مبانی کامپیوتر و برنامه‌نویسی به زبان C

فصل چهارم: ساختارهای انتخاب و تکرار، تابع خواندن

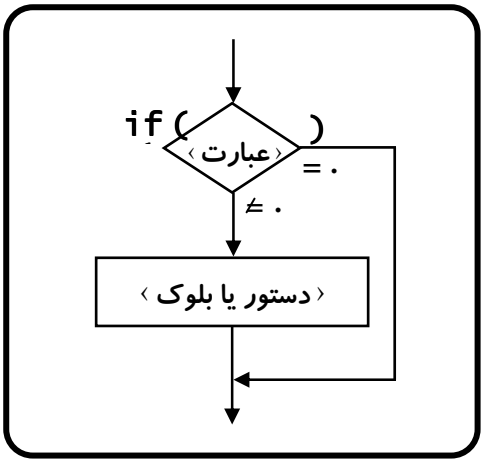
۴-۱ مقدمه

- عبارتها و قواعد ترتیب و تقدم عملگرها تعیین کنندهٔ ترتیب انجام عملیات در داخل یک دستور.
- نیاز به امکاناتی در سطح بالاتر برای کنترل ترتیب اجرای دستورهای تشکیل دهندهٔ یک برنامه.
- ساختارهای انتخاب و تکرار، وجود در غالب برنامه ها برای تصمیم‌گیری و تکرار، چرا ساختار؟
- عملگرهای رابطه‌ای و منطقی برای ساختن شرطها.
- خواندن داده‌ها از ورودی.

۴-۲ دستور شرطی یا if ساده

قالب کلی `if (‹عبارت›) ‹دستور یا بلوک›`

نحوهٔ اجرا



شکل ۴-۱: نحوهٔ اجرای دستور if ساده.

- یکی از حالتهای ساختار انتخـاب (بخش ۳-۴-۱ در سمت راست شکل ۳-۴)
- شرط دستور (عبارت داخل پرانتز)، توجه به آثار جانبی در صورت وجود.
- مفهوم بلوک در زبان C: مجموعهٔ تعدادی دستور که به وسیلهٔ یک زوج آکولاد احاطه شده است.
- عدم وجود عبارت از نوع منطقی با حاصل درست یا نادرست (عبارات فقط از نوع عددی).
- مقدار صفر به معنی ارزش نادرست (جواب خیر) و مقدار غیرصفر به معنی ارزش درست (جواب بله).

مثال: با فرض اینکه مقادیر متغیرهای صحیح a و b به ترتیب ۱۰ و ۵ باشد.

```
if (a - b) printf("%d\n", a);
if (a = a - b) printf("%d\n", a);
```

- لزوم توجه به علامت = به معنی عملگر تخصیص و نه بررسی تساوی دو مقدار.

عملگرهای رابطه‌ای

- عملگرهای دوتایی، بدون اثر جانبی، برای مقایسهٔ دو عملوند هریک در قالب یک عبارت درست.
- انجام مقایسه بین حاصل دو عبارت.
- ارائه مقدار یک درصورت مثبت (درست) بودن جواب مقایسه و صفر درصورت منفی (نادرست) بودن.

جدول ۴-۱: عملگرهای رابطه‌ای همراه با نحوهٔ عمل آنها.

ردیف	شکل عملگر	نحوهٔ کار عملگر
۱	<	اگر حاصل عملوند سمت چپ کوچکتر از حاصل عملوند سمت راست باشد نتیجهٔ عمل یک است وگرنه نتیجه صفر است.
۲	<=	اگر حاصل عملوند سمت چپ کوچکتر از یا مساوی با حاصل عملوند سمت راست باشد نتیجهٔ عمل یک است وگرنه نتیجه صفر است.
۳	>=	اگر حاصل عملوند سمت چپ بزرگتر از یا مساوی با حاصل عملوند سمت راست باشد نتیجهٔ عمل یک است وگرنه نتیجه صفر است.
۴	>	اگر حاصل عملوند سمت چپ بزرگتر از حاصل عملوند سمت راست باشد نتیجهٔ عمل یک است وگرنه نتیجه صفر است.
۵	==	اگر حاصل دو عملوند سمت چپ و راست مساوی باشند نتیجهٔ عمل یک وگرنه نتیجه صفر است.
۶	!=	اگر حاصل دو عملوند سمت چپ و راست مساوی نباشند نتیجهٔ عمل یک وگرنه نتیجه صفر است.

- لزوم توجه به علامت = به معنی عملگر تخصیص و == به معنی بررسی تساوی دو مقدار.
- یک کلمه بودن عملگرهای دو کاراکتری.

تقدم اجرای عملگرها (توضیح در مورد عملگرهای نقیض، عطف و فصل در بخش بعدی).

جدول ۴-۲: تقدم و ترتیب اجرای عملگرها در عبارات (سطر اول دارای بالاترین تقدم است).

تقدم اجرا	ترتیب اجرا	گروه عملگرهای با تقدم مساوی	توضیح
اول	از راست به چپ	! - +	نقیض، منفی‌کردن، مثبت‌کردن
دوم	از چپ به راست	* / %	ضرب، تقسیم، باقیمانده
سوم	از چپ به راست	+ -	جمع، تفریق
چهارم	از چپ به راست	< <= > >=	مقایسه‌ها
پنجم	از چپ به راست	== !=	مقایسه‌های تساوی
ششم	از چپ به راست	&&	عطف
هفتم	از چپ به راست	||	فصل
هشتم	از راست به چپ	=	تخصیص

مثال: با فرض مقادیر ۲، ۵، ۳ و ۱ به ترتیب برای‌متغیرهای صحیح a، b، c و d.

```
if(a+b>=c+d)
{   a=a+1;
    b=b*2;
}
```

```
if(a+2==(c=c+d))
{   a=a*2;
    b=c/2;
}
```

```
if((d=a+b*2!=(d=a+c+1))>a-b)
{   c=c*2;
    d=d-2;
}
```

- عدم نیاز به مقایسهٔ عبارات با صفر.
- معادل بودن `...if(a+b)` با `...if(a+b!=0)` و `...if(!(a+b))` با `...if(a+b==0)`.

عملگرهای منطقی برای ترکیب شرطها

جدول ۴-۳: عملگرهای منطقی همراه با نحوهٔ عمل آنها.

ردیف	شکل عملگر	نوع عملگر	نحوهٔ کار عملگر
۱	&&	دوتایی	اگر حاصل هر دو عملوند سمت چپ و راست غیرصفر باشد، نتیجهٔ این عمل یک است وگرنه نتیجه صفر است.
۲	||	دوتایی	اگر حاصل حداقل یکی از عملوندهای سمت چپ و راست غیرصفر باشد، نتیجهٔ این عمل یک است وگرنه نتیجه صفر است.
۳	!	یکتایی	اگر حاصل عملوند این عملگر غیرصفر باشد، نتیجهٔ عمل صفر است وگرنه نتیجه یک است.

تقدم و ترتیب اجرای این عملگرها در جدول ۴-۲.

جدول ۴-۴: نمونه‌هایی از عبارات و نحوهٔ محاسبهٔ آنها با فرض مقادیر ۱، ۴ و ۱۴ به ترتیب. برای x، y و z.

عبارت	حاصل	توضیح
`x <= ۱ && y == ۳`	۰	حاصل رابطهٔ سمت راست صفر است پس حاصل عملگر && هم صفر می‌شود.
`x <= ۱ || y==۳`	۱	حاصل رابطهٔ سمت چپ یک است پس حاصل عملگر || هم یک می‌شود.
`!(x > ۱)`	۱	حاصل رابطهٔ داخل پرانتز صفر است پس حاصل عملگر ! یک می‌شود.
`!x > ۱`	۰	مقدار x غیرصفر است پس نقیض آن صفر می‌شود که از یک بزرگتر نیست پس حاصل کل عبارت صفر می‌شود.
`(z = !(y - ۴)) > ۰`	۱	مقدار پرانتز داخلی صفر و نقیض آن یک است که داخل متغیر z ذخیره می‌شود، و چون این مقدار از صفر بزرگتر است حاصل کل عبارت یک می‌شود.
`!(x <= ۱ || y == ۳)`	۰	عبارت داخل پرانتز مشابه سطر دوم است که حاصل آن یک می‌شود و نقیض آن صفر خواهد شد.
`x>=۱&&!(y==۳||z+x>=۱۴)||x+y>۲`	۱	مقدار داخل پرانتز یک و نقیض آن صفر است در نتیجه حاصل عملگر && صفر خواهد شد ولی چون حاصل مقایسهٔ آخر یک است نتیجهٔ کل عبارت یک می‌باشد.

