مبانی کامپیوتر و برنامه‌نویسی به زبان C

فصل ششم: تکمیل ساختارهای انتخاب و تکرار

۶-۱ مقدمه

- پایین‌ترین سطح کنترل اجرای عملیات: در خلال یک دستور از طریق قواعد ترتیب و تقدم
- دومین سطح کنترل: ترتیب اجرای دستورها در یک تابع از طریق ساختارهای ترتیب، انتخاب و تکرار.
- بالاترین سطح کنترل: ترتیب اجرای توابع از طریق احضار و برگشت بین توابع.
- مطالب این فصل: سایر ساختارهای انتخاب و تکرار و نحوهٔ استفاده از آنها.
- انواع حلقه‌های تکرار: نوع اول حلقه‌های تکراری که خاتمه آنها در اثر برقراری شرط خاص است.
- نوع دوم: حلقه‌های تکرار با شمارنده،خاتمهٔ حلقه بر اساس تعداد دفعات تکرار.
- نوع سوم: تلفیق دو نوع قبلی.

۶-۲ حلقهٔ تکرار با شمارنده

- عدم وجود امکاناتی برای تعیین تعداد دفعات تکرار در دستورهای `while` و `do`.
- لزوم گنجاندن دستورهای اضافی درصورت نیاز به شمارش تعداد دفعات تکرار حلقه.
- وجود دستور خاص برای ایجاد حلقه‌های تکرار با شمارنده تقریباً در همهٔ زبانهای برنامه‌نویسی.
- دستور `for` در زبان C برای این کار با امکانات خیلی بیشتر.

۶-۲-۱ دستور for

چند نمونه ساده

```
for (j = ۰; j < ۱۰; ++j) printf(“%d”, j);


for (j = ۰, k = ۱۰; j < ۱۰; ++j, --k)
   printf(“%d, %d”, j, k);


for (s = ۱, p = ۱, j = ۱; j <= n; j++, s += j, p *= j)
{  printf(“j=%d, \tsum=%ld\n”, j, s);
   printf(“product=%ld\n”, p);
}
```

- ساختار تکرار پیچیده‌تر از دستورهای while و do.
- دارای امکاناتی برای تعیین و مقداردهی اولیهٔ شمارنده، تست مقدار شمارنده، تغییر شمارنده.

قالب کلی دستور for

for (‹ عبارت آغازین › ; ‹ عبارت شرط خاتمه › ; ‹ عبارت پایانی ›) ‹ دستور یا بلوک ›

الگوریتم اجرا

۱- ‹ عبارت آغازین › محاسبه می‌گردد.

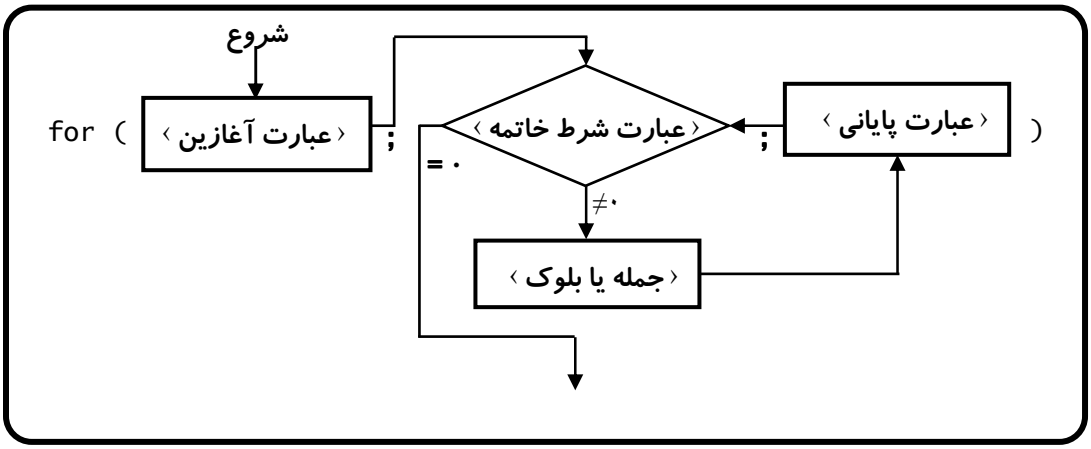
۲- ‹ عبارت شرط خاتمه › محاسبه می‌شود.

۳- اگر مقدار حاصل از محاسبهٔ ‹ عبارت شرط خاتمه › صفر باشد حلقه ترک می‌شود یعنی تکرار متوقف شده، اجرا از دستور بعد از for ادامه می‌یابد.

۴- در غیر این صورت (نتیجهٔ محاسبهٔ ‹ عبارت شرط خاتمه › غیرصفر باشد) ‹ دستور یا بلوک › بعد از پرانتز که به آن بدنهٔ حلقه گویند اجرا می‌شود.

۵- ‹ عبارت پایانی › محاسبه می‌گردد.

۶- عملیات از مرحلهٔ ۲ تکرار می‌شود.



شکل ۶-۱: نحوهٔ اجرای دستور for.

عبارات آغازین و پایانی معمولاً شـامل یک یا چند عبارت (عملگر کاما) ازجمله احضار تابع.

- **عبارت شرط خاتمه معمولا یک مقایسه (در حالت کلی می‌تواند هر عبارتی باشد).**
- **توجه به شرایط مرزی: عدم ورود به حلقه در صورت صفر بودن عبارت شرط از همان آغاز کار.**
- **لزوم دقت روی حدود اجرای حلقه.**

مثال:

```
for (j = ۰; j < ۱۰; ++j) printf(“%d”, j);
for (j = ۱; j <= ۱۰; ++j) printf(“%d”, j);
```

- **لزوم توجه دقیق به آثار جانبی عبارات سه گانه.**
- **نیاز به صفر شدن مقدار عبارت شرط پس از تعداد دفعات محدودی تکرار حلقه.**
- **آزاد بودن شمارنده و عبارات در طول اجرای بدنهٔ حلقه.**
- **اختیاری بودن سه عبارت آغازین، شرط خاتمه و پایانیِ داخل پرانتز.**

قالب کلی دستور for با استفاده از دستور while

‹ عبارت آغازین › ;

while (‹ عبارت شرط خاتمه ›)

{ ‹ دستور یا بلوک ›

‹ عبارت پایانی ›

}

مثال به همراه معادل با دستور while:

```
for (s = ۱, p = ۱, j = ۱; j <= n; j++, s += j, p *= j)
{  printf("j=%d, \tsum=%ld\n", j, s);
   printf("product=%ld\n", p);
}
```

```
s = ۱;
p = ۱;
j = ۱;
while (j <= n)
{  printf("j=%d, \tsum=%ld\n", j, s);
   printf("product=%ld\n", p);
   j++;
   s += j;
   p *= j;
}
```

- **توجه به عملگر کاما و عملوندهای آن.**
- **توجه به وابستگی بین عبارتهای عملگر کاما (اجزای عبارت پایانی).**
- **لزوم تاثیر نتیجهٔ محاسبهٔ عبارتها در شرط حلقه.**

شکلهای دیگر معادل مثال قبل

شکل اول:

```
s = ۱;
p = ۱;
for (j = ۱; j <= n;)
{  printf("j=%d, \tsum=%ld\n", j, s);
   printf("product=%ld\n", p);
   s += ++j;
   p *= j;
}
```

شکل دوم:

```
s = ۱;
p = ۱;
j = ۱;
for (; j <= n;)
{  printf("j=%d, \tsum=%ld\n", j, s);
   printf("product=% ld\n", p);
```

```
    s += ++j, p *= j;
}
```

شکل سوم:

```
s = ۱, p = ۱, j = ۱;
for (;;)
{  printf("j=%d, \tsum=%ld\n", j, s);
   printf("product=%ld\n", p);
   s += ++j;
   p *= j;
   if (j > n) return (۰);
}
```

دستور for با بدنهٔ خالی

```
for (nd = ۰, sd = ۰; m > ۰; nd++, sd += m % ۱۰, m /= ۱۰)
   ;

for(s=۱,p=۱,j=۱;j<=n; printf("j=%d, \tsum=%ld\n", j, s),
                      printf("product=%ld\n", p),j++,s+=j,p*=j)
   ;
```

- تذکر: پسندیده نبودن دستورهای طولانی به شکل اخیر در برنامه‌نویسی صحیح
- اجتناب از قرار دادن دستورهایی که ارتباطی به خود دستور for ندارند در داخل زوج پرانتز آن.
- نتیجه: خوانا و قابل فهم بودن برنامه‌ها از دید مستندسازی.

