第9章 指针技术

§1 指针的概念

1.变量的指针与直接访问

- 变量的指针：变量的内存地址称为变量的指针
- 变量的直接访问：按变量指针存取变量值的方式

2. 指针变量与间接访问

- 指针变量：用于存放变量指针值的变量
- 变量的间接访问：访问指向变量的指针变量，从中获得所需访问变量的指针，再访问变量值的方式

```
eg1: int i=3,j;
     float f=0.5;
     ......
     j=i;
```

```
eg2:  int i=3,j;
      float f=0.5;
      int *i_pointer;
       ......
      i_pointer=&i;
```

§2 指针变量

一、指针变量定义

类型说明符 *指针变量名1, *指针变量名2, … ;

— C语言中，指针变量类型特指其所指向变量的类型

```
eg1: int *pi;   /*pi是整型指针变量*/
```

```
eg2: float *p1, *p2;  /*定义两个实型指针变量*/
```

```
eg3: char *ps;    /*ps是字符型指针变量*/
```

```
eg4:  int  i,  j, *pi, *pj;
      pi=&i;  pj=&j;      /*pi指向i, pj指向j*/
```

```
eg5:  int  i,  j, *pi =&i, *pj =&j; /*用初始化建立指向关系*/
```

§2 指针变量

一、指针变量定义

类型说明符 *指针变量名1, *指针变量名2, … ;

— C语言中，指针变量类型特指其所指向变量的类型

```
eg6:  int i,*pi,*pj;
      pi=&i;
      pj=pi;  /*指针传递*/
```

```
eg7: int i,  j, *pi=&i, *pj=&j, *pt;
     pt=pi;
     pi=pj;
     pj=pt;   /*指针交换*/
```

```
eg8: int i;
     float *pf=&i;
```

错误

```
eg9: int i,*pi=&i;
     float *pf ;
     pf=pi;
```

§2 指针变量

二、相关运算

1.取址运算符&

&变量— 获得变量的地址(指针)值

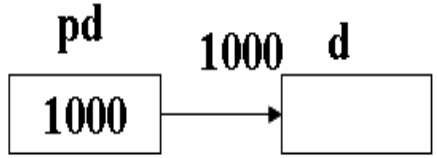
——不能用整型量和任何非地址类型的数给指针变量赋值

——只能取变量的地址，不可取常量和表达式的地址

eg1:double d,*pd=&d;

或

eg2: double d,*pd;
pd=&d;



eg3: int i=500;
double *pd;
pd=2*i;

错误

eg4: double *pd;
pd=1000;

eg5: &233

错误

eg6: &(i+233)

§2 指针变量

二、相关运算

2.指针(间访)运算符*

*指针变量— 获得指针变量所指对象之值

eg1: 若有int x=100, y=10, *px=&x, *py=&y;则 :

```
        y=x;        /*对x和y的直接访问*/
等价于: y=*px;      /*对x一级间访，对y直接访问*/
等价于: *py=x;      /*对x直接访问，对y一级间访*/
等价于: *py=*px;    /*对x和y的一级间访*/
```

y=x+3; ⟩⟩ y=*px+3; ⟩⟩ *py=x+3; ⟩⟩ *py=*px+3;

```
eg2: int x=100, y=10, *px =&x, *py =&y;
     printf("%d%d",x,y); /* 等价于printf("%d%d",*px,*py);*/
```

§2 指针变量

&、*算符同优先级，满足右结合性

严禁使用未初始化的指针变量

通式1：

若有： type a, *p=&a;

则： *&a ≡ a

&*p ≡ p

```
#include <stdio.h>
main( )
{  int *iptr;  /*iptr是悬空指针*/
   *iptr=421;
   printf("*iptr=%d\n", *iptr); }
```

错误

§2 指针变量

例1：已知：int i=0,j=1,*p=&i,*q=&j;错误的语句是__D__。

A. i=*&j;　　B. p=&*&i;　　C. j=*p++;　　D. i=*&q;

例2:说出下列程序的运行结果

```
#include <stdio.h>
main( )
{ int i=4,j=6,k=8,*p=&i,*q=&j,*r=&k;
   int x,y,z;
   x=p==&i;   /*x=1*/
   y=3*-*p/(*q)+7; /*y=5*/
   z=*(r=&k)=*p**q;  /*z=k=24*/
   printf("%d,%d,%d",x,y,z);
}
```

1,5,24

§2 指针变量

三、变量的指针作函数实参

例1： 编程按逆序输出a, b之值， 要求采用交换法实现。

其中： main—输入a,b并将其按逆序输出

swap—交换a,b之值， 使大者在a中

```
#include <stdio.h>
main( )
{ int a, b;
  void swap(int,int);
  scanf("%d%d", &a, &b);
 if(a<b)  swap(a, b);
 printf("max=%d,min=%d\n",a,b);
}
```

错误

```
void  swap(int x, int y)
{ int t;
  t=x; x=y; y=t;
}
```

§2 指针变量

三、变量的指针作函数实参

例1： 编程按逆序输出a, b之值， 要求采用交换法实现。

其中： main—输入a,b并将其按逆序输出

swap—交换a,b之值， 使大者在a中

```
#include <stdio.h>
main( )
{ int a, b;
  void swap (int *, int *);
  scanf("%d%d", &a, &b);
  printf("a=%d,b=%d\n",a,b);
  if(a<b) swap(&a, &b);
  printf("max=%d,min=%d",a,b); }
```

正确

```
void swap(int *p1 ,int *p2)
 {  int t;
     t=*p1;
    *p1=*p2;
    *p2=t;
   /*交换p1,p2所指之值*/
  }
```

§2 指针变量

例2（考题）：

程序输出第一行是 48 ，

第二行是 62 ，

第三行是 65 。

```
#include <stdio.h>
 void  fun(int *a, int b, int *c)
   { *a=++b; b=*c; *c=*a; }
 main( )
  { int a=10, b=20, c=30;
     {  int a=4;
        c=a+b;
         printf("%d\n", a+b+c);
         {  int c=b;
            fun(&a, b, &c);
            printf("%d\n", a+b+c);
         }
          printf("%d\n", a+b+c);
     }
  }
```

§3 一维数组与指针变量

一、一维数组中的指针表示

- 数组的指针是指数组的起始地址
- 数组元素的指针是指该元素的地址
- 数组名代表数组指针，是指针常量

int a[8];

1000		a[0]
1002		a[1]
1004		a[2]
1006		a[3]
1008		a[4]
1010		a[5]
1012		a[6]
1014		a[7]

float f[5];

2000		f[0]
2004		f[1]
2008		f[2]
2012		f[3]
2016		f[4]

