

## 一、 单项选择题

1. ( **A** ) 是构成 C 语言程序的基本单位。  
A、函数          B、过程          C、子程序          D、子例程
2. C 语言程序从 **C** 开始执行。  
A) 程序中第一条可执行语句    B) 程序中第一个函数  
C) 程序中的 main 函数          D) 包含文件中的第一个函数
- 3、以下说法中正确的是 ( **C** )。  
A、C 语言程序总是从第一个定义的函数开始执行  
B、在 C 语言程序中，要调用的函数必须在 main() 函数中定义  
C、C 语言程序总是从 main() 函数开始执行  
D、C 语言程序中的 main() 函数必须放在程序的开始部分
4. 下列关于 C 语言的说法错误的是 ( **B** )。  
A) C 程序的工作过程是编辑、编译、连接、运行  
B) C 语言不区分大小写。  
C) C 程序的三种基本结构是顺序、选择、循环  
D) C 程序从 main 函数开始执行
5. 下列正确的标识符是 ( **C** )。  
A.-a1          B.a[i]          C.a2\_i          D.int t

### 5~8 题为相同类型题

考点：标识符的命名规则

- (1) 只能由字母、数字、下划线构成
- (2) 数字不能作为标识符的开头
- (3) 关键字不能作为标识符

选项 A 中的“-”，选项 B 中“[”与“]”不满足 (1)；选项 D 中的 int 为关键字，不满足 (3)

6. 下列 C 语言用户标识符中合法的是 ( **B** )。  
A)3ax    B)x    C)case    D)-e2    E)union  
选项 A 中的标识符以数字开头不满足 (2)；选项 C, E 均为为关键字，不满足 (3)；选项 D 中的“-”不满足 (1)；

7. 下列四组选项中，正确的 C 语言标识符是 ( **C** )。  
A) %x          B) a+b          C) a123          D) 123  
选项 A 中的“%”，选项 B 中“+”不满足 (1)；选项 D 中的标识符以数字开头不满足 (2)

- 8、下列四组字符串中都可以用作 C 语言程序中的标识符的是 ( **A** )。  
A、print    \_3d    db8    aBc          B、I\am    one\_half    start\$it    3pai  
C、str\_1    Cpp    pow    while          D、Pxq    My->book    line#    His.age  
选项 B 中的“\”，”\$”，选项 D 中“>”，”#”，”.”，”-”不满足 (1)；选项 C 中的 while 为关键

字，不满足(3)

9.C语言中的简单数据类型包括(D)。

- A、整型、实型、逻辑型      B、整型、实型、逻辑型、字符型  
C、整型、字符型、逻辑型      D、整型、实型、字符型

10.在C语言程序中，表达式 $5\%2$ 的结果是C。

- A)2.5      B)2      C)1      D)3

详见教材 P52~53.

%为求余运算符，该运算符只能对整型数据进行运算。且符号与被模数相同。 $5\%2=1$ ； $5\%(-2)=1$ ； $(-5)\%2=-1$ ； $(-5)\%(-2)=-1$ ；

/为求商运算符，该运算符能够对整型、字符、浮点等类型的数据进行运算， $5/2=2$

11. 如果  $\text{int } a=3, b=4$ ；则条件表达式" $a<b? a:b$ "的值是A。

- A) 3      B) 4      C) 0      D) 1

详见教材 P97.

表达式 1? 表达式 2: 表达式 3

先计算表达式 1，

若表达式 1 成立，则选择计算表达式 2，并将表达式 2 的值作为整个大表达式的值；

若表达式 1 不成立，则选择计算表达式 3，并将表达式 3 的值作为整个大表达式的值

此题中的  $a<b$  相当于表达式 1， $a$  相当于表达式 2， $b$  相当于表达式 3.

$a$  为 3， $b$  为 4。 $a<b$  表达式 1 成立，因此计算表达式 2，并将表达式 2 的值即  $a$  中的值，并作为整个表达式的值，因此整个表达式的值为 3

12. 若  $\text{int } x=2, y=3, z=4$  则表达式  $x<z?y:z$  的结果是(B)。

- A)4      B)3      C)2      D)0      E)1

13. C语言中，关系表达式和逻辑表达式的值是(B)。

- A) 0      B) 0 或 1      C) 1      D) 'T'或'F'

14. 下面(D)表达式的值为 4.

- A)  $11/3$       B)  $11.0/3$   
C)  $(\text{float})11/3$       D)  $(\text{int})(11.0/3+0.5)$

14~16 题为同一类型

详见教材 P54~56.

(1) 相同数据类型的元素进行数学运算(+、-、\*、/)得到结果还保持原数据类型。

(2) 不同数据类型的元素进行数学运算，先要统一数据类型，统一的标准是低精度类型转换为高精度的数据类型。

选项 A，11 与 3 为两个整数， $11/3$  结果的数据类型也应为整数，因此将 3.666666 的小数部分全部舍掉，仅保留整数，因此  $11/3=3$ 。

选项 B，11.0 为实数，3 为整数，因此首先要统一数据类型，将整型数据 3 转换为 3.0，转换后数据类型统一为实型数据，选项 B 变为  $11.0/3.0$ ，结果的数据类型也应为实型数据，因此选项 B  $11.0/3=3.666666$

选项 C，先将整数 11 强制类型转换，转换为实型 11.0，因此选项 C 变为  $11.0/3$ ，其后计算过程、结果与选项 B 同



素; .....;a[9] 表示组中第 10 个元素.

选项 A, 超过了数组 a 的逻辑地址下标范围;

选项 B, 逻辑地址下标只能为整数

选项 C, 逻辑地址下标只能放在[]中

21.若有以下数组说明, 则 i=10;a[a[i]]元素数值是 ( C )。

int a[12]={1,4,7,10,2,5,8,11,3,6,9,12};

A.10

B.9

C.6

D.5

先算 a[a[i]]内层的 a[i], 由于 i=10,因此 a[i]即 a[10].

a[10]对应下面数组中的元素为 9. 因此 a[a[i]]即为 a[9]

a[9]对应下面数组中的元素为 6. 因此 a[9]即为 6

22.若有说明: int a[][3]={ {1,2,3},{4,5},{6,7}}; 则数组 a 的第一维的大小为: ( B )

A. 2

B. 3

C. 4

D.无确定值

5 7

D) 3 6 9

二维数组的一维大小, 即指二维数组的行数, 在本题中, 按行对二维数组赋值, 因此内层有几个大括号, 数组就有几行

23.对二维数组的正确定义是 ( C )

详见教材 P149~152, 二维数组的定义、初始化

类型符 数组名 [常量表达式][常量表达式]

二维数组可以看做是矩阵

类型符是指数组中数组元素的类型; 数组名要符合标识符命名规则; 第一个常量表达式是指数组的行数; 第二个常量表达式是指数组的列数; 常量表达式的值只能是整数, 不可以是变量, 而且从 1 开始计数。

一维数组初始化时可以省略数组长度

二维数组初始化时可以省略行数, 但不能省略列数

选项 A,B, 都省略了列数

选项 D, 不符合二维数组定义的一般形式, 行、列常量表达式应该放在不同的[]中

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

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

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

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

24. 已知 int a[3][4];则对数组元素引用正确的是\_\_C\_\_

A)a[2][4]

B)a[1,3]

C)a[2][0]

D)a(2)(1)

详见教材 P150, 数组元素的引用

数组名[下标][下标]

引用数组元素时, []中的下标为逻辑地址下标, 只能为整数, 可以为变量, 且从 0 开始计数

第一个[下标]表示行逻辑地址下标, 第二个[下标]表示列逻辑地址下标。

本题图示详见 P149 图 6.7

因此 a 的行逻辑地址范围 0~2; a 的列逻辑地址范围 0~3;

选项 A, 列逻辑地址下标超过范围

选项 B,D, 的引用形式不正确。

25.C 语言中函数返回值的类型是由\_\_\_\_\_A\_\_\_\_\_ 决定的.

- A)函数定义时指定的类型                      B) return 语句中的表达式类型  
C) 调用该函数时的实参的数据类型   D) 形参的数据类型

26. 在 C 语言中, 函数的数据类型是指(A)

- A 函数返回值的数据类型                      B. 函数形参的数据类型  
C 调用该函数时的实参的数据类型      D.任意指定的数据类型

27. 在函数调用时, 以下说法正确的是 ( B )

- A.函数调用后必须带回返回值  
B.实际参数和形式参数可以同名  
C.函数间的数据传递不可以使用全局变量  
D.主调函数和被调函数总是在同一个文件里

28. 在 C 语言中, 表示静态存储类别的关键字是: ( C )

- A) auto      B) register      C) static      D) extern

29. 未指定存储类别的变量, 其隐含的存储类别为 ( A )。

- A)auto    B)static    C)extern    D)register

30. 若有以下说明语句:

```
struct student
{ int num;
  char name[ ];
  float score;
}stu;
```

则下面的叙述不正确的是: ( D )

- A. struct 是结构体类型的关键字  
B. struct student 是用户定义的结构体类型  
C. num, score 都是结构体成员名  
D. stu 是用户定义的结构体类型名

31.若有以下说明语句:

```
struct date
{ int year;
  int month;
  int day;
}brithday;
```

则下面的叙述不正确的是\_\_\_\_\_C\_\_\_\_\_.

- A) struct 是声明结构体类型时用的关键字  
B) struct date 是用户定义的结构体类型名  
C) brithday 是用户定义的结构体类型名  
D) year,day 都是结构体成员名

32. 以下对结构变量 stu1 中成员 age 的非法引用是\_\_\_\_\_B\_\_\_\_\_

```
struct student
{ int age;
```

```
int num;
}stu1,*p;
p=&stu1;
```

A) stu1.age    B) student.age    C) p->age    D) (\*p).age

33.设有如下定义:

```
struct sk
{
    int a;
    float b;
}data;
int *p;
```

若要使 P 指向 data 中的 a 域, 正确的赋值语句是 C

A) p=&a;    B) p=data.a;    C) p=&data.a;    D)\*p=data.a;

34.设有以下说明语句:

```
typedef struct stu
{
    int a;
    float b;
} stutype;
```

则下面叙述中错误的是 ( D )。

- A、struct 是结构类型的关键字
- B、struct stu 是用户定义的结构类型
- C、a 和 b 都是结构成员名
- D、stutype 是用户定义的结构体变量名

35. 语句 int \*p;说明了 C。

- A)p 是指向一维数组的指针
- B)p 是指向函数的指针,该函数返回一 int 型数据
- C)p 是指向 int 型数据的指针 // 指针的定义教材 P223
- D)p 是函数名,该函数返回一指向 int 型数据的指针

36. 下列不正确的定义是 ( A )。

- A. int \*p=&i,i;    B.int \*p,i;
- C. int i,\*p=&i;    D.int i,\*p;

选项 A 先定义一个整型指针变量 p, 然后将变量 i 的地址赋给 p。然而此时还未定义变量 i 因此编译器无法获得变量 i 的地址。(A 与 C 对比, 选项 C 先定义变量 i, 则在内存中为 i 分配空间, 因此 i 在内存空间的地址就可以确定了; 然后再定义 p, 此时可以为 p 赋 i 的地址, C 正确)

37. 若有说明: int n=2,\*p=&n,\*q=p,则以下非法的赋值语句是: ( D )

- A) p=q    B) \*p=\*q    C) n=\*q    D) p=n

p,q 同为整型指针变量, 二者里面仅能存放整型变量的地址。

选项 A, q 中为地址, 因此可将此地址赋给 p

选项 B, \*p 表示 p 所指向对象 n 的内容, 即一个整数; \*q 表示 q 所指向对象的内容, 由于在定义 q 时为其初始化, 将 p 中 n 的地址给 q, 因此 p 中存放 n 的地址, \*q 表示 q 所指向对象 n 的内容.因此\*p=\*q 相当于 n=n;

选项 C, n=\*q 等价于 n=n;

选项 D, p 中只能存放地址, 不能将 n 中的整数值赋给 p

38. 有语句: `int a[10];` 则 B 是对指针变量 p 的正确定义和初始化。

A) `int p=*a;` B) `int *p=a;` C) `int p=&a;` D) `int *p=&a;`

选项 A, a 是数组名, 不是指针变量名, 因此不可用 \* 标注数组名 a

选项 C, a 是数组名, 数组名就是地址, 无需再用地址符号。而且在定义指针变量 p 时, 应在变量名前加 \*, 表明 p 是指针变量

选项 D, a 是数组名, 数组名就是地址, 无需再用地址符号。

39. 若有说明语句 “`int a[5], *p=a;`”, 则对数组元素的正确引用是 (C)。

A. `a[p]` B. `p[a]` C. `*(p+2)` D. `p+2`

首先定义一个整型数组 a, a 的长度为 5, 然后定义一个指针变量 p, 并同时 p 进行初始化, 将数组 a 的地址赋给 p。因此此时 p 中存放的数组 a 的首地址, 即数组中第一个元素 `a[0]` 的地址。