۶-۲-۲ دستورهای break و continue

- نیاز به ترک حلقه (دستورهای for، do و while) تحت شرایطی، قبل خاتمه طبیعی حلقه.

قالب کلی

```
break;
```

- نیاز به قطع اجرای یک دور حلقه (دستورهای for، do و while) و ادامه اجرا با دور بعدی.

قالب کلی

```
continue;
```

- دستور while: انتقال اجرا به اول بدنه حلقه.

- **دستور do: انتقال اجرا به آخر حلقه و بررسی شرط مربوط به ادامهٔ حلقهٔ تکرار.**
- **دستور for: محاسبه عبارت پایانی و بعد بررسی عبارت شرط، سپس تصمیم‌گیری برای ادامه یا خاتمهٔ حلقه.**

مثال: برنامهٔ نمونهٔ ۶-۱

با استفاده از حلقهٔ تکرار for برنامه‌ای بنویسید که مقدار n را از ورودی بخواند و اعداد اول از ۳ تا n را محاسبه کرده، چاپ نماید.

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

main()
{  long prm, dvd, n;

   scanf ("%ld", &n);
   for (prm = ۳; prm <= n; prm += ۲)                    /* حلقهٔ تکرار ساختن اعداد */
   {  for(dvd = ۲; dvd < (long)sqrt(prm) + ۱; dvd++)
         if (!(prm % dvd))                              /* حلقهٔ تکرار بررسی اول بودن */
            break;
      if (dvd == (long)sqrt(prm) + ۱)
         printf("%ld\n", prm);
   }
   return ۰;
}
```

شکل ۶-۵ الف: متن برنامهٔ ۶-۱، چاپ اعداد اول از ۳ تا n با استفاده از دستور for در قالب یک تابع.

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

int isprime(long p)                                 /* تابعی برای بررسی اول بودن یک عدد */
{  long dvd;

   for(dvd = ۲; dvd < (long)sqrt(p) + ۱; dvd++)
      if (!(p % dvd))
            return ۰;                               /* برگرداندن جواب صفر اگر عدد اول نیست */
   return ۱;                                        /* برگرداندن جواب یک اگر عدد اول است */
}


main()                                              /* تابع اصلی برای انجام سایر امور */
{  long prm, n;

   scanf ("%ld", &n);
   for (prm = ۳; prm <= n; prm += ۲)                /* حلقهٔ تکرار ساختن اعداد */
      if (isprime(prm))                             /* احضار تابع بررسی اول بودن */
         printf("%ld\n", prm);                      /* چاپ عدد درصورتی که اول باشد */
   return ۰;
}
```

شکل ۶-۵ ب: متن برنامهٔ ۶-۱، چاپ اعداد اول از ۳ تا n با استفاده از دو تابع.

مثال: برنامهٔ نمونهٔ ۶-۲

برنامه‌ای بنویسید که تعداد n عدد صحیح و مثبت حداکثر هشت رقمی (n قبل از همهٔ اعداد داده می‌شود) را از ورودی بخواند و برای هر مقدار درصورتی که می‌تواند یک عدد در مبنای هشت باشد آن را به مبنـای ده برده، همراه با عدد اولیه چاپ نماید وگرنه عدد را همراه با پیغام مناسب چاپ کند.

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

main()                                          /* برنامهٔ تبدیل مبنا از هشت به ده */
{  long m, i, newm, tm;
   int n;

   printf("\fNumber in Base ۸   Number in Base ۱۰\n");
   scanf("%d", &n);
   for (; n > ۰; n--)                           /* حلقهٔ تکرار خواندن عددها و انجام عملیات */
   {  scanf("٪۸ld", &m);
      for (tm = m, i = ۰, newm = ۰; tm > ۰; i++) /* حلقهٔ تکرار تبدیل مبنا */
      {  if (tm % ۱۰ > ۷)
            break;
         newm += tm % ۱۰ * pow(۸, i);
         tm /= ۱۰;
      }
      if (tm == ۰)
         printf("٪۱۰ld٪۸ld\n", m, newm);
      else
         printf("٪۱۰ld           not in base ۸\n", m);
   }
   return (۰);
}
```

داده‌های ورودی برنامه:

```
۷
۱۲
۲۰۰۰
۹۸۱
۱۰۰۰
۱۲۳۴۵
۱
۱۲۰
```

خروجی برنامه:

```
Number in Base ۸   Number in Base ۱۰
        ۱۲                 ۱۰
```

شکل ۶-۶: متن برنامهٔ ۶-۲، تبدیل عدد از مبنای هشت به ده همراه با نمونهٔ اجرای برنامه.

۶-۳ سایر ساختارهای انتخاب

۶-۳-۱ دستور شرطی تودرتو

- روش مناسب نوشتن دستورهای شرطی تودرتو.
- فشرده‌سازی `if`های تودرتو با کنگره‌بندی مناسب.
- توجه به لزوم جدا از هم نوشتن کلمه‌های `else` و `if` در این قالب.

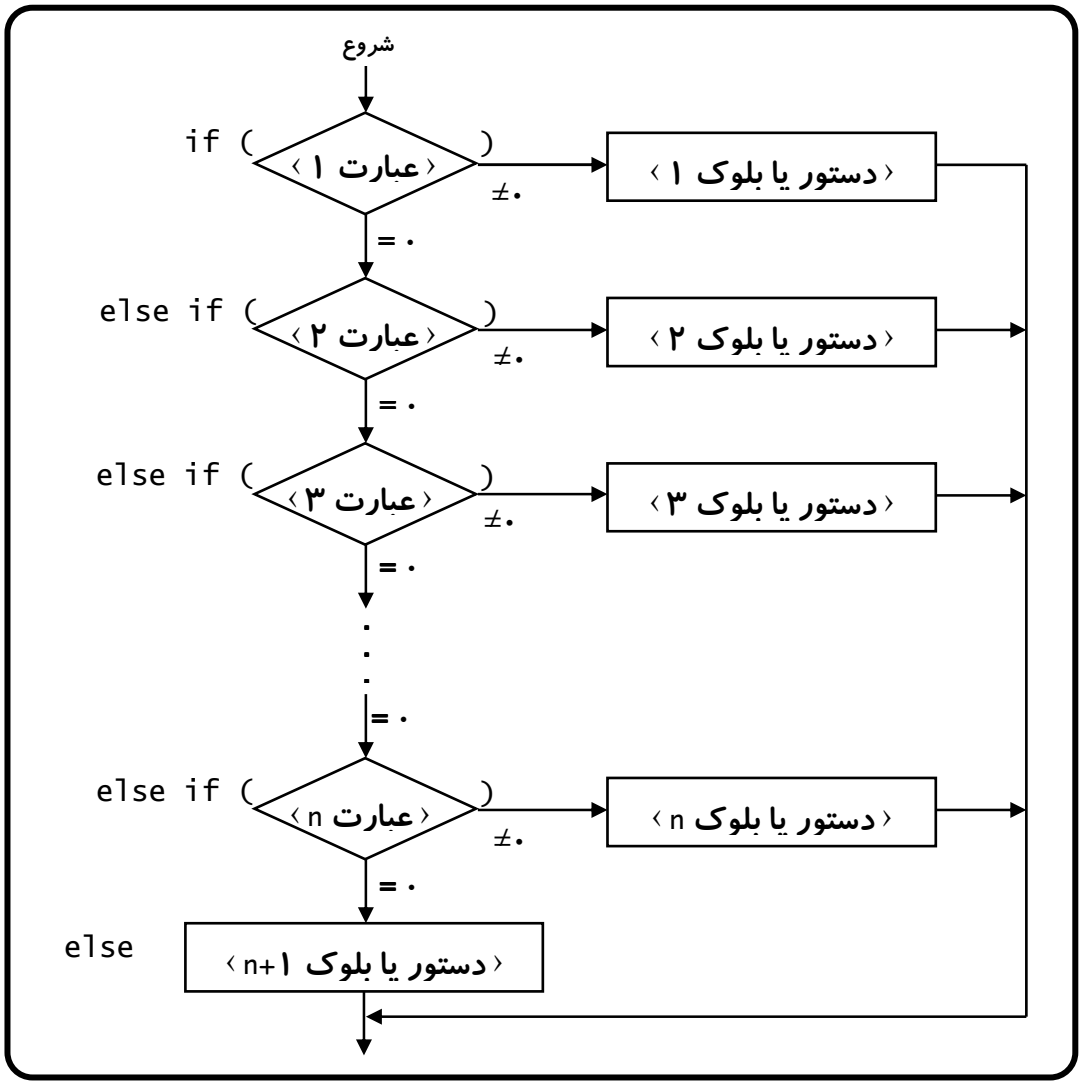
قالب مناسب