§3 一维数组与指针变量

二、指向数组元素的指针变量

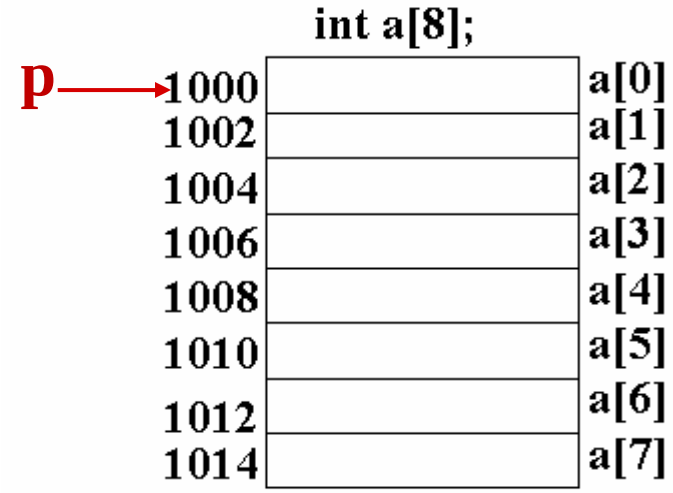
arraytype *指针变量名;

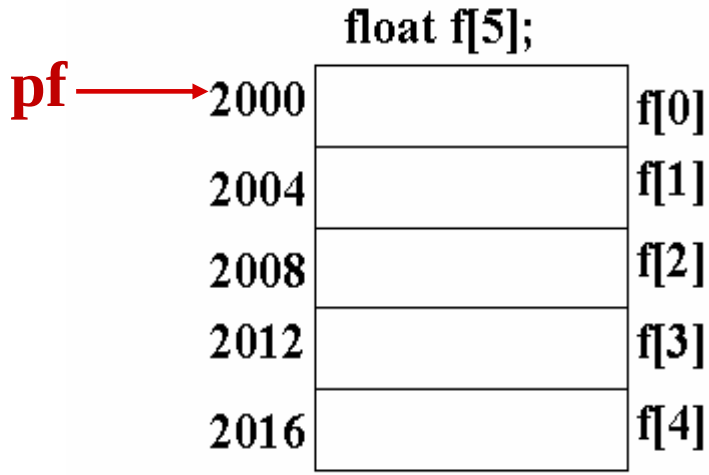
eg1: int a[8],*p;
p=&a[0];

eg1: int a[8],*p;
p=a;

eg2: float f[5];
float *pf=f;

eg2: float f[5],*pf;
pf=f;





§3 一维数组与指针变量

三、相关指针运算

1.++p、p ++ 、 p+=1、p=p+1; --p、p--、 p-=1、p=p-1

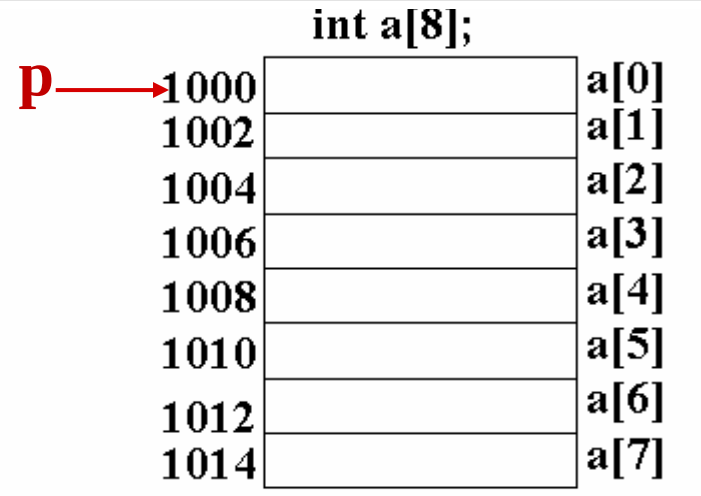
— 表示指针变量加（减）所指元素内存字节数

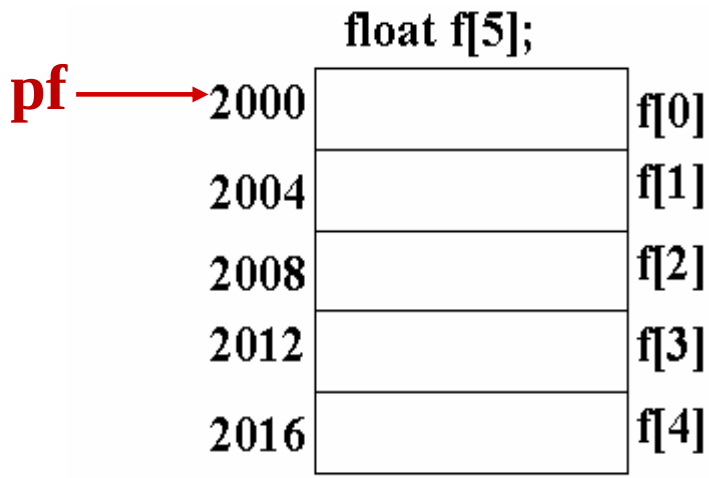
— 对于 p=&a[i];执行p加1操作后， p=&a[i+1]

— 对于 p=&a[i];执行p减1操作后， p=&a[i-1]

2.p+=i 、p=p+i; p-=i、p=p-i

—表示加/减: i*sizeof(arraytype)





§3 一维数组与指针变量

三、相关指针运算

1.++p、p ++ 、 p+=1、p=p+1; --p、p--、 p-=1、p=p-1

— 表示指针变量加（减）所指元素内存字节数

— 对于 p=&a[i];执行p加1操作后， p=&a[i+1]

— 对于 p=&a[i];执行p减1操作后， p=&a[i-1]

2.p+=i 、p=p+i; p-=i、p=p-i

—表示加/减: i*sizeof(arraytype)

```
arrtype a[n],*p;
for(p=a;p<=a+n-1;p++)
  printf("% ",*p);
```

正向遍历a数组

```
arrtype a[n],*p;
for(p=a+n-1;p>=a;p--)
  printf("% ",*p);
```

反向遍历a数组

§3 一维数组与指针变量

三、相关指针运算

1.++p、p ++ 、 p+=1、p=p+1; --p、p--、 p-=1、p=p-1

— 表示指针变量加（减）所指元素内存字节数

— 对于 p=&a[i];执行p加1操作后， p=&a[i+1]

— 对于 p=&a[i];执行p减1操作后， p=&a[i-1]

2.p+=i 、p=p+i; p-=i、p=p-i

—表示加/减: i*sizeof(arraytype)

