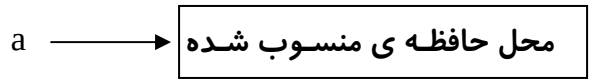
مبانی کامپیوتر و برنامه‌نویسی به زبان C

فصل هفتم: آرایه ها

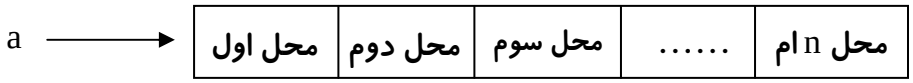
۷-۲-۱ آرایه های یک بعدی

تعریف یک خانه در حافظه



اکنون می خواهیم بدانیم که آیا می توان به جای یک خانه ی حافظه، تعدادی خانـه را بـه یـک نـام منسـوب نمود و حالتی به شکل زیر را به وجود آورد.

محل های حافظه ی منسوب شده به



نحوه ی تعریف

[« تعداد محل های مورد نیاز »] « نام آرایه »

```
#define MAXSTD  ۵۰
int  stno[۱۰۰], m, n,  hours[MAXSTD];
char first-name [۲۰],  last-name [۳۰];
long double  total[MAXSTD \ ۵];
```

نحوه ی استفاده

```
stno[۰]
```

```
stno[MAXSTD]
hourse[( k-۱ ) * j] و total[k-j]
```

عملگر زیرنویس (subscript)

مثال های زیر همه به خانه ی خاصی از آرایه ی stno رجوع می کنند.

```
stno[۲۵] stno[m] stno[m*n+۲] stno[stno[۳۱]] stno[m*hours[stno[۱۲]*n]]
```

به برنامه کوچک زیر توجه کنید:

```
int marks[۱۰۰], i = ۱۰۰۰;
marks[i] = i \ ۲;
int i; numbers[۵۰];
for (i = ۰; i < ۵۰; i++)
     scanf("%d", &numbers[i]);
for  (i = ۴۹; i >= ۰; i--)
     printf ("%d \n", numbers[i]);
```

مقدار دهی اولیه

```
int num[۵] = {۱۲, ۲, -۴, ۰, ۶۷۸}, sum[] = {۰, ۶, ۱, ۱۰, ۱۲};
float total[۵۰] = {۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸, ۹, ۱۰};
char  s[۵] = {'a', 'b', 'c', 'd', 'e'};
float  p[۵] = {۸, ۷, ۶, ۵, ۴, ۳, ۲, ۱};      /* دستور غلط */
```

۷-۲-۲ مرتب کردن مقادیر

```
#include <stdio.h>

main()
{  long a0, a1, a2,. . . , a99, t;

   scanf("%ld%ld . . . %ld",
         &a0, &a1, . . . , &a99);
   if (a0 > a1)
   {  t = a0; a0 = a1; a1 = t;}
   if (a0 > a2)
   {  t = a0; a0 = a2; a2 = t;}
   .
   .
   .
   if (a0 > a99)
   { t = a0; a0 = a99; a99 = t;}
   if (a1 > a2)
   { t = a1; a1 = a2; a2 = t;}
   .
   .
   .
   if (a1 > a99)
   { t = a1; a1 = a99; a99 = t;}
   if (a2 > a3)
   { t = a2; a2 = a3; a3 = t;}
   .
   .
   .
   if (a2 > a99)
   { t = a2; a2 = a99; a99= t;}
   .
   .
   .
   if (a97 > a98)
   { t = a97; a97 = a98; a98 = t;}
   if (a97 > a99)
   { t = a97; a97 = a99; a99 = t;}
   if (a98 > a99)
   { t = a98; a98 = a99; a99 = t;}
   printf("%ld,%ld, . . . ,%ld\n",
          a0, a1, . . . , a99);
   return 0;
}
```

```
#include <stdio.h>

main()
{  long a, b, c, t;

   scanf("%ld%ld%ld", &a, &b, &c);
   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("%ld, %ld, %ld\n", a, b, c);
   return 0;
}
```

شکل ۷-۱: متن برنامهٔ مرتب کردن سه عدد.

```
#include <stdio.h>
#define MAXSIZE ۱۰۰

main()        /* برنامهٔ مرتب کردن مقادیر با استفاده از روش اندیس‌دهی برای دستیابی به خانه‌های آرایه */
{  long a[MAXSIZE], t;              /* تعریف آرایهٔ محل ذخیرهٔ مقادیر */
   int i, j, n;

   scanf("%d", &n);                 /* خواندن تعداد عددها و خود آنها */
   for (i = ۰; i < n; i++)
      scanf("%ld", &a[i]);
   for (i = ۰; i < n - ۱; i++)      /* حلقه‌های تکرار مقایسه برای مرتب کردن */
      for (j = i + ۱; j < n; j++)
         if (a[i] > a[j])           /* مقایسه و درصورت لزوم جابه‌جا کردن دو مقدار مقایسه شده */
         {  t = a[i];
            a[i] = a[j];
            a[j] = t;
         }
   for (i = ۰; i < n; i++)          /* حلقهٔ تکرار چاپ مقادیر مرتب شده */
      printf("%ld\n", a[i]);
   return ۰;
}
```

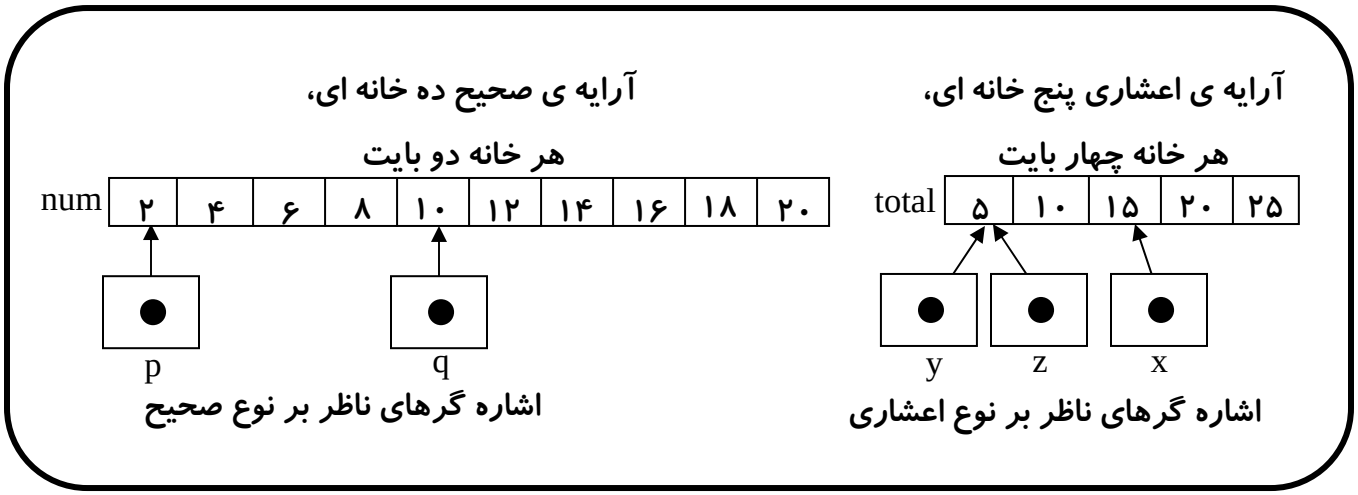
شکل ۷-۷: متن برنامهٔ ۷-۱، مرتب کردن تعدادی عدد به روش اندیس‌دهی برای دستیابی به خانه‌های آرایه.

```
#include <stdio.h>
#define MAXSIZE ۱۰۰

main()                                  /* برنامهٔ بررسی مرتب بودن اعداد */
{  long a[MAXSIZE];
   int i, j, n;

   scanf("%d", &n);                     /* خواندن تعداد عددها و خود آنها */
   for (i = ۰; i < n; i++)
      scanf("٪۸ld", &a[i]);
                                        /* حلقهٔ تکرار کنارگذاشتن مقادیر متوالی و مساوی */
   for (i = ۰; a[i] == a[i + ۱] && i < n - ۱; i++)
      ;
   if (i == n - ۱)                      /* حالت استثنایی که همهٔ عددها مساوی هستند */
      printf("All numbers are equal.\n");
   else if (a[i] < a[i + ۱])            /* بررسی با فرض ترتیب صعودی */
   {                                    /* حلقهٔ تکرار بررسی ترتیب صعودی */
      for (j = i; a[j] <= a[j + ۱] && j < n - ۱; j++)
         ;
      if (j == n - ۱)                   /* ترتیب صعودی برقرار است */
         printf("Numbers are in ascending order.\n");
      else                              /* ترتیب صعودی از یک خانهٔ خاصی به بعد به هم خورده است */
         printf("Numbers are not ordered.\n");
   }
   else                                 /* بررسی با فرض ترتیب نزولی */
   {                                    /* حلقهٔ تکرار بررسی ترتیب نزولی */
      for (j = i; a[j] >= a[j + ۱] && j < n - ۱; j++)
         ;
      if (j == n - ۱)                   /* ترتیب نزولی برقرار است */
         printf("Numbers are in descending order.\n");
      else                              /* ترتیب نزولی از یک خانهٔ خاصی به بعد به هم خورده است */
         printf("Numbers are not ordered.\n");
   }
   return ۰;
}
```

شکل ۷-۸: متن برنامهٔ ۷-۲، خواندن، بررسی مرتب بودن و تعیین نوع ترتیب تعدادی عدد.

۱-۳-۷ کار با آرایه ها از طریق اشاره گرها

```
float *x , *y, *z, total[] = {۵, ۱۰, ۱۵, ۲۰, ۲۵};

int  num[۱۰] = {۲, ۴, ۶, ۸, ۱۰, ۱۲, ۱۴, ۱۶, ۱۸, ۲۰};

int *p, *q;
p = num;
q = num + ۴;

x = total + ۱;

y = &total[۰];

++x;
z = x - ۱;

z --;
```



شکل ۴-۷ : نمونه ای از ارتباط اشاره گرها و آرایه ها.

۷-۳-۲ محاسبات بر روی اشاره گرها

```
int num [۱۰] = {۲, ۴, ۶, ۸, ۱۰, ۱۲, ۱۴, ۱۶, ۱۸, ۲۰}, *p, *q
p = num;
q = num + ۴;
printf ("%d, %d, %d, %d \n", num[۰], p[۵], &num[۰], &p[۵]);
printf ("%d, %d, %d, %d \n", *p, *(num+۵), q - ۴, p + ۵);
```

به عنوان مثالی دیگر به دستورهای زیر و نتیجه ی اجرای آن ها توجه کنید.

```
long a[] = {۱۰, ۲۰, ۳۰, ۴۰, ۵۰, ۶۰}, *lp;
lp = a + ۳;
printf ("%ld %ld %ld %ld %ld %ld\n", a[۳], lp[۰], a[۰], lp[-۳],
lp[۲], a[۵]);
printf ("%ld %ld %ld %ld %ld %ld \n", *(a+۳), *lp, *a, *(lp-۳),
*(lp+۲), *(a+۵)) ;
printf ("%ld %ld %ld %ld %ld %ld ", *a+۵, *lp-۳, a, a+۱, lp, lp-۳ )
;
```

نتیجه ی اجرا:

```
۴۰   ۴۰   ۱۰   ۱۰   ۶۰   ۶۰
۴۰   ۴۰   ۱۰   ۱۰   ۶۰   ۶۰
۱۵   ۳۷   ۷۳۳۴   ۷۳۳۸   ۷۳۴۶   ۷۳۳۴
```

```
#include <stdio.h>
#define MAXSIZE ۱۰۰

main()                                  /* برنامهٔ مرتب کردن مقادیر با استفاده از اشاره‌گرها */
{  long a[MAXSIZE], t, *lptr, *kptr;
   int n;
```

```
#include <stdio.h>
#define MAXSIZE ۱۰۰

main()                                  /* برنامهٔ بررسی مرتب بودن اعداد با استفاده از اشاره‌گرها */
{  long a[MAXSIZE], *pi, *pj;
   int n;

   scanf("%d", &n);                     /* خواندن تعداد عددها و خود آنها */
   for (pi = a; pi < a + n; pi++)
      scanf("%ld", pi);
   for (pi = a; *pi == *(pi + ۱) && pi < a + n - ۱; pi++)
      ;                                 /* حلقهٔ تکرار کنارگذاشتن مقادیر متوالی و مساوی */
   if (pi == a + n - ۱)                 /* حالت استثنایی که همهٔ عددها مساوی هستند */
      printf("All numbers are equal.\n");
   else if (*pi < *(pi + ۱))            /* بررسی با فرض ترتیب صعودی */
   {  for (pj = pi; *pj <= *(pj + ۱) && pj < a + n - ۱; pj++)
         ;                              /* حلقهٔ تکرار بررسی ترتیب صعودی */
      if (pj == a + n - ۱)              /* ترتیب صعودی برقرار است */
         printf("Numbers are in ascending order.\n");
      else                              /* ترتیب صعودی از یک خانهٔ خاصی به بعد به هم خورده است */
         printf("Numbers are not ordered.\n");
   }
   else                                 /* بررسی با فرض ترتیب نزولی */
   {  for (pj = pi; *pj >= *(pj + ۱) && pj < a + n - ۱; pj++)
         ;                              /* حلقهٔ تکرار بررسی ترتیب نزولی */
      if (pj == a + n - ۱)              /* ترتیب نزولی برقرار است */
         printf("Numbers are in descending order.\n");
      else                              /* ترتیب نزولی از یک خانهٔ خاصی به بعد به هم خورده است */
         printf("Numbers are not ordered.\n");
   }
   return ۰;
}
```

شکل ۷-۱۱: متن برنامهٔ ۷-۵، خواندن، بررسی مرتب بودن و نوع ترتیب اعداد با استفاده از اشاره‌گرها.

جدول ۷-۱ : نمونه هایی از عملیات روی آرایه ها و اشاره گرها.

سطرهای برنامه	نام خانه ی مورد تغییر	محتویات خانه پس از اجرای دستور	آدرس خانه ی مورد تغییر
main ( )			
{char info[۳] = {'x', 'y', 'z'};	info[۰]	'x'	۱۲۳۴
	info[۱]	'y'	۱۲۳۵
	info[۲]	'z'	۱۲۳۶
char *infp = &info [۰];	infp	۱۲۳۴	۱۰۰۰
char tmp;	tmp		۹۸۷۶
tmp = info[۰];	tmp	'x'	۹۸۷۶
tmp = *(info + ۲);	tmp	'z'	۹۸۷۶
tmp = *(infp + ۱);	tmp	'y'	۹۸۷۶
tmp = *infp;	tmp	'x'	۹۸۷۶
infp = info + ۱;	infp	۱۲۳۵	۱۰۰۰
tmp = *infp;	tmp	'y'	۹۸۷۶
tmp = *(infp + ۱);	tmp	'z'	۹۸۷۶
infp = info;	infp	۱۲۳۴	۱۰۰۰
tmp = *++infp;	infp	۱۲۳۵	۱۰۰۰
	tmp	'y'	۹۸۷۶
tmp = ++*infp;	info[۱]	'z'	۱۲۳۵
	infp	۱۲۳۵	۱۰۰۰
	tmp	'z'	۹۸۷۶
tmp = *infp++;	infp	۱۲۳۶	۱۰۰۰
	tmp	'z'	۹۸۷۶
tmp = *infp;	tmp	'z'	۹۸۷۶
return ۰;			
}			

۷-۴ آرایه های چند بعدی

۷-۴-۱ : نحوه ی تعریف و مقدار دهی اولیه

```
#define MAXROW ۵۰
#define MAXCOL ۶۰
int m[۱۰][۲۰], log[۵][۶][۸];
float res[MAXROW][MAXCOL];
char names[MAXROW + MAXCOL][MAXCOL - ۱۰];
```

تجسم برنامه نویس از آرایه

تخصیص حافظه به آرایه ها

```
int  a[۳][۴] = { {۲, ۴, ۶, ۸}, {۱, ۳, ۵, ۷}, {۱, ۲, ۳, ۴}};
int  a[۳][۴] = {۲, ۴, ۶, ۸, ۱, ۳, ۵, ۷, ۱, ۲, ۳, ۴}};
```

۷-۴-۲ نحوه ی دسترسی به عناصر آرایه

	ستون صفرم	ستون یکم	ستون دوم	ستون سوم
سطر صفرم	a[۰][۰]	a[۰][۱]	a[۰][۲]	a[۰][۳]
سطر یکم	a[۱][۰]	a[۱][۱]	a[۱][۲]	a[۱][۳]
سطر دوم	a[۲][۰]	a[۲][۱]	a[۲][۲]	a[۲][۳]

آرایه ی a

نحوه ی تجسم آرایه ی دو بعدی [۴] [۳] a در قالب یک جدول همراه با اندیس عناصر آن

a[۰]	a[۰][۰]	a[۰][۱]	a[۰][۲]	a[۰][۳]
a[۱]	a[۱][۰]	a[۱][۱]	a[۱][۲]	a[۱][۳]
a[۲]	a[۲][۰]	a[۲][۱]	a[۲][۲]	a[۲][۳]

سطر صفرم / سطر یکم / سطر دوم

نحوه ی تجسم آریه ی دو بعدی a[۳][۴] در قالب آرایه ای از آرایه ها همراه با اندیس عناصر آن

- a یک ثابت اشاره گر است که آدرس خانه ی صفرم آرایه ی اصلی می باشد.
- a معادل است با &a[۰] یعنی آدرس شروع آرایه ای که طول هر خانه ی آن معادل طول چهار خانه از نوع int است. a[۰]، a[۱] و a[۲] ثابت اشاره گر هستند و به ترتیب آدرس خانه ی صفرم از سطرهای صفرم، اول و دوم (یا آدرس خانه ی صفرم از آرایه های موجود در خانه های آرایه ی اصلی) هستند.

آدرس خانه ی شروع آرایه ی دو بعدی &a[۰][۰]

- برای رجوع به خانه ی سطر iام و ستون jام از آرایه ی a می توان به جای a[i][j] از عبارت *(a[i]+j) نیز استفاده نمود. a[i]+j معادل است با &a[i][j] و *(a[i]) خانه ی صفرم سطر iام را مورد دستیابی قرار می دهد.
- (*(a + i))[j] که در آن عبارت *(a + i) معادل a[i] می باشد و زیرنویس [j] به آن اضافه شده است.
- *(*(a + i) + j) که در آن عبارت *(a + i) معادل a[i] می باشد که آدرس شروع سطر iام است سپس به اندازه ی j خانه جلو رفته و با استفاده از عملگر * محتویات آن مورد دستیابی قرار می گیرند.
- *(&a[۰][۰] + ۴*i + j) که در آن به تعداد خانه های تخصیص یافته از شروع آرایه جلو رفته و سپس خانه ی مزبور با استفاده از عملگر * مورد دستیابی قرار گرفته است.

روش اول

```
for  (i = ۰; i < ۳; i++)
       for (j = ۰; j < ۴; a[i][j] = i * j,  j++)
for (i = ۰; i < ۳; i++)
 {   for (j = ۰; j < ۴; j++)
           printf ("%d", a[i][j]);
     printf ("\n");
}
```

روش دوم

```
for  (i = ۰;  i < ۳;  i ++)
       for (j = ۰; j < ۴; *(a[i] + j) = i  *  j, j++);
for (i = ۰; i < ۳; i++)
{    for (j = ۰; i < ۴; j++)
           printf ("%d ", *(a[i] + j));
      printf ("\n");
}
```

روش سوم

```
for (i = ۰;  i < ۳;  i++)
     for (j = ۰; j < ۴; (*(a + i))[j] = i * j, j++);
for (i = ۰; i < ۳; i++)
{    for (j = ۰; j < ۴; j++)
           printf ("%d ", (*(a + i))[j]);
      printf ("\n");
}
```

روش چهارم

```
for (i = ۰; i < ۳; i++)
      for (j = ۰; j < ۴; *(*(a + i)) + j) = i * j,  j++);
for (i = ۰; i < ۳; i++)
{     for (j = ۰; j < ۴; j++)
```

```
            printf ("%d ", *(*(a + i)) + j));
        printf ("\n") ;
}
```

روش پنجم

```
for (i = ۰;  i < ۳;  i++)
      for (j = ۰; j < ۴;  *(&a[۰][۰] + i * ۴ + j) = i * j, j++ );
for (i = ۰; i < ۳; i++)
{    for (j = ۰; j < ۴;  j++)
            printf ("%d ", *( &a[۰][۰] + i * ۴ + j);
      printf ("\n");
}
```

```
#include <stdio.h>
#define PROVINCES ۳۰
#define YEARS     ۱۰
#define STARTYEAR ۷۱
#define ENDYEAR   ۸۰

main()                    /* برنامهٔ تهیهٔ گزارش تولید ده سالهٔ گندم در استانهای کشور */
{  long crop[PROVINCES][YEARS], totprov[PROVINCES], totyear[YEARS];
   long grandtot = ۰;
   int i, j;

   for (i = ۰; i < PROVINCES; i++)/* حلقه‌های تکرار خواندن داده‌ها و ذخیره در آرایه */
      for (j = ۰; j < YEARS; j++)
         scanf("٪ld", &crop[i][j]);
   for (i = ۰; i < PROVINCES; i++)/* حلقه‌های تکرار محاسبهٔ جمع تولید ده سالهٔ هر استان */
   {  totprov[i] = ۰;
      for (j = ۰; j < YEARS; j++)
         totprov[i] += crop[i][j];
      grandtot += totprov[i];     /* محاسبهٔ جمع تولید ده سالهٔ کشور */
   }
   for (i = ۰; i < YEARS; i++)     /* حلقه‌های تکرار محاسبهٔ جمع تولید هر سالهٔ کشور */
   {  totyear[i] = ۰;
      for (j = ۰; j < PROVINCES; j++)
         totyear[i] += crop[j][i];
   }
   printf("\f      Wheat production report for year"
          " ٪۲d to ٪۲d\n", STARTYEAR, ENDYEAR);/* چاپ عنوان گزارش */
   printf("                  ");
   for (i = STARTYEAR; i <= ENDYEAR; i++)   /* حلقهٔ تکرار چاپ تیتر سالها */
      printf("   ٪۳d  ", i);
   printf("   Totals\n");
   for (i=۰; i<PROVINCES; i++)/* حلقه‌های تکرار چاپ تولیدات سالانه و جمع تولید هر استان*/
   {  printf("province no ٪۲d ", i + ۱);
      for (j = ۰; j < YEARS; j++)
         printf("٪۶ld   ", crop[i][j]);
      printf("٪۸ld \n", totprov[i]);
   }
   printf("   Totals    ");
   for (i = ۰; i < YEARS; i++)     /* حلقهٔ تکرار چاپ تولید ملی در هر سال */
```

شکل ۷-۱۲: متن برنامهٔ ۷-۶، چاپ گزارش تولید ده سالهٔ گندم در استانهای کشور.

۷-۵ آرایه ها و توابع

۷-۵-۱ آرایه ها به عنوان پارامتر

روش ارسال آرگومان ها همان روش انتقال مقدار است منتها به جای انتقال مقادیر موجود در خانه های آرایه، آدرس ابتدای آن به برنامه ی احضار شونده منتقل می گردد.

در برنامه ی احضار شونده، امکان تغییر محتویات خانه های آرایه نیز وجود دارد. حال اگر بخواهیم از اعمال این گونه تغییرات جلوگیری کنیم. کافی است در تعریف پارامتر مربوطه در برنامه ی احضار شونده قبل از نوع پارامتر کلمه ی const اضافه گردد.

```
#include <stdio.h>

double sum(double vector[], int n)   /* معرفی تابع و پارامترهای آن */
{  int i;
   double sum = ۰.۰;

   for (i = ۰; i <  n; i++)
      sum +=  vector[i];              /* جمع زدن محتویات خانه های آرایه */
   return sum ;
}

main()
{  double numbers[۱۰۰], total;       /* تعریف متغیرها از جمله آرایه و تخصیص حافظه به آنها */
   int  n ;

   for (n = ۰; n < ۱۰۰;  n++)        /* خواندن یکصد عدد از ورودی */
      scanf ("%lf", &numbers[n]);
   total = sum(numbers, n);          /* محاسبه ی مجموع محتویاتخانه های صفرم تا نودو نهم */
   printf("sum of all numbers:%f\n", total);
   total = sum (numbers, ۵۰);        /* محاسبه ی مجموع محتویات خانه های صفرم تا چهل و نهم */
   printf("sum of the first ۵۰ numbers:%f\n", total);
   total = sum(&numbers[۵۰], ۵۰);    /* محاسبه ی مجموع محتویات خانه های پنجاهم تا نودو نهم */
   printf("sum of the last ۵۰ numbers:%f\n", total);
   total = sum (numbers + ۲۰, ۳۰) ; /* محاسبه ی مجموع محتویات خانه های بیستم تا چهل و نهم */
   printf("sum of ۳۰ numbers after the first ۲۰ numbers:%f\n", total);
   return ۰;
```

شکل ۷-۶ الف: روشهای مختلف ارسال تمام یک بخشی از یک آرایه به عنوان آرگومان در احضار تابع.

۷-۵-۲ تابع احضار شونده

به تابع sum توجه نمایید سطر اول این تابع به شکل

```
double sum (double vector [۱۰۰], int n)
```

است. که

```
double sum(double *vector, int n)
```

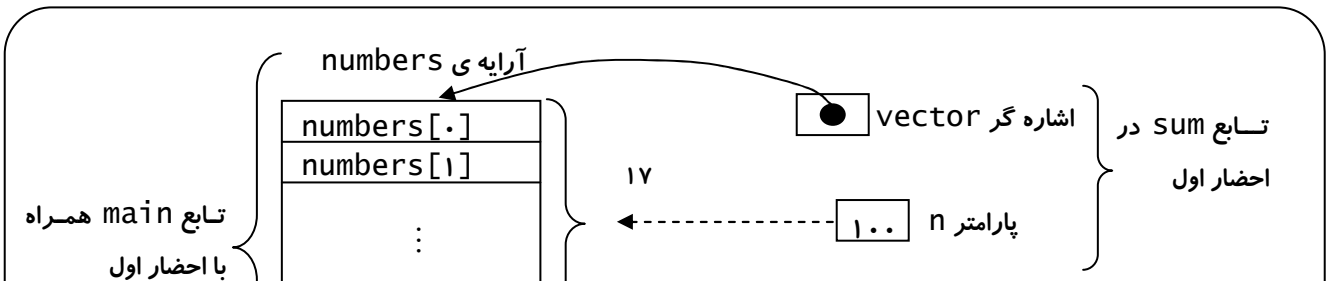
هم همان معنی را خواهد داد. اگر اشاره گر تعریف کنیم باید اشاره گر مزبور را در داخل یک زوج پرانتز قرار داد و تعداد عنصر ابعاد دوم به بعد را در داخل زوج کروشه بعد از پرانتز آورد.

در دستیابی به محتویات خانه های آن ها می توان به جای اندیس دهی از محاسبات روی اشاره گرها استفاده نمود که در این صورت در تابع sum بالا دستور داخل حلقه ی تکرار به شکل

```
sum +=*(vector + i);
```

نوشته می شود.

۳-۵-۷ تابع احضار کننده و مراحل احضار



به برنامه های زیر توجه کنید:

برنامه ی نمونه ی ۷-۷

برنامه ی نمونه ی ۷-۸

برنامه ی نمونه ی ۷-۹

```
#include <stdio.h>
#define MAXSIZE ۱۰۰

                /* . . .  محل قرار دادن تابع بررسی کنندهٔ مرتب بودن مقادیر موجود در آرایه  . . . */
main()                                  /*  برنامهٔ بررسی مرتب بودن اعداد  */
{  long a[MAXSIZE];
   int i, j, n, result;        ۱۸

   scanf("%d", &n);                     /*  خواندن تعداد عددها و خود آنها  */
   for (i = ۰; i < n; i++)
```

```
int check_order                          /* تابعی برای بررسی مرتب بودن مقادیر موجود در یک آرایه */
    (long a[], int n)
{  int i, j;

                                         /* حلقهٔ تکرار کنارگذاشتن مقادیر متوالی و مساوی */
   for (i = ۰; a[i] == a[i + ۱] && i < n - ۱; i++)
      ;
   if (i == n - ۱)                       /* حالت استثنایی که همهٔ عددها مساوی هستند */
      return -۲;                         /* و مقدار ۲- به عنوان جواب برگردانده می‌شود */
   else if (a[i] < a[i + ۱])             /* بررسی با فرض ترتیب صعودی */
   {                                     /* حلقهٔ تکرار بررسی ترتیب صعودی */
      for (j = i; a[j] < a[j + ۱] && j < n - ۱; j++)
         ;
      if (j == n - ۱)                    /* ترتیب صعودی برقرار است، مقدار ۱+ برگردانده می‌شود */
         return ۱;
      else                               /* ترتیب صعودی از یک خانهٔ خاصی به بعد به هم خورده است */
         return ۰;                       /* مقدار صفر به عنوان جواب برگردانده می‌شود */
   }
   else                                  /* بررسی با فرض ترتیب نزولی */
   {                                     /* حلقهٔ تکرار بررسی ترتیب نزولی */
      for (j = i; a[j] > a[j + ۱] && j < n - ۱; j++)
         ;
      if (j == n - ۱)                    /* ترتیب نزولی برقرار است، مقدار ۱- برگردانده می‌شود */
         return -۱;
      else                               /* ترتیب نزولی از یک خانهٔ خاصی به بعد به هم خورده است */
         return ۰;                       /* مقدار صفر به عنوان جواب برگردانده می‌شود */
   }
}
```

شکل ۷-۱۳ ب: متن برنامهٔ ۷-۷، تابع بررسی کنندهٔ مرتب بودن مقادیر و تعیین نوع ترتیب آنها.

```
#include <stdio.h>
#define PROVINCES ۳۰
#define YEARS     ۱۰
#define STARTYEAR ۷۱
#define ENDYEAR   ۸۰

/* . . .  محل قرار گرفتن توابع مربوط به خواندن داده‌ها، محاسبهٔ جمع یک سطر یا یک ستون و چاپ گزارش  . . .*/

main()                                  /*  برنامهٔ تهیهٔ گزارش تولید ده سالهٔ گندم در استانهای کشور  */
{  long crop[PROVINCES][YEARS], totprov[PROVINCES], totyear[YEARS];
   long grandtot = ۰;
   int i;

   read_in(crop);                       /*  احضار تابع اول برای خواندن داده‌های ورودی  */
                        /*  حلقهٔ تکرار محاسبهٔ جمع تولید ده سالهٔ هر استان از طریق احضار تابع دوم  */
                        /*  یکبار برای هر استان و محاسبهٔ جمع تولید ده سالهٔ کشور  */
   for (i = ۰; i < PROVINCES; i++)
      grandtot += find_total(crop, totprov, i, 'R');
                        /*  حلقهٔ تکرار محاسبهٔ جمع تولید هر سالهٔ کشور  */
   for (i = ۰; i < YEARS; i++)
      find_total(crop, totyear, i, 'C');  /*  احضار تابع دوم یکبار برای هر سال  */
                        /*  احضار تابع سوم برای چاپ گزارش  */
   print_out(crop, totprov, totyear, grandtot);
   return ۰;
}
```

شکل ۷-۱۷ ب: متن برنامهٔ ۷-۱۱، تابع اصلی چاپ گزارش تولید گندم با استفاده از سه تابع شکل ۷-۱۷ پ.

```
void read_in(long crop[][YEARS])    /* تابعی برای خواندن داده‌های ورودی */
{  int i, j;

   for (i = ۰; i < PROVINCES; i++)  /* حلقه‌های تکرار خواندن داده‌ها و ذخیره در آرایه */
      for (j = ۰; j < YEARS; j++)
         scanf("٪ld", &crop[i][j]);
   return;
}
       /* تابعی برای محاسبهٔ جمع یک سطر (محصول سالانهٔ یک استان) یا جمع یک ستون (محصول ملی در یک سال) */
long find_total (long crop[][YEARS],long total[],int rc_no,char sw)
{  int i, last;

   total[rc_no] = ۰;
   last = (sw == 'R' ? YEARS : PROVINCES);  /* تعیین تعداد خانه‌های سطر یا ستون */
   for (i = ۰; i < last; i++)    /* حلقهٔ تکرار محاسبهٔ جمع محتویات یک سطر یا یک ستون */
      total[rc_no] += (sw=='R' ? crop[rc_no][i] : crop[i][rc_no]);
   return total[rc_no];          /* برگرداندن حاصل‌جمع محاسبه شده */
}

void print_out                   /* تابعی برای چاپ گزارش مورد نظر */
   (long crop[][YEARS],long totprov[],long totyear[],long grandtot)
{  int i, j;

   printf("\f      Wheat production report for year "
          "١٣٪٢d to ١٣٪٢d\n", STARTYEAR, ENDYEAR);/* چاپ عنوان گزارش */
   printf("                    ");
   for (i = STARTYEAR; i <= ENDYEAR; i++) /* حلقهٔ تکرار چاپ تیتر سالها */
      printf("   ١٣٪٢d  ", i);
   printf("   Totals\n");
   for (i=۰; i < PROVINCES; i++)/* حلقه‌های تکرار چاپ تولیدات سالانه و جمع تولید هر استان*/
   {  printf("province no ٪٢d ", i + ١);
      for (j = ۰; j < YEARS; j++)
         printf("٪٩ld   ", crop[i][j]);
      printf("٪٨ld \n", totprov[i]);
   }
   printf("   Totals    ");
   for (i = ۰; i < YEARS; i++)   /* حلقهٔ تکرار چاپ تولید ملی در هر سال */
      printf("٪٨ld ", totyear[i]);
   printf(" ٪٩ld", grandtot);    /* چاپ کل تولید ده سالهٔ کشور */
   return;
}
```

برنامه ی ۷-۸: تابعی بنویسید که آرایه ی دو بعدی text را که m سطر از آن پرشده وتعداد ستونهای آن با ثابت نمادی ROWSIZE تعریف شده است و همچنین آرایه ی یک بعدی pattern با p خانه، هـر دو از نوع char همراه با m و p را به عنوان پارامتر بگیرد. با فرض اینکه در خانه های هـر دو آرایـه کـد عـددی معادل کاراکترهای خاصی ذخیره شده اسـت تعـداد دفعـات تکـرار محتویـات آرایـه ی pattern حـاوی چهارکاراکتر F، I، N و D در خانه های صفرم تا سوم خود باشد، این تـابع بایـد تعـداد دفعـاتی کـه کلمـه ی FIND در چهارخانه ی متوالی یک سطر ظاهر شده است را یافته به عنوان نتیجه برگرداند (کلمه باید به طور کامل روی یک سطر باشد.)

```
#define ROWSIZE ۸۰
                                        /*  تابع برای شمارش تعداد دفعات تکرار یک الگو در یک متن  */
int count_part
    (char text[][ROWSIZE], char pattern[], int m, int p)
{   int i, j, k, count = ۰;

    for (i = ۰; i < m; i++) /*  حلقه تکرار پیمایش سطرهای آرایه ی حاوی متن  */
        for (j = ۰;j < ROWSIZE-p+۱; j++)   /*  حلقه تکرار پیمایش کاراکترهای موجود در یک سطر  */
        {   for (k = ۰; k < p && text[i][j + k] == pattern[k]; k++)
                ;                                /*  حلقه تکرار بررسی تطبیق الگو با کاراکترهای داخل سطر مورد پیمایش  */
            Count += (k == p);          /*  شمارش در صورت احراز تطبیق کامل  */
        }
    Return count;
}
```

شکل ۷-۱۴: متن برنامهٔ ۷-۸، تابعی برای یافتن تعداد دفعات تطبیق یک الگو در یک متن.

```
#include<stdio.h>
#define ROWSIZE ۶
                /* . . . محل قرار دادن تابع شمارش تعداد دفعات تکرار یک الگو در یک متن . . . */
main()
{                                        /* تعریف و مقداردهی متن مورد بررسی */
   char t[۱۰][۶]={'A', 'R', 'A', 'R', 'A', 'T',
                  'A', 'B', 'C', 'A', 'R', 'A',
                  'X', 'A', 'R', 'A', 'R', 'Q',
                  'S', 'A', 'R', 'T', 'R', 'Z'};
   char p[۳]={'A', 'R', 'A'};    /* تعریف و مقداردهی الگوی مورد نظر */
   int n, i, j;

   printf("The text is: ");          /* چاپ متن مورد بررسی */
   for (i = ۰; i < ۴; i++)
   {  for (j = ۰;j < ۶; j++)
         printf("%c", t[i][j]);
      printf("\n               ");
   }
   printf("\nThe pattern is: ");     /* چاپ الگوی مورد نظر */
   for (i = ۰; i < ۳; i++)
      printf("%c", p[i]);
   printf("\n");
   n = count_pat(t, p, ۴, ۳);        /* احضار تابع برای شمارش تعداد تطبیق الگو در متن */
   printf("\nThe number of matches found is: %d\n", n);     /* چاپ نتیجه */
   return ۰;
}
```

خروجی برنامه:

```
The text is: ARARAT
             ABCARA
             XARARQ
             SARTRZ
The pattern is: ARA
The number of matches found is: ۴
```

شکل ۷-۱۵: متن برنامهٔ ۷-۹، بررسی تطبیق الگو با استفاده از تابع برنامهٔ ۷-۸ همراه با نتیجهٔ اجرا.

نحوه نوشتن توابعی که باید چند مقدار به عنوان نتیجه برگرداند:

```
#include <stdio.h>

/* تابع مجزا برای استخراج ب م م و ک م م دو عدد با دو پارامتر ورودی (اول و دوم) و دو پارامتر خروجی (سوم و چهارم) */
int gcdlcm
   (int m, int n, int *gcd, int *lcm)
{  int tm, tn, tr;

   if (m < ۰ || n < ۰)  /* بررسی مثبت بودن مقدار پارامترها */
      return ۰;                 /* برگشت به تابع احضار کننده با مقدار صفر دال بر عدم موفقیت */
   tm = m;
   tn = n;
   while (tr = tm % tn)         /* حلقهٔ محاسبهٔ ب م م دو عدد */
   {  tm = tn;
      tn = tr;
   }
   *gcd = tn;                   /* ذخیرهٔ ب م م دو عدد در محل مورد اشاره توسط پارامتر سوم */
   *lcm = m * n / tn;           /* محاسبهٔ ک م م و ذخیرهٔ آن در محل مورد اشاره توسط پارامتر چهارم */
   return ۱;              /* برگشت به تابع احضار کننده با مقدار یک دال بر موفقیت */
}

main()                          /* تابع اصلی برای خواندن زوج عددها، احضار تابع فوق و چاپ نتیجه */
{  int m, n, gcd, lcm, success;

   while (scanf("%d%d", &m, &n) != EOF)/* خواندن زوج عددها و بررسی پایان داده‌ها */
   {           /* احضار تابع و ارسال دو مقدار به عنوان پارامتر ورودی و دو آدرس برای ذخیرهٔ ب م م و ک م م در آنها */
      success = gcdlcm(m, n, &gcd, &lcm);
      if (!success)             /* بررسی نتیجه و چاپ پیغام درصورت عدم موفقیت */
         printf("Invalid input data:%d, %d\n", m, n);
      else                      /* چاپ ب م م و ک م م درصورت موفقیت */
      {  printf("GCD(%d, %d) = %d\n", m, n, gcd);
         printf("LCM(%d, %d) = %d\n", m, n, lcm);
      }
   }
   return ۰;
}
```

شکل ۷-۱۸: متن برنامهٔ ۷-۱۲، استخراج ب م م و ک م م دو عدد با استفاده از یک تابع فرعی و تابع اصلی.