- **محاسبه منطقی: عدم محاسبه عملوند دوم اگر با توجه به حاصل عملوندی که اول محاسبه می‌شود، نتیجهٔ عمل قابل تشخیص باشد.**
- **با فرض محاسبه عملوند سمت چپ قبل از عملوند سمت راست.**
- **در عمل &&: مقدار عملوند سمت چپ صفر، نتیجهٔ نهایی هم صفر، عدم نیاز به ادامهٔ محاسبه.**
- **در عمل | |: مقدار عملوند سمت چپ یک، نتیجهٔ نهایی هم یک، عدم نیاز به ادامهٔ محاسبه.**
- **اولویت نحوهٔ محاسبهٔ عملوندها در عملگرهای && و | | نسبت به اعمال تقدمها.**

مثال با فرض محاسبهٔ عملوند سمت چپ قبل از عملوند سمت راست

نمونهٔ اول:

```
int m;
m = ۱;
if (m == ۵ && printf("Right operand\n") > ۱۰)
   printf("Entered if\n");
```

نمونهٔ دوم:

```
int m, n, k;
m = ۱; n = ۹۹; k = ۸۸;
if (m == ۵ && (n = ۵) > k)
   printf("Entered if\n");
printf("n = %d\n", n);
```

نتیجهٔ چاپ:

```
n = ۹۹
```

نمونهٔ سوم:

```
int m, n, k;
m = ۱; n = ۹۹; k = ۸۸;
if (m == ۱ || (k = ۲) > n && (n = ۵))
   printf("Entered if\n");
printf("n = %d k = %d\n", n, k);
```

نتیجهٔ چاپ:

```
Entered if
n = ۹۹ k = ۸۸
```

نمونهٔ چهارم:

```
int m, n, k;
m = ۱; n = ۹۹; k = ۸۸;
if (m == ۴ || (k = ۲) > n && (n = ۵))
   printf("Entered if\n");
printf("n = %d k = %d\n", n, k);
```

نتیجهٔ چاپ:

```
n = ۹۹ k = ۲
```

نمونهٔ پنجم:

```
int m, n, k;
m = ۱; n = ۹۹; k = ۸۸;
if (m == ۴ || (k = ۲) < n && (n = ۵))
   printf("Entered if\n");
printf("n = %d k = %d\n", n, k);
```

نتیجهٔ چاپ:

```
Entered if
n = ۵ k = ۲
```

نمونهٔ ششم:

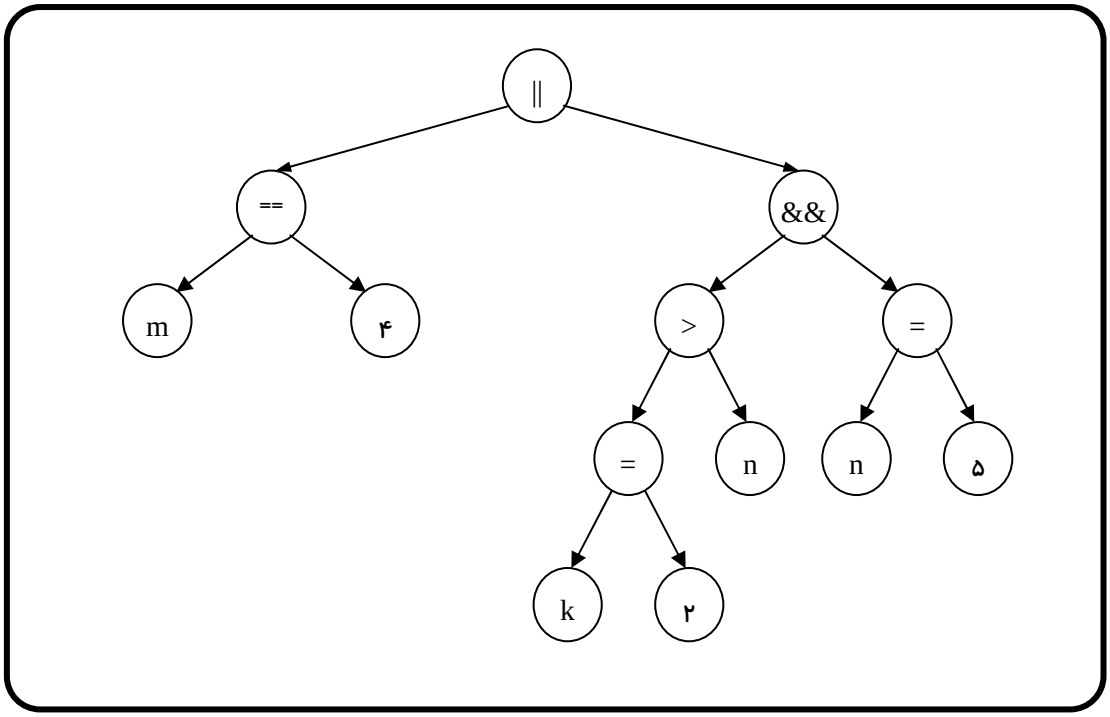
```
int a = ۳, b = ۷, c = ۴, d = ۵, e = ۱۱,f = ۲۲;
if (d = a + b != a + c || (e = c > a) && (f = !(a > b)))
{  a = a + ۱;
   b = b * ۲;
}
printf("a=%d b=%d \nc=%d d=%d e=%d f=%d\n",a,b,c,d,e,f);
```

نتیجهٔ چاپ:

```
a=۴ b=۱۴
c=۴ d=۱ e=۱۱ f=۲۲
```

استفاده از درخت تجزیه برای تعیین روند محاسبه عبارت شرط نمونه چهارم

```
if (m == ۴ || (k = ۲) > n && (n = ۵))
```



شکل ۴-۲: درخت تجزیهٔ متناظر با عبارت شرط در نمونهٔ چهارم از نحوهٔ اعمال عملگرهای منطقی.

مثال: برنامهٔ نمونهٔ ۴-۱

برنامهٔ ۴-۱: برنامه‌ای بنویسید که در آن نخست ضرایب یک معادلهٔ درجه دوم تعریف شود و پس از چاپ ضرایب از اول یک صفحه (هر ضریب روی یک سطر با توضیح مناسب)، معادله حل شده، با توجه به علامت دلتا نتیجه در قالب ریشه‌های مختلف، ریشهٔ مضاعف و بدون ریشهٔ حقیقی روی یک سطر مجزا با یک سطر فاصله از سطر قبلی با توضیحات مربوطه چاپ گردد.

```
#include <stdio.h>
#include <math.h>                                   /* اضافه‌کردن فایل سرآمد حاوی توابع ریاضی */

main()                                              /* برنامهٔ حل معادلهٔ درجه دوم */
{  float a, b, c;
   float delta, x1, x2, x;

   a = 2;
   b = 5.5;
   c = 1.7;
   printf("\fa=%f\n", a);                           /* چاپ ضرایب در این سطر و دو سطر بعدی */
   printf("b=%f\n", b);
   printf("c=%f\n", c);
   delta = b * b - 4 * a * c;                       /* محاسبهٔ دلتا */
   if (delta > 0)                                   /* محاسبه و چاپ ریشه‌های مختلف */
   {  x1 = (-b + sqrt(delta)) / (2 * a);
      x2 = (-b - sqrt(delta)) / (2 * a);
      printf("\nX1=%6.2f\tX2=%6.2f\n", x1, x2);
   }
   if (delta == 0)
   {  x = -b / (2 * a);                             /* محاسبه و چاپ ریشهٔ مضاعف */
      printf("\nX1=X2=%6.2f\n", x);
   }
   if (delta < 0)                                   /* چاپ پیغام عدم وجود ریشهٔ حقیقی */
      printf("\nThe equation has no real root!\n");
   return (0);
}
```

شکل ۴-۷: متن برنامهٔ ۴-۱، حل معادلهٔ درجه دوم با استفاده از دستور شرطی ساده.

- **مبتنی بر روندنمای ۲-۲ در بخش روندنماهای نمونهٔ فصل دوم (شکل ۲-۷).**
- **تنها تفاوت آن با روندنمای مزبور تخصیص ضرایب به جای خواندن از ورودی.**

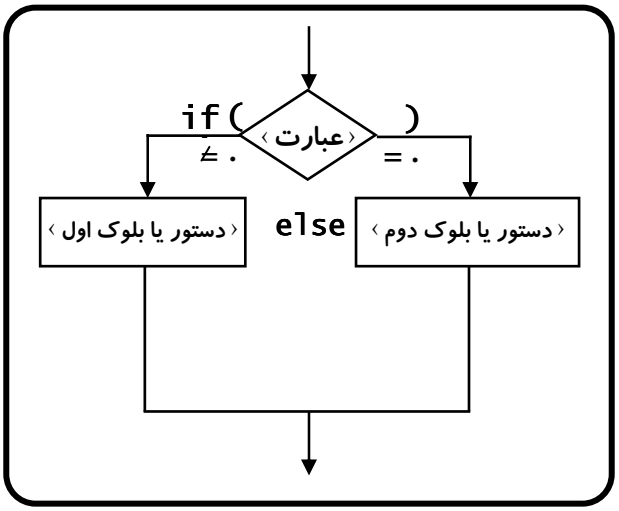
۴-۳ دستور شرطی یا if کامل

- دستور if ساده حالت اگر- آن‌گاه
- دستور if کامل به صورت اگر-آن‌گاه- وگرنه
- پیاده‌سازی حالت دیگری از ساختار انتخاب (بخش ۳-۴-۱ فصل سوم سمت چپ شکل ۳-۴).

قالب کلی دستور if کامل.

if (‹ عبارت ›) ‹ دستور یا بلوک اول › else ‹ دستور یا بلوک دوم ›

نحوهٔ اجرا



شکل ۴-۳: نحوهٔ اجرای دستور if کامل.

مثال: با فرض مقادیر ۱۰ و ۵ به ترتیب برای متغیرهای صحیح a و b.

```
if (a - b)
   printf("%d\n", a);
else
   printf("%d\n", b);
printf("End of if\n");


if (b = (a = a - b) == ۱۰)
   printf("%d\n", a);
else
{  a = b * b;
   printf("%d\t%d\n", a, b);
}
printf("if statement is done\n");
```

مثال: با فرض مقادیر ۵، ۳، ۱، ۷، ۴، ۵ و ۱ به ترتیب برای متغیرهای a، b، c، d، x، y و z.

- **حاصل عبارت داخل پرانتز if برابر یک شده، دستور بعد از پرانتز اجرا می‌گردد.**
- **جملهٔ The compound condition was successful در اثر آن چاپ خواهد شد.**

```
if (a+b >= c*۶ && x >= !(y+z) && a > b || c > d || b < ۳ && x > y)
   printf("The compound condition was successful\n");
else
{  printf("The compound condition was not successful\n");
   a = a * b;
   x = x / y;
}
```

دقت در نوشتن دستورهای if به صورت تودرتو

- **عدم تطبیق شکل ظاهری با مفهوم.**

```
if (a > b)
   if (b > c)
   {  a = a - ۱;
      b = b + ۱;
   }
else
   c = a + b;
```

- **نیاز به یک else تهی برای if دوم یا یک بلوک اطراف if داخلی.**

```
if (a > b)
   if (b > c)
   {  a = a - ۱;

      b = b + ۱;
   }
   else
      ;
else
   c = a + b;
```

```
if (a > b)
{  if (b > c)
   {  a = a - ۱;

      b = b + ۱;
   }
}
else
   c = a + b;
```

مثال: برنامهٔ نمونهٔ ۴-۲

در برنامهٔ ۴-۱ برای به دست آوردن جوابهای معادلهٔ درجه دوم از دستورهای شرطی ساده استفاده گردید. برنامهٔ مزبور را مجدداً بازنویسی نمایید با این تفاوت که برای حل معادلهٔ درجه دوم مورد بحث از دستور شرطی کامل استفاده کنید.

```
#include <stdio.h>
#include <math.h>                          /* اضافه‌کردن فایل سرآمد حاوی توابع ریاضی */

main()                                     /* برنامهٔ حل معادلهٔ درجه دوم */
{  float a, b, c;
   float delta, x1, x2, x;

   a = 2;
   b = 5.5;
   c = ۱.۷;
   printf("\fa=%f\n", a);                  /* چاپ ضرایب در این سطر و دو سطر بعدی */
   printf("b=%f\n", b);
   printf("c=%f\n", c);
   delta = b * b - ۴ * a * c;              /* محاسبهٔ دلتا */
   if (delta > ۰)                          /* محاسبه و چاپ ریشه‌های مختلف */
   {  x۱ = (-b + sqrt(b * b - ٤ * a * c)) / (۲ * a);
      x۲ = (-b - sqrt(b * b - ٤ * a * c)) / (۲ * a);
      printf("\nX۱=٪۶.۲f\tX۲=٪۶.۲f\n", x۱, x۲);
   }
   else
      if (delta == ۰)
      {  x = -b / (۲ * a);                 /* محاسبه و چاپ ریشهٔ مضاعف */
         printf("\nX۱=X۲=٪۶.۲f\n", x);
      }
      else                                 /* چاپ پیغام عدم وجود ریشهٔ حقیقی */
         printf("\nThe equation has no real root!\n");
   return (۰);
}
```

شکل ۴-۸: متن برنامهٔ ۴-۲، حل معادلهٔ درجه دوم با استفاده از دستور شرطی کامل.

- **مقایسهٔ این برنامه با برنامهٔ ۴-۱.**
- **توجه به تفاوتهای نحوهٔ استفاده از دو نوع دستور `if` و به‌ویژه `if` های تودرتو.**

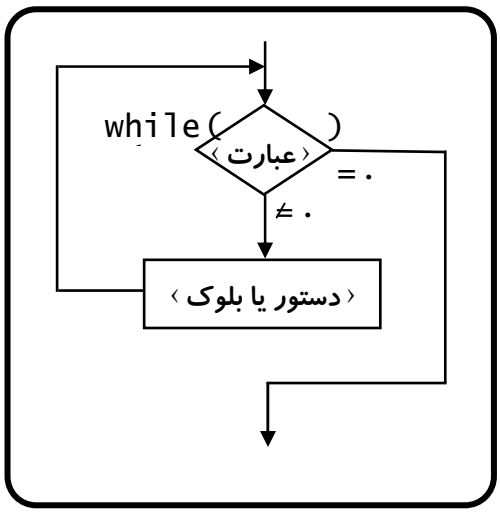
۴-۴ حلقهٔ تکرار while

- نوع اول حلقه‌های تکرار: تعداد دفعات تکرار مشخص نبوده، خاتمهٔ تکرار بر مبنای نتیجهٔ شرط خاص.
- یکی از حالتهای ساختـــار تکرار (بخش ۳-۴-۱ از فصل سوم سمت چپ شکل ۳-۵).

قالب کلی دستور while برای ساختن حلقه‌های تکرار نوع اول.

while (‹ عبارت ›) ‹ دستور یا بلوک ›

نحوهٔ اجرا



شکل ۴-۴: نحوهٔ اجرای دستور while.

- عبارت داخل پرانتز شرط حلقه.
- بعد از پرانتز یک دستور تنها یا تعدادی دستور در قالب یک بلوک (بدنهٔ حلقه).
- بروز حلقهٔ تکرار نامحدود در صورت عدم تغییر مقدار عبارت شرط حلقه در اثر تکرار بدنهٔ حلقه.
- عدم اجرای بدنهٔ حلقه حتی یک بار درصورت عدم برقراری شرط در شروع کار.
- اهمیت شرایط مرزی در حلقه های تکرار.

مثال: چاپ مقادیر یک تا ۱۰۰۰ همراه با جذرشان

```
K = ۱;
while(k <= ۱۰۰۰)
{  s = sqrt(k);
   printf("k=٪۴d\tsqrt(k)=٪۶.۳f\n", k, s);
   k = k + ۱;
```

}

مثال: برنامهٔ نمونهٔ ۴-۳

برنامه‌ای بنویسید که جدولی حاوی تبدیل درجه حرارت از سانتیگراد به فارنهایت و کلوین استخراج نموده، چاپ نماید. در این برنامه مقادیر اولیه و نهایی برای درجه حرارت سانتیگراد و همچنین قدرنسبت تغییر آن را در آغاز در داخل متغیرهای مناسب ذخیره نمایید. فرض کنید همیشه قدر نسبت مثبت است و برای این برنامه مقادیر اولیه و نهایی صفر تا صد را با قدرنسبت دو در نظر بگیرید.

```
#include <stdio.h>

main()                    /* برنامهٔ چاپ جدول تبدیل درجه حرارت از سانتیگراد به فارنهایت و کلوین */
{  int start, end, step;
   float cent, fahr, kel;

   start = ۰;                              /* تعیین مقادیر اولیه، نهایی و قدر نسبت */
   end = ۱۰۰;
   step = ۲;
   printf("\f   TEMPRATURE TRANSFER TABLE\n\n");
   printf("Celsius   Fahrenheit   Kelvin \n");
   printf("------------------------------\n");
   cent = start;
   while (cent <= end)              /* حلقهٔ تکرار استخراج و چاپ درجه حرارت‌ها */
   {  fahr = (۹.۰/۵.۰) * cent + ۳۲;
      kel = cent + ۲۷۳;
      printf("%۵.۰f        %۶.۲f        %۴.۰f\n", cent, fahr, kel);
      cent = cent + step;
   }
   return (۰);
}
```

شکل ۴-۹: متن برنامهٔ ۴-۳، استخراج و چاپ جدول تبدیل درجه حرارت با استفاده از دستور while.

- **چاپ یک جدول خالی در صورت غیرمنطقی بودن مقدار اولیه و نهایی یعنی متغیرهای start و end.**
- **توجه به نحوهٔ چاپ عنوان جدول، عناوین ستونها، تنظیم فاصله بین کلمات در عناوین و مقادیر.**
- **استفاده از تعداد مناسب خط فاصله برای رسم خط افقی زیر عناوین.**
- **استفاده از دقت صفر در مشخصهٔ تبدیل برای چاپ اعداد سانتیگراد و کلوین بدون رقم اعشاری.**

مثال: برنامهٔ نمونهٔ ۴-۵

برنامه‌ای بنویسید که اعداد اول از ۳ تا ۱۰۰۰ را استخراج و چاپ نماید. مراحل انجام این کار نخست در روندنمای ۲-۴ در بخش روندنماهای نمونهٔ فصل دوم (شکل ۲-۹) به صورت عادی و سپس در مبحث برنامه‌نویسی ساختیافته از بخش ۳-۴ فصل سوم در قالب شکل ۳-۶ به صورت ساختیافته نوشته شد. این برنامه تفاوتهای جزئی با دو روندنمای یاد شده دارد از جمله این که در روندنماها اعداد اول از ۳ تا n استخراج و چاپ می‌شود.

```
#include <stdio.h>

main()                                          /* برنامهٔ چاپ اعداد اول از ۳ تا ۱۰۰۰ */
{  int prm, dvd, n, rem;

   n = ۱۰۰۰;
   prm = ۳;
   while (prm <= n)                             /* حلقهٔ تکرار ساختن اعداد */
   {  dvd = ۲;
      rem = ۱;
      while (rem > ۰ && dvd < prm)              /* حلقهٔ تکرار بررسی اول بودن */
      {  rem = prm % dvd;
         dvd = dvd + ۱;
      }
      if (rem > ۰)
         printf("%d\n", prm);
      prm = prm + ۲;
   }
   return (۰);
}
```

شکل ۴-۱۱: متن برنامهٔ ۴-۵، استخراج و چاپ اعداد اول از ۳ تا ۱۰۰۰ با استفاده از دستور while.

- **دو حلقه تکرار تودرتو یکی برای ساختن اعداد فرد از ۳ تا ۱۰۰۰ و دیگری برای بررسی اول بودن آن.**
- **شرط مرکب با استفاده از عملگر && درحلقهٔ داخلی.**
- **دستور شرطی ساده بعد از حلقهٔ داخلی برای بررسی نحوه خروج از حلقه**
- **نیاز به مقدار اولیه یک برای متغیر rem.**

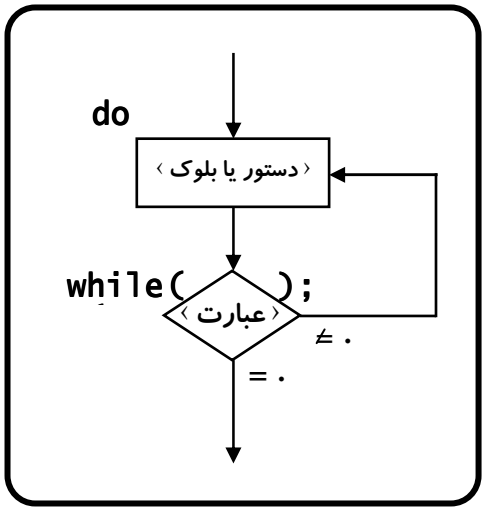
۵-۴ حلقهٔ تکرار do

- یکی دیگر از حالتهای ساختار تکرار (بخش ۳-۴-۱ فصل سوم در سمت راست شکل ۳-۵).
- تفاوت با حلقهٔ تکرار while: بررسی شرط خاتمهٔ حلقه بعد از هر بار اجرای بدنهٔ حلقه.
- اجرای حلقه حداقل یک بار.
- توجه به شرایط مرزی.
- مورد استفادهٔ کمتر از دستور while.
- لزوم قرار دادن علامت پایان دستور یعنی ; بعد از زوج پرانتز حاوی عبارت شرط.

قالب کلی دستور do برای ساختن حلقه‌های تکرار نوع دوم

do ‹ دستور یا بلوک › while (‹ عبارت ›);

نحوه اجرا



شکل ۴-۵: نحوهٔ اجرای دستور do.

مثال:

```
K = ۱;
do
{  s = sqrt(k);
   printf("k=٪۴d\tsqrt(k)=٪۶.۳f\n", k, s);
   k = k + ۱;
} while (k <= ۱۰۰);
```

مثال: برنامهٔ نمونهٔ ۴-۴

برنامهٔ چاپ جدول تبدیل درجه حرارت (برنامهٔ ۴-۳) را مجدداً بنویسید ولی این بار به جای دستور while در آن از دستور do برای ساختن حلقهٔ تکرار استفاده نمایید. ضمناً فرض کنید در این برنامه قدرنسبت می‌تواند مثبت یا منفی باشد که لازم است منطقی بودن مقادیر اولیه، نهایی و قدرنسبت کنترل گردد. در سایر موارد این برنامه با برنامهٔ قبلی تفاوتی ندارد.

```
#include <stdio.h>
main()                    /* برنامهٔ چاپ جدول تبدیل درجه حرارت از سانتیگراد به فارنهایت و کلوین */
{  int start, end, step;
   float cent, fahr, kel;

   start = ۰;                          /*  تعیین مقادیر اولیه، نهایی و قدرنسبت  */
   end = ۱۰۰;
   step = ۲;
   printf("\f   TEMPRATURE TRANSFER TABLE\n\n");
   printf("Celsius   Fahrenheit   Kelvin \n");
   printf("-----------------------------\n");
   cent = start;
                                       /*  بررسی غیرمنطقی بودن مقادیر اولیه، نهایی و قدرنسبت  */
   if (step == ۰ || step>۰ && start>end || step<۰ && start<end)
   {  printf("Incorrect start/end/step!\n");
      return ۰;
   }
   do                                  /*  حلقهٔ تکرار استخراج درجه حرارت‌ها  */
   {  fahr = (۹.۰/۵.۰) * cent + ۳۲;
      kel = cent + ۲۷۳;
      printf("٪۵.۰f        ٪۶.۲f        ٪۴.۰f\n", cent, fahr, kel);
      cent = cent + step;
   } while (step > ۰ && cent <= end || step < ۰ && cent >= end);
   return (۰);
}
```

شکل ۴-۱۰: متن برنامهٔ ۴-۴، استخراج و چاپ جدول تبدیل درجه حرارت با استفاده از دستور do.

- **شرط توقف حلقه در پایان بدنهٔ حلقه.**
- **شرط اضافی در شروع برنامه برای اطمینان از منطقی بودن مقادیر اولیه، نهایی و قدرنسبت.**
- **توجه به چگونگی ترکیب عملگرهای منطقی بدون نیاز به زوج پرانتز (با توجه به تقدمها).**
- **درنظر گرفتن محاسبه منطقی.**
- **تغییر شرط توقف حلقه به دلیل امکان وجود قدرنسبت مثبت یا منفی.**

مثال: برنامهٔ نمونهٔ ۴-۶

در برنامهٔ ۴-۵ محاسبه و چاپ اعداد اول از ۳ تا ۱۰۰۰ با استفاده از دستور while انجام گرفت، هدف از این برنامه تکرار عمل مزبور با استفاده از دستور do به جای دستور while می‌باشد.

```
#include <stdio.h>

main()                                          /* برنامهٔ چاپ اعداد اول از ۳ تا ۱۰۰۰ */
{  int prm, dvd, n, rem;

   n = ۱۰۰۰;
   prm = ۳;
   do                                           /* حلقهٔ تکرار ساختن اعداد */
   {  dvd = ۲;
      do                                        /* حلقهٔ تکرار بررسی اول بودن */
      {  rem = prm % dvd;
         dvd = dvd + ۱;
      } while (rem > ۰ && dvd < prm);
      if (rem > ۰)
         printf("%d\n", prm);
      prm = prm + ۲;
   } while (prm <= n);
   return (۰);
}
```

شکل ۴-۱۲: متن برنامهٔ ۴-۶، استخراج و چاپ اعداد اول از ۳ تا ۱۰۰۰ با استفاده از دستور do.

- **دو حلقه تکرار تودرتو از نوع do با شرط خاتمهٔ در پایان بعد از اجرای بدنهٔ حلقه.**
- **عدم نیاز به مقدار اولیه یک در متغیر rem قبل از شروع حلقه.**

۴-۶ خواندن از ورودی

- انجام عملیات ورودی و خروجی در زبان C با استفاده از توابع آماده.
- روشهای متنوع خواندن داده ها و با استفاده از منابع مختلف.
- ورودی استاندارد هر برنامه در حالت معمولی صفحهٔ کلید.

۴-۶-۱ خواندن با فرمت، روش متداول

استفاده از تابع scanf با قالب کلی احضار زیر.

scanf(‹رشتهٔ فرمت› , ‹لیست آدرسهای متغیرها›)

- خواندن کاراکترها از ورودی.
- تبدیل طبق مشخصه‌های تبدیل فرمت ارائه شده در رشته فرمت.
- ذخیره در متغیرهایی که آدرس آنها اعلام گردیده به ترتیب از چپ به راست.

شکل ۴-۶: نحوهٔ کار تابع scanf (خواندن با فرمت).

- نتیجه تابع تعداد فقره داده‌های خوانده شده در قالب عدد صحیح (مثبت، صفر یا ۱-).
- خواندن تا: انجام به طور کامل یا عدم امکان ادامهٔ خواندن یا اعلام خاتمهٔ داده‌های ورودی.
- ثابت نمادی EOF معادل حالت پایان فایل(۱-) برای حفظ استقلال از ماشین.
- کلیدهای Ctrl و z همزمان از طریق صفحهٔ کلید، برای اعلام پایان فایل.
- تشخیص پایان فایل به طور اتوماتیک در خواندن از فایل.
- امکان تشخیص اینکه عمل خواندن تا چه حد موفق بوده برمبنای مقدار نتیجه تابع.

مثال ساده: scanf(“%d%f”, &m, &x); (مانند یک عبارت)

ورودی: ۴۵۶ ۱۲۳ نتیجه: m حاوی عدد صحیح ۱۲۳ و x حاوی عدد اعشاری ۴۵۶.۰

- امکان ذکرتوضیحات و فاصله‌های سفید (فاصلهٔ خالی، \t، \n، \f و \r) در رشته فرمت.

- لزوم تطبیق سه عامل درگیر در خواندن با فرمت: داده‌های ورودی، مشخصه‌های تبدیل و نوع متغیرها.
- عملگر استخراج آدرس &، عملگر یکتایی با بالاترین تقدم، خطر فراموش کردن آن.

۴-۶-۲ رشتهٔ فرمت و مشخصه‌های تبدیل

- رشتهٔ فرمت مشخص کنندهٔ نوع داده‌ها، اندازهٔ داده‌ها و محل داده‌های مورد تخصیص به متغیرها.
- لزوم قرار دادن رشتهٔ فرمت به صورت ثابت رشته‌ای به طور کامل روی یک سطر.
- امکان تعریف رشتهٔ فرمت با حالت پویا و قابل تغییر در زمان اجرا.
- انجام کل عمل خواندن بر اساس رشتهٔ فرمت.
- خواندن کاراکترهای ورودی، تبدیل بر اساس مشخصه تبدیل، ذخیره نتیجهٔ در آدرس متغیر متناظر.
- خواندن کامل: انجام عمل خواندن تا جایی که رشتهٔ فرمت تمام شود.
- خواندن ناتمام: عدم تطبیق کاراکترهای ورودی با مشخصهٔ تبدیل فرمت، یا پایان فایل.
- عدم وجود تعداد کافی آدرس متغیر در لیست آدرس‌ها.
- وجود تعداد آدرسهای بیشتر از تعداد مشخصه‌های تبدیل فرمت.

مثال:

```
k = scanf("%d%f", &m, &x);
if (k == ۲)
   printf("Reading data was successful.\n");
else
   if(K == EOF)
      printf("End of file encountered in input!\n");
   else
      printf("Insufficient data supplied in input!\n");
```

- خواندن بخشی از یک سطر ورودی در یک احضار و خواندن بقیهٔ آن سطر ورودی در احضار بعدی.

```
int m, n, j;
float a, b;
j = scanf("%d%d", &m, &n);
printf("%d, %d, %d\n", j, m, n);
j = scanf("%f%f", &a, &b);
printf("%d, %f, %f", j, a, b);
```

داده‌های ورودی

۱۲ ۳.۴۵

۶.۷ ۸.۹

نتیجه خروجی

۲, ۱۲, ۳

۲, ۰.۴۵۰۰۰۰, ۶.۷۰۰۰۰۰

موارد قابل ارائه در رشتهٔ فرمت

- **فاصله‌های سفید: تطبیق با ورودی متناظر در صورت وجود وگرنه صرف نظر از آنها.**

مثال: با فرض x اعشاری و m صحیح باشد.

```
scanf("%d \n %f", &m, &x);
scanf("%d \t\r\n  \f  %f", &m, &x);
scanf("%d\n%f\n", &m, &x);
scanf("%d%f\n", &m, &x);
```

- **دو احضار اول: بین دو عدد ورودی حداقل یک فاصله برای جدا سازی یا هر تعداد و ترکیبی از فاصله‌های سفید.**
- **یک راه دیگر وجود علامت برای مقدار دوم.**
- **اعلام پایان داده‌های ورودی با فشار دادن کلید Enter.**
- **دو احضار پایینی: لزوم درج حداقل یک کاراکتر غیرفاصلهٔ سفید بعد دو عدد و بعد کلید Enter.**

کاراکترمعمولی

- **جداکنندهٔ بین فقره‌های داده در ورودی به جای فاصله‌های سفید.**
- **نیاز به وجود عین آنها در محل مناسب ازداده های ورودی جهت تطبیق.**

.

```
scanf("%d,%f", &m, &x);
scanf("m=%d x= %f", &m, &x);
scanf("%d next%flast", &m, &x);
scanf("%d next%f last", &m, &x);
```

- **احضار اول: بین دو عدد ورودی یک علامت , (کاما)، امکان وجود فاصله‌های سفید بعد از این علامت.**
- **عدم وجود هیچ کاراکتری قبل از آن یعنی بین آخرین رقم عدد و علامت کاما.**
- **احضار دوم: شروع ورودی الزاماً با m=، بعد یک عدد صحیح، وجود فاصله قبل و بعد از عدد بلامانع.**
- **بعد از عدد اول و قبل از عدد دوم لزوم وجود دو کاراکتر x=، و نهایتاً عدد دوم باید یک عدد اعشـاری.**
- **صورت درست: x= ۴۵.۶۷ m= ۱۲۳. خروجی تابع مقدار ۲**
- **صورت غلط: x= ۴۵.۶۷ m =۱۲۳ باشد خروج تابع مقدار صفر.**
- **احضار سوم: عدم نیاز به کلمهٔ last در ورودی بعد از عدد دوم، خاتمه ورودی با کلید Enter.**

- **احضار چهارم: بعد از عدد دوم و قبل از کلید Enter حداقل یک فاصلهٔ سفید و سپس یک کاراکتر غیرفاصلهٔ سفید (به جای کلمهٔ last)**

مشخصهٔ تبدیل

- **شروع با علامت %.**
- **ادامه با هر کدام از سه مورد زیر، به ترتیبِ ذکر شده.**

علامت نادیده گرفتن ورودی: علامت * (اختیاری).

طول میدان ماکزیمم (اختیاری)

- **عدد صحیح بدون علامت، حداکثر ستونهای مورد خواندن متناظر با مشخصهٔ تبدیل.**
- **عدم استفاده از آن برای کد c، خطرناک در صورت استفاده.**
- **ملاک نبودن جداکننده برای مشخصهٔ تبدیل فرمت c.**
- **امکان کوتاه‌تر بودن طول فقره داده در مورد مشخصه‌های تبدیل d و f با حضور جداکننده**
- **خواندن کاراکترها به تعداد طول میدان ماکزیمم یا برخورد با جداکننده یا کاراکتر غیرقابل قبول.**
- **جداسازی بر مبنای طول میدان بدون حضور جداکننده.**

علامت تبدیل

- **d برای تبدیل به عدد صحیح.**
- **f برای تبدیل به عدد اعشاری.**
- **c برای تبدیل به معادل عددی کاراکتر در مثلا کد ASCII.**
- **الزامی بودن علامت تبدیل در مشخصهٔ تبدیل فرمت.**
- **توقف خواندن روی کاراکتر غلط در صورت عدم تطبیق کاراکترهای ورودی با علامت تبدیل.**
- **مبنای عمل خواندن مشخصه‌های تبدیل در رشتهٔ فرمت.**
- **لزوم وجود یک آدرس در لیست آدرسها به ازای هر مشخصهٔ تبدیل (علامت % و ملحقات آن).**
- **غیرقابل پیش‌بینی بودن نتیجهٔ عمل خواندن اگر تعداد آدرسها کمتر از تعداد مشخصه‌های تبدیل باشد.**
- **غیرقابل پیش‌بینی بودن نتیجهٔ عدم وجود مشخصهٔ تبدیل درست بعد از علامت %.**
- **صرف‌نظر از آدرسهای اضافی اگر تعداد آدرسهای ارائه شده بیشتر از تعداد مشخصه‌های تبدیل باشد.**

جدول ۴-۵ الف: نمونه‌هایی از فرمت خواندن با فرض اینکه m متغیر صحیح، x و y متغیرهای اعشاری و p و q متغیرهای صحیح یک بایتی باشند. علامت " در دو طرف داده‌های ورودی برای تعیین دقیق آنها است و بخشی از آن داده‌ها نیست.

آرگومان‌های تابع scanf	داده‌های ورودی	نتیجهٔ خواندن	توضیح
(“٪۲d٪۴f”,&m,&x)	“۱۲۳۴۵۶”	m:۱۲ x:۳۴۵۶.۰	از طول میدان استفاده شده است.
(“٪۲d\n٪۴f”,&m,&x)	“۱۲۳۴۵۶”	m:۱۲ x:۳۴۵۶.۰	چون جدا کننده وجود ندارد، طول میدان ملاک است و برای \n تطبیقی انجام نمی‌شود.
(“٪۲d\r٪۴f”,&m,&x)	“۱ ۲۳۴۵۶”	m:۱ x:۲۳۴۵.۰	چون جدا کنندهٔ فاصلهٔ خالی وجود دارد، برای m طول میدان ملاک نبوده، \r با فاصلهٔ خالی تطبیق می‌یابد.
(“٪۲d\t٪۴f”,&m,&x)	“۱ ۲۳.۴۵۶”	m:۱ x:۲۳.۴	مثل مورد بالا ولی \t با علامت New Line تطبیق یافته است (ورودی در دو سطر).
(“٪۲d\f٪۴f”,&m,&x)	“۱۰ ۲۳.۴۵۶”	m:۱ x:?	به دلیل وجودکاراکتر غلط (نقطه) بعد از ۱، خواندن m روی آن متوقف شده، New Line به خواندن خاتمه می‌دهد (ورودی در دو سطر).
(“٪۲d\n٪۴f”,&m,&x)	“-۱ -۲q۴۵۶”	m:-۱ x:-۲.۰	به دلیل وجودکاراکتر غیرقابل قبول (q) بعد از ۲، برای x تا قبل از q خوانده می‌شود.
(“m is٪۲dx=٪۴f”, &m,&x)	“m is۱۲x=۱.۵”	m:۱۲, x:۱.۵	تطبیق بین کاراکترها در فرمت و در ورودی وجود دارد.
(“m is٪۲dx=٪۴f”, &m,&x)	“mis۱۲x= ۱.۵”	m:۱۲ x:۱.۵	فاصلهٔ بین m و is در فرمت نیازی به تطبیق ندارد و فاصلهٔ قبل از عددها در ورودی هم اشکالی ندارد.
(“m is٪۲dx=٪۴f”, &m,&x)	“mis۱۲ x= ۱.۵”	m:۱۲ x:?	بعد از۱۲ در ورودی باید x باشد که فاصلهٔ خالی است، فقط m خوانده می‌شود.
(“m is٪۲dx=٪۴f”, &m,&x)	“mi s۱۲ x= ۱.۵”	m:? x:?	کلمهٔ is با ورودی تطبیق نمی‌کند و خواندن متوقف می‌شود.

مثال: برنامهٔ نمونهٔ ۴-۷

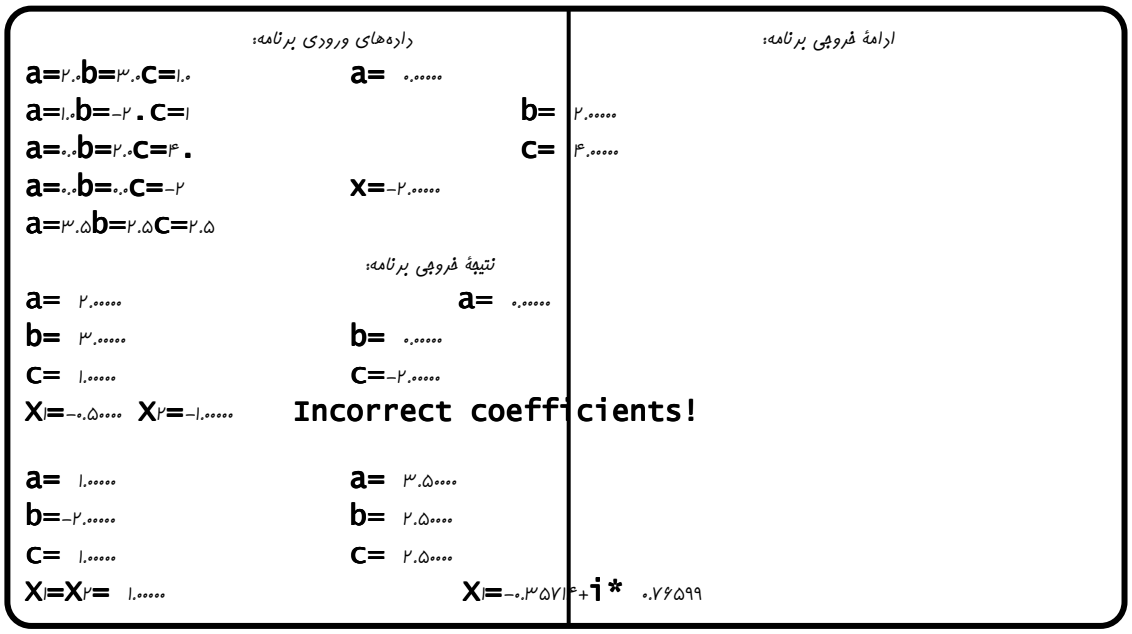
برنامه‌ای بنویسید که در آن ضرایب تعدادی معادلهٔ درجه دوم از ورودی خوانده شود (هر ضریب به صورت یک عدد اعشاری در حداکثر سه ستون درج شده که اسم آن همراه با یک علامت مساوی در سمت چپش ذکر شده است). پس از چاپ ضرایب هر معادله (هر ضریب روی یک سطر با توضیح مناسب و یک سطر خالی قبل از چاپ ضرایب)، معادله حل شده، با توجه به علامت دلتا نتیجه در قالب ریشه‌های مختلف، ریشه‌های مضاعف و ریشه‌های مختلط روی یک سطر مجزا با توضیحات مناسب چاپ گردد. در این برنامه لازم است غیرمنطقی بودن ضرایب و درجه اول بودن هر معادله نیز بررسی گردد. شروع چاپ نتایج برنامه از اول صفحه باشد.

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

main()                                          /* برنامهٔ حل تعدادی معادلهٔ درجه دوم */
{ float a, b, c;
  float delta, x۱, x۲, x;

  printf("\f");
                                  /* حلقهٔ تکرار اصلی خواندن ضرایب همراه با توضیحات و انجام عملیات خواسته شده */
  while (scanf("\na=%۳fb=%۳fc=%۳f", &a, &b, &c) != EOF)
  { printf("\na=%۸.۵f\nb=%۸.۵f\nc=%۸.۵f\n", a, b, c);/* چاپ ضرایب */
    if (a == ۰)
      if (b == ۰)
        printf("Incorrect coefficients!\n");
      else
      { x = -c / b;  printf("x=%۸.۵f\n", x); }
    else
    { delta = b * b – ۴ * a * c;          /* محاسبهٔ دلتا */
      if (delta > ۰)                          /* محاسبه و چاپ ریشه‌های مختلف */
      { x۱ = (-b + sqrt(delta)) / (۲ * a);
        x۲ = (-b - sqrt(delta)) / (۲ * a);
        printf("X۱=%۸.۵f\tX۲=%۸.۵f\n", x۱, x۲);
      }
      else
        if (delta == ۰)
        { x = -b / (۲ * a);               /* محاسبه و چاپ ریشه‌های مضاعف */
          printf("X۱=X۲=%۸.۵f\n", x);
        }
        else                                    /* محاسبه و چاپ ریشه‌های مختلط */
        { delta = -delta;  x = -b / (۲ * a);
          printf("X۱=%۸.۵f+i*%۸.۵f\n",x, sqrt(delta) / (۲ * a));
          printf("X۲=%۸.۵f-i*%۸.۵f\n",x, sqrt(delta) / (۲ * a));
        }
  }
```

شکل ۴-۱۳ الف: متن برنامهٔ ۴-۷، حل تعدادی معادلهٔ درجه دوم و چاپ جوابها (حتی جواب مختلط).



شکل ۴-۱۳ ب: نمونهٔ اجرای برنامهٔ ۴-۷، حل کامل معادلهٔ درجه دوم.

- **توجه به مقایسهٔ نتیجهٔ تابع خواندن با ثابت نمادی EOF برای بررسی خاتمهٔ داده‌ها (شرط while).**
- **خاتمهٔ داده‌ها از صفحهٔ کلید دو کلید Ctrl و z را با هم، سپس کلید Enter.**
- **استفاده از دستورهای شرطی کامل و تودرتو در داخل حلقه برای بررسی حالات مختلف ضرایب.**

مثال برنامهٔ نمونه ۴-۸: برنامه‌ای بنویسید که تعدادی عدد صحیح و مثبت حداکثر پنج رقمی را از ستونهای ۱۱ تا ۱۵ هر سطر بخواند و برای هر مقدار درصورتی که آن مقدار می‌تواند یک عدد در مبنای هشت باشد آن را به مبنای ده برده، همراه با عدد اولیه چاپ نماید، در غیر این صورت عدد را همراه با پیغام مناسب چاپ کند. خروجی برنامه باید دارای عنوان‌بندی مناسب باشد.

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

main()                                  /* برنامهٔ تبدیل عدد از مبنای هشت به ده */
{  int m, n, i, ir, newn;

   printf("\fnumber in Base ۸   Number in Base ۱۰\n");
   printf("-----------------------------------\n");
   while (scanf("%۵d", &m) != EOF)   /* حلقهٔ تکرار اصلی خواندن عددها و انجام عملیات */
   {  n = m;
      i = ۰;
      newn = ۰;
      while (m > ۰ && (ir = m % ۱۰) <= ۷)     /* حلقهٔ تکرار تبدیل مبنا */
      {  newn = newn + ir * pow(۸, i);
         m = m / ۱۰;
         i = i + ۱;
      }
      if (m == ۰)
         printf("%۱۰d%۱۸d\n", n, newn);
      else
         printf("%۱۰d           not in base ۸\n", n);
   }
   return (۰);
}
```

داده‌های ورودی برنامه:

```
۱۲
۲۰۰۰
۹۸۱
۱۰۰۰
۱۲۳۴۵
۱
```

نتایج خروجی برنامه:

```
number in Base ۸   Number in Base ۱۰
-----------------------------------
        ۱۲                    ۱۰
```

شکل ۴-۱۴: متن برنامهٔ ۴-۸، تبدیل عدد از مبنای هشت به ده همراه با نمونهٔ اجرای برنامه.

- **متن برنامهٔ منطبق با روندنمای ۲-۸ در بخش روندنماهای نمونهٔ فصل دوم، شکل ۲-۱۳.**

- استفاده از دو حلقهٔ تکرار تودرتو، یکی برای خواندن داده‌ها و دیگری برای تبدیل مبنای هر عدد ضمن بررسی درستی ارقام عدد.

مثال برنامهٔ نمونه ۴-۹: برنامه‌ای بنویسید که در یک حلقهٔ تکرار هر بار سه عدد صحیح و مثبت حداکثر چهار رقمی را از ورودی بخواند و بزرگ‌ترین مقسوم‌علیه مشترک هر سه عدد را محاسبه نموده، همراه با خود آن سه عدد با توضیح مناسب در خروجی چاپ نماید. توجه داشته باشید که هر گروه سه‌تایی اعداد در ورودی هیچ‌گونه ترتیبی از نظر بزرگی و کوچکی ندارند.

```
#include <stdio.h>

main()                                  /* برنامهٔ محاسبهٔ بزرگ‌ترین مقسوم‌علیه مشترک سه عدد */
{  int m, n, l, gcd, ir;

                                        /* حلقهٔ تکرار خواندن عددها و انجام عملیات */
   while (scanf("%d%d%d", &m, &n, &l) != EOF)
   {  printf("GCD(%d,%d,%d)= ", m, n, l);/* چاپ عددهای خوانده شده */
      while (ir = m % n)      /* حلقهٔ محاسبهٔ ب م م دو عدد اول */
      {  m = n;
         n = ir;
      }
      while (ir = n % l)          /* حلقهٔ محاسبهٔ ب م م عدد سوم با ب م م دو عدد اول */
      {  n = l;
         l = ir;
      }
      gcd = l;
      printf("%d\n", gcd);
   }
   return (0);
}
```

داده‌های ورودی برنامه:

```
۱۲  ۴۵  ۳۴
۳۶  ۶۰  ۸۴
۱۱  ۷   ۱۹
۱۲۰  ۴۰۰  ۵۰۰
۱۰۰۰  ۱۵۰۰  ۲۲۰۰
```

نتایج خروجی برنامه:

```
GCD(  ۱۲,  ۴۵,  ۳۴)=   ۱
GCD(  ۳۶,  ۶۰,  ۸۴)=  ۱۲
GCD(  ۱۱,  ۷,  ۱۹)=   ۱
GCD(  ۱۲۰,  ۴۰۰,  ۵۰۰)=  ۲۰
```

شکل ۴-۱۵: متن برنامهٔ ۴-۹، محاسبهٔ ب م م سه عدد همراه با نمونهٔ اجرای برنامه.

- منطبق با روندنمای ۲-۵ در بخش روندنماهای نمونهٔ فصل دوم (شکل ۲-۱۰).
- یک حلقهٔ تکرار برای خواندن دادهها، دو حلقهٔ داخلی، اولی برای محاسبهٔ ب م م دو عدد اول و حلقهٔ دوم برای محاسبهٔ ب م م این مقدار با عدد سوم.
- توجه به نحوهٔ استفاده از عبارتهای محاسبهای به عنوان شرط در حلقههای `while`.

۸-۴ اشتباهات متداول برنامهنویسی

- در زبان C عملگر = برای تخصیص و عملگر == برای بررسی تساوی تعریف شده است، پس باید دقت نمود که در عبارتهایی که به عنوان شرط در دستورهای شرطی و حلقههای تکرار استفاده میشود عملگر درست را در جای خود استفاده نمود. استفاده از عملگر = به جای عملگر == غلط نحوی محسوب نشده و معنای متفاوت دارد.
- استفاده از عملگر == برای مقایسهٔ دو مقدار اعشاری توصیه نمیشود زیرا این مقادیر به دلیل تبدیلات بین مبناهای ده و دو و ذخیره به صورت توان علمی معمولاً با تقریب همراه هستند.
- تعدادی از عملگرهای زبان C از دو کاراکتر تشکیل شدهاند (==، =!، =>، =<، &&، ||)، در بین این دو کاراکتر نباید فاصلهٔ خالی قرار داده شود وگرنه غلط نحوی محسوب میشود.
- در هر دو دستور `do` و `while`، در صورتی حلقه تکرار میشود که حاصل عبارت شرط غیرصفر (درست) باشد و تنها تفاوت آنها محل انجام بررسی عبارت شرط است که در `while` قبل از انجام بدنهٔ حلقه و در `do` بعد از انجام بدنهٔ حلقه میباشد. به این ترتیب بدنهٔ هر حلقهٔ `do` حداقل یک بار اجرا میشود ولی بدنهٔ `while` ممکن است اصلاً اجرا نشود. به این ترتیب در کاربرد این دو دستور دقت کافی لازم است.
- افرادی که با زبانهای دیگر برنامهنویسی کردهاند باید دقت نمایند که در زبان C در دستور `while` نیازی به نوشتن کلمهٔ `do` بعد از عبارت شرط نیست (در بعضی از زبانها این کار لازم است) و در صورت نوشتن آن دستورهای `while` و `do` مخلوط میشوند که غلط نحوی است. علاوه بر این بعد از عبارت شرط در دستور `if` هم نیازی به کلمهٔ `then` نمیباشد و در صورتی که مثل خیلی از زبانهای دیگر این کلمه گذاشته شود دستور از نظر نحوی غلط خواهد بود.
- قرار دادن پرانتز در دو طرف عبارت شرط در دستورهای `if`، `while` و `do` اجباری است، نبودن آن باعث غلط شدن دستور از نظر نحوی میگردد.

۴-۹ پرسشها

توصیه اکید در مورد انجام پرسشهای ۴-۱۱ و ۴-۱۲

تکلیف چهارم: مهلت دو هفته

انجام پرسش۴-۱۰

وارد کردن در محیط زبان C و غلط گیری

اجرا با داده های واقعی و اطمینان از درستی آن

تحویل فایل برنامه به زبان C و فایل زبان ماشین قابل اجرا

***۴-۱۰- شرکت تعاونی دانشجویی در نظر دارد امور حسابداری خود را به شما به عنوان برنامه‌نویس واگذار نماید. برای این‌کار در پایان هر ماه داده‌هایی شامل اجناس وارده و صادرهٔ خود آماده می‌کند. سطر اول داده‌ها حاوی تعداد اجناس (عدد دو رقمی) و تاریخ روز (عدد شش رقمی با قالب** yymmdd**) و سپس برای هر قلم جنس یک سطر داده به صورتی که در بالای شکل ۴-۱۷ آمده است، وجود دارد و مقادیر درج شده روی آن بدون فاصله می‌باشند. شرکت تعاونی از شما می‌خواهد برنامه‌ای بنویسید که این داده‌ها را بخواند و گزارشی از وضعیت فروش ماهانهٔ شرکت به صورت ارائه شده در پایین شکل ۴-۱۷ چاپ نماید.**

سود هر جنس با توجه به کد آن محاسبه می‌گردد که برای کدهای A **و** B **و** C **مقدار سود ده درصد، برای کدهای** D **و** E **و** F **این مقدار پانزده درصد و برای سایر کدها بیست درصد می‌باشد. در هر صفحه از گزارش داده‌های مربوط به پانزده قلم جنس به صورت دو خط در میان چاپ شده، در پایان هر صفحه جمع دو ستون آخر مربوط به اقلام همان صفحه زیر ستونهای مربوطه درج شود. در هنگام چاپ تاریخ در بالای هر صفحه، عدد سال که به شکل دو رقمی از ورودی خوانده شده، باید به شکل چهار رقمی چاپ گردد. در پایان لیست جمع دو ستون آخر مربوط به کل لیست در زیر ستونهای خود و داده‌های مربوط به اجناس دارای بیشترین و کمترین فروش بعد از آنها چاپ گردد. در نوشتن این برنامه کلیهٔ عملیات به صورت صحیح انجام می‌شود و فرض کنید کلیهٔ مقادیر محاسبه شده در اثر عملیات خواسته شده از ۳۲۷۶۷ کمتر است.**

قالب کلی داده‌های ورودی

کد جنس	قیمت واحد خرید	موجودی آخر ماه	تعداد وارده در طول ماه	موجودی اول ماه	شمارهٔ جنس
یک حرف	چهار رقم	سه رقم	سه رقم	سه رقم	پنج رقم

قالب کلی گزارش خروجی

Monthly Sales Report for Student's Cooperative Shop

Date: yyyy/mm/dd تاریخ با قالب **Page:** شمارهٔ صفحه

Item No.	Amount in Stock	Amount Received	Amount Sold	Monthly Balance	Unit Price	Total Sold	Total Profit
شمارهٔ جنس	موجودی اول ماه	تعداد وارده در طول ماه	تعداد فروش در طول ماه	ماندهٔ موجودی آخر ماه	قیمت واحد فروش	کل فروش ماهانهٔ این جنس	مقدار سود این جنس در ماه
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Totals: جمع کل فروش ماهانه جمع کل سود ماهانه

Most Sold Item No & Amount: شماره و مبلغ جنسی که بیشترین فروش را از نظر مبلغ داشته

Least Sold Item No & Amount: شماره و مبلغ جنسی که کمترین فروش را از نظر مبلغ داشته

شکل ۴-۱۷: قالب کلی داده‌های ورودی و گزارش خروجی مربوط به پرسش ۴-۱۰.

***۴-۱۱- برنامهٔ ارائه شده در شکل ۴-۱۸ را دنبال کرده، نتیجهٔ اجرا و حاصل چاپ آن را مشخص نمایید. سپس برنامه یاد شده را روی کامپیوتر اجرا کنید و پاسخ خود را با حاصل اجرا توسط کامپیوتر مقایسه نموده، در صورت وجود تفاوت بین آنها، اشتباه یا اشتباهات خود را تعیین نمایید.**

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

main()
{  int m, n, j, k, n1;
   float a, b, c;

   scanf("%3f%2d%*2d%3f%3f%2d", &a, &k, &b, &c, &m);
   m = sqrt(a) + b * 1 / 2 - c;
   while (a * a - a * (b - 1.1) + sqrt(pow(a, 2) + m * 3) > m - 1)
   {  c = c * k - 4;
      printf("%4.1f%5.1f%3.1f%3d\n", a, b, c, m);
      scanf("%f%f%*d", &a, &b);
      k = k - 2;
   }
   scanf("%3f%1d%*2f%3f", &a, &n, &b);
   if (a < b)
   {  j = 1;   n1 = n;
      do
      {  n = n * 2;
         printf("%2d%4.1f%4.1f%2f\n", n, a, b, pow(k, 2) * 2);
         j = j + n1;
      } while (j <= -k);
   }
   else {  n = n * 100;   printf("%d\n", n);
   }
   printf("\f%4d%3d\n%4f\nEnd of Program\n", j, n, a);
   return (0);
}
```

داده‌های ورودی برنامه:

```
  4  12455.6
1.5  2  1  5.1  5
```

شکل ۴-۱۸: متن برنامهٔ مربوط به پرسش ۴-۱۱.

***۴-۱۲- در شکل ۴-۱۹برنامه‌ای به همراه داده‌های ورودی آن ارائه شده است. این برنامه را دنبال کرده، نتیجهٔ اجرا و حاصل چاپ آن را مشخص نمایید. سپس برای اطمینان از درستی پاسخهای خود، برنامه یاد شده را روی کامپیوتر اجرا کنید و پاسخ خود را با نتیجهٔ اجرا توسط کامپیوتر مقایسه نموده، در صورت وجود تفاوت، در مورد اشتباه یا اشتباهات خود تحقیق نمایید.**

```
#include <stdio.h>

main()
{  int m, n;
   float x, y;
   char c;

   m = ۱۰; n = -۱۵;
   x= ۱۵.۱; y = ۱.۷۵;
   c = 'x';

   y = m =
      (m*۱۰۰-n) / ۱۰۰ + (m*۱۰۰-n) % ۱۰۰ * ۱۰۰+۰.۹۹;
   printf("٪۵f,%.۱f\n٪۷.۵d<%d>",y, x + ۸۴.۸۵, m, -n);
   scanf("%f%*۵d٪۴f%d٪۴d",&x, &y, &n, &m);
   if (m = ۰)
      x = x / ۱۰;
   else if (n <= ۰)
   {  n = -۱۰۰۰ + (m = ۵);
      m = m == m;
   }
   else   x = x * ۱۰;
   printf("٪۲d\f٪۶.۵d\t٪۵.۰f", n, m, x + y - ۰.۲);
   while (m + n <= m && x < y)
   {  x = y = m - n;
      y =۱۰۰۰ - y;
   }
   scanf("٪۶f٪۴f%c",&y, &x, &c);
   printf("٪۵.۱f\tc%c\nn",y - x, c - ۲);
   return ۰;
}


-۲.۳  -۹۹.۹۹  -۱۱۵+۱۲.۳۰۳      داده‌های ورودی:
۲۱۲DIGE۰XA۱.۲۱e+۲
```

شکل ۴-۱۹: متن برنامهٔ مربوط به پرسش ۴-۱۲.