对于数组元素下标的引用 (详见 p144), 一般形式 数组名[下标] 其中下标为逻辑地址下标, 从 0 开始计数, 方括号中的下标可以是变量, 可以是表达式, 但结果一定要是整数。

选项 A, p 中存放的是地址, 不是整数, 不能做数组元素的下标

选项 B, a 是数组名, 数组名就是地址, 不是整数, 不能做数组元素的下标

选项 C, (重点!!! 详见 p231~234) `p+2` 表示指向同一数组中的下两个元素的地址, 当前 p 指向 `a[0]`, 则 `p+2` 表示 `a[2]` 的地址, 因此 `*(p+2)` 表示 `a[2]` 的内容

40. 有如下程序

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

则数值为 9 的表达式是 B

A) `*P+9` B) `*(P+8)` C) `*P+=9` D) `P+8`

(重点!!! 详见 p231~234)

首先定义一个整型数组 a, a 的长度为 5, 然后定义一个指针变量 P, 并同时 P 进行初始化, 将数组 a 的地址赋给 P。因此此时 P 中存放的数组 a 的首地址, 即数组中第一个元素 `a[0]` 的地址。

数组中 9 对应的是 `a[8]`, 选项 B, `P+8` 表示数组中后 8 个元素的地址, 即 `a[8]` 的地址。`*(P+8)` 则表示该地址内所存放的内容, 即 `a[8]` 的值。

选项 A, `*P` 表示 P 所指向对象的内容, 此时 P 指向 `a[0]`, `*P` 即 `a[0]` 的值 1。 `*P+9=1+9=10`

选项 C, `*P` 表示 P 所指向对象的内容, 此时 P 指向 `a[0]`, `*P` 即 `a[0]` 的值。因此 `*P+=9` 即 `*P=*P+9`, 等价于 `a[0]=a[0]+9`。

选项 D, `P+8` 表示数组中后 8 个元素的地址, 即 `a[8]` 的地址, 而非 `a[8]` 中的值。

41. 在 C 语言中, 以 D 作为字符串结束标志

A) `'\n'` B) `' '` C) `'0'` D) `'\0'`

42. 下列数据中属于“字符串常量”的是 (A)。

A. `"a"` B. `{ABC}` C. `'abc\0'` D. `'a'`

若干个字符构成字符串

在 C 语言中, 用单引号标识字符; 用双引号标识字符串

选项 B, C, 分别用 `{}` 和 `"` 标识字符串

选项 D, 标识字符。

43. 已知 `char x[]="hello", y[]={ 'h','e','a','b','e' };`, 则关于两个数组长度的正确描述是 B。



A)相同      B)x 大于 y      C)x 小于 y      D)以上答案都不对

C 语言中，字符串后面需要一个结束标志位'\0'，通常系统会自动添加。

对一维数组初始化时可采用字符串的形式（例如本题数组 x），也可采用字符集合的形式（例如本题数组 y）。在以字符串形式初始化时，数组 x 不仅要存储字符串中的字符，还要存储字符串后的结束标志位，因此数组 x 的长度为 6；在以字符集合形式初始化时，数组 y，仅存储集合中的元素，因此数组 y 长度为 5

## 二、 读程序

### 基本输入输出及流程控制

1.

```
#include <stdio.h>
main()
{ int a=1,b=3,c=5;
  if (c==a+b)
      printf("yes\n");
  else
      printf("no\n");
}
```

运行结果为: **no**

详见教材 p89 选择结构

详见教材 p91 关系符号

详见附录 D p378 符号的优先级

==表示判断符号两边的值是否相等；=表示将符号右边的值赋给左边的变量

本题考点是选择结构 3 种基本形式的第二种

选择结构三种一般形式中的“语句”皆为复合语句，复合语句要用{ }括起来，只有当复合语句中只包括一条语句时可以省略{ }，此题即如此，因此两个 printf 操作没有加{ }

若 c==a+b 成立，则执行 printf("yes\n");

否则（即 c==a+b 不成立），执行 printf("no\n");

+的优先级高于==，因此先算 a+b,值为 4，表达式 5==4 不成立，因此执行 printf("no\n");即输出字符串 no

2.

```
#include <stdio.h>
main()
{ int a=12, b= -34, c=56, min=0;
```



```

min=a;
if(min>b)
    min=b;
if(min>c)
    min=c;
printf("min=%d", min);
}

```

运行结果为: **min=-34**

详见教材 p89 选择结构

本题考点是选择结构 3 种基本形式的第一种

一共包含了两个选择结构（两个 if 语句）

定义变量，并赋值 此时  $a=12$ ,  $b=-34$ ,  $c=56$ ,  $min=0$

将  $a$  中值拷贝，赋给  $min$ ，覆盖了  $min$  中的 0，此时  $min$  中的值被更新为 12。

若  $min>b$  成立，则执行  $min=b$ ;

若  $min>c$  成立，则执行  $min=c$ ;

输出  $min$  中的值

12 大于 -34, 第一个 if 语句的表达式成立，因此执行  $min=b$ ; 执行后  $min$  中的值被更新为 -34.

-34 小于 56, 第二个 if 语句的表达式不成立，因此不执行  $min=c$ ;

最后输出  $min$  中的值，为 -34.

3.

```

#include <stdio.h>
main()
{ int x=2,y=-1,z=5;
    if(x<y)
        if(y<0)
            z=0;
        else
            z=z+1;
    printf("%d\n",z);
}

```

运行结果为: **5**

遇到选择结构，首先要明确条件表达式成立时执行哪些操作。本题中，第一个 if 语句，其后的复合语句没有大括号 { }，说明复合语句中只包含一条语句，进而省略了 { }。内层的 if...else... 是选择结构的第二种基本形式，在结构上视为一条语句。因此内层的 if...else... 作为第一个 if 语句的复合语句。

若表达式  $x<y$  成立，则继续判断

若  $y<0$ ，则执行  $z=0$ ;

否则（即  $y>=0$ ），执行  $z=z+1$ ;

输出 z

2>-1, 表达式  $x < y$  不成立, 因此不执行内层的 if...else....进而 z 中的值没有被改变。

输出 z 中的值为 5

4.

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

```
main()
```

```
{ float a,b,c,t;
```

```
    a=3;
```

```
    b=7;
```

```
    c=1;
```

```
    if(a>b)
```

```
        {t=a;a=b;b=t;}
```

```
    if(a>c)
```

```
        {t=a;a=c;c=t;}
```

```
    if(b>c)
```

```
        {t=b;b=c;c=t;}
```

```
    printf("%5.2f,%5.2f,%5.2f",a,b,c);
```

```
}
```

运行结果为: 1.00, 2.00, 7.00

详见教材 p72 数据的输出形式

本题包含了 3 个 if 语句, 每个 if 语句后的 { } 都不可省略, 因为每个 { } 中都包含了多条语句

若表达式  $a > b$  成立, 则执行 {t=a;a=b;b=t;}

若表达式  $a > c$  成立, 则执行 {t=a;a=c;c=t;}

若表达式  $b > c$  成立, 则执行 {t=b;b=c;c=t;}

输出 a,b,c 中的值, 要求输出的每个数据宽度为 5 个空格, 小数部分保留 2 位, 数据右对齐

3 小于 7, 因此表达式  $a > b$  不成立, 因此不执行 {t=a;a=b;b=t;}

3 大于 1, 因此表达式  $a > c$  成立, 则执行 {t=a;a=b;b=t;}. 第一句, 将 a 中的 3 拷贝, 粘贴到 t 中; 第二句, 将 c 中的 1 拷贝, 粘贴到 a 中, 覆盖掉先前的 3; 第三句. 将 t 中的 3 拷贝到 c 中, 覆盖掉 c 中先前的 1. 执行完复合语句后实现了 a,c 元素的值的互换, a 为 1, c 为 3, t 为 3,。

7 大于 c 中的 3, 因此  $b > c$  成立, 执行则执行 {t=b;b=c;c=t;}, 过程同上, 执行后 b 为 3, c 为 7, t 为 7

此时输出 a,b,c 中的值为 1.00, 2.00, 7.00

5.

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

```
main()
```

```
{ float c=3.0, d=4.0;
```

```

if ( c>d ) c=5.0;
else
    if ( c==d ) c=6.0;
    else c=7.0;
printf ( "%.1f\n",c );
}

```

运行结果为: **7.0**

此题为 if...else...语句的嵌套, 第二 if...else...作为第一个 if...else...语句 else 部分的复合语句。

若表达式  $c>d$  成立, 则执行  $c=5.0$ ;

否则 (表达式  $c>d$  不成立)

若表达式  $c==d$  成立, 则执行  $c=6.0$ ;

否则, 执行  $c=7.0$ ;

输出  $c$  中的值

3.0 小于 4.0, 因此表达式  $c>d$  不成立, 执行第二个 if...else...。

3.0 不等于 4.0, 因此表达式  $c==d$  不成立, 执行  $c=7.0$ , 将 7.0 赋给  $c$ , 覆盖掉  $c$  中的 3.0, 此时  $c$  中的值为 7.0

输出此时的  $c$  中的值

6.

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

```
main()
```

```

{   int m;
    scanf("%d", &m);
    if (m >= 0)
        {   if (m%2 == 0)printf("%d is a positive even\n", m);
            else      printf("%d is a positive odd\n", m); }
    else
        {   if (m % 2 == 0) printf("%d is a negative even\n", m);
            else      printf("%d is a negative odd\n", m); }
}

```

若键入 -9, 则运行结果为: **-9 is a negative odd**

7.

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

```
main()
```

```

{ int num=0;
  while(num<=2){ num++; printf("%d\n",num); }
}

```

运行结果为:

**1**

**2**

### 3

#### 详见教材 p115 循环结构

当循环条件 `num<=2` 成立的时候，执行循环体{ `num++`; `printf("%d\n",num)`; }中的语句。

循环初值 `num` 为 0;

循环条件 `num<=2` 成立

第 1 次循环：执行 `num++`;即将 `num` 中的值加 1，执行后 `num` 为 1;

执行 `printf("%d\n",num)`; 在屏幕上输出 `num` 中的值，即输出 1，之后换行

此时 `num` 中的值为 1，循环条件 `num<=2` 成立

第 2 次循环：执行 `num++`;即将 `num` 中的值加 1，执行后 `num` 为 2;

执行 `printf("%d\n",num)`; 在屏幕上输出 `num` 中的值，即输出 2，之后换行

此时 `num` 中的值为 2，循环条件 `num<=2` 成立

第 3 次循环：执行 `num++`;即将 `num` 中的值加 1，执行后 `num` 为 3;

执行 `printf("%d\n",num)`; 在屏幕上输出 `num` 中的值，即输出 3，之后换行

此时 `num` 中的值为 3，循环条件 `num<=2` 不成立，结束循环。

### 8.

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

```
main( )
```

```
{ int sum=10,n=1;
```

```
    while(n<3)    {sum=sum-n;  n++; }
```

```
printf(“%d,%d”,n,sum);
```

```
}
```

运行结果为：3,7

当循环条件 `n<3` 成立的时候，执行循环体{`sum=sum-n`; `n++`; }中的语句。

循环初值 `sum` 为 10,`n` 为 1;

循环条件 `n<3` 成立

第 1 次循环：执行 `sum=sum-n=10-1=9`;

执行 `n++`;即将 `n` 中的值加 1，执行后 `n` 为 2;

此时 `n` 中的值为 2，`sum` 中的值为 9，循环条件 `n<3` 成立，继续执行循环

第 2 次循环：执行 `sum=sum-n=9-2=7`;

执行 `n++`;即将 `n` 中的值加 1，执行后 `n` 为 3;

输出此时 `n,sum` 中的值，即为 3,7。需要注意，在 `printf(“%d,%d”,n,sum)`; 中要求输出的数据彼此间用逗号间隔，因此结果的两个数据间一定要有逗号

9.

```
#include <stdio.h>
main()
{ int num,c;
  scanf("%d",&num);
  do {c=num%10; printf("%d",c); }while((num/=10)>0);
  printf("\n");
}
```

从键盘输入 23，则运行结果为： **32**

详见教材 p117 循环结构； p60 复合的赋值运算符

**do{ }while(表达式);**

先无条件执行循环体，再判断循环条件。注意 while（表达式）后有分号

定义整型变量 num， c;

为 num 赋一个整型值；

执行{c=num%10; printf("%d",c); }直到循环条件(num/=10)>0 不成立；

输出换行

已知为 num 赋值 23

第 1 次执行循环体

执行 c=num%10=23%10=3;

执行 printf("%d",c);输出 3

判断循环条件 num/=10 等价于 num=num/10; 因此 num=23/10=2, 2 大于 0，因此循环条件 (num/=10)>0 成立，继续执行循环体。执行完第 1 次循环时，num 为 2，c 为 3

第 2 次执行循环体

执行 c=2%10=2;

执行 printf("%d",c);再输出 2

判断循环条件 num=2/10=0,0 等于 0，因此循环条件(num/=10)>0 不成立。结束循环

10

```
#include <stdio.h>
main()
{ int s=0,a=5,n;
  scanf("%d",&n);
  do { s+=1; a=a-2; }while(a!=n);
  printf("%d, %d\n",s,a);
}
```

若输入的值 1，运行结果为： **2,1**

详见教材 p117 循环结构； p60 复合的赋值运算符

执行{ s+=1; a=a-2; }直到循环条件 a!=n 不成立；

已知为 n 赋值 1, s 为 0, a 为 5

第 1 次执行循环体

执行  $s+=1$ ; 等价于  $s=s+1=0+1$

执行  $a=a-2$ ;  $a=5-2=3$

判断循环条件, 3 不等于 1, 因此循环条件  $a!=n$  成立, 继续执行循环体。

执行完第 1 次循环时, s 为 1, a 为 3

第 2 次执行循环体

执行  $s+=1$ ; 等价于  $s=s+1=1+1=2$

执行  $a=a-2$ ;  $a=3-2=1$

判断循环条件, 1 等于 1, 因此循环条件  $a!=n$  不成立, 结束循环。

执行完第 2 次循环时, s 为 2, a 为 1

输出此时 s, a 中的值, 结果为 2, 1

11.

```
#include "stdio.h"
```

```
main()
```

```
{char c;
```

```
c=getchar();
```

```
while(c!='?') {putchar(c); c=getchar(); }
```

```
}
```

如果从键盘输入 abcde? fgh (回车)

运行结果为: **abcde**

12.