```
if ( ‹عبارت ۱› )
   ‹دستور یا بلوک ۱›
else if ( ‹عبارت ۲› )
   ‹دستور یا بلوک ۲›
else if ( ‹عبارت ۳› )
   ‹دستور یا بلوک ۳›
      .
      .
      .
else if ( ‹عبارت n› )
   ‹دستور یا بلوک n›
else
   ‹دستور یا بلوک n+۱›
```

قالب نامناسب

```
if ( ‹عبارت ۱› )
   ‹دستور یا بلوک ۱›
else
   if ( ‹عبارت ۲› )
      ‹دستور یا بلوک ۲›
   else
      if ( ‹عبارت ۳› )
         ‹دستور یا بلوک ۳›
         .
         .
         .
               else
                  if ( ‹عبارت n› )
                     ‹دستور یا بلوک n›
                  else
                     ‹دستور یا بلوک n+۱›
```

شکل ۶-۲ الف: قالبهای کلی مناسب و نامناسب برای دستورهای `if` تودرتو.

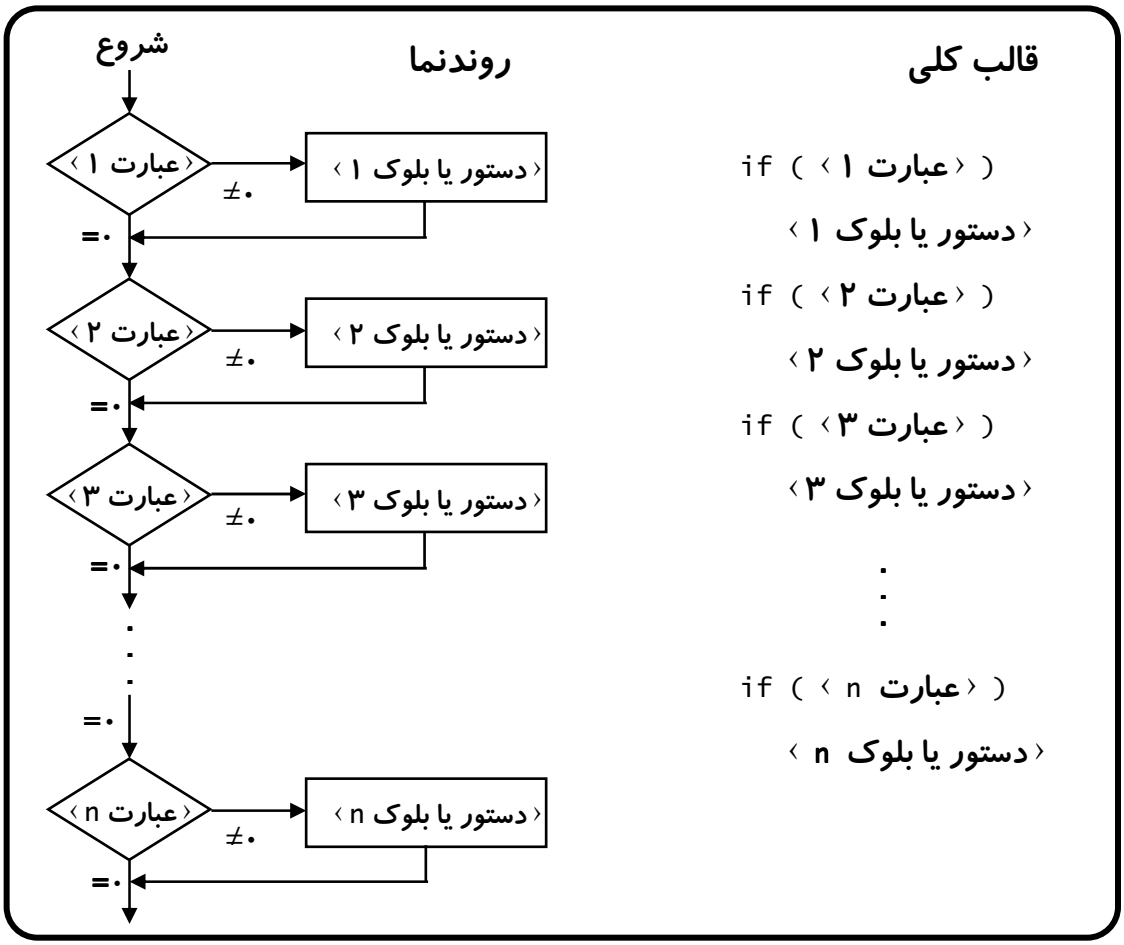


شکل ۶-۲ ب: نحوی اجرای دستور if تودرتو.

- اجرای ‹دستور یا بلوک i› در صورت غیرصفر بودن ‹عبارت i› و سپس ترک ساختار، برای i<n.
- اجرای ‹دستور یا بلوک n+۱› (اگر موجود باشد) در صورت صفر بودن همهٔ عبارتها.
- توجه به تفاوت کلی این ساختار با ساختارهای ارائه شده در شکلهای ۶-۳ الف، ۶-۳ ب و ۶-۳ پ

ساختار الف مجموعه‌ای از دستورهای شرطی مستقل از یکدیگر.

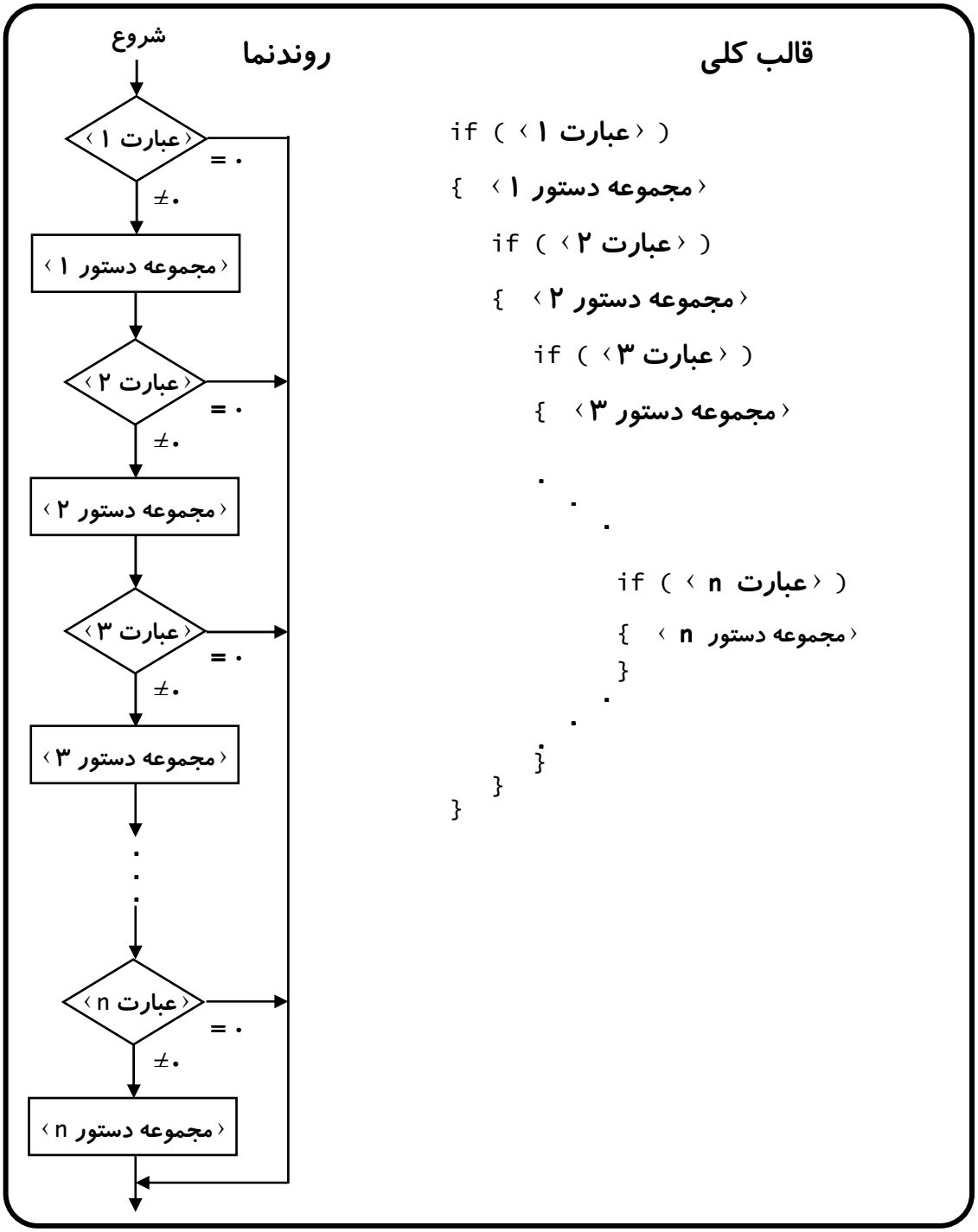
- **اجرای دستور یا بلوک مربوط به هر عبارت در صورت غیرصفر بودن حاصل آن عبارت (چه یکی و چه بیشتر).**



شکل ۶-۳الف: روندنما و قالب کلی دستورهای if مستقل.

ساختار ب: دستورهای if با بلوکهای تو در تو و ظاهر خطی

- تودرتو بودن در قسمت if (غیرصفر بودن عبارت)به جای قسمت else.



شکل ۶-۳ ب: روندنما و قالب کلی دستورهای if با بلوکهای تو در تو و ظاهر خطی.

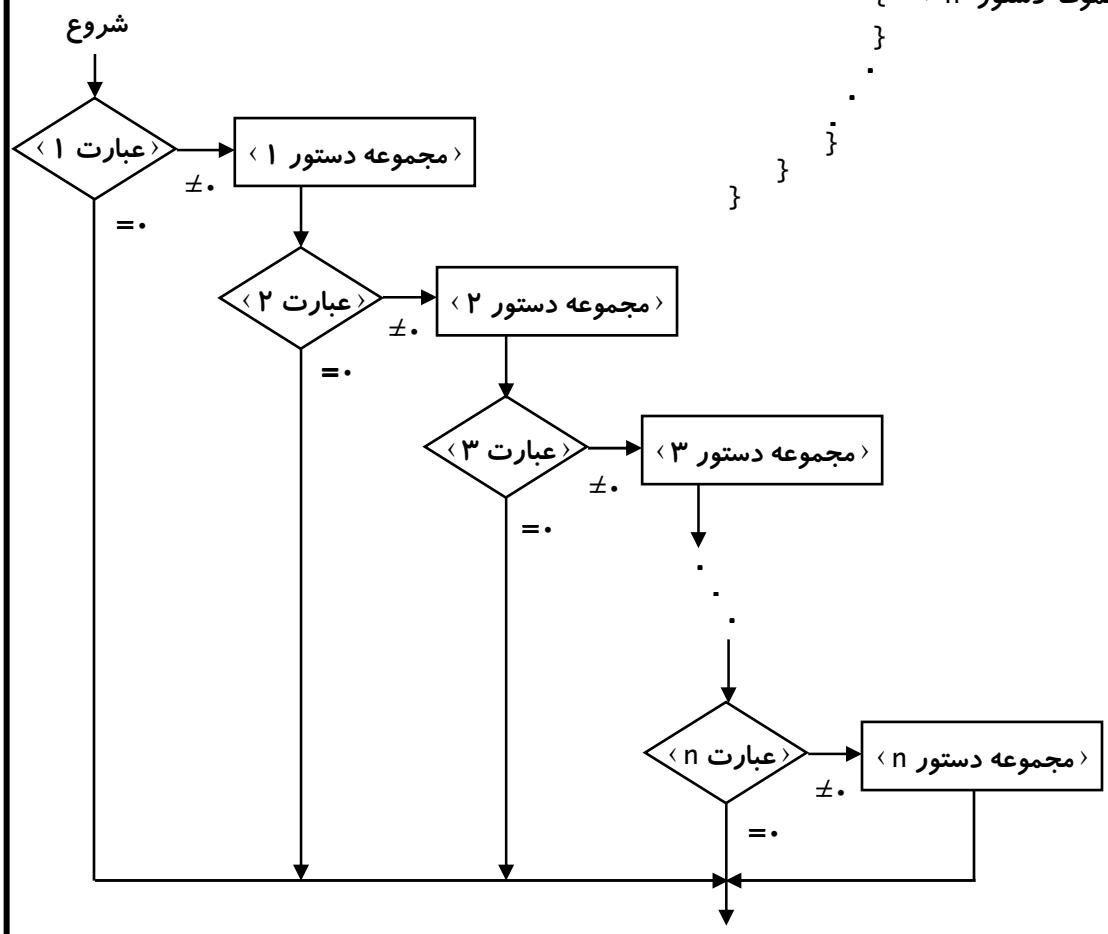
ساختار پ: دستورهای if با بلوکهای تو در تو و ظاهر پله‌ای، دارای معنای مشابه ساختار ب.

- **مورد استفاده در برنامهٔ نمونهٔ ۶-۳ برای حل تعدادی معادلهٔ درجه دوم.**

قالب کلی

```
if ( ‹ عبارت ۱ › )
{  ‹مجموعه دستور ۱›
   if ( ‹عبارت ۲› )
   {  ‹مجموعه دستور ۲›
      if ( ‹عبارت ۳› )
      {  ‹مجموعه دستور ۳›
         .
           .
             .
         if ( ‹ n عبارت› )
         {  ‹مجموعه دستور n ›
         }
           .
             .
      }
   }
}
```

روندنما



شکل ۶-۳ پ: روندنما و قالب کلی دستورهای if با بلوکهای تو در تو و ظاهر پله‌ای.

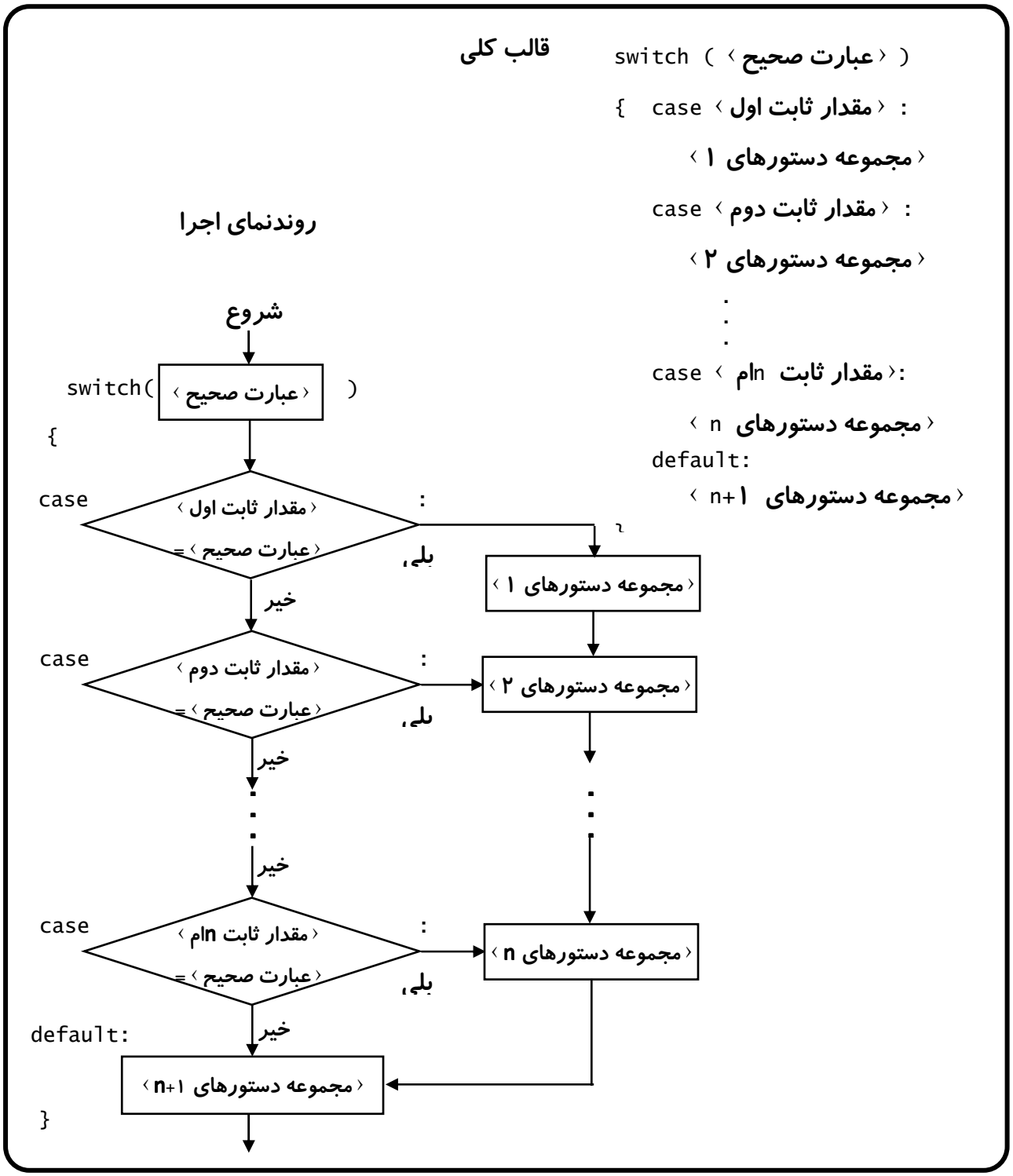
مثال برنامهٔ نمونهٔ ۶-۳: برنامه‌ای بنویسید که در آن ضرایب تعدادی معادلهٔ درجه دوم از ورودی خوانده شود. تعداد معادله‌ها در آغاز داده شده و هر ضریب به صورت یک عدد اعشاری در حداکثر پنج ستون درج گردیده است. در مورد هر معادله نخست پس از رها کردن یک سطر خالی، هر ضریب روی یک سطر با توضیح مناسب چاپ گردد. سپس معادله حل شده، با توجه به علامت دلتا جوابهای معادله در قالب ریشه‌های مختلف، ریشه‌های مضاعف و ریشه‌های مختلط محاسبه گردد. نهایتاً ریشه‌های محاسبه شده روی یک سطر مجزا با توضیحات مناسب چاپ شود. در این برنامه باید غیرمنطقی بودن ضرایب و همچنین درجه اول بودن هر معادله نیز بررسی شده و عملیات مناسب انجام گیرد. علاوه بر این لازم است شروع چاپ نتایج برنامه از اول یک صفحهٔ جدید باشد.

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

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

   scanf("%d", &n);
   printf("\f");
   for (j = ۱; j <= n; ++j)
   {  scanf("٪۵f٪۵f٪۵f", &a, &b, &c);
      printf("\na=٪۸.۵f\nb=٪۸.۵f\nc=٪۸.۵f\n", a, b, c);
      if (a == ۰)
         if (b == ۰)
            if (c != ۰)                         /* تشخیص غلط بودن ضرایب */
               printf("Incorrect coefficients!\n");
            else
               ;
         else                                   /* تشخیص معادلهٔ درجه اول */
         {  x = -c / b; printf("X=٪۸.۵f\n", x); }
      else
      {  delta = b * b - ۴ * a * c;
         if (delta > ۰)                         /* تشخیص دو ریشهٔ 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)                    /* تشخیص ریشهٔ مضاعف */
         {  x = -b / (۲ * a); printf("X۱=X۲=٪۸.۵f\n", x); }
         else                                   /* تشخیص ریشه‌های مختلط */
         {  delta = -delta;
            x = -b / (۲ * a);                   /* محاسبه و چاپ ریشه‌های مختلط */
            printf("X۱=٪۸.۵f+i*٪۸.۵f\t", x, sqrt(delta) / (۲ * a));
            printf("X۲=٪۸.۵f-i*٪۸.۵f\n", x, sqrt(delta) / (۲ * a));
         }
      }
   }
```

شکل ۶-۷: متن برنامهٔ ۶-۳، حل تعدادی معادلهٔ درجه دوم با استفاده از دستور شرطی چندگانه.

۶-۳-۲ دستور switch

- نوع دیگری از تصمیم‌گیریهای چندگانه براساس نتیجهٔ محاسبهٔ یک عبارت صحیح.
- مقایسهٔ نتیجهٔ عبارت با مقادیر ثابت دلخواه و اجرای بخش یا بخشهایی در صورت تساوی.



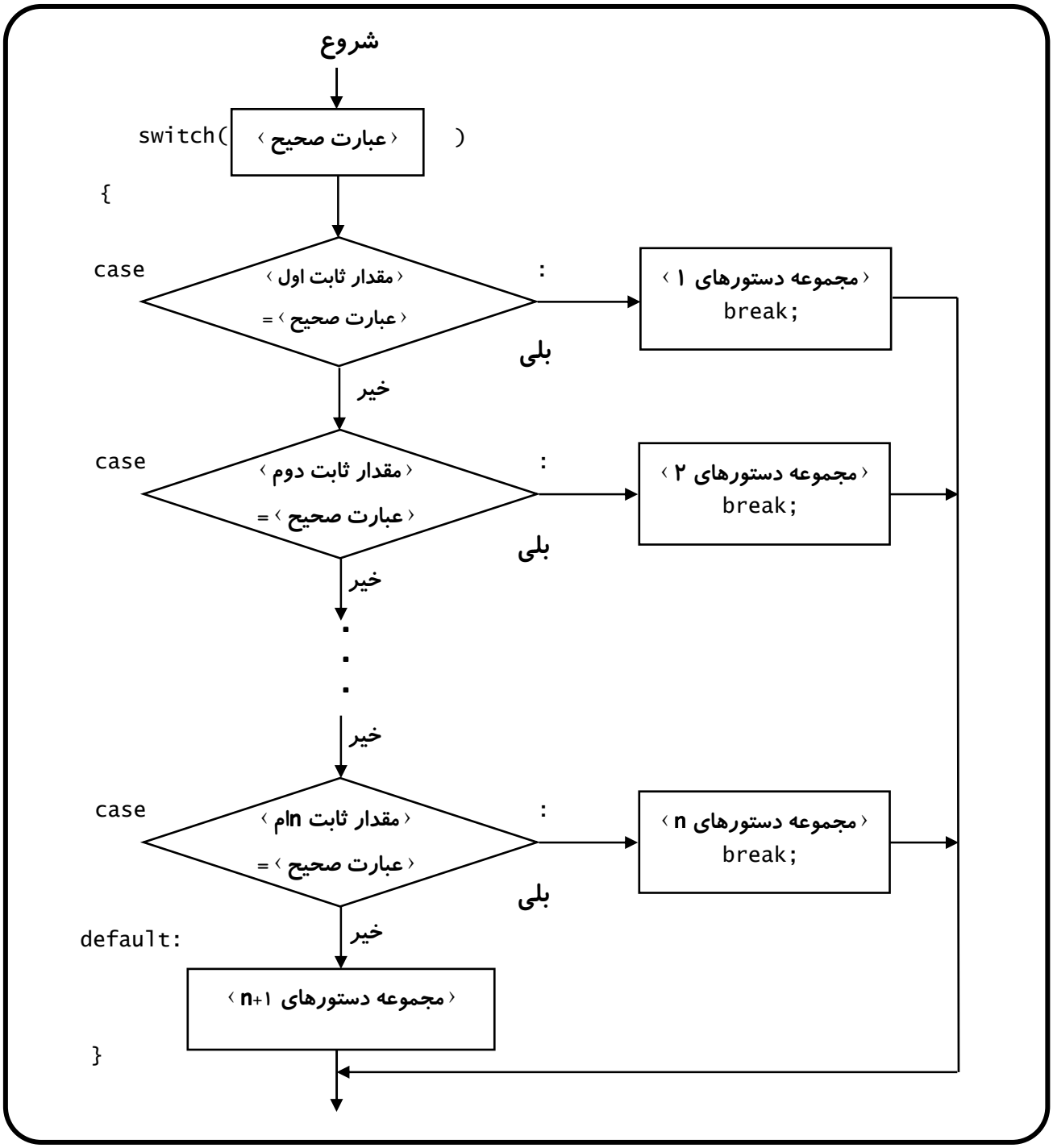
شکل ۶-۴ الف: قالب کلی و نحوهٔ اجرای دستور switch.

- دستور شرطی چندگانه، تصمیم‌گیری بر پایهٔ مقدار یک عبارت شرط مشترک از نـوع صحیح.
- محاسبه مقـدار ‹ عبارت صحیح › در آغـاز و مقـایسه با مقادیر ثـــابت اول تا nام.
- ادامهٔ اجـرای از مجمـوعه دستور بعـد از caseهای kام در صـورت تساوی مقدار عبارت با مقدار ثـابت kام.
- ادامهٔ اجـرای ‹ مجمـوعه دستورهـای k › تـا پـایـان بلوک switch.
- ادامهٔ اجرا از مجموعه دستورهای بعد از default (در صورت وجـود) در حالـت عـدم تسـاوی نتیجـهٔ عبارت با هیچیک از مقادیر ثابت اول تا nام.

نکات مهم در رابطه با دستور switch

- لزوم متفاوت بودن مقادیر ثابت بعد از case های مختلف (عدد صحیح، ثابـت نمـادی یـا شمارشـی یـا عبارتی متشکل از آنها.
- اختیاری و قابل حذف بودن حالت default، امکان عدم تساوی مقدار عبـارت بـا هیچیـک از مقـادیر ثابت اول تا nام و عدم اجرای هیچ‌کدام از مجموعه دستورها.
- عدم نیاز به قرار دادن مجموعه دستورهای هر case در داخل بلوک (علائم { }).
- پشت سرهم نوشتن دو یا چند case درصورتی که عملیات مربوط به آنها یکسان باشد.
- عدم نیاز به نوشتن caseهای مختلف و حتی بلوک default با ترتیب خاص.
- نوشتن default به طور قراردادی در پایان.
- لزوم قرار دان یک دستور break; در پایان مجموعه دسـتورهای هـر caseای کـه قـرار اسـت فقـط مجموعه دستورهای مربوط به همان case اجرا گردد و سپس ساختار switch ترک شود.
- معمول بودن این‌کار در دستور switch.

روندنمای دستور switch با فرض وجود دستور break; در آخر هر مجموعه دستور.



شکل ۶-۴ ب: نحوهٔ اجرای دستور switch همراه با دستور break در بلوکهای آن.

مثال: کاربرد ساده‌ای از دستور switch.

```
switch (x)
{  case '۱':    /* نه کیس با دستور قابل اجرای مشترک */
   case '۲':
   case '۳':
   case '۴':
   case '۵':
   case '۶':
   case '۷':
   case '۸':
   case '۹':
      x = x - '۱' + 'a';
      break;   /* switch ترک دستور */
   case 'a':   /* دو کیس با دستور قابل اجرای مشترک */
   case 'b':
      x = x -'a' + 'A';
      break;   /* switch ترک دستور */
   case ۱۲۳:    /* ادامهٔ اجرا به داخل کیس بعدی بعد از اجرای جملهٔ این کیس */
      printf("%d\n",x);
   case '>':
      ++x;
      break;   /* switch ترک دستور */
   default:
      x = ۱۰۰;
      break;   /* switch ترک دستور */
}
```

مثال: برنامهٔ نمونهٔ ۶-۵ چاپ یک تقویم برای سال خورشیدی

برنامه‌ای بنویسید که در آغاز داده‌های مربوط به این‌که روز اول سال چه روزی از هفته است و همچنین کبیسه یا عادی بودن سال را از ورودی گرفته، یک تقویم خورشیدی چاپ نماید. بخشی از یک نمونه خروجی مورد انتظار از این برنامه که تقویم سال ۱۳۸۳ را نشان می‌دهد، در شکل ۶-۹ الف ارائه شده است. نام روزهای هفته باید به صورت افقی در بالا و شمارهٔ روزها به صورت دقیق در زیر نام هر روز چاپ شود. همچنین قبل از شروع به چاپ نام روزهای هفته، باید نام ماه با حروف بزرگ بعد از رها شدن دو سطر خالی چاپ گردد و بعد از چاپ نام ماه هم یک سطر خالی رها شود.

```
FARVARDIN
Shanbe     Yekshanbe Doshanbe Seshanbe  Charshanbe Panjshanbe Jomeh
                                                                ۱
   ۲          ۳         ۴         ۵          ۶          ۷         ۸
   ۹          ۱۰        ۱۱        ۱۲         ۱۳         ۱۴        ۱۵
   ۱۶         ۱۷        ۱۸        ۱۹         ۲۰         ۲۱        ۲۲
   ۲۳         ۲۴        ۲۵        ۲۶         ۲۷         ۲۸        ۲۹
   ۳۰         ۳۱

ORDIBEHESHT
Shanbe     Yekshanbe Doshanbe Seshanbe  Charshanbe Panjshanbe Jomeh
                                 ۱          ۲          ۳         ۴
   ۵          ۶         ۷         ۸          ۹          ۱۰        ۱۱
   ۱۲         ۱۳        ۱۴        ۱۵         ۱۶         ۱۷        ۱۸
   ۱۹         ۲۰        ۲۱        ۲۲         ۲۳         ۲۴        ۲۵
   ۲۶         ۲۷        ۲۸        ۲۹         ۳۰         ۳۱

KHORDAD
Shanbe     Yekshanbe Doshanbe Seshanbe  Charshanbe Panjshanbe Jomeh
                                                                ۱
   ۲          ۳         ۴         ۵          ۶          ۷         ۸
   ۹          ۱۰        ۱۱        ۱۲         ۱۳         ۱۴        ۱۵
   ۱۶         ۱۷        ۱۸        ۱۹         ۲۰         ۲۱        ۲۲
   ۲۳         ۲۴        ۲۵        ۲۶         ۲۷         ۲۸        ۲۹
   ۳۰         ۳۱

      .          .          .
      .          .          .
      .          .          .

ESFAND
Shanbe     Yekshanbe Doshanbe Seshanbe  Charshanbe Panjshanbe Jomeh
   ۱          ۲         ۳         ۴          ۵          ۶         ۷
```

شکل ۶-۹ الف: بخشی از نتیجهٔ مورد انتظار از برنامهٔ ۶-۵، چاپ تقویم هجری خورشیدی.

- **توجه به نحوهٔ در خواست شمارهٔ روز شروع و نوع سال، خواندن و بررسی درستی آنها در یک حلقه. استفاده از این ساختار به عنوان الگویی برای این‌گونه عملیات.**
- **استفاده از دستور `switch` برای چاپ نام هر ماه و محاسبهٔ تعداد روزهای آن.**
- **توجه به نحوه تنظیم شروع چاپ شمارهٔ روزهای ماه با استفاده از یک محاسبهٔ ساده و یک حلقه `for`.**
- **حلقهٔ تکرار چاپ تقویم یک ماه و نهایتاً محاسبهٔ شمارهٔ روز شروع ماه بعد.**
- **رها کردن سطر در این دستور برای ماه‌هایی است که روز آخر آن جمعه نیست (چرا؟).**

```
#include <stdio.h>
main()                          /* برنامهٔ چاپ تقویم خورشیدی */
{  int day_code, days_in_month, day, month;
/* شمارهٔ روز شروع ماه (۰=شنبه، ۱=یکشنبه ... ۶=جمعه)، تعداد روز ماهی که تقویم آن چاپ می‌شود، شمارنده‌های روز و ماه */
  char leap_year;               /* A=سال عادی   K=سال کبیسه */
  do                            /* حلقهٔ تکرار خواندن کد روز اول سال و کبیسه بودن به صورت قابل قبول */
  { printf
      ("\nPlease enter starting day number and leap year code:\n ");
    scanf("%d%c", &day_code, &leap_year);
  } while (day_code<۰||day_code>۶||leap_year!='A'&&leap_year!='K');
  printf("\f");
  for (month = ۱; month <= ۱۲; month++)/* حلقهٔ تکرار آماده‌سازی داده‌های هر ماه */
  { switch (month)                      /* چاپ اسم ماه و تعیین تعداد روز هر ماه */
    { case ۱:
        printf("\n\nFARVARDIN");   days_in_month = ۳۱; break;
      case ۲:
        printf("\n\nORDIBEHESHT"); days_in_month = ۳۱; break;
      case ۳:
        printf("\n\nKHORDAD");     days_in_month = ۳۱; break;
      case ۴:
        printf("\n\nTIR");         days_in_month = ۳۱; break;
      case ۵:
        printf("\n\nMORDAD");      days_in_month = ۳۱; break;
      case ۶:
        printf("\n\nSHAHRIVAR");   days_in_month = ۳۱; break;
      case ۷:
        printf("\n\nMEHR");        days_in_month = ۳۰; break;
      case ۸:
        printf("\n\nABAN");        days_in_month = ۳۰; break;
      case ۹:
        printf("\n\nAZAR");        days_in_month = ۳۰; break;
      case ۱۰:
        printf("\n\nDEY");         days_in_month = ۳۰;  break;
      case ۱۱:
        printf("\n\nBAHMAN");      days_in_month = ۳۰; break;
      case ۱۲:
        printf("\n\nESFAND"); days_in_month =
                              leap_year == 'K' ? ۳۰ : ۲۹; break;
    }
    printf("\n\nShanbe    Yekshanbe  Doshanbe   Seshanbe    ");
    printf("Charshanbe Panjshanbe Jomeh\n");
    for (day=۱;day<۱+day_code*۱۱;day++)printf(" ");   /* تنظیم چاپ روز اول ماه*/
    for (day=۱; day <= days_in_month; day++)          /* چاپ روزهای یک ماه */
    { printf("%۴d", day);
      if((day+day_code)%۷ > ۰) printf("      ");    /*                    */
```

شکل ۶-۹ ب: متن برنامهٔ ۶-۵، چاپ تقویم هجری خورشیدی برای هر سال دلخواه در قالب یک تابع اصلی.

برنامه ۶-۵: متن برنامهٔ چاپ تقویم خورشیدی که کلاً در قالب یک تابع اصلی (main) در شکل ۶-۹ ب نوشته شد در قالب دو تابع، یکی با نام find_days برای چاپ نام هر ماه و محاسبهٔ تعداد روزهای آن و دیگری یک تابع اصلی برای انجام سایر موارد در شکلهای ۶-۹ پ و ۶-۹ ت ارائه شده است.

```
#include <stdio.h>
                /* . . . find_days محل قرار دادن تابع . . . */
main()                  /* برنامهٔ چاپ تقویم خورشیدی */
{ int day_code, days_in_month, day, month;
/* شمارهٔ روز شروع ماه (۰=شنبه، ۱=یکشنبه ... ۶=جمعه)، تعداد روز ماهی که تقویم آن چاپ می‌شود، شمارنده‌های روز و ماه */
  char leap_year;       /* A=سال عادی   K=سال کبیسه */
  do                    /* حلقهٔ تکرار خواندن کد روز اول سال و کبیسه بودن به صورت قابل قبول */
  { printf
    ("\nPlease enter starting day number and leap year code:\n ");
    scanf("%d%c", &day_code, &leap_year);
  } while (day_code<۰||day_code>۶||leap_year!='A'&&leap_year!= 'K');
  printf("\f");
  for (month = ۱; month <= ۱۲; month++)/* حلقهٔ تکرار آماده‌سازی داده‌های هر ماه */
  { days_in_month = find_days(month, leap_year);
                                /* چاپ اسم ماه و تعیین تعداد روز هر ماه */
    printf("\n\nShanbe    Yekshanbe  Doshanbe  Seshanbe    ");
    printf("Charshanbe Panjshanbe Jomeh\n");
    for (day = ۱; day < ۱ + day_code * ۱۱; day++) printf(" ");
                                /* تنظیم چاپ اولین روز ماه */
    for (day = ۱; day <= days_in_month; day++)   /* چاپ روزهای یک ماه */
    { printf("%۴d", day);
      if((day + day_code)%۷>۰) printf("      "); /* تنظیم چاپ روز بعدی*/
      else printf("\n");          /* ادامهٔ چاپ در سطر بعدی */
    }             /* جملهٔ شرطی بعدی برای تنظیم روز اول ماه بعد و رها کردن سطر در صورت لزوم */
      if(day_code = (day_code + days_in_month) % ۷) printf("\n");
   }
   return ۰;
}
```

شکل ۶-۹ پ: متن برنامهٔ ۶-۵، تابع اصلی مربوط به چاپ تقویم هجری خورشیدی.

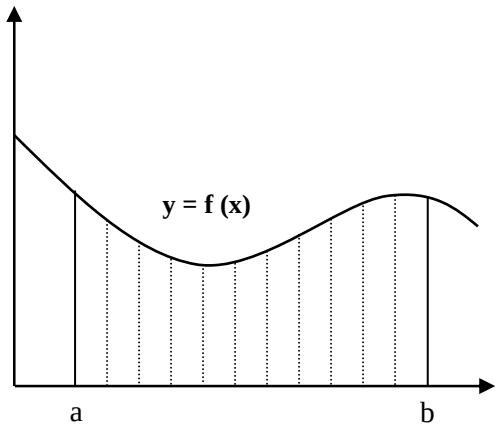
```
int find_days(int month, char leap_year)
                                        /*  تابعی برای چاپ اسم یک ماه و محاسبهٔ تعداد روز آن  */
{  int days_in_month;

switch (month)                          /*  چاپ اسم ماه و تعیین تعداد روز هر ماه  */
   {  case ۱:
         printf("\n\nFARVARDIN");
         days_in_month = ۳۱;
         break;
      case ۲:
         printf("\n\nORDIBEHESHT");
         days_in_month = ۳۱;
         break;
      case ۳:
         printf("\n\nKHORDAD");
         days_in_month = ۳۱;
         break;
      case ۴:
         printf("\n\nTIR");
         days_in_month = ۳۱;
         break;
      case ۵:
         printf("\n\nMORDAD");
         days_in_month = ۳۱;
         break;
      case ۶:
         printf("\n\nSHAHRIVAR");
         days_in_month = ۳۱;
         break;
      case ۷:
         printf("\n\nMEHR");
         days_in_month = ۳۰;
         break;
      case ۸:
         printf("\n\nABAN");
         days_in_month = ۳۰;
         break;
      case ۹:
         printf("\n\nAZAR");
         days_in_month = ۳۰;
         break;
      case ۱۰:
         printf("\n\nDEY");
         days_in_month = ۳۰;
         break;
      case ۱۱:
         printf("\n\nBAHMAN");
         days_in_month = ۳۰;
         break;
```

شکل ۶-۹ ت: متن برنامهٔ ۶-۵، تابع محاسبهٔ تعداد روز و چاپ اسم هر ماه در تقویم هجری

- **محل متن تابع `find_days` در بالای تابع `main`**
- **امکان قرار دادن توابع مختلف در فایلهای مختلف، ترجمه مجزا و تلفیق در زمان اتصال و پیوند.**

۶-۴ برنامه‌های نمونه

برنامهٔ ۶-۴: محاسبهٔ انتگرال معین در یک فاصلهٔ مشخص بـه روش ذوزنقـه. هـدف نوشـتن برنامـه‌ای است که تابع f(x) در آن تعریف شود و پس از خواندن مقـادیر a، b و n از ورودی مقـدار S را بـا اسـتفاده از روش مورد بحث محاسبه کرده، همراه با پیغام مناسب چاپ نماید. مسلماً فرض بر این است که تابع مـورد نظر در فاصلهٔ مزبور تعریف شده و پیوسته است.



$$S = \int_a^b f(x)\,dx \Rightarrow S \approx \sum_{i=1}^{n} S_i$$

$$S_i = \frac{h}{2} \times (f(a + (i-1) \times h) + f(a + i \times h))$$

$$h = \frac{b-a}{n}$$

شکل ۶-۸ الف: روش ذوزنقه برای محاسبهٔ تقریبی انتگرال معین.

```
#include <stdio.h>
#define f(x) (۲ * (x) * (x) + ۳ * (x) - ۵) /* تعریف تابع */

main()                                  /* برنامهٔ محاسبهٔ انتگرال معین به روش ذوزنقه */
{  double a, b, h, s = ۰;
   int i, n;

   scanf("%lf%lf%d",&a, &b, &n);
   h = (b - a) / n;
   for (i = ۱; i <= n; ++i)             /* حلقهٔ تکرار محاسبه و جمع زدن مساحت ذوزنقه‌ها */
      s += h * (f(a + (i - ۱) * h) + f(a + i * h)) / ۲;
   printf("For a = %f, b = %f , n = %d, the value of S =
%f\n",
          a, b, n, s);
   return (۰);
}
```

شکل ۶-۸ ب: متن برنامهٔ ۶-۴، محاسبهٔ انتگرال معین به روش ذوزنقه.

برنامهٔ ۶-۶: داده‌های مربوط به کارمندان دانشگاه به صورتی که در بالای شکل ۶-۱۰ الف نشان داده شده است آماده گردیده و قبل این داده‌ها تعداد کارمندان به صورت یک عدد صحیح حداکثر سه رقمی قرار دارد. هدف نوشتن مجموعه برنامه‌ای است که داده‌های هر کارمند را بخواند و پس از انجام عملیات لازم طبق ضوابطی که در ادامه گفته شده است، مبالغ مالیات، بازنشستگی، بیمهٔ درمان، بیمهٔ عمر و پس‌انداز مسکن او را محاسبه کرده، آن داده‌ها را همراه با نتایج محاسبات و خالص دریافتی در یک پرفراژ حقوقی به صورتی که در پایین شکل ۶-۱۰ الف نشان داده شده است از اول یک صفحه چاپ نماید.

قالب کلی سطر اصلی داده‌های ورودی

تعداد اضافه‌کاری	تعداد عائله	فوق‌العاده	مزایای شغل	حقوق پایه	شمارهٔ کارمند
۲ رقم	۲ رقم	۷ رقم	۷ رقم	۷ رقم	۵ رقم

قالب کلی پرفراژ حقوقی

```
                        PAYMENT SLIP

EMPLOYEE NO:  شمارهٔ کارمند

BASE PAY        :  حقوق پایه          TAX                :  مالیات
BONUS           :  مزایای شغل         HEALTH INSURANCE:  بیمهٔ درمان
EXTRA BONUS     :  فوق‌العاده         PENSION            :  بازنشستگی
FAMILY BENEFIT:  حق عائله‌مندی       LIFE INSURANCE     :  بیمهٔ عمر
OVERTIME WORK :  اضافه‌کاری           SAVING             :  پس‌انداز
TOTAL PAYMENTS:  جمع پرداختیها        TOTAL DEDUCTIONS:  جمع کسور

                NET PAYMENT:  خالص دریافتی
```

شکل ۶-۱۰ الف: قالب داده‌های ورودی و پرفراژ حقوقی مورد نیاز در برنامهٔ محاسبهٔ حقوق.

محاسبات مربوط به حقوق

- ۸٪ از مجموع حقوق پایه، مزایا، فوق‌العاده و اضافه‌کاری به عنوان بازنشستگی محاسبه می‌شود.
- به‌ازای هر یک نفر عائلهٔ تحت تکفل مبلغ ۵۰۰۰ ریال بیمه درمان در نظر گرفته می‌شود.
- ۲۰٪ از حقوق پایه به عنوان پس‌انداز مسکن کسر می‌گردد.
- مبلغ ۱۵۰۰۰ ریال برای بیمهٔ عمر کسر می‌گردد.
- به مجموع حقوق پایه، مزایای شغل، فوق‌العاده و اضافه‌کاری مالیات به شرح زیر تعلق می‌گیرد.
- تا مبلغ ۱۰۰۰۰۰ ریال از مالیات معاف است.
- از ۱۰۰۰۰۱ ریال تا ۵۰۰۰۰۰ ریال ۱۰٪ مالیات نسبت به مازاد ۱۰۰۰۰۰ ریال محاسبه می‌شود.
- از ۵۰۰۰۰۱ ریال تا ۱۰۰۰۰۰۰ ریال ۱۵٪ مالیات نسبت به مازاد ۵۰۰۰۰۰ ریال محاسبه می‌شود.
- از ۱۰۰۰۰۰۱ ریال به بالا ۲۰٪ مالیات نسبت به مازاد ۱۰۰۰۰۰۰ ریال محاسبه می‌شود.
- به‌ازای هر نفر عائله غیر از خود شخص که جزء تعداد عائله است، مبلغ ۱۰۰۰۰ ریال پرداخت می‌شود.
- درصورت غیرصفر بودن تعداد اضافه‌کاری، به این تعداد سطرهایی حاوی ساعات اضافه‌کاری و نرخ یک ساعت به صورت دو عدد صحیح بعد از سطر اصلی داده‌ها وجود دارد که باید خوانده شده، مبلغ اضافه‌کاری با استفاده از آنها محاسبه گردد.
- خالص دریافتی عبارت است از مجموع حقوق پایه، مزایای شغل، فوق‌العاده، حق عائله و اضافه‌کاری پس از کسر مجموع مالیات، بازنشستگی، بیمهٔ درمان، بیمهٔ عمر و پس‌انداز مسکن.

تقسیم بندی برنامه

- فرمانهای پیش‌ترجمه در شکل ۶-۱۰ ب.
- تابع فرعی برای گرفتن مبلغ مشمول مالیات، محاسبهٔ مالیات برگرداندن آن در شکل ۶-۱۰ پ.
- تابع اصلی برای انجام کلیهٔ موارد غیر از محاسبهٔ مالیات در شکل ۶-۱۰ ت.
- نمونه‌ای از اجرای برنامه در شکل ۶-۱۰ ث.

چند نکته مهم

- تعریف ثابت نمادی برای پله‌های محاسبهٔ مالیات، درصد بازنشستگی و غیره که قابل تغییر باشند.
- تعریف ماکرو برای عبارات مشمول مالیات و پس‌انداز.
- تعریف ثابت شمارشی برای درصدهای مربوط به پله‌های محاسبهٔ مالیات.
- استفاده اجباری از عملگر تغییر نوع در سه مورد مبلغ اضافه‌کاری، مبلغ عائله‌مندی و مبلغ بیمهٔ درمان.

```
                /*  فرمانهای پیش‌ترجمه برای برنامهٔ محاسبه و چاپ پرفراژ حقوق کارمندان  */

#include <stdio.h>
#define PNSN_RT ۰.۰۸                    /*  درصد بازنشستگی  */
#define SVNG_RT ۰.۲۰                    /*  درصد پس‌انداز  */
#define HLT_INS ۵۰۰۰                    /*  مبلغ سرانهٔ بیمهٔ درمان  */
#define LIF_INS ۱۵۰۰۰                   /*  مبلغ بیمهٔ عمر  */
#define FML_BNF ۱۰۰۰۰                   /*  مبلغ سرانهٔ عائله‌مندی  */
#define STEP۱   ۱۰۰۰۰۰          /*  مبلغ پلهٔ اول محاسبهٔ مالیات  */
#define STEP۲   ۵۰۰۰۰۰          /*  مبلغ پلهٔ دوم محاسبهٔ مالیات  */
#define STEP۳   ۱۰۰۰۰۰۰         /*  مبلغ پلهٔ سوم محاسبهٔ مالیات  */
                                 /*  سطر بعدی، عبارت مشمول مالیات  */
#define TAXABLE (base + bonus + extra + overtime)
#define SAVABLE (base)                   /*  عبارت مشمول پس‌انداز  */
                                 /*  سطر بعدی، عبارت جمع کسور  */
#define DEDUCTIONS (tax+pension+hlt_insr+lif_insr+saving)
```

شکل ۶-۱۰ ب: متن برنامهٔ ۶-۶، فرمانهای پیش‌ترجمهٔ مورد نیاز برای محاسبه و چاپ پرفراژ حقوق.

```
long tax_comp(long amount)                  /*  تابع محاسبهٔ مالیات  */
{  long tax;
   enum tax_percents {PRCT۱ = ۱۰, PRCT۲ = ۱۵, PRCT۳ = ۲۰};

   if (amount <= STEP۱)                     /*  معاف از مالیات  */
      tax = ۰;
   else if (amount <= STEP۲)                /*  پلهٔ اول محاسبهٔ مالیات  */
      tax = (amount - STEP۱) * PRCT۱ / ۱۰۰.۰;
   else if (amount <= STEP۳)                /*  پلهٔ دوم محاسبهٔ مالیات  */
      tax = (STEP۲ - STEP۱) * PRