除 A；函数调用时，函数名称是需要区分大小写的，排除 B；函数中可以多次出现 return 语句，排除 D；故选 C。

3、【正解】B

【解析】在某些情况下，函数名可以作为实参传递，排除 A；函数中可以多次出现 return 语句，排除 C；递归

调用需要明确结束条件，否则就会变成死循环，排除 D，故选 B。

4、【正解】B

【解析】函数的值能通过 return 语句返回主调函数，可以允许有多个 return 语句，但每次只能有一个 return 语句被执行，故只能返回一个返回值。无返回值的函数，可以定义为“空类型”，因此选 B。

5、【正解】B

【解析】由题意可知，函数的两个形参都是指针的格式，因此，在函数的声明时，也必须是两个指针类型。

6、【正解】A

【解析】函数调用中发生的数据传递是单向的。即只能把实参的值传递给形参，而不能把形参的值反向地传递给实参。因此在函数调用时，形参的值发生改变，而实参中的值不会发生变化。

7、【正解】C

【解析】本题考察重点在于函数的声明和调用；当 i 等于 30 时循环停止，总共经历三次循环，最后 a 等于 503。

8、【正解】A

【解析】在 fun(&b, a) 中，*c='a', d=65。*c+1='b', d+1=66, printf("%c,c, ", *c, d); 输出 b, B, ; 因为指针指向地址的值为 b，此时 b=*c='b'; 函数返回执行 printf("%c,%c\n", b, a); 输出 b, A，故选 A。

9、【正解】B

【解析】在 f(int *p, int *q) 函数中，执行 p=p+1 是将 p 所对应的地址加 1，而 *q=*q+1 是将 q 所指向的 n 的地址所对应的值加 1，所以 m 的地址所对应的值没有改变，而 n 的值则为 3 了。故选 B。

10、【正解】A

【解析】因为 fun() 是一个递归函数，所以主函数 fun(7) 经过 3 次递归调用，其过程可描述为“fun(7)=7-fun(5)=7-(5-fun(3))=7-(5-(3-fun(1)))=2”，所以最后的输出结果为 2，故选 A。

提高篇

1、【正解】TurboC++

Java

Basical

【解析】函数执行的功能是把输入的若干字符串按照从大到小的规则排序。输出时要注意不是从第一个字符串开始输出。

2、【正解】15

【解析】此题考察的是程序的嵌套。首先明确 runc(runc(x,y),z) 只有两个参数，一个 z，另一个 runc(x,y)，所以只要知道它们的值，就能解出此题。

3、【正解】#include<stdio.h>

#include<math.h>

int s(int a[], int n)

{

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

{if(a[i]%7==0) //判断这个数是否能够被 7 整除

{printf("%d", i+1); //输出这个数的位置为 i+1

return a[i+1];

break;

}

if(i==n)

return 0;

```

    }
}
void main()
{
    int a[]={1,2,3,4,5,6,77,3,7,8};
    int m=s(a,10);
    printf("%d",m);
}

```

4、【正解】 void draw_triangle(int line)

```

{
    int i,j;
    for(i=1;i<=line;i++)
    {
        for (j=1;j<=line-i;j++)
            printf(" ");
        for(j=1;j<=i;j++)
            printf("*");
        printf("\n");
    }
}

```

5、【正解】 #include<stdio.h>

```

double ftoc(double f)
{
    double c=0;
    c=5*(f-32)/9;
    return c;
}
void main()
{
    double f;
    scanf("%f",&f);
    printf("%f",ftoc(f));
}

```

专题九参考答案

基础篇

1、【正解】 C

2、【正解】 D

【解析】 p 指向 a 数组的第一个元素，所以 p->x 为 20，然后 p=p->y 后，p 指向数组 a 的第二个元素，所以输出为 15，故选 D。

3、【正解】 D

【解析】要输出为 6，由题意可知，由 $5+1=6$ 得出，A 的结果为 7；B、C 的结果都为 5。

4、【正解】C

【解析】dt 为结构体数组，那么指针 p 指向了结构体数组的一个元素，所以 p->x 为 1，p->y 为 2，故选 C

5、【正解】B

【解析】sizeof 是内存容量度量函数，该结构中 char 型变量共占 10 个字节，int 型占 2 个字节，double 占 8 个字节，共计 10 个字节。

6、【正解】D

【解析】由于函数是将实参的值传递给形参，因此，形参值的改变并不影响实参的值，因此结构体中的成员并不发生改变，故选 D。

提高篇

1、【正解】B

【解析】typedef 并不是增加一种新的类型，而是将原有的类型给予一个新的名字。

2、【正解】B

【解析】typedef 定义时格式为 typedef 原类型名 新类型名；新类型名定义后，原类型名可以继续使用，故选 B。

3、【正解】B

【解析】本题中 int* 用 T 来代替，所以定义 T a[10]; 就相当于 int *a[10]; 故选 B。

4、【正解】A

【解析】删除链表中的节点的操作方法是：将要删除节点的上一个节点指向要删除节点的下一个节点，然后释放该要删除的节点，故选 A。

5、【正解】A

【解析】本题考察 malloc 函数，题目中要求 p 指向一个 int 型的动态存储单元，那么就应该将分配单元换为 int，故选 A。

6、【正解】

(1) struct POINT

```
{
    double x;
    double y;
};
```

(2) int in_circle(struct POINT *point)

```
{
    if(point->x*point->x+point->y*point->y< 1)
        return 1;
    else
        return 0; }
```

7、【正解】

```

#include <stdio.h>
#define N 3
struct student
{
    char id[15];
    char name[10];
    char sex[5];
    float score;
    struct student *next;
} STUDENT;
void PassRate(struct student *head);
int Sort(struct student **head);
int main(void)
{
    struct student *stu=NULL;
    struct student *head=NULL,*tmp=NULL;
    int i,j;

    /*get and save the data*/
    for(i=0;i<N;i++)
    {
        stu=(struct student*)malloc(sizeof(struct student));
        scanf("%s",stu->id);
        scanf("%s",stu->name);
        scanf("%s",stu->sex);
        scanf("%f",&stu->score);
        stu->next=head;
        head=stu;          //head-insert list
    }
    PassRate(head);
    i=0;
    struct student *tmp;
    if( Sort(head))
    {
        tmp=head;
        for(i=0;i<N;i++)
        {
            printf("%s ",tmp->id);
            printf("%s ",tmp->name);
            printf("%s ",tmp->sex);
            printf("%f\n",tmp->score);
            tmp=tmp->next;
        }
    }
    return 0;
}

}
/*get and output the pass rate*/
void PassRate(struct student *head)
{
    int i;
    float count=0;
    struct student *tmp=head;
    for(i=0;i<N;i++)
    {
        if((tmp!=NULL)&&(tmp->score>=60))
            count++;
        tmp=tmp->next;
    }
    count=count/N;
    printf("The number of pass rate is %f\n",count);
}
int Sort(struct student **head)
{
    int i=0;
    struct student *prior1,*prior2,*p1,*p2,*t;
    p1=(struct student*)malloc(sizeof(struct student));
    p1->next=*head;
    (*head)=prior1=p1;
    for(p1=prior1->next;p1->next=NULL;prior1=p1,p1=p1->next)
    {
        for(p2=p1->next;prior2=p1;p2!=NULL;prior2=p2,p2=p2->next)
        {
            if(p1->score<p2->score){
                t=p2->next;
                prior1->next=p2;
                prior2->next=p1;
                p2->next=p1->next;
                p1->next=t;
                t=p1;p1=p2;p2=t;
            }
        }
        p1=(*head);
        (*head)=(*head)->next;
        free(p1);
        return 1;
    }
}

```

8、【正解】

#include<stdio.h> //头文件、宏定义、结构体定义、声明 1 分

#include<stdlib.h>

#define WORKER_NUMBER 5//员工个数

typedef struct worker

{

char name[20];

char sex;

int work_age;

int wage;

char money[6];

} WORKER;

void input(WORKER *,int);

void output(WORKER *,int);

void main(void) //main 函数中框架输入、输出 2 分

{

WORKER workman[WORKER_NUMBER];

input(workman,WORKER_NUMBER);

output(workman,WORKER_NUMBER);

}

void input(WORKER *workman,int number) //输入函数 3 分

{

int i;

char tmp[15];

char money_tmp;

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

{

printf("Name:");

gets((workman+i)->name);

printf("Sex(1-male,0-female):");

gets(tmp);

(workman+i)->sex=*tmp;

printf("Work age:");

gets(tmp);

(workman+i)->work_age=atoi(tmp);

printf("Wage:");

gets(tmp);

(workman+i)->wage=atoi(tmp);

//计算各种面值钱币张数

(workman+i)->money[0]=(workman+i)->wage/100;

money_tmp=(workman+i)->wage%100;

(workman+i)->money[1]=money_tmp/50;

money_tmp=money_tmp%50;

(workman+i)->money[2]=money_tmp/20;

money_tmp=money_tmp%20;

(workman+i)->money[3]=money_tmp/10;

money_tmp=money_tmp%10;

```

    (workman+i)->money[4]=money_tmp/5;
    (workman+i)->money[5]=money_tmp%5;
}
}
void output(WORKER *workman,int number)//输出函数 2 分
{
    int i;
    int j;
    int unit[]={100,50,20,10,5,1}; //为方便循环输出钱币张数而定义
    printf("%-15s%-4s%-9s%-5s%-6s\n","name","sex","work_age","wage","money");
    for(i=0;i<number;i++)
    {
        if((workman+i)->work_age>20&&(workman+i)->wage>5000&&(workman+i)->sex=='1')
        {
            printf("%-15s%-4c%-9d%-5d",(workman+i)->name,(workman+i)->sex,
            (workman+i)->work_age,(workman+i)->wage);
            for(j=0;j<6;j++)
            {
                printf("%3d:%-2d",unit[j],(workman+i)->money[j]);
            }
            printf("\n");
        }
    }
}
}
}

```

专题十参考答案

基础篇

1、【正解】C

【解析】选项 A 的方式是只能读，不能写，无法修改。选项 B 是以追加方式"ab+"打开文件读写，以这种方式打开时，新写入的数据只能追加在文件原有内容之后，但可以对以前的数据读出。可见，按此种方式打开文件不能实现文件内容的修改。选项 D 以"w+"方式打开文件，此时，原文件中已存在的内容都被清除。但新写入文件的数据可以被再次读出或再次写入，故也不能实现对文件的修改。只有以"r+"方式打开文件时，才允许将文件原来数据读出，也允许在某些位置上再写入，从而实现对文件的修改。

2、【正解】rewind()

3、【正解】C

【解析】fseek()函数的功能是将fp的文件指针从SEEK_END开始，移动0个字节，也就是未产生移动，指针位于文件尾；ftell()函数的功能是返回文件的文件位置指针当前所指位置（字节数），所以这个语句段的作用是计算文件的长度。

4、【正解】二进制

5、【正解】A

【解析】B 选项中打开一个已存在的文件并进行写操作后，原有文件中的全部数据不一定被覆盖，也可以对源文件进行追加操作等。C 选项中在一个程序中对文件进行写操作后，不是先关闭该文件然后再打开才能读到第一个数据，用fseek()函数进行重新定位即可。D 选项中，C 语言中的文件可以进行随机读写。

6、【正解】C

提高篇

1、【正解】`ch!=EOF` 或 `!feof(fp)`

`fclose(fp)`

2、【正解】D

【解析】函数原型：`unsigned fwrite(const void *buffer,unsigned size,unsigned count,FILE *fp)`

`buffer` 是待输出数据块的起始地址；`size` 表示每个数据块的大小（待输出的每个数据块的字节数）；`count` 最多允许写入的数据块个数（每个数据块 `size` 个字节）。

3、【正解】D

4、【正解】`#include<stdio.h>`

```
void main()
{   FILE *f1,*f2;
    char c;
    if((f1=fopen("a.txt ", "r "))==NULL)
    {   printf("Can't open file 1. ");
        return;
    }
    if((f2=fopen("b.txt ", "w "))==NULL)
    {   printf("Can't open file 2. ");
        return;
    }
    while((c=fgetc(f1))!=EOF)
        if(c>= 'A ' && c<= 'Z ')
            fputc(c+32,f2);
    fclose(f1);
    fclose(f2);
}
```

终极题目

请输入目前的分数，并输出学完《C 语言考试宝典》的结果！

参考答案：

```
#include<stdio.h>

void main()

{

    int x;

    scanf("%d",&x);

    printf("100 分");

}
```