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

```
main()
```

```
{ char c;
```

```
while((c=getchar())!='$')
```

```
{ if('A'<=c&& c<='Z') putchar(c);
```

```
else if('a'<=c&& c<='z') putchar(c-32); }
```

```
}
```

当输入为 ab\*AB%cd#CD\$时, 运行结果为: **ABABCD**

13.

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

```
main()
```

```
{ int x, y=0;
```

```
for(x=1; x<=10; x++)
```

```
{ if(y>=10)
```

```
break;
```

```
y=y+x;
```

```
}
```

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

}

运行结果为: 10 5

详见教材 p120 for 语句

详见教材 p126~128 break, continue 语句

for( 表达式 1;表达式 2;表达式 3)

{

}

(1) 先求解表达式 1

(2) 求解表达式 2, 若其值为真, 执行循环体, 然后执行 (3). 若为假, 则结束循环, 转到(5)

(3) 求解表达式 3

(4) 转回上面 (2)继续执行

(5) 循环结束, 执行 for 语句下面的一个语句

**break**, 跳出循环体; **continue**, 结束本次循环 (第 i 次循环), 继续执行下一次循环(第 i+1 次循环)

此题 表达式 1 为  $x=1$ , 表达式 2 (循环条件) 为  $x \leq 10$ , 表达式 3 为  $x++$   
初值  $x$  为 1,  $y$  为 0, 循环条件 (即表达式 2)  $x \leq 10$  成立, 进入循环体

第 1 次循环

执行 if 语句。0 小于 10, if 语句的条件表达式不成立, 不执行 break;

执行  $y=y+x; y=0+1=1$

转向表达式 3, 执行  $x++$ ,  $x=x+1=1+1=2$ 。循环条件  $x \leq 10$  成立, 进入第 2 次循环

第 2 次循环

执行 if 语句。1 小于 10, if 语句的条件表达式不成立, 不执行 break;

执行  $y=y+x; y=1+2=3$

转向表达式 3, 执行  $x++$ ,  $x=x+1=2+1=3$ 。循环条件  $x \leq 10$  成立, 进入第 3 次循环

第 3 次循环

执行 if 语句。3 小于 10, if 语句的条件表达式不成立, 不执行 break;

执行  $y=y+x; y=3+3=6$

转向表达式 3, 执行  $x++$ ,  $x=x+1=3+1=4$ 。循环条件  $x \leq 10$  成立, 进入第 4 次循环

第 4 次循环

执行 if 语句。6 小于 10, if 语句的条件表达式不成立, 不执行 break;

执行  $y=y+x; y=6+4=10$

转向表达式 3, 执行  $x++$ ,  $x=x+1=4+1=5$ 。循环条件  $x \leq 10$  成立, 进入第 5 次循环

第 5 次循环

执行 if 语句。10 等于 10, if 语句的条件表达式成立, 执行 break, 跳出循环。

从 break 跳出至 for 语句的下一条语句。执行 `printf("%d %d", y, x);`  
输出当前的  $y$  与  $x$ 。结果为 10 5



14.

```
#include<stdio.h>
main( )
{   char ch;
    ch=getchar();
    switch(ch)
    {   case  'A': printf("%c",'A');
        case  'B': printf("%c",'B'); break;
        default: printf("%s\n","other");
    }
}
```

当从键盘输入字母 A 时，运行结果为：AB

详见教材 p103，switch 语句

switch（表达式）

```
{   case  常量 1 : 语句 1
    case  常量 2 : 语句 2
        :       :       :
    case  常量 n : 语句 n
    default      : 语句 n+1
}
```

其中表达式，常量 1，...，常量 n 都为整型或字符型

case 相当于给出执行程序的入口和起始位置，若找到匹配的常量，则从此处开始往下执行程序，不再匹配常量，直至遇到 break 或 switch 结束

本题过程：

首先从键盘接收一个字符'A'并将其放在变量 ch 中。

执行 switch 语句。Switch 后面的条件表达式为 ch,因此表达式的值即为字符'A'。用字符'A'依次与下面的 case 中的常量匹配。

与第 1 个 case 后的常量匹配，则从其后的语句开始往下执行程序（在执行过程中不再进行匹配。）因此先执行 printf("%c",'A')，屏幕上输出 A；再往下继续执行 printf("%c",'B')，屏幕上输出 B；再继续执行 break，此时跳出 switch 语句。

15.

```
#include <stdio.h>
main( )
{ int a=1,b=0;
  scanf("%d",&a);
  switch(a)
  {   case 1: b=1; break;
      case 2: b=2; break;
      default : b=10;  }
```

```
    printf("%d ", b);
}
```

若键盘输入 5，运行结果为：10

本题过程：

首先用 scanf 函数为变量 a 赋值为 5。

执行 switch 语句。switch 后面的条件表达式为 a,因此表达式的值即为 5. 用 5 依次与下面 case 中的常量匹配。没有找到匹配的常量，因此两个 case 后的语句都不执行。执行 default 后面的语句 b=10; 将 10 赋给变量 b。

输出变量 b,结果为 10

16.

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

```
main()
```

```
{ char grade='C';
```

```
    switch(grade)
```

```
    {
```

```
        case 'A': printf("90-100\n");
```

```
        case 'B': printf("80-90\n");
```

```
        case 'C': printf("70-80\n");
```

```
        case 'D': printf("60-70\n"); break;
```

```
        case 'E': printf("<60\n");
```

```
        default : printf("error!\n");
```

```
    }
```

```
}
```

运行结果为：

70-80

60-70

本题过程：

首先从键盘接收一个字符'C'并将其放在变量 grade 中。

执行 switch 语句。switch 后面的条件表达式为 grade,因此表达式的值即为字符'C'. 用字符'C'依次与下面的 case 中的常量匹配。

与第 3 个 case 后的常量匹配，则从其后的语句开始往下执行程序（在执行过程中不再进行匹配。）因此先执行 printf("70-80\n");，屏幕上输出 70-80,并换行；再往下继续执行 printf("60-70\n");，屏幕上输出 60-70，并换行；再继续执行 break，此时跳出 switch 语句。

17.

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

```
main()
```

```
{ int y=9;
```

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

```
        if(y%3==0)
```

```

        { printf("%d",-y);
        }
    }

```

运行结果为:

**852**

**详见教材 p53, 自增自减符号**

此题 表达式 1 被省略, 表达式 2 (循环条件) 为  $y > 0$ , 表达式 3 为  $y--$   
初值  $y$  为 9, 循环条件 (即表达式 2)  $y > 0$  成立, 进入循环体

第 1 次循环

执行 if 语句。  $9 \% 3 == 0$ , if 语句的条件表达式成立, 执行 `printf("%d",-y)`, 即  $y$  先自减 1 变为 8, 然后在输出, 因此屏幕上输出 8

转向表达式 3, 执行  $y--$ ,  $y = y - 1 = 8 - 1 = 7$ 。循环条件  $y > 0$  成立, 进入第 2 次循环

第 2 次循环

执行 if 语句。  $7 \% 3$  不为 0, if 语句的条件表达式不成立, 不执行 `printf("%d",-y)`

转向表达式 3, 执行  $y--$ ,  $y = y - 1 = 7 - 1 = 6$ 。循环条件  $y > 0$  成立, 进入第 3 次循环

第 3 次循环

执行 if 语句。  $6 \% 3 == 0$ , if 语句的条件表达式成立, 执行 `printf("%d",-y)`, 即  $y$  先自减 1 变为 5, 然后在输出, 因此屏幕上输出 5

转向表达式 3, 执行  $y--$ ,  $y = y - 1 = 5 - 1 = 4$ 。循环条件  $y > 0$  成立, 进入第 4 次循环

第 4 次循环

执行 if 语句。  $4 \% 3$  不为 0, if 语句的条件表达式不成立, 不执行 `printf("%d",-y)`

转向表达式 3, 执行  $y--$ ,  $y = 4 - 1 = 3$ 。循环条件  $y > 0$  成立, 进入第 5 次循环

第 5 次循环

执行 if 语句。  $3 \% 3 == 0$ , if 语句的条件表达式成立, 执行 `printf("%d",-y)`, 即  $y$  先自减 1 变为 2, 然后在输出, 因此屏幕上输出 2

转向表达式 3, 执行  $y--$ ,  $y = y - 1 = 2 - 1 = 1$ 。循环条件  $y > 0$  成立, 进入第 5 次循环

第 6 次循环

执行 if 语句。  $1 \% 3$  不为 0, if 语句的条件表达式不成立, 不执行 `printf("%d",-y)`

转向表达式 3, 执行  $y--$ ,  $y = 1 - 1 = 0$ 。循环条件  $y > 0$  不成立, 循环结束。

**18.**

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

```
main()
```

```
{ int i,sum=0; i=1;
```

```
do{ sum=sum+i; i++; }while(i<=10);
```

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

```
}
```

运行结果为: 55

19.

```
#include <stdio.h>
#define N 4
main()
{ int i;
  int x1=1,x2=2;
  printf("\n");
  for(i=1;i<=N;i++)
  { printf("%4d%4d",x1,x2);
    if(i%2==0)
      printf("\n");
    x1=x1+x2;
    x2=x2+x1;
  }
}
```

运行结果为:

1   2   3   5  
8   13   21   34

此题 首先为整型变量赋初值  $x1=1, x2=2$

表达式 1 为  $i=1$ , 表达式 2 (循环条件) 为  $i \leq N$  即  $i \leq 4$ , 表达式 3 为  $i++$   
循环变量初值  $i$  为 1, 循环条件 (即表达式 2)  $i \leq 4$  成立, 进入第 1 次循环

第 1 次循环

执行 `printf("%4d%4d",x1,x2);`因此屏幕上输出 1   2

执行 if 语句。1%2 不为 0, if 语句的条件表达式不成立, 不执行 `printf("\n");`

执行  $x1=x1+x2=1+2=3$ ;此时  $x1$  中的值已变为 3

执行  $x2=x2+x1=2+3=5$ 。

转向表达式 3, 执行  $i++$ ,  $i$  为 2。循环条件  $i \leq 4$  成立, 进入第 2 次循环

第 2 次循环

执行 `printf("%4d%4d",x1,x2);`因此屏幕上输出 3   5

执行 if 语句。2%2==0, if 语句的条件表达式成立, 执行 `printf("\n");`换行

执行  $x1=x1+x2=3+5=8$ ;此时  $x1$  中的值已变为 8

执行  $x2=x2+x1=5+8=13$ 。

转向表达式 3, 执行  $i++$ ,  $i$  为 3。循环条件  $i \leq 4$  成立, 进入第 3 次循环

第 3 次循环

执行 `printf("%4d%4d",x1,x2);`因此屏幕上输出 8   13

执行 if 语句。3%2 不为 0, if 语句的条件表达式不成立, 不执行 `printf("\n");`

执行  $x1=x1+x2=8+13=21$ ;此时  $x1$  中的值已变为 21

执行  $x2=x2+x1=21+13=34$ 。

转向表达式 3，执行  $i++$ ， $i$  为 4。循环条件  $i \leq 4$  成立，进入第 4 次循环

第 2 次循环

执行 `printf("%4d%4d",x1,x2);`因此屏幕上输出 21 34

执行 if 语句。 $4\%2==0$ ，if 语句的条件表达式成立，执行 `printf("\n");`换行

执行  $x1=x1+x2=21+34=55$ ;此时  $x1$  中的值已变为 55

执行  $x2=x2+x1=34+55=89$ 。

转向表达式 3，执行  $i++$ ， $i$  为 5。循环条件  $i \leq 4$  不成立，结束循环

20

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

```
main( )
```

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

```
for(x=30, y=0; x>=10, y<10; x--, y++)
```

```
    x/=2, y+=2;
```

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

```
}
```

运行结果为:

**x=0,y=12**

21.

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

```
#define N 4
```

```
main( )
```

```
{ int i,j;
```

```
for(i=1;i<=N;i++)
```

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

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

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

```
printf("\n");
```

```
}}
```

运行结果为:

\*

\*

\*

\*

**详见教材 P41 符号常量**

用宏处理指令定义符号常量  $N$  为 4，在编译过程中，遇到  $N$  即视为整数 4。

外层 for 循环，表达式 1 为  $i=1$ ，表达式 2（循环条件）为  $i \leq N$ ，表达式 3 为  $i++$

内层 for 循环，表达式 1 为  $j=1$ ，表达式 2（循环条件）为  $j < i$ ，表达式 3 为  $j++$

首先计算外层循环的表达式 1,  $i$  为 1, 使得循环条件  $i \leq 4$  成立, 进入外层 for 循环体

外层 for 循环第 1 次 此时  $i$  为 1

内层循环  $j=1$ , 使得循环条件  $j < i$  不成立, 因此不执行内层循环体 (不输出空格)

执行 `printf("*");`

执行 `printf("\n");` 换行

至此外层循环体执行完, 计算外层循环的表达式 3,  $i++$ , 此时  $i$  为 2. 使得循环条件  $i \leq 4$  成立, 再次进入外层 for 循环体

外层 for 循环第 2 次 此时  $i$  为 2

内层循环  $j=1$ , 使得循环条件  $j < i$  成立

第 1 次执行内层循环体 `printf(" ");`

执行内层循环表达式 3,  $j++$  为 2,  $j < i$  不成立, 跳出内层循环

执行 `printf("*");`

执行 `printf("\n");` 换行

至此外层循环体执行完, 计算外层循环的表达式 3,  $i++$ , 此时  $i$  为 3. 使得循环条件  $i \leq 4$  成立, 进入外层 for 循环体

外层 for 循环第 3 次 此时  $i$  为 3

内层循环  $j=1$ , 使得循环条件  $j < i$  成立

第 1 次执行内层循环体 `printf(" ");`

执行内层循环表达式 3,  $j++$  为 2,  $j < i$  成立, 再次执行内层循环

第 2 次执行内层循环体 `printf(" ");`

执行内层循环表达式 3,  $j++$  为 3,  $j < i$  不成立, 跳出内层循环

执行 `printf("*");`

执行 `printf("\n");` 换行

至此外层循环体执行完, 计算外层循环的表达式 3,  $i++$ , 此时  $i$  为 4. 使得循环条件  $i \leq 4$  成立, 进入外层 for 循环体

外层 for 循环第 4 次 此时  $i$  为 4

内层循环  $j=1$ , 使得循环条件  $j < i$  成立

第 1 次执行内层循环体 `printf(" ");`

执行内层循环表达式 3,  $j++$  为 2,  $j < i$  成立, 再次执行内层循环

第 2 次执行内层循环体 `printf(" ");`

执行内层循环表达式 3,  $j++$  为 3,  $j < i$  成立, 再次执行内层循环

第 3 次执行内层循环体 `printf(" ");`

执行内层循环表达式 3,  $j++$  为 4,  $j < i$  不成立, 跳出内层循环

执行 `printf("*");`

执行 `printf("\n");` 换行

至此外层循环体执行完, 计算外层循环的表达式 3,  $i++$ , 此时  $i$  为 5. 使得循环条件  $i \leq 4$  不成立, 跳出外层 for 循环体

## 数组

1.

```
#include <stdio.h>
main()
{   int    i, a[10];
    for(i=9;i>=0;i--)
        a[i]=10-i;
    printf(“%d%d%d”,a[2],a[5],a[8]);
}
```

运行结果为:

852

详见 p143-146. 例题 6.1 一定看懂!

首先定义整型变量 i, 整型数组 a, a 的长度为 10, 即 a 中包含 10 个整型元素 (整型变量)

执行 for 循环语句

初值 i=9, 使得循环条件 i>=0 成立, 执行循环体

第 1 次循环

执行 a[i]=10-i 等价于 a[9]=10-9=1

计算表达式 3, 即 i--, i 为 8, 使得循环条件 i>=0 成立, 继续执行循环体

第 2 次循环

执行 a[i]=10-i 等价于 a[8]=10-8=2

计算表达式 3, 即 i--, i 为 7, 使得循环条件 i>=0 成立, 继续执行循环体

第 3 次循环

执行 a[i]=10-i 等价于 a[7]=10-7=3

计算表达式 3, 即 i--, i 为 6, 使得循环条件 i>=0 成立, 继续执行循环体

第 4 次循环

执行 a[i]=10-i 等价于 a[6]=10-6=4

计算表达式 3, 即 i--, i 为 5, 使得循环条件 i>=0 成立, 继续执行循环体

第 5 次循环

执行 a[i]=10-i 等价于 a[5]=10-5=5

计算表达式 3, 即 i--, i 为 4, 使得循环条件 i>=0 成立, 继续执行循环体

第 6 次循环

执行 a[i]=10-i 等价于 a[4]=10-4=6

计算表达式 3, 即 i--, i 为 3, 使得循环条件 i>=0 成立, 继续执行循环体

第 7 次循环

执行 a[i]=10-i 等价于 a[3]=10-3=7

计算表达式 3, 即 i--, i 为 2, 使得循环条件 i>=0 成立, 继续执行循环体

第 8 次循环

执行 a[i]=10-i 等价于 a[2]=10-2=8



计算表达式 3, 即  $i--$ ,  $i$  为 1, 使得循环条件  $i \geq 0$  成立, 继续执行循环体  
第 9 次循环

执行  $a[i]=10-i$  等价于  $a[1]=10-1=9$

计算表达式 3, 即  $i--$ ,  $i$  为 0, 使得循环条件  $i \geq 0$  成立, 继续执行循环体  
第 10 次循环

执行  $a[i]=10-i$  等价于  $a[0]=10-0=10$

计算表达式 3, 即  $i--$ ,  $i$  为 -1, 使得循环条件  $i \geq 0$  不成立, 跳出循环体

## 2.

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

```
main()
```

```
{ int i,a[6];
```

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

```
        a[i]=i;
```

```
    for (i=5; i>=0 ; i--)
```

```
        printf("%3d",a[i]);
```

```
}
```

运行结果为:

**5 4 3 2 1 0**

首先定义整型变量  $i$ , 整型数组  $a$ ,  $a$  的长度为 6, 即  $a$  中包含 6 个整型元素 (整型变量)

执行第一个 for 循环语句

初值  $i=0$ , 使得循环条件  $i<6$  成立, 执行循环体

第 1 次循环

执行  $a[i]=i$  等价于  $a[0]=0$

计算表达式 3, 即  $i++$ ,  $i$  为 1, 使得循环条件  $i<6$  成立, 继续执行循环体

第 2 次循环

执行  $a[i]=i$  等价于  $a[1]=1$

计算表达式 3, 即  $i++$ ,  $i$  为 2, 使得循环条件  $i<6$  成立, 继续执行循环体

第 3 次循环

执行  $a[i]=i$  等价于  $a[2]=2$

计算表达式 3, 即  $i++$ ,  $i$  为 3, 使得循环条件  $i<6$  成立, 继续执行循环体

第 4 次循环

执行  $a[i]=i$  等价于  $a[3]=3$

计算表达式 3, 即  $i++$ ,  $i$  为 4, 使得循环条件  $i<6$  成立, 继续执行循环体

第 5 次循环

执行  $a[i]=i$  等价于  $a[4]=4$

计算表达式 3, 即  $i++$ ,  $i$  为 5, 使得循环条件  $i<6$  成立, 继续执行循环体

第 6 次循环

执行  $a[i]=i$  等价于  $a[5]=5$

计算表达式 3, 即  $i++$ ,  $i$  为 6, 使得循环条件  $i<6$  不成立, 结束循环

执行第二个 for 循环语句

初值  $i=5$ , 使得循环条件  $i \geq 0$  成立, 执行循环体

第 1 次循环

执行 `printf("%3d",a[i]);` 即输出 `a[5]` 的值  
计算表达式 3, 即 `i--`, `i` 为 4, 使得循环条件 `i>=0` 成立, 继续执行循环体  
第 2 次循环  
执行 `printf("%3d",a[i]);` 即输出 `a[4]` 的值  
计算表达式 3, 即 `i--`, `i` 为 3, 使得循环条件 `i>=0` 成立, 继续执行循环体  
第 3 次循环  
执行 `printf("%3d",a[i]);` 即输出 `a[3]` 的值  
计算表达式 3, 即 `i--`, `i` 为 2, 使得循环条件 `i>=0` 成立, 继续执行循环体  
第 4 次循环  
执行 `printf("%3d",a[i]);` 即输出 `a[2]` 的值  
计算表达式 3, 即 `i--`, `i` 为 1, 使得循环条件 `i>=0` 成立, 继续执行循环体  
第 5 次循环  
执行 `printf("%3d",a[i]);` 即输出 `a[1]` 的值  
计算表达式 3, 即 `i--`, `i` 为 0, 使得循环条件 `i>=0` 成立, 继续执行循环体  
第 6 次循环  
执行 `printf("%3d",a[i]);` 即输出 `a[0]` 的值  
计算表达式 3, 即 `i--`, `i` 为 -1, 使得循环条件 `i>=0` 不成立, 结束循环

### 3.

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

```
main( )
```

```
{ int i,k,a[10],p[3];
```

```
    k=5;
```

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

```
        a[i]=i;
```

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

```
        p[i]=a[i*(i+1)];
```

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

```
        k+=p[i]*2;
```

```
    printf("%d\n",k);
```

```
}
```

运行结果为: **21**

首先定义整型变量 `i`, `k`, 整型数组 `a`, `a` 的长度为 10, 整型数组 `p`, `p` 的长度为 3

`k` 初值为 5

第一个 `for` 循环语句为数组 `a` 进行初始化

执行完第一个 `for` 语句后, `a[0]=0`, `a[1]=1`, `a[2]=2`, `a[3]=3`, `a[4]=4`, `a[5]=5`, `a[6]=6`, `a[7]=7`, `a[8]=8`, `a[9]=9` (循环过程略)

第二个 `for` 循环语句为数组 `p` 进行初始化

初值 `i=0`, 使得循环条件 `i<3` 成立, 执行循环体

第 1 次循环

执行 `p[i]=a[i*(i+1)];` 即 `p[0]=a[0*(0+1)]=a[0]=0`

计算表达式 3, 即 `i++`, `i` 为 1, 使得循环条件 `i<3` 成立, 继续执行循环体

第 2 次循环

执行  $p[i]=a[i*(i+1)]$ ; 即  $p[1]=a[1*(1+1)]=a[2]=2$

计算表达式 3, 即  $i++$ ,  $i$  为 2, 使得循环条件  $i<3$  成立, 继续执行循环体

第 3 次循环

执行  $p[i]=a[i*(i+1)]$ ; 即  $p[2]=a[2*(2+1)]=a[6]=6$

计算表达式 3, 即  $i++$ ,  $i$  为 3, 使得循环条件  $i<3$  不成立, 结束循环

第三个 for 循环语句

初值  $i=0$ , 使得循环条件  $i<3$  成立, 执行循环体

第 1 次循环

执行  $k+=p[i]*2$ ; 即  $k=5+p[0]*2=5+0=5$

计算表达式 3, 即  $i++$ ,  $i$  为 1, 使得循环条件  $i<3$  成立, 继续执行循环体

第 2 次循环

执行  $k+=p[i]*2$ ; 即  $k=5+p[1]*2=5+2*2=9$

计算表达式 3, 即  $i++$ ,  $i$  为 2, 使得循环条件  $i<3$  成立, 继续执行循环体

第 3 次循环

执行  $k+=p[i]*2$ ; 即  $k=9+p[2]*2=9+6*2=21$

计算表达式 3, 即  $i++$ ,  $i$  为 3, 使得循环条件  $i<3$  不成立, 结束循环

#### 4.

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

```
int m[3][3]={1},{2},{3}};
```

```
int n[3][3]={1,2,3};
```

```
main( )
```

```
{ printf(“%d,”, m[1][0]+n[0][0]);
```

```
printf(“%d\n”,m[0][1]+n[1][0]);
```

```
}
```

运行结果为:

3,0

详见教材 P149~152, 图 6.7 看懂!

首先定义整型二维数组  $m$ ,  $m$  为 3 行, 3 列的二维矩阵, 并对其以行的形式初始化

$m[0][0]=1$   $m[0][1]=0$   $m[0][2]=0$

$m[1][0]=2$   $m[1][1]=0$   $m[1][2]=0$

$m[2][0]=3$   $m[2][1]=0$   $m[2][2]=0$

定义整型二维数组  $n$ ,  $n$  为 3 行, 3 列的二维矩阵

$n[0][0]=1$   $n[0][1]=2$   $n[0][2]=3$

$n[1][0]=0$   $n[1][1]=0$   $n[1][2]=0$

$n[2][0]=0$   $n[2][1]=0$   $n[2][2]=0$

因此  $m[1][0]+n[0][0]=2+1=3$

$m[0][1]+n[1][0]=0+0=0$

5.

```
#include <stdio.h>
main()
{ int i;
  int x[3][3]={1,2,3,4,5,6,7,8,9};
  for (i=1; i<3; i++)
    printf("%d  ",x[i][3-i]);
}
```

运行结果为:

**6 8**

首先按存储顺序为数组 x 初始化

x[0][0]=1 x[0][1]=2 x[0][2]=3

x[1][0]=4 x[1][1]=5 x[1][2]=6

x[2][0]=7 x[2][1]=8 x[2][2]=9

初值 i=1, 使得循环条件 i<3 成立, 执行循环体

第 1 次循环

执行 printf("%d ",x[i][3-i]), 打印出 x[i][3-i], 即 x[1][2]的值

计算表达式 3, 即 i++, i 为 2, 使得循环条件 i<3 成立, 继续执行循环体

第 2 次循环

执行 printf("%d ",x[i][3-i]), 打印出 x[i][3-i], 即 x[2][1]的值

计算表达式 3, 即 i++, i 为 3, 使得循环条件 i<3 成立, 结束循环

6.

```
#include <stdio.h>
main( )
{ int n[3][3], i, j;
  for(i=0; i<3; i++ )
    {for(j=0; j<3; j++ )
      {n[i][j]=i+j;
       printf("%d  ", n[i][j]);
      }
    }
}
```

运行结果为:

**0 1 2**

**1 2 3**

**2 3 4**

循环变量 i 为 0, 循环条件 i<3 成立, 执行循环体

外层 for 第 1 次循环 相当于输出第 1 行

内层 for 循环 j 初值为 0, 循环条件  $j < 3$  成立, 执行循环体

内层 for 第 1 次循环

执行  $n[i][j]=i+j$ ; 即  $n[0][0]=0+0=0$ ;

执行  $\text{printf}(\text{"%d"}, n[i][j])$ ;

执行内层循环表达式 3,  $j++$ ,  $j$  为 1,  $j < 3$  成立, 继续执行内层循环体

内层 for 第 2 次循环

执行  $n[i][j]=i+j$ ; 即  $n[0][1]=0+1=1$ ;

执行  $\text{printf}(\text{"%d"}, n[i][j])$ ;

执行内层循环表达式 3,  $j++$ ,  $j$  为 2,  $j < 3$  成立, 继续执行内层循环体

内层 for 第 3 次循环

执行  $n[i][j]=i+j$ ; 即  $n[0][2]=0+2=2$ ;

执行  $\text{printf}(\text{"%d"}, n[i][j])$ ;

执行内层循环表达式 3,  $j++$ ,  $j$  为 3,  $j < 3$  不成立, 结束内层循环

执行  $\text{printf}(\text{"\n"})$ ;

执行外层 for 语句的表达式 3,  $i++$ ,  $i$  为 1,  $i < 3$  成立, 继续执行外层循环体

外层 for 第 2 次循环 相当于输出第 2 行

内层 for 循环 j 初值为 0, 循环条件  $j < 3$  成立, 执行循环体

内层 for 第 1 次循环

执行  $n[i][j]=i+j$ ; 即  $n[1][0]=1+0=1$ ;

执行  $\text{printf}(\text{"%d"}, n[i][j])$ ;

执行内层循环表达式 3,  $j++$ ,  $j$  为 1,  $j < 3$  成立, 继续执行内层循环体

内层 for 第 2 次循环

执行  $n[i][j]=i+j$ ; 即  $n[1][1]=1+1=2$ ;

执行  $\text{printf}(\text{"%d"}, n[i][j])$ ;

执行内层循环表达式 3,  $j++$ ,  $j$  为 2,  $j < 3$  成立, 继续执行内层循环体

内层 for 第 3 次循环

执行  $n[i][j]=i+j$ ; 即  $n[1][2]=1+2=3$ ;

执行  $\text{printf}(\text{"%d"}, n[i][j])$ ;

执行内层循环表达式 3,  $j++$ ,  $j$  为 3,  $j < 3$  不成立, 结束内层循环

执行  $\text{printf}(\text{"\n"})$ ;

执行外层 for 语句的表达式 3,  $i++$ ,  $i$  为 1,  $i < 3$  成立, 继续执行外层循环体

外层 for 第 2 次循环 相当于输出第 3 行

内层 for 循环 j 初值为 0, 循环条件  $j < 3$  成立, 执行循环体

内层 for 第 1 次循环

执行  $n[i][j]=i+j$ ; 即  $n[2][0]=2+0=2$ ;

执行  $\text{printf}(\text{"%d"}, n[i][j])$ ;

执行内层循环表达式 3,  $j++$ ,  $j$  为 1,  $j < 3$  成立, 继续执行内层循环体

内层 for 第 2 次循环

执行  $n[i][j]=i+j$ ; 即  $n[2][1]=2+1=3$ ;

执行  $\text{printf}(\text{"%d"}, n[i][j])$ ;

执行内层循环表达式 3,  $j++$ ,  $j$  为 2,  $j < 3$  成立, 继续执行内层循环体

内层 for 第 3 次循环

执行  $n[i][j]=i+j$ ; 即  $n[2][2]=2+2=4$ ;

执行内层循环表达式 3, j++, j 为 3, j<3 不成立, 结束内层循环  
执行 printf("\n");  
执行外层 for 语句的表达式 3, i++, i 为 3, i<3 不成立, 结束外层循环

7.

```
#include <stdio.h>
main()
{
    char diamond[][5]={{'_','_','*','*'},{'_','*','_','*'},
    {'*','_','_','_','*'},{'_','*','_','*','*'},{'_','_','_','*'}};
    int i,j;
    for(i=0;i<5;i++)
    {
        for(j=0;j<5;j++)
            printf("%c",diamond[i][j]);
        printf("\n");
    }
```

}注: “\_”代表一个空格。

运行结果为:

```

      *
    *  *
  *    *
 *      *
*        *

```

8.

```
#include <stdio.h>
main()
{ int i, f[10];
  f[0]=f[1]=1;
  for(i=2;i<10;i++)
      f[i]=f[i-2]+f[i-1];
  for(i=0;i<10;i++)
      { if(i%4==0)
          printf("\n");
        printf("%d  ",f[i]);
      }
}
```

运行结果为:

```
1  1  2  3
5  8 13 21
34 55
```

9.

```
#include "stdio.h"
func(int b[])
{ int j;
  for(j=0;j<4;j++)
    b[j]=j;
}
main()
{ int a[4], i;
  func(a);
  for(i=0; i<4; i++)
    printf("%2d",a[i]);
}
```

运行结果为:

**0 1 2 3**

**详见教材 P194**

定义函数 func

函数头: 未定义函数的类型, 则系统默认为 int 型。函数 func 的形参为整型数组名, 即只接收整型数组地址。

函数体: 定义整型变量 j

循环变量初值 (表达式 1) j=0, 使得循环条件 (表达式 2) j<4 成立, 执行循环体第 1 次循环

执行 b[j]=j; 即 b[0]=0;

执行循环变量自增 (及表达式 3) j++, j 为 1, 使得 j<4 成立, 继续执行循环体第 2 次循环

b[1]=1;

j++, j 为 2, 使得 j<4 成立, 继续执行循环体第 3 次循环

b[2]=2;

j++, j 为 3, 使得 j<4 成立, 继续执行循环体第 4 次循环

b[3]=3;

j++, j 为 4, 使得 j<4 不成立, 结束循环

main 函数:

定义整型变量 i 和数组 a, 其长度为 4,

func(a);表示调用函数 func, 并以数组名 a 作为调用的实参 (数组名在 C 语言中表示数组所在内存空间的首地址, 在以数组名作为实参时, 形参与实参公用存储空间, 因此对数组 b 的操作, 即对数组 a 的操作。)



10.

```
#include <stdio.h>
main ()
{float fun(float x[]);
  float ave,a[3]={4.5, 2, 4};
  ave=fun (a) ;
  printf("ave=%7.2f",ave);
}
float fun (float x[])
{int j;
  float aver=1;
  for (j=0;j<3;j++)
    aver=x[j]*aver;
  return (aver);
}
```

运行结果为:

**ave= 36.00**

11.

```
#include <stdio.h>
main()
{int a[2][3]={{1,2,3},{4,5,6}};
  int b[3][2],i,j;
  for(i=0;i<=1;i++)
    {for(j=0;j<=2;j++)
      b[j][i]=a[i][j];
    }
  for(i=0;i<=2;i++)
    {for(j=0;j<=1;j++)
      printf("%5d",b[i][j]);
    }
}
```

运行结果为:

**1 4 2 5 3 6**

12.

```
#include <stdio.h>
f(int b[],int n)
{int i,r;
  r=1;
  for (i=0;i<=n;i++)
    r=r*b[i];
  return (r);
}
```

```

}
main()
{int x,a[]={1,2,3,4,5,6,7,8,9};
  x=f(a,3);
  printf("%d\n",x);
}

```

运行结果为:

**24**

13.

```

#include "stdio.h"

main()
{int j,k;
  static int x[4][4],y[4][4];
  for(j=0;j<4;j++)
    for(k=j;k<4;k++)
      x[j][k]=j+k;
  for(j=0;j<4;j++)
    for(k=j;k<4;k++)
      y[k][j]=x[j][k];

  for(j=0;j<4;j++)
    for(k=0;k<4;k++)
      printf("%d,",y[j][k]);
}

```

运行结果为:

**0,0,0,0,1,2,0,0,2,3,4,0,3,4,5,6**

## 函数

1.

```

#include <stdio.h>
int Sub(int a, int b)
{return (a- b);}
main()
{int  x, y, result = 0;
scanf("%d,%d", &x,&y );
result = Sub(x,y );
printf("result = %d\n",result);
}

```

当从键盘输入:6,3 运行结果为:

**result =3**

2.

```

#include <stdio.h>
int min( int x, int y )
{ int m;
  if ( x>y ) m = x;
else      m = y;
  return(m);
}
main() {
  int a=3,b=5,abmin ;
  abmin = min(a,b);
  printf("min is %d", abmin);
}

```

运行结果为:

**min is 5**

3.

```

#include<stdio.h>
func(int x) {
  x=10;
  printf("%d,",x);
}
main( )
{ int x=20;
  func(x);
  printf("%d", x);
}

```

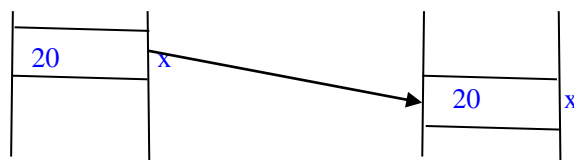
运行结果为:

**10, 20**

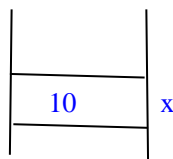
在 main 函数中调用函数 func，main 函数将 20 作为实参穿给 func，并转向开始执行 func.

main()对应的内存

func ( ) 对应的内存



func()执行 x=10;, 其内存中 x 变为 10.



func()执行 printf(“%d, ”,x); 即输出 func 函数对应内存中 x 的值,输出的是 10。  
至此, func 函数执行结束, 返回 main 函数。

main 函数执行 printf(“%d”, x);此时输出 main 函数对应内存中的 x, 即 20

4.

```
#include <stdio.h>
int m=4;
int func(int x,int y)
{   int m=1;
    return(x*y-m);
}
main()
{int a=2,b=3;
    printf("%d\n",m);
    printf("%d\n",func(a,b)/m);
}
```

运行结果为:

4  
1

整型变量 m 在函数外定义, 因此 m 为全局变量, 其作用于范围为其定义位置开始, 一直到整个程序结束。因此 func 与 main 函数都可以访问 m

程序首先执行 main 函数

执行 printf(“%d\n”,m); 即输出 m 中的值 4, 并换行。

执行 printf(“%d\n”,func(a,b)/m);即输出表达式 func(a,b)/m 的值, 为了计算该表达式, 需要调用函数 func。此时 main 将 a,b 中的 2 和 3 值作为实参传递给 func 的 x 和 y  
程序开始转向执行 func 函数, 此时 func 中的 x 为 2, y 为 3

执行 int m=1; 此句定义了一个局部变量 m 并赋值为 1。m 的作用域为其所在的复合语句, 即 func 的函数体, 因此在 func 的函数体中, 有限访问局部变量 m。

执行 return(x\*y-m); 即 return (2\*3-1) ;返回的是整数 5.

func 函数返回至 main 函数中的被调用处

main 函数中 func(a,b)的值为 5, func(a,b)/m=5/4=1,注意, 在 main 函数中访问的 m 为全局变量 m, 此时 main 函数无法访问 func 中的 m, 因为不在 func 中 m 的作用域。

5.

```
#include <stdio.h>
int fun(int a, int b)
```

```

{ if(a>b)    return(a);
  else      return(b);
}
main()
{ int x=15, y=8, r;
  r= fun(x,y);
  printf("r=%d\n", r);
}

```

运行结果为: **r=15**

程序首先执行 main 函数

执行 r= fun(x,y);即将 func(x,y)的值赋给 r, 为了计算该表达式, 需要调用函数 func。此时 main 将 x,y 中的 15 和 8 值作为实参传递给 func 的 a 和 b

程序开始转向执行 func 函数, 此时 func 中的 a 为 15, b 为 8

执行 if 语句;判断 if 后面的表达式, a>b 成立, 因此执行相应的操作 return(a); 即返回 a 的值。

func 函数返回至 main 函数中的被调用处

main 函数中 func(x,y)的值为 15, 即将 15 赋给 r。

执行 printf("r=%d\n", r); 即输出 r=15

6.

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

```
int fac(int n)
```

```
{ int f=1,i;
```

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

```
      f=f * i;
```

```
  return(f);
```

```
}
```

```
main()
```

```
{ int j,s;
```

```
  scanf("%d",&j);
```

```
  s=fac(j);
```

```
  printf("%d!=%d\n",j,s);
```

```
}
```

如果从键盘输入 3, 运行结果为: **3!=6**

程序首先执行 main 函数

执行 r= fun(x,y);即将 func(x,y)的值赋给 r, 为了计算该表达式, 需要调用函数 func。此时 main 将 x,y 中的 15 和 8 值作为实参传递给 func 的 a 和 b

程序开始转向执行 func 函数, 此时 func 中的 a 为 15, b 为 8

执行 if 语句;判断 if 后面的表达式, a>b 成立, 因此执行相应的操作 return(a); 即返回 a 的值。

func 函数返回至 main 函数中的被调用处

main 函数中 func(x,y)的值为 15, 即将 15 赋给 r。

执行 printf("r=%d\n", r); 即输出 r=15

7.

```
#include <stdio.h>
unsigned fun6(unsigned num)
{ unsigned k=1;
  do
  { k*=num%10;
    num/=10;
  }while(num);
  return k;
}
main()
{ unsigned n=26;
  printf("%d\n",fun6(n));
}
```

运行结果为: **12**

程序首先执行 main 函数

执行 `printf("%d\n",fun6(n));` 即输出表达式 `func(6)` 的值, 为了计算该表达式, 需要调用函数 `func`。此时 `main` 将 `n` 中的 26 作为实参传递给 `func` 的 `num`

程序开始转向执行 `func` 函数, 此时 `func` 中的 `num` 为 26

执行 `do-while` 语句

第 1 次循环

执行 `k*=num%10`, 即 `k=k*(num%10)=1*(26%10)=6`

执行 `num/=10`; 即 `num=num/10=26/10=2`

`while` 后面循环条件为 `num`, 此时 `num` 为 2, 是非 0 值, 即表示循环条件成立, 继续执行循环体。此时 `k` 为 6

第 2 次循环

执行 `k*=num%10`, 即 `k=k*(num%10)=6*(2%10)=12`

执行 `num/=10`; 即 `num=num/10=2/10=0`

`while` 后面循环条件为 `num`, 此时 `num` 为 0, 表示循环条件不成立, 结束循环

执行 `return k`; 即返回至 `main` 函数中的被调用处

执行 `main` 函数

继续执行 `printf("%d\n",fun6(n));` 即输出 12

8.

```
#include <stdio.h>
int max(int x, int y);
main()
{ int a,b,c;
  a=7;b=8;
  c=max(a,b);
  printf("Max is %d",c);
}
```

```

}
max(int x, int y)
{ int z;
z=x>y? x : y;
return(z) ;
}

```

运行结果为:

**Max is 8**

## 指针

1.

```

#include <stdio.h>
main ( )
{ int x[] = {10, 20, 30, 40, 50 };
int *p ;
p=x;
printf ( “%d”, *(p+2) );
}

```

运行结果为:

**30**

首先定义一个整型数组 `x`，`x` 的长度为 5;然后定义一个指针变量 `p`;对 `p` 进行初始化，将数组 `x` 的地址赋给 `p`。因此此时 `p` 中存放的数组 `x` 的首地址，即数组中第一个元素 `x[0]` 的地址。

然后执行 `printf` 语句，输出表达式 `*(p+2)` 的值。`p+2` 表示以 `p` 当前指向的位置起始，之后第 2 个元素的地址，即 `a[2]` 的地址。`*(p+2)` 则表示该地址内所存放的内容，即 `a[2]` 的值 30，因此输出 30

2.

```

#include <stdio.h>
main( )
{ char s[]="abcdefg";
char *p;
p=s;
printf("ch=%c\n",*(p+5));
}

```

运行结果为:

**ch=f**

首先定义一个字符型数组 `s`，并用字符串 `abcdefg` 对 `s` 进行初始化; 然后定义一个字符型指针变量 `p`; 对 `p` 进行初始化，将数组 `s` 的地址赋给 `p`。因此此时 `p` 中存放的数组 `s` 的首地址，即数组中第一个元素 `s[0]` 的地址。

然后执行 `printf` 语句，输出表达式 `*(p+5)` 的值。`p+5` 表示以 `p` 当前指向的位置起始，之后



第 5 个元素的地址，即 a[5]的地址。\*(p+5)则表示该地址内所存放的内容，即 a[5]的值 f，因此输出 ch=f

3.

```
#include<stdio.h>
main ( )
{ int a[]={1, 2, 3, 4, 5} ;
  int x, y, *p;
  p=a;
  x=*(p+2);
  printf("%d:  %d \n", *p, x);
}
```

运行结果为:

**1:3**

首先定义一个整型数组 a，并对 a 进行初始化；然后定义整型变量 x,y，整型指针变量 p；再将数组 a 的地址赋给 p。因此此时 p 中存放的数组 a 的首地址，即数组中第一个元素 a[0]的地址。执行 x=\*(p+2)；p+2 表示以 p 当前所指向的位置起始，之后第 2 个元素的地址，即 a[2]的地址。\*(p+2)则表示该地址内所存放的内容，即 a[2]的值 3，然后再把 3 赋给 x

然后执行 printf 语句，先输出表达式\*p 的值。此时\*p 表示的是 p 所指向变量的内容，即 a[0]的值 1。再输出一个冒号。然后再输出 x 中的值 3。

4.

```
#include<stdio.h>
main()
{ int arr[ ]={30,25,20,15,10,5}, *p=arr;
  p++;
  printf("%d\n",*(p+3));
}
```

运行结果为: **10**

首先定义一个整型数组 arr，并对 arr 进行初始化；然后定义整型指针变量 p；再将数组 arr 的地址赋给 p。因此此时 p 中存放的数组 arr 的首地址，即数组中第一个元素 a[0]的地址。

执行 p++，即 p=p+1。p+1 表示以 p 当前所指向的位置起始，之后第 1 个元素的地址，即 arr[1]的地址，然后再将 arr[1]的地址赋给 p，执行完此语句后，p 不再指向 arr[0]，而是指向 arr[1]。

然后执行 printf 语句，输出表达式\*(p+3)的值。p+3 表示以 p 当前指向的位置起始(此时 p 指向 arr[1])，之后第 3 个元素的地址，即 arr[4]的地址。\*(p+3)则表示该地址内所存放的内容，即 arr[4]的值 10，因此输出 10

5.

```
#include <stdio.h>
main( )
{ int a[ ]={1, 2, 3, 4, 5, 6};
```

```

int x, y, *p;
p = &a[0];
x = *(p+2);
y = *(p+4);
printf("p=%d, x=%d, y=%d\n", *p, x, y);
}

```

运行结果为:

**\*p=1, x=3, y=5**

首先定义一个整型数组 a, 并对 a 进行初始化; 然后定义整型变量 x,y, 整型指针变量 p; 再将数组元素 a[0]的地址赋给 p。

执行 x=\*(p+2); p+2 表示以 p 当前所指向的位置起始, 之后第 2 个元素的地址, 即 a[2]的地址。\*(p+2)则表示该地址内所存放的内容, 即 a[2]的值 3, 然后再把 3 赋给 x

执行 y = \*(p+4); p+4 表示以 p 当前所指向的位置起始, 之后第 4 个元素的地址, 即 a[4]的地址。\*(p+4)则表示该地址内所存放的内容, 即 a[4]的值 5, 然后再把 5 赋给 y

执行 printf 语句, 先输出表达式\*p 的值。此时\*p 表示的是 p 所指向变量的内容, 即 a[0]的值 1。再输 x 的值 3。再输出 y 的值 5。

## 6.

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

```

main( )
{ static char a[ ]="Program", *ptr;
  for(ptr=a, ptr<a+7; ptr+=2)
    putchar(*ptr);
}

```

运行结果为:

**Porm**

首先定义一个字符型数组 a, 并对 a 进行初始化; 然后定义字符型指针变量 p;

执行 for 语句 ptr=a 为表达式 1, 将数字 a 的地址赋给 ptr; 表达式 2 (循环条件) ptr<a+7; 表达式 3 为 ptr+=2, 即 ptr= ptr+2;

第 1 次执行循环体

执行 putchar(\*ptr); 即输出\*ptr 所对应的字符。此时 ptr 指向数组中的第 1 个元素, 即 a[0],因此\*ptr 表示 a[0]中的值, 即'P'.

执行完循环体, 转向执行表达式 3, 即 ptr= ptr+2。ptr+2 表示以 ptr 当前所指向的位置起始, 之后第 2 个元素的地址,即 a[2]的地址, 然后将 a[2]的地址赋给 ptr。a[2]的地址等价于 a+2, 因此循环条件 ptr<a+7 成立, 继续执行循环体

第 2 次执行循环体

执行 putchar(\*ptr); 即输出\*ptr 所对应的字符。此时 ptr 指向数组中的第 3 个元素, 即 a[2],因此\*ptr 表示 a[2]中的值, 即'o'.

执行完循环体, 转向执行表达式 3, 即 ptr= ptr+2。ptr+2 表示以 ptr 当前所指向的位置起始, 之后第 2 个元素的地址,即 a[4]的地址, 然后将 a[4]的地址赋给 ptr。a[4]的地址等价于 a+4, 因此循环条件 ptr<a+7 即 a+4<a+7 成立, 继续执行循环体

第 3 次执行循环体

执行 putchar(\*ptr); 即输出\*ptr 所对应的字符。此时 ptr 指向数组中的第 5 个元素, 即

a[4],因此\*ptr 表示 a[4]中的值,即'r'.

执行完循环体,转向执行表达式 3,即 ptr= ptr+2。ptr+2 表示以 ptr 当前所指向的位置起始,之后第 2 个元素的地址,即 a[6]的地址,然后将 a[6]的地址赋给 ptr。a[6]的地址等价于 a+6,因此循环条件 ptr<a+7 即 a+6<a+7 成立,继续执行循环体

第 4 次执行循环体

执行 putchar(\*ptr); 即输出\*ptr 所对应的字符。此时 ptr 指向数组中的第 7 个元素,即 a[6],因此\*ptr 表示 a[6]中的值,即'm'.

执行完循环体,转向执行表达式 3,即 ptr= ptr+2。ptr+2 表示以 ptr 当前所指向的位置起始,之后第 2 个元素的地址,即 a[8]的地址,然后将 a[8]的地址赋给 ptr。a[6]的地址等价于 a+8,因此循环条件 ptr<a+7 即 a+8<a+7 不成立,结束循环。

## 7.

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

```
char s[]="ABCD";
```

```
main()
```

```
{ char *p;
```

```
  for(p=s;p<s+4;p++)
```

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

```
}
```

运行结果为:

**A**   **ABCD**

**B**   **BCD**

**C**   **CD**

**D**   **D**

首先定义一个字符型数组 s,并对 s 进行初始化;数组 s 是全局变量,其有效范围从其定义开始至整个程序结束。

执行 main 函数

定义一个字符型指针 p。

执行 for 语句 p=s 为表达式 1,将数字 s 的首地址赋给 p;表达式 2(循环条件)p<s+4;表达式 3 为 p++,即 p=p+1;

第 1 次执行循环体

执行 printf("%c %s\n",\*p,p);即以字符%c 形式输出\*p 所对应的字符。此时 p 指向数组中的第 1 个元素,即 s[0],因此\*p 表示 a[0]中的值,即'A'。然后再以字符串%s 的形式输出以 p 中地址为首地址的整个字符串,即输出 ABCD

执行完循环体,转向执行表达式 3,即 p= p+1。p+1 表示以 p 当前所指向的位置起始,之后 1 个元素的地址,即 s[1]的地址,然后将 a[1]的地址赋给 p。s[1]的地址等价于 s+1,因此循环条件 p<s+4 成立,继续执行循环体

第 2 次执行循环体

执行 printf("%c %s\n",\*p,p);即以字符%c 形式输出\*p 所对应的字符。此时 p 指向数组中的第 2 个元素,即 s[1],因此\*p 表示 s[1]中的值,即'B'。然后再以字符串%s 的形式输出以 p 中地址为首地址的整个字符串,此时 p 指向 s[1],即从 s[1]开始,依次输出后面的字符串,因此又输出 BCD

执行完循环体，转向执行表达式 3，即  $p = p + 1$ 。 $p + 1$  表示以  $p$  当前所指向的位置起始，之后 1 个元素的地址，即  $s[2]$  的地址，然后将  $a[2]$  的地址赋给  $p$ 。 $s[2]$  的地址等价于  $s + 2$ ，因此循环条件  $p < s + 4$  成立，继续执行循环体

#### 第 3 次执行循环体

执行 `printf("%c %s\n", *p, p);` 即以字符  $\%c$  形式输出  $*p$  所对应的字符。此时  $p$  指向数组中的第 3 个元素，即  $s[2]$ ，因此  $*p$  表示  $s[2]$  中的值，即 'C'。然后再以字符串  $\%s$  的形式输出以  $p$  中地址为首地址的整个字符串，此时  $p$  指向  $s[2]$ ，即从  $s[2]$  开始，依次输出后面的字符串，因此又输出 CD

执行完循环体，转向执行表达式 3，即  $p = p + 1$ 。 $p + 1$  表示以  $p$  当前所指向的位置起始，之后 1 个元素的地址，即  $s[2]$  的地址，然后将  $s[2]$  的地址赋给  $p$ 。 $s[2]$  的地址等价于  $s + 3$ ，因此循环条件  $p < s + 4$  成立，继续执行循环体

#### 第 4 次执行循环体

执行 `printf("%c %s\n", *p, p);` 即以字符  $\%c$  形式输出  $*p$  所对应的字符。此时  $p$  指向数组中的第 4 个元素，即  $s[3]$ ，因此  $*p$  表示  $s[3]$  中的值，即 'D'。然后再以字符串  $\%s$  的形式输出以  $p$  中地址为首地址的整个字符串，即输出 D

执行完循环体，转向执行表达式 3，即  $p = p + 1$ 。 $p + 1$  表示以  $p$  当前所指向的位置起始，之后 1 个元素的地址，即  $s[3]$  的地址，然后将  $s[3]$  的地址赋给  $p$ 。 $s[3]$  的地址等价于  $s + 4$ ，因此循环条件  $p < s + 4$  不成立，结束循环

## 结构体

### 1.

```
#include<stdio.h>
struct st
{ int x;
  int y;
} a[2]={5, 7, 2, 9};
main()
{
printf("%d\n",a[0].y*a [1].x);
}
```

运行结果是：

14

首先是定义结构体  $st$ ， $st$  中共有两个整型成员  $x$ ， $y$ 。

然后定义一个  $st$  类型的数组  $a$ ， $a$  的长度为 2，即数组中含有两个  $st$  类型的元素，分别是  $a[0]$  和  $a[1]$ 。对  $a$  进行初始化，此题是按照储存顺序进行初始化，即将 5 赋给  $a[0]$  中的  $x$ （即  $a[0].x = 5$ ）；将 7 赋给  $a[0]$  中的  $y$ （即  $a[0].y = 7$ ）；将 2 赋给  $a[1]$  中的  $x$ （即  $a[1].x = 2$ ）；将 9 赋给  $a[1]$  中的  $y$ （即  $a[1].y = 9$ ）；

执行  $main$  函数，输出表达式  $a[0].y * a [1].x$  的值，即  $7 * 2$  的值

5	a[0].x } a[0]
7	a[0].y }
2	a[1].x } a[1]
9	a[1].y }

2.

```
#include<stdio.h>
main( )
{struct stu
    {int num;
    char a[5];
    float score;
    }m={1234,"wang",89.5};
printf("%d,%s,%f",m.num,m.a,m.score);
}
```

运行结果是:

**1234,wang,89.5**

3.

```
#include<stdio.h>
struct cmplx
{ int x;
  int y;
} cnum[2]={1, 3, 2, 7};
main( )
{
    printf("%d\n", cnum[0].y * cnum[1].x );
}
```

运行结果是: **6**

与第一题解法同

4.

```
#include <stdio.h>
struct abc
{ int a, b, c; };
main()
{ struct abc s[2]={1,2,3},{4,5,6}};
  int t;
  t=s[0].a+s[1].b;
  printf("%d \n",t);
}
```

运行结果是: **6**

与第一题解法同

### 三、 程序填空

1. 输入一个字符，判断该字符是数字、字母、空格还是其他字符。

```
main( )
{ char ch;
  ch=getchar();
  if( ch>='a'&&ch<='z' || ch>='A'&&ch<='Z' )
    printf("It is an English character\n");
  else if( ch>='0'&&ch<='9' )
    printf("It is a digit character\n");
  else if( ch==' ' )
    printf("It is a space character\n");
  else
    printf("It is other character\n"); }
```

第 1 空：字符在计算机中以 ASCII 码的形式存储。所以当输入的字符，即 ch 中字符所对应的 ASCII 码的范围在英文字母的 ASCII 码的范围内即可，参照 p377。由于英文字母又分为大写字母和小写字母，因此此处用一个逻辑或表达式，表示 ch 中是小写字母或者大写字母，都能使得表达式成立。ch>=97&&ch<=122 || ch>=65&&ch<=90

需要注意的是，对于本题区间所对应的表达式，不可写作 97<=ch<=122，也不可写作 'A'<=ch<='Z'。对于 97<=ch<=122 因为在计算此表达式时的顺序是从左向右，因此先计算 97<=ch。无论 ch 中的取值如何，表达式 97<=ch 的值只有两种情况：0 或 1。所以无论是 0 还是 1，都小于 122，因此 97<=ch<=122 恒成立。

第 3 空，判断 ch 中是否为空格，也是通过 ch 中字符与空格字符的 ASCII 码来判断。在判断表达式的值是否相等时，用关系符号==；不要用赋值符号=。

2. 下列程序的功能是从输入的整数中，统计大于零的整数个数和小于零的整数个数。用输入 0 来结束输入，用 i,j 来放统计数，请填空完成程序。

```
void main()
{ int n,i=0,j=0;
  printf("input a integer,0 for end\n");
  scanf("%d",&n);
  while ( n 或 n!=0 ) {
    if(n>0) i= i+1 ;
```

```

        else j=j+1;
    }
    printf("i=%4d,j=%4d\n",i,j);
}

```

此题用  $i$  来记录大于零的整数，用  $j$  记录小于零的整数。所以循环条件是  $n$ （或者  $n!=0$ ）即当  $n$  不为 0 时执行循环体。在循环体中是一个选择语句。如果  $n>0$ ，则令  $i$  加 1，相当于令正整数的个数加 1；否则（即  $n<0$ ），令  $j$  加 1，相当于令负整数的个数加 1。

### 3. 编程计算 $1+3+5+\dots+101$ 的值

```

#include <stdio.h>

void main()
{
    int i, sum = 0;
    for (i = 1; i<=101; i=i+2;)
        sum = sum + i;
    printf("sum=%d\n", sum);
}

```

for 语句的一般形式详见 p120.

表达式 1 为  $i = 1$ ，为循环变量赋初值，即循环从 1 开始，本题从 1 到 101，因此终值是 101，表达式 2 是循环条件，用来控制循环的结束，因此循环条件为  $i \leq 101$ ；表达式 3 为循环变量的自增，本题是

### 4. 编程计算 $1+3+5+\dots+99$ 的值

```

main()
{
    int i, sum = 0;
    i=1;
    while (i<100)
    {
        sum = sum + i;
        i=i+2 ;
    }
    printf("sum=%d\n", sum);
}

```

### 5. 从键盘输入一个字符，判断它是否是英文字母。

```

#include <stdio.h>

void main()
{
    char c;
    printf("input a character:");
    c=getchar();
}

```

```

    if(c>= 'A' && c<= 'Z' || c>='a' && c<= 'z')        printf("Yes\n");
    else        printf("No");
}

```

6. 下面程序的功能是在 a 数组中查找与 x 值相同的元素所在位置,请填空。

```

#include <stdio.h>

void main()
{
    int a[10],i,x;
    printf("input 10 integers: ");
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    printf("input the number you want to find x: ");
    scanf("%d",&u>&x);
    for(i=0;i<10;i++)
        if(x==a[i])
            break;
    if(i<10)
        printf("the pos of x is: %d\n",i);
    else printf("can not find x! \n");
}

```

7. 程序读入 20 个整数，统计非负数个数，并计算非负数之和。

```

#include <stdio.h>

main()
{
    int i, a[20], s, count;
    s=count=0;
    for(i=0; i<20; i++)
        scanf("%d", &a[i]);
    for(i=0; i<20; i++)
    {
        if( a[i]<0 )    continue ;
        s+=a[i];
        count++;
    }
    printf("s=%d\t count=%d\n", s, count);
}

```

8. 输入一个正整数 n (1<n≤10)，再输入 n 个整数，用选择法将它们从小到大排序后输出。



```

#include <stdio.h>
int main(void){
    int i, index, k, n, temp;
    _____ /* 定义 1 个数组 a, 它有 10 个整型元素*/
    printf("Enter n: ");
    _____
    printf("Enter %d integers: ", n);
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for(k = 0; k < n-1; k++){ /* 对 n 个数排序 */
        index = k;
        for( _____ )
            if( _____ ) index = i;
        _____
    }
    printf("After sorted: ");
    for(i = 0; i < n; i++) /* 输出 n 个数组元素的值 */
        _____
    return 0;}

```

## 四、 程序改错

一、 下面每个程序的划线处有语法或逻辑错误，请找出并改正，使其得到符合题意的执行结果。

1. 求  $1 \times 2 \times 3 \times 4 \times \dots \times n$

```

main()
{ long int sum; //若定义变量的语句有错误，常见考点有两个：（1）变量的类型，
（2）在定义用于存放运算结果的变量时，一定要赋初值。一般赋值 0 或者循环初值。
    int n,i=1;
    scanf("%d",n); //若 scanf 语句有错误，常见考点有两个：（1）格式声明符号要与
后面欲赋值的变量的类型一致，此题%d与n的类型int一致（详见p69-78）；（2）变量的
前面要有地址符号&
    printf("\n");
    while(i<n) // 循环条件用于控制循环的次数，若以 i<n 为循环条件，则意味着
i 的终值为 n-1, 由于且 i 初值为 1, 因此一共能够循环 n-1 次。比要求少了 1 次，因此应
改为 i<=n 或者 i<n+1
    { sum=sum*i; // 若不为 sum 赋初值，则此处无法计算 sum*i。
      i++;
    }
    printf("sum=%d",sum); //若 printf 语句有错误，常见考点有 1 个：格式声明符号
要与后面欲输出的变量的类型一致，此题%d与sum的类型long int不一致，应改为%ld

```

(详见 p69-78);

```
}  
sum 应初始化 即加入 sum=1  
第四行改为: scanf("%d",&n);  
第六行改为: while(i<=n)或者 while(i<n+1)  
第十行改为: printf("sum=%ld",sum);
```

## 2. 求一个数组中最大值及其下标。

```
main()  
{ int max,j,m;  
  int a[5];  
  for(j=1;j<=5;j++) //j=1 为循环变量 j 赋初值为 1, 同时用 j 作为数字元素的逻辑地址下标。因此输出的时候只能从 a[1]开始输出, 无法输出 a[0].因此应将 j 赋初值 0,  
  相应的循环条件改为 j<5 或者 j<=4 用于控制循环执行 5 次
```

scanf("%d",a); //若 scanf 语句有错误, 常见考点有两个: (1) 格式声明符号要与后面欲赋值的变量的类型一致, 此题%d 与 a 的类型 int 一致 (详见 p69-78); (2) 变量的前面要有地址符号&

```
    max=a[0];  
    for(j=1;j<=5;j++) //修改思路与上一个 for 语句同  
    {  
        if(max<a[j])  
        { max=a[j];  
          m=j;  
        }  
    }  
    printf("下标: %d\n 最大值:%d", j, max) //j 为 for 语句的循环变量, 当 for 语句  
    执行完之后, j 中的值为 6, 并非最大值下标, 在执行某一次循环的比较过程中, 将当时最大值的下标存在了 m 里  
}
```

第四行改为: for(j=0;j<5;j++)  
第五行改为: scanf("%d",&a[j]);  
第七行改为: for(j=1;j<5;j++)  
第八行改为: if(max<a[j])  
第十三行改为: printf("下标: %d\n 最大值:%d", m,max)

## 3. 用一个函数求两个数之和。

```
sum(x,y) //函数定义的一般形式 p173-174  
{ float z;  
  z=x+y;  
  return; //return 语句后面可以返回 0、常量、变量和表达式的值。  
}
```

```

main()
{ float a,b;
  int c; //若定义变量的语句有错误，常见考点有两个：（1）变量的类型，（2）在定义用于存放运算结果的变量时，一定要赋初值。一般赋值 0 或者循环初值。
  scanf("%f,%f",&a,&b);
  c=sum(a,b);
  printf("\nSum is %f",sum);
}

```

第一行改为: float sum(float x, float y);  
 第四行改为: return(z);或者 return z;  
 第八行: float c;  
 第十一行: printf("\nSum is %f",c);

#### 4. 程序读入 20 个整数，统计非负数个数，并计算非负数之和.

#include "stdio.h"

main()

```

{
  int i, s, count, n=20;
  int a[n]; //数组定义的一般形式，详见 p143, 其中的常量表达式不能为变量
  s=count=1;
  for( i=1, i<20, i- ) // for 语句的格式，三个表达式之间用分号，且分号不可省略
  {
    scanf("%d", &a[i]); //若 scanf 语句有错误，常见考点有两个：（1）格式声明符号要与后面欲赋值的变量的类型一致，此题%d 与 n 的类型 int 一致（详见 p69-78）；（2）变量的前面要有地址符号&
    for(i=0; i<20; i++)
    {
      if(a[i]<0)
        break; // break 与 continue 的区别 p128. 在改错题中若错误出现在 break 语句，则通常是将 break 换为 continue；反之，若错误出现在 continue，通常是将其换为 break
      s +=a[i];
      count++;
    }
    printf("s=%f count=%f\n", s, count); //若 printf 语句有错误，常见考点有 1 个：格式声明符号要与后面欲输出的变量的类型一致
  }
}

```

答案: int a[20]

```

s=count=0;
for(i=0; i<20; i++)
  scanf("%d", &a[i]);
  continue;
  printf("s=%d count=%d\n", s, count);

```

#### 5. 从键盘输入整数 x 的值,并输出 y 的值.

```
main()
{ float x,y;
  scanf("%d",&x);
  y=3.5+x;
  printf("y=%d");
}
```

正确的:           int x; float y;  
                      printf("y=%f",y);

6 编程计算下面分段函数, 输入 x, 输出 y

$$y = \begin{cases} x-1 & x < 0 \\ 2x-1 & 0 \leq x \leq 10 \\ 3x-11 & x > 10 \end{cases}$$

```
main()
{ int x,y;
  printf("\n Input x:\n");
  scanf("%d", x); // 错误同上题 scanf
  if(x<0)
    y=x-1;
    else if(x>=0||x<=10) // ||表示逻辑或, 当左边表达式成立或者右边表达式成立
    时, 整个表达式成立。 &&表示逻辑与, 当左边表达式和右边表达式同时成立时, 整
    个表达式成立。此处用逻辑表达式来表示 x 的区间[0,10], 因此应改用逻辑与符号
    y=2x-1; // C 语言中乘号不能省略, 且用*表示乘法运算
  else
    y=3x-1; // C 语言中乘号不能省略, 且用*表示乘法运算
    printf("y=%d",&y); //printf 与 scanf 不用, printf 后面给出的是变量名列表或表
    达式列表, 无需地址符号
}
```

第一处改为: scanf("%d",&x);

第二处改为: x>=0&& x<=10

第三处改为: y=2\*x-1;

第四处改为: y=3\*x-1;

第五处改为: printf("y=%d",y);

7. 求 100~300 间能被 3 整除的数的和。

```
main()
```

```

{ int n;
  long sum; //若定义变量的语句有错误，常见考点有两个：（1）变量的类型，（2）在
              定义用于存放运算结果的变量时，一定要赋初值，一般赋值 0 或者循环初值。
  for(n=100,n<=300,n++) // for 语句的格式，三个表达式之间用分号，且分号不可省略
  {
    if(n%3=0) // = 是赋值符号，用于将右边的值赋给左边的变量；== 是
              关系符号，用来判断两个值是否相等。改错中 if 后面表达式中的赋值符号是常见的
              考点。
    sum=sum*n;
  }
  printf("%ld ",sum);
}

```

第一处改为：long sum=0;

第二处改为：for(n=100;n<=300;n++)

第三处改为：if(n%3==0)

第四处改为：sum=sum+n;

8. 求表达式  $c = \sqrt{ab}$  的值

```

#include <stdio.h>
#include <math.h>
int fun(int x, int y);
main()
{   int a,b;   float f;
    scanf("%d,%d",&a,&b); //与改错第 1 题中的 scanf 错误相同
    if(ab>0){ // C 语言中乘号不能省略，且用*表示乘法运算
        fun(a,b); // 调用带有返回值的函数，应将函数的返回值保存在变量里
        printf("The result is:%d\n",&f) //与第 6 题中 printf 错误相同
    }
    else printf("error!");
fun(x,y) // 定义函数的一般形式 p173-174
{   float result;
    result = sqrt(a+b);
    return; //return 语句后面可以返回 0、常量、变量和表达式的值。
}

```

第一处改为：if(a\*b>0)

第二处改为：f= fun(a,b);

第三处改为：printf("The result is:%d\n",f);

第四处改为：float fun(int x, int y)

第五处改为：f= fun(a,b);

第六处改为: result = sqrt(a\*b);

第七处改为: return result;

## 五、编程题

1. 输入 2 个整数，求两数的平方和并输出。

```
#include <stdio.h>
int main(void)
{
    int a, b, s;
    printf("please input a,b:\n");
    scanf("%d%d", &a, &b);
    s = a*a + b*b;
    printf("the result is %d\n", s);
    return 0;
}
```

2. 输入一个圆半径 r，当 r>=0 时，计算并输出圆的面积和周长，否则，输出提示信息。

```
#include <stdio.h>
#define PI 3.14
int main(void)
{
    double r, area, girth;
    printf("please input r:\n");
    scanf("%lf", &r);
    if (r >= 0)
    {
        area = PI * r * r;
        girth = 2 * PI * r;
        printf("the area is %.2f\n", area);
        printf("the girth is %.2f\n", girth);
    }
    else
    {
        printf("Input error!\n");
        return 0;
    }
}
```

3、已知函数  $y=f(x)$ ，编程实现输入一个 x 值，输出 y 值。

$$y = \begin{cases} 2x+1 & (x < 0) \\ 0 & (x = 0) \\ 2x-1 & (x > 0) \end{cases}$$

```
#include <stdio.h>
void main()
{
    int x, y;
    scanf("%d", &x);
    if (x < 0) y = 2 * x + 1;
```

```

else if(x>0) y=2*x-1;
else y=0;
printf("%d",y);
}

```

- 4.从键盘上输入一个百分制成绩 **score**，按下列原则输出其等级：**score**≥90，等级为 **A**；**80**≤**score**<90，等级为 **B**；**70**≤**score**<80，等级为 **C**；**60**≤**score**<70，等级为 **D**；**score**<60，等级为 **E**。

```

#include <stdio.h>
void main(){
    int    data;
    char   grade;
    printf("Please enter the score:");
    scanf("%d", &data);
    switch(data/10)
    {   case 10:
        case 9:  grade='A';  break;
        case 8:  grade='B';   break;
        case 7:  grade='C';   break;
        case 6:  grade='D';   break;
        default: grade='E';
    }
    printf("the grade is %c",grade);
}

```

5. 编一程序每个月根据每个月上网时间计算上网费用，计算方法如下：

$$\text{费用} = \begin{cases} 30\text{元} & \leq 10\text{小时} \\ \text{每小时}3\text{元} & 10 - 50\text{小时} \\ \text{每小时}2.5\text{元} & \geq 50\text{小时} \end{cases}$$

要求当输入每月上网小时数,显示该月总的上网费用(6分)

```

#include <stdio.h>
void main()
{ int hour;
  float fee;
  printf("please input hour:\n");
  scanf("%d",&hour);
  if(hour<=10)
      fee=30;
  else if(hour>=10&&hour<=50)
      fee=3*hour;
  else fee=hour*2.5;
  printf("The total fee is %f",fee);
}

```

6. 从键盘输入10个整数，统计其中正数、负数和零的个数，并在屏幕上输出。

```
#include <stdio.h>

void main( ) {
    int a, i, p=0, n=0, z=0;
    printf("please input number");
    for(i=0; i<10; i++){
        scanf("%d", &a);
        if (a>0)      p++;
        else if (a<0)  n++;
        else z++;
    }
    printf("正数: %5d, 负数: %5d, 零: %5d\n", p, n, z);
}
```

7. 编程序实现求 1-10 之间的所有数的乘积并输出。

```
#include <stdio.h>

void main()
{
    int i;
    long sum=1;
    for(i=1; i<=10; i=i+1)
        sum=sum*i;
    printf("the sum of odd is :%ld", sum);
}
```

8. 从键盘上输入 10 个数，求其平均值。

```
#include <stdio.h>

void main(){
    int a, i, sum=0;
    float ave;;
    for(i=0; i<10; i++){
        scanf("%d", &a);
        sum+=a;
    }
    ave=(float)sum/10;
    printf("ave = %f\n", ave);
}
```

9. 编程序实现求 1-1000 之间的所有奇数的和并输出。

```
#include <stdio.h>

void main( )
{
    int i, sum=0;
    for(i=1; i<1000; i=i+2)
        sum=sum+i;
    printf("the sum of odd is :%d", sum);
}
```



10.有一个分数序列：2/1， 3/2， 5/3， 8/5， 13/8， .....编程求这个序列的前 20 项之和。

```
#include <stdio.h>
void main(){
    int i,t,n=20;
    float a=2,b=1,s=0;
    for(i=1;i<=n;i++)
        {s=s+a/b;
         t=a;
         a=a+b;
         b=t;
        }
    printf("sum=%6.2f",s);
}
```

11. 从键盘输入两个数，求出其最大值（要求使用函数完成求最大值，并在主函数中调用该函数）

```
#include <stdio.h>
float max(float x,float y);
void main()
{   float a,b,m;
    scanf("%f,%f",&a,&b);
    m=max(a,b);
    printf("Max is %f\n",m);
}
float max(float x,float y)
{
    if (x>=y)
        return x;
    else
        return y;
}
```

12. 编写程序，其中自定义一函数，用来判断一个整数是否为素数，主函数输入一个数，输出是否为素数。

```
#include <math.h>
#include <stdio.h>
int IsPrimeNumber(int number)
{   int i;
    if (number <= 1)
        return 0;
    for (i=2; i<=sqrt(number); i++)
        {   if ((number % i) == 0)
                return 0;   }
}
```

```

        return 1;}
void main()
{ int n;
  printf("Please input n:");
  scanf("%d",&n);
  if(IsPrimeNumber(n))
    printf("\n%d is a Prime Number",n);
  else printf("\n%d is not a Prime Number",n);}

```

### 13、从键盘输入 n 个数存放在数组中，将最小值与第一个数交换，输出交换后的 n 个数。

```

#include <stdio.h>
int main(void){
    int i,n,iIndex,temp;
    int a[10];
    printf("Enter n: ");
    scanf("%d", &n);
    printf("Enter %d integers:\n ",n);
    for(i=0;i<n;i++)
        scanf("%d", &a[i]);
    iIndex=0;
    for(i=1;i<n;i++){
        if(a[i]<a[iIndex])    iIndex=i;
    }
    temp=a[0];a[0]=a[iIndex];a[iIndex]=temp;
    for(i=0;i<n;i++)
        printf("%5d", a[i]);
    printf("\n");
    return 0;
}

```

#### 第二种解法 利用函数

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

```

int comp(int array[], int n)
{
    int i,index,temp;
    printf("为数组赋值: \n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
    for(i=1,index=0;i<=n-1;i++)
    {
        if(array[i]<array[index])

```

```

        {    index=i;
        }
    }
    temp=array[0];array[0]=array[index];array[index]=temp;
    for(i=0;i<n;i++)
    {
        printf("%d  ",array[i]);
    }
    return 0;
}
main()
{    int n;
    int a[10];
    printf("为 n 赋值: \n");
    scanf("%d",&n);
    comp(a,n);}

```