3.若二指针变量指向同一数组，则可进行减法和比较运算

```
eg:  int a[10],*p=a,*q=&a[2];
     printf("%d\n",q-p);  /* 2 */
     q+ +;   /* q=&a[3] */
     printf("%d\n", p-q);  /*  -3  */
     printf("%d,%d,%d\n",p<q,p==q,p>q); /*1,0,0*/
```

§3 一维数组与指针变量

三、相关指针运算

通式2：

若有：arraytype a[n], *p=a;

则：&a[i] ≡ a+i≡ p+i

a[i] ≡ *&a[i] ≡ *(a+i) ≡*(p+i) ≡ p[i]

在C语言中，[ ]是变址运算符，其计算功能是：

地址[整数] ≡ *(地址+整数)

例:编程计算s数组中串的实际长度

```
#include <stdio.h>
main( )
{ char s[81], *ps=s;
  gets(s);
 while(*ps!='\0')  ps++;     /*将ps移至串尾\0处*/
 printf("The string length is %d\n", ps-s);  }
```

§3 一维数组与指针变量

三、相关指针运算

通式2：

若有：arraytype a[n], *p=a;

则：&a[i] ≡ a+i≡ p+i

a[i] ≡ *&a[i] ≡ *(a+i) ≡*(p+i) ≡ p[i]

在C语言中，[ ]是变址运算符，其计算功能是：

地址[整数] ≡ *(地址+整数)

通式3：

若有：arraytype a[n], *p；且：p=&a[i];

则：*(p++) → a[i++]

*(++p) → a[++i]

*(p- -) → a[i- -]

*(- -p) → a[- -i]

§3 一维数组与指针变量

四、数组元素a[i]访问的全部方法

通式4： 若有：arraytype a[n], *p=a;则数组元素a[i]的访问方法：

①a[i] —下标法 ② *(a+i)或*(p+i) — 指针常量法

③p[i] — 带下标的指针变量法 ④ *p++ — 指针变量加1法

例1：以输出一个int a[10]数组的元素值为例

①法：

```
#include <stdio.h>
main( )
 { int  a[10], i;
   /*给a数组赋值*/
   for(i=0; i<10; i++)
     printf("%d,", a[i]); }
```

②法：

```
#include <stdio.h>
main( )
 { int  a[10], i,*p=a;
   /*给a数组赋值*/
   for(i=0; i<10; i++)
     printf("%d,", *(p+i));  }
```

§3 一维数组与指针变量

四、数组元素a[i]访问的全部方法

通式4： 若有：arraytype a[n], *p=a;则数组元素a[i]的访问方法：

①a[i] —下标法　　② *(a+i)或*(p+i) — 指针常量法

③p[i] — 带下标的指针变量法　　④ *p++ — 指针变量加1法

例1：以输出一个int a[10]数组的元素值为例

④法之一解：

```
#include <stdio.h>
main( )
 { int  a[10], i,*p=a;
   /*给a数组赋值*/
   for(i=0; i<10; i++)
     printf("%d,", *p++); }
```

④法之二解：

```
#include <stdio.h>
main( )
 { int  a[10], *p;
   /*给a数组赋值*/
   for(p=a; p<=p+9; p++)
     printf("%d,", *p);  }
```

§3 一维数组与指针变量

四、 数组元素a[i]访问的全部方法

例2(考题): 以下程序的输出结果是_____。

```
main( )
{  int a[ ]={1, 2, 3, 4, 5, 6}, *p=a;
   *(p+3)+=2;
   printf("%d,", *++p);
   printf("%d\n", *(p+3));
 }
```

2,5

§3 一维数组与指针变量

四、数组元素a[i]访问的全部方法

例3(考题)：以下程序的功能是____，其输出结果是____。

```
main( )
{  static int a[ ]={7,4,5,3,10};
    int m=a[0],k,*p=a;
    for(k=0;k<5;k++)
      m=(*(p+k)>m)?*(p+k):m;
   printf("m=%d\n",m);
 }
```

求出a数组中的最大值

m=10

§3 一维数组与指针变量

五、数组指针作函数实参

例1:编子函数inv将数组int a[10]反置。

I解:

```
#include <stdio.h>
main( )
{ int i , a[10];
  void inv(int x[ ], int n);
 /*输入输出反置前的a数组*/
   inv( a, 10);
   for( i=0; i<10; i++)
   printf("%6d",*(a+i)); }
```

```
void inv( int  x[ ], int n)
 { int i, j, t;
   for(i=0,j=n-1;i<j;i++,j--)
    { t=x[i]; x[i]=x[j]; x[j]=t;}
 }
```

§3 一维数组与指针变量

五、数组指针作函数实参

例1:编子函数inv将数组int a[10]反置。

II解:

```
#include <stdio.h>
main( )
{ int i , a[10];
  void inv(int *x, int n);
 /*输入输出反置前的a数组*/
   inv( a, 10);
   for( i=0; i<10; i++)
   printf("%6d",*(a+i));  }
```

```
void inv( int *x, int n)
 { int t, *i, *j;
   for(i=x,j=x+n-1;i<j;i++,j--)
   { t=*i; *i=*j; *j=t;}
 }
```

§3 一维数组与指针变量

五、数组指针作函数实参

例1:编子函数inv将数组int a[10]反置。

III解:

```
#include <stdio.h>
main( )
{ int i , a[10],*p=a;
  void inv(int x[], int n);
 /*输入输出反置前的a数组*/
   inv( p, 10);
   for( i=0; i<10; i++)
   printf("%6d",*(a+i));  }
```

```
void inv( int  x[ ], int n)
 { int i, j, t;
   for(i=0,j=n-1;i<j;i++,j--)
    { t=x[i]; x[i]=x[j]; x[j]=t;}
 }
```

§3 一维数组与指针变量

五、数组指针作函数实参

例1:编子函数inv将数组int a[10]反置。

III解:

```
#include <stdio.h>
main( )
{ int i , a[10],*p=a;
  void inv(int *x, int n);
 /*输入输出反置前的a数组*/
   inv( p, 10);
   for( i=0; i<10; i++)
   printf("%6d",*(a+i));  }
```

```
void inv( int *x, int n)
 { int t, *i, *j;
   for(i=x,j=x+n-1;i<j;i++,j--)
   { t=*i; *i=*j; *j=t;}
 }
```

§3 一维数组与指针变量

通式5：

主调函数中有： arraytype a[n], *p=a;

被调函数欲获得a数组中全部或部分数据，

则通信界面的4种构造方法：

函数调用的实参	被调函数首部形参
数组名a	数组名x
数组名a	指针变量x
指针变量p (指向数组a任一位置)	数组名x
指针变量p (指向数组a任一位置)	指针变量x

§3 一维数组与指针变量

例2: 编写max_min_val子函数，从实参数组num[10]中找出最大值和最小值

```
#include <stdio.h>
int Max, Min;
void  max_min_val(int *parr, int n)
 { int *p;
   Max=Min=*parr;
   for(p=parr+1;p<=(parr+n-1);p++)
     if(*p>Max)  Max=*p;
     else if(*p<Min)  Min=*p;
  }
```

```
main( )
{ int i, num[10];
   /*输入num数组*/
  max_min_val (num, 10);
printf("%d,%d",Max,Min);
 }
```

§4 指针变量与二维数组

一、二维数组的各类指针表示

1. 对于 a[m][n]数组，有：

a ——— 0行首地址；
a+1 ——— 1行首地址；
a+2 ——— 2行首地址；
……
a+i ——— i行首地址；
……
a+m-1 ——— m-1行首地址；

2. 对于 a[m][n]数组，有：

a+1 → a+1*n*sizeof(arrtype)
a+i → a+i*n*sizeof(arrtype)

行控制
行跳跃

§4 指针变量与二维数组

3. 对于a[m][n]，存在着名为a[0]、a[1]、… a[m-1]的m个一维数组，它们分别表示该一维数组的首地址。即：a[i]表示第i行0列地址

a[0] —— 0行0列地址，即&a[0][0]；

a[1] —— 1行0列地址，即&a[1][0]；

……

a[m-1]—— m-1行0列地址，即&a[m-1][0]；

4. a[i]代表a数组第i行0列地址,a[i]+j表示a数组第i行j列地址

a[i]+1 → a[i]+1*sizeof(arrtype)

a[i]+j → a[i]+j*sizeof(arrtype)

列控制

列跳跃

§4 指针变量与二维数组

5. 由行控制转为列控制：

行控制	列控制
a —— 0行首地址	*a —— 0行0列地址
a+1—— 1行首地址	*(a+1) —— 1行0列地址
……	……
a+i—— i行首地址	*(a+i) —— i行0列地址
……	……
a+m-1——m-1行首地址	*(a+m-1) —— m-1行0列地址

§4 指针变量与二维数组

6. 由列控制转为行控制：

列控制	行控制
a[0] —— 0行0列地址	&a[0] —— 0行首地址
a[1] —— 1行0列地址	&a[1] —— 1行首地址
……	……
a[i] —— i行0列地址	&a[i] —— i行首地址
……	……
a[m-1]—— m-1行0列地址	&a[m-1] —— m-1行首地址

§4 指针变量与二维数组

通式6：

对于二维数组arraytype a[m][n]，有：

1. a+i ≡ &a[i] ——— 行控制

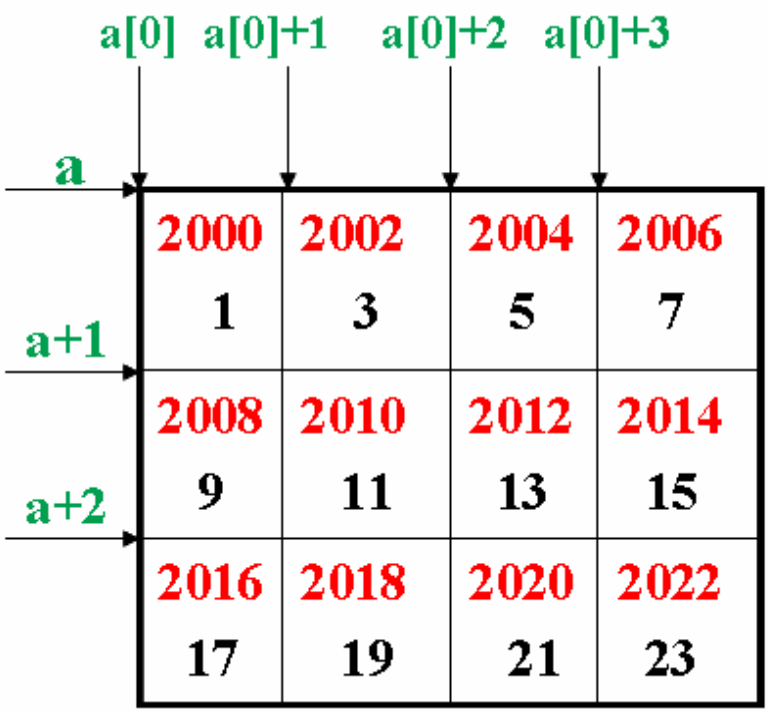
2. a[i] ≡ *(a+i) ≡ &a[i][0] ——— 列控制

3. &a[i][j] ≡ a[i]+j ≡ *(a+i)+j

4. a[i][j] ≡*&a[i][j] ≡ *(a[i]+j) ≡ *(*(a+i)+j)

§4 指针变量与二维数组

例1:已知a数组如图所示，解释下表的含义和值



表示形式	地址
a	2000, 横
a[0], *(a+0), *a	2000, 纵
a+1, &a[1]	2008, 横
a[1], *(a+1)	2008, 纵
a[1]+2, *(a+1)+2, &a[1][2]	2012, 纵
(a[1]+2),(*(a+1)+2), a[1][2]	元素值13

§4 指针变量与二维数组

例2：若有定义：int a[2][3];则对a数组的第i行第j列（假设i,j已正确说明并赋值）元素地址的正确引用为 D 。

A. *(a[i]+j)　　B. (a+i)+j　　C. *(a+i+j)　　D. a[i]+j

例3（考题）：设有数组说明" int a[4][4];"，则不能等价表示数组元素a[3][3]的是 D 。

A. *(a[3]+3)　　B. *(*(a+3)+3)

C. *&a[3][3]　　D. (*(*(a+3))+3)

§4 指针变量与二维数组

二、指向二维数组的指针变量

1.指向数组元素的指针变量

arraytype *指针变量名

例1(考题)： 设有变量说明“static int a[2][3]={1,2,3,4,5,6};
int m,*ptr=&a[0][0];”,执行语句m=(*ptr)*(*(ptr+2))*(*(ptr+4));后， m的值为 A 。

A. 15 B. 48 C. 24 D. 60

例2(考题):设有说明static int a[3][3]={{1},{4,5},{7,8,9}},*pa=a[2];则*(pa-2)的值为 5 。

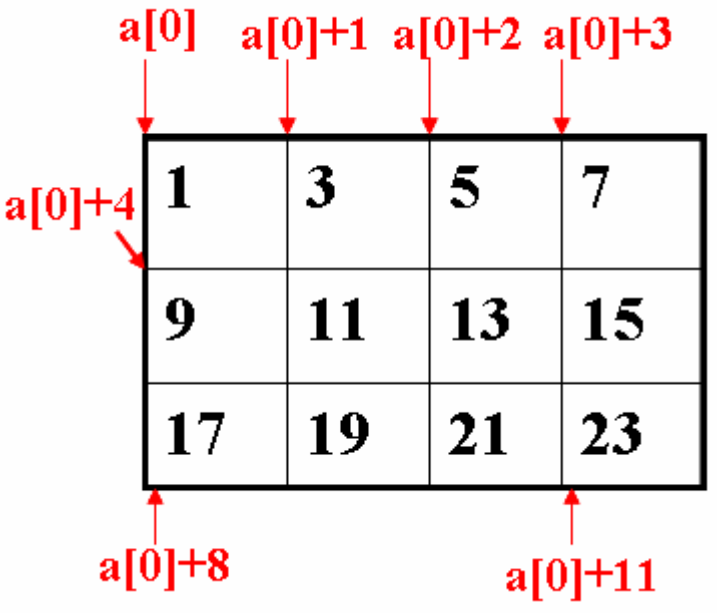
例3(考题)若有:int a[ ][4]={1,2,3,4,5,6,7,8,9,10},*p=*(a+1);则值为9的表达式是 B 。

A. p+=3, *p++ B. p+=4,*(p++) C. p+=4, *++p D. p+=4, ++*p

§4 指针变量与二维数组

二、指向二维数组的指针变量

例4：用指针变量方法输出以下数组a[3][4]各元素之值

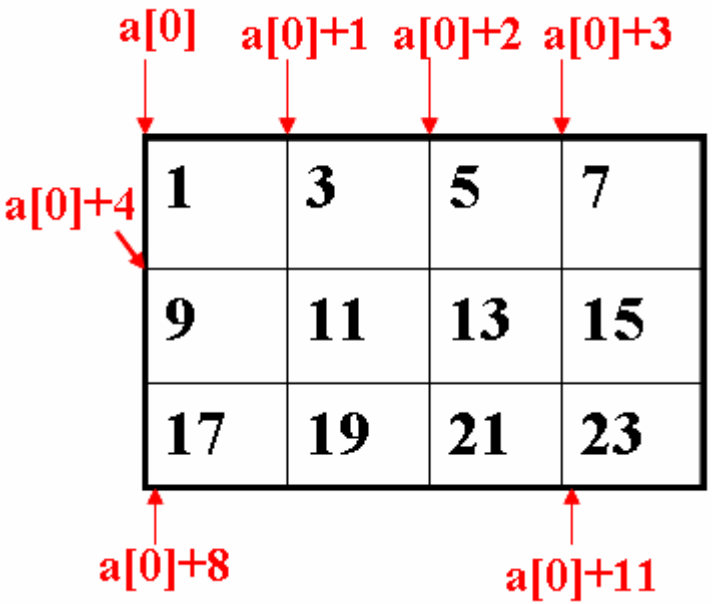



```
#include <stdio.h>
main( )
{ int a[3][4]={1,3,..19,21,23};
   int *p;
   for(p=a[0]; p<＝a[0]+11; p++)
    { if ((p-a[0])%4==0)  printf(“\n”);
       printf(“%8d”, *p);
     }
 }
```

§4 指针变量与二维数组

二、指向二维数组的指针变量

例5：打印以下a[3][4]数组的a[i][j]元素之值




```
#include <stdio.h>
main( )
{ int a[3][4]={1,3,5,7,9,11,13,15,
                17,19,21,23};
  int  i, j, *p=&a[0][0];
  printf("Input i,j:  0≤i≤2,0≤j≤3\n");
  scanf("%d,%d",&i, &j);
  printf("a[%d][%d]=%d\n", i, j,
                *(p+i*4+j)); }
```

通式7： 在数组a[m][n]中， a[i][j]的绝对位置是 i*n+j+1

a[i][j]相对a[0][0]的相对位置是 i*n+j

§4 指针变量与二维数组

二、指向二维数组的指针变量

2.指向长n的一维数组的指针变量

arraytype (*指针变量名)[n];

eg1: int a[3][4] ,(*p)[4] ;
p=a; /*p指向第0行*/

eg2:int a[3][4] ,(*p)[4] ;
p=&a[0]; /*p指向第0行*/

正确

eg3: int a[3][4] ,(*p)[4] ;
p=*a;

eg4: int a[3][4] ,(*p)[4] ;
p=a[0];

错误

eg5: int a[3][4] ,(*p)[4], *q1,*q2;
p=a;
q1=p+1; /*在编译时导致warning， 不推荐*/
q2=*p+1; /*q2指向a的0行1列， 即&a[0][1]*/

§4 指针变量与二维数组

二、指向二维数组的指针变量

通式8：

对于arrtype a[m][n],*p1=a[0],(*p2)[n]=a;则有：

1. &a[i][j] ≡ a[i]+j ≡ *(a+i)+j

≡a[0]+i*n+j ≡p1+i*n+j ≡*(p2+i)+j

2. a[i][j] ≡*&a[i][j] ≡ *(a[i]+j) ≡ *(*(a+i)+j)

≡*(a[0]+i*n+j) ≡*(p1+i*n+j) ≡*(*(p2+i)+j)

§4 指针变量与二维数组

二、指向二维数组的指针变量

例1:若有以下定义和语句,则对a数组元素地址的正确引用为 C 。

```
int a[2][3],(*p)[3];  p=a;
```

A. *(p+2)　　B. p[2]　　C. p[1]+1　　D. (p+1)+2

例2：若有以下定义和语句，则对b数组的第i行第j列（假设i,j已正确说明并赋值）元素的非法引用为 B 。

```
int b[2][3]={0},(*p)[3];　　p=b;
```

A. *(*(p+i)+j)　　B. *(p+i)+j　　C. (*(p+i))[j]　　D. *(p[i]+j)

例3：若有以下定义：

```
int x[4][3]={1,2,3,4,5,6,7,8,9,10,11,12} ,(*p)[3]=x ;
```

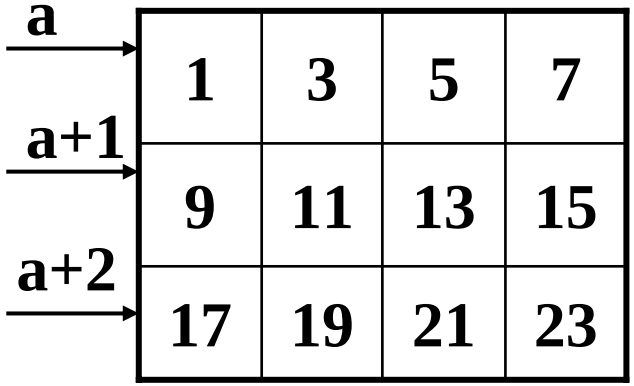
则能够正确表达数组元素x[1][2]的表达式是 D 。

A. *((*p+1) [2])　B. (*p+1)+2　　C. *(*(p+5))　　D.*(*(p+1)+2)

§4 指针变量与二维数组

二、指向二维数组的指针变量

例4：输出整型数组a[3][4]

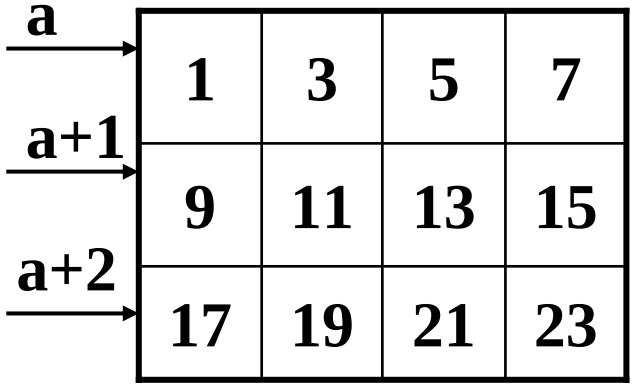



```
#include <stdio.h>
main( )
{ int a[3][4]={1,3,5…,21,23};
  int (*p)[4], j;
  for(p=a; p<=a+2; p++)
   { for( j=0; j<=3; j++)
       printf("%8d", *(*p+j));
     printf("\n");
    }
 }
```

§4 指针变量与二维数组

二、指向二维数组的指针变量

例5：输出整型数组a[3][4]的a[i][j]元素之值




```
#include <stdio.h>
main( )
{ int a[3][4]={1,3,…,23};
  int (*p)[4], i, j;
  p=a;
  scanf("%d,%d", &i, &j);
  printf("\n a[%d][%d]=%d\n",
          i, j, *(*(p+i)+j));
 }
```

§4 指针变量与二维数组

三、二维数组指针作函数参数

1.用纵向指针做实参

用指向数组元素的指针变量作形参

通信界面的构造方法

主调函数	被调函数
:	fun(arraytype *p)
arraytype a[m][n];	{ … }
fun(a[0]);	
¦	

§4 指针变量与二维数组

三、二维数组指针作函数参数

2.用横向指针作实参

用指向一维数组的指针变量作形参

通信界面的构造方法

主调函数	被调函数
┆	fun(arraytype (*p)[n])
arraytype a[m][n];	{ … }
┆	
fun(a);	
┆	

§4 指针变量与二维数组

例1：已知main中有实参数组score[3][4]如下所示：

(1) 编average 打印平均成绩；

(2) 编search搜索并输出第i个学生成绩(假定学号从0计算)

65.0	57.	70.	60.
58.	87.	90.	81.
90.	99.	100	98.

```
#include <stdio.h>
main( )
{ void average( float *p, int n);
  void search( float (*p)[4],int i);
  float score[3][4]={ 65.0, 57., … };
  average(*score, 12);
  search(score, 2);
}
```

§4 指针变量与二维数组

例1：已知main中有实参数组score[3][4]如下所示：

(1) 编average 打印平均成绩；

(2) 编search搜索并输出第i个学生成绩(假定学号从0计算)

```
void average(float *p,int n)
{ float *p_end,sum=0.,aver;
  p_end=p+n-1;
  for( ;p<=p_end;p++)
    sum+=(*p);
  aver=sum/n;
  printf("%.2f\n",aver);
}
```

```
#include <stdio.h>
main( )
{ void average( float *p, int n);
   void search( float (*p)[4],int i);
  float score[3][4]={ 65.0, 57., … };
  average(*score, 12);
   search(score, 2);
}
```

§4 指针变量与二维数组

例1：已知main中有实参数组score[3][4]如下所示：

(1) 编average 打印平均成绩；

(2) 编search搜索并输出第i个学生成绩(假定学号从0计算)

```
void search(float (*p)[4],int i)
{ int j;   /*搜索i行各列*/
  printf("The score of NO.");
  printf("%d are:\n",i);
  for(j=0;j<=3;j++)
   printf("%10.2f",*(*(p+i)+j));
  printf("\n");
}
```

```
#include <stdio.h>
main( )
{ void average( float *p, int n);
   void search( float (*p)[4],int i);
  float score[3][4]={ 65.0, 57., … };
  average(*score, 12);
   search(score, 2);
}
```

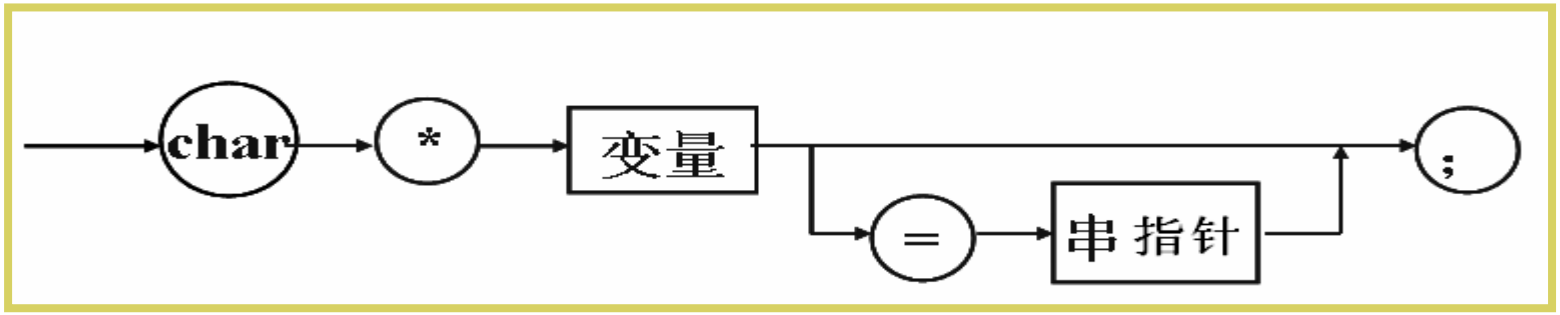
§4 指针变量与二维数组

例2(考题):下列程序输出的第一行是 1,1,2 ，第二行是 3,5,8 ，输出的第三行是 13,21,34 。

```
main( )
{ int i, j, a[3][3]={1, 1}, *p1, *p2, *p3;
  p1=a[0]; p2=a[0]+1; p3=a[0]+2;
  for(i=2; i<9; i++)
     func(p1++, p2++, p3++);
 for(i=0; i<3; i++)
 { for(j=0; j<3; j++)
     printf("%d,", a[i][j]); printf("\n");}
 }
func( int *q1, int *q2, int *q3)
 { *q3=*q1+*q2; }
```

§5 指针变量与字符数组

一、字符指针变量定义及初始化



eg1: char s[ ]="I love china!", *p=s;

eg3: char *p="I love china!";

eg2: char s[ ]="I love china!", *p;
p=s;

eg4: char *p;
p="I love china!";

eg5: char s[20];
s="I love china!";

错误

eg6: char s[20];
strcpy(s,"I love china!");

正确

§5 指针变量与字符数组

一、字符指针变量定义及初始化

例1：说出下列程序的运行结果

```
main( )
  { char *p="I love china!";
    printf("The sixth character is %c.\n", p[5]);
    p=p+7;
    printf("%s\n", p);
    printf("%c\n", *p);}
```

The sixth character is e.
china!
c

§5 指针变量与字符数组

一、字符指针变量定义及初始化

例2：分析以下程序的运行结果

```
#include <string.h>
main( )
{ char *p,*p1;
  p1=p="abcdefghijk";  p+=3;
  printf(" %d\n",strlen(strcpy(p,"ABCD")));
  printf("%d\n",strlen(p1));
  printf("%s\n%s\n",p,p1);   }
```

```
4
7
ABCD
abcABCD
```

例3（考题）：以下能将字符串"good!"正确存放在字符数组s中，或使指针s能指向这个字符串的是__D__。

A. char s[4]={'g', 'o', 'o', 'd', '!'};　　B. char s[5]; s="good!";

C. int s[5]="good!";　　D. char *s; s="good!";

§5 指针变量与字符数组

例4：分析以下程序的运行结果

```
#include <stdio.h>
char b[ ]="computer";
char *a="COMPUTER";
main( )
{ int i=0;
   printf("%c,%s",*a, b+1);
   while(putchar(*(a+i)))  i++;
   putchar('\n');
   printf("%d", i);
   while(--i)  putchar(*(b+i));
   putchar('\n');
   printf("%s\n", &b[3]);
 }
```

```
C,omputerCOMPUTER
8retupmo
puter
```

§5 指针变量与字符数组

二、字符数组指针作函数参数

实参	形参
字符数组名s1	字符数组名s2
字符数组名s1	字符指针变量p
字符指针变量p1(指向串首或串的任一位置)	字符数组名s2
字符指针变量p1 (指向串首或串的任一位置)	字符指针变量p2

§5 指针变量与字符数组

二、字符数组指针作函数参数

例1：编strcmp子函数，比较主函数中串s和t的大小

```
#include <stdio.h>
main( )
{char s[81], t[81];
  int val;
  int strcmp(char *,char *);
  gets(s); gets(t);
  printf("Str s:%s\nStr t:%s\n",s,t);
  val=strcmp(s, t);
  if(val>0) printf("s>t %d\n",val);
  else if(val==0) printf("s=t\n");
  else printf("s<t %d\n",val);  }
```

```
int strcmp( s1, t1)
char *s1, *t1;
{for(; *s1==*t1;s1++,t1++)
    if(*s1=='\0') return 0;
  return *s1-*t1;
 }
```

§5 指针变量与字符数组

例2(考题)：输入一个不包含空格的字符串，判断输入的字符串是否为回文。回文是相对中心左右对称的字符串。
例如：level、abccba均是回文串。

```
#include <stdio.h>
int f(char *p)
{ char *p1, *p2;
  p1=p2=p;
  while(*p2++);
        (1) ;
  while( (2) )
   { if( (3) ) return 0;
     p1++; p2--;
   }
 return 1;
}
```

```
main( )
{ char s[200];
  printf("输入一个字符串：");
  scanf("%s", s);
  if( (4) ) printf("字符串%s是回文!",s);
  else printf("字符串%s不是回文!",s);
}
```

（1）p2=p2-2　　（2）p1<p2

（3）*p1!=*p2　　（4）f(s)

§6 指针变量与函数

1. 编译系统为一个C函数分配的入口地址称为该函数的指针

2.C语言中，函数名代表函数的指针

3.指向函数的指针变量

——语法I： funtype (*指针变量名)( );

——语法II： funtype (*指针变量名)(形参类型表);

——语法III： funtype (*指针变量名)(形参定义表);

eg1: int (*p)( );

eg2: int (*p)(int,int);

eg3: int (*p)(int x,int x);

eg4: p=max; /*p指向max函数*/

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

§6 指针变量与函数

4. 函数调用方法(函数名法、函数指针变量法)

——语法I： 函数名(实参表)

——语法II： (*函数指针变量)(实参表)

```
#include <stdio.h>
main( )
{ int max(int,int);
  int a, b, c;
   scanf("%d,%d", &a, &b);
   c=max(a, b);
  printf("a=%d,b=%d,max=%d",a,b,c);
 }
```

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

§6 指针变量与函数

4. 函数调用方法(函数名法、函数指针变量法)

——语法I： 函数名(实参表)

——语法II： (*函数指针变量)(实参表)

——语法III： 函数指针变量(实参表)

```
#include <stdio.h>
main( )
{ int max(int,int);
  int a, b, c, (*p)(int,int);
   scanf("%d,%d", &a, &b);
   c=(*p)(a, b);   /*可写为: c=p(a, b); */
  printf("a=%d,b=%d,max=%d",a,b,c);
 }
```

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

§6 指针变量与函数

例(考题)：以下程序输出的第一行是 10,20,30 ，

第二行是 10,20,200 。

```
void  f1(int x,int y,int *sum)
{ *sum=x+y;
   x++; y++;
 }
void f2(int a,int b,int *p)
{ *p=a*b;
   a+=b; b-=a;
}
```

```
main( )
{ void (*f )( );
   int a=10, b=20, c=100;
   f=f1;
   f(a, b, &c);
   printf("%d,%d,%d\n", a, b, c);
   f=f2;
   f(a, b, &c);
   printf("%d,%d,%d\n", a, b, c);
}
```

§7 返回指针值的函数

K&R传统语法

```
类型名 *函数名([形参名])
[形参定义]
{ [内部变量定义]
   [语句集合]
}
```

ANSI原型语法

```
类型名 *函数名([形参定义])
{ [内部变量定义]
   [语句集合]
}
```

```
eg1: int *a(int x,int y)
        {……}
```

```
eg2: int (*a)(int x,int y)
        {……}
```

错误

```
eg3:  int *a(int x,int y );
```

声明

```
eg4:  int (*a) (int x ,int y);
```

定义

§7 返回指针值的函数

例1（考题）：以下程序的输出结果是 hane 。

```
char *fun(char *s)
{ int i,j;
  for(i=j=0;s[i]!='\0';i++)
    if(s[i]!='c') s[j++]=s[i];
  s[j]='\0';
 return s;
}
main( )
{ printf("%s",fun("chance"));
}
```

§7 返回指针值的函数

例2：有成绩数组score[3][4]，编写fail函数，找出有不及格课程的学生供main打印。

```
#include <stdio.h>
int*fail((*p)[4])
{ int *pt, j;
  pt=*(p+1);
  /*假设无不及格*/
  for(j=0;j<=3;j++)
    if(*(*p+j)<60)  pt=*p;
  return pt;
 }
```

```
main( )
{ int i, j, *p;    /*p接收fail返回地址*/
  int score[ ][4]={60,70,…,66};
  int *fail(int (*p)[4]);
  for(i=0;i<=2;i++)
  { p=fail(score+i);
    if(p==*(score+i))
    {for(j=0;j<=3;j++)
      printf("%6d",*(p+j));
      printf("\n");}
  } }
```

§8 指针数组和多级指针

一、指针数组

1.含义：指针数组是同类型指针变量(值)的集合

2.定义方法：type *数组名a[数组长度];

eg1: float *f1[100];

eg2: float (*f2)[100];

```
eg3: int a=2, b=3, c, d;
     int  *p[4];
     p[0]=&a;  p[1]=&b; p[2]=&c;  p[3]=&d;
     *p[2]=*p[0]+*p[1]   ←——c=a+b;
```

eg4: char *lineptr[100];

§8 指针数组和多级指针

例1(考题)： 若有以下声明和语句：

```
int  t[3][3],*pt[3],k;
  for(k=0;k<3;k++)  pt[k]=&t[k][0];
```

则表达式*(*(pt+1)+2)所引用的是 C 。

A. t[2][0]　　B. &t[2][0]　　C. t[1][2]　　D. &t[1][2]

例2(考题)：下列程序输出的第一行是 6,6 ， 第二行是 9,9 。

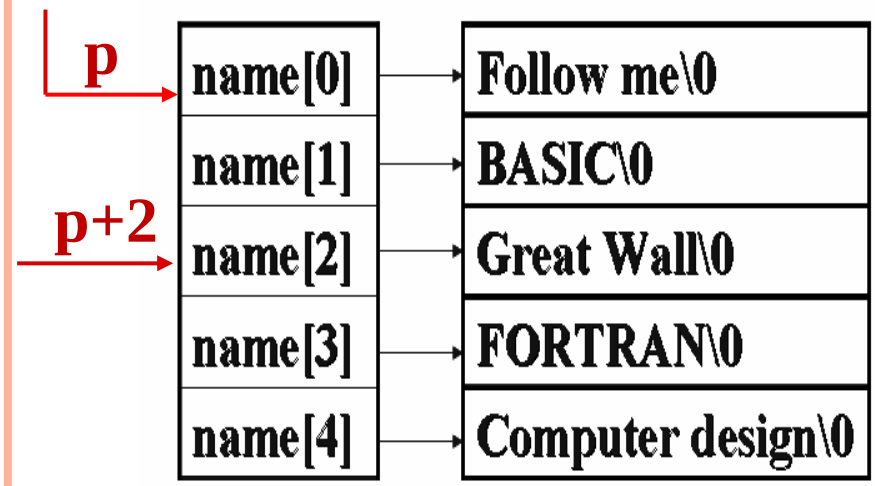
```
main( )
{ int i, p[3][3]={1,2,3,4,5,6,7,8,9},*p1[3],(*p2)[3];
  for( i=0; i<3; i++)  p1[i]=p[i];
  p2=p;
  for(i=1; i<3; i++)
    printf("\n%d,%d", *(*(p1+i)+1)+1, *(*++p2+1)+1);
}
```

§8 指针数组和多级指针

二、多级指针

类型标识符 **指针变量名；

```
main( )
{ char *name[5]={“Follow me”,
   ”BASIC”,”Great Wall”,
   “FORTRAN”,
   ”Computer design”};
   char **p;
   p=&name[0];
   p=p+2;
   printf(“%s\n”, *p);
   printf(“%c\n”,**++p);}
```



Great Wall
F

§8 指针数组和多级指针

例1（考题）：以下程序运行时输出的第一行是 8,4 ，

第二行是 5,8 。

```
main( )
{ int i, p[9]={1,2,3,4,5,6,7,8,9}; *p1[3], **p2;
  for(i=0; i<3; i++)   p1[i]=&p[6-3*i];
  p2=p1+2;
  for(i=1; i<3; i++)
   printf("%d,%d\n", *(++p1[i]+2)+1,
                          *(*p2--+1)+2);
 }
```

§8 指针数组和多级指针

例2(考题)：以下程序的运行结果是____。

```
main( )
{ static char c[ ][6]={“QUICK”,”FOX”,”JEMP”,”DAZY”};
  static char *cp[ ]={c[0],c[1],c[2],c[3]},**cpp=cp;
  printf(“%c”,**++cpp);}
```

A.F　　B.O　　C.A　　D.E

A

例3(考题)：以下程序的运行结果是____。

```
main( )
{ int  x[5]={2,4,6,8,10},*p,**pp;
  p=x;
  pp=&p;
  printf(“%d”,*(p++));
  printf(“%3d\n”,**pp); }
```

2 4

§8 指针数组和多级指针

例4(考题)：以下程序中，函数encrypt的功能是对第一个形参指向的字符串做加密处理，函数返回加密后字符串的首地址。

加密算法：判断字符串中每个字符是否为英文字母，若不是字母则保持原字符不变；若是大写字母，则用字母表中该大写字母对应的小写字母之后的第n个小写字母取代原字母；若是小写字母，则用字母表中该小写字母对应的大写字母之后的第n个大写字母取代该字母。

大写字母表和小写字母表均被看成是首尾相连的环形表。例如，当n=3时，若原字母是a，则加密后该字符被D取代；若原字母是Y，则加密后该字符被b取代。

例：n=3时，

dLLA加密为Good

例：n=4时，

hQYG加密为Luck

§8 指针数组和多级指针

```
#include <stdio.h>
#include <ctype.h>
char *encrypt( ___(1)___ ,int n)
{ int i,t;
  for(i=0;a[i]!='\0';i++)
  {  if(isalpha(a[i]))
     {  t=(toupper(a[i])-'A'+n)%26;
        a[i]= ___(2)___ ?'A'+t:'a'+t;
     }
  }
      ___(3)___ ;
}
void main( )
{ char *s[2]={"dLLA","hQYG"};
  printf("%s\n",encrypt(s[0],3));   /*输出Good*/
  printf("%s\n",encrypt(s[1],4));   /*输出Luck*/  }
```

(1) char *a或 char a[ ]

(2) a[i]>='a'&&a[i]<='z'

或 islower(a[i])

(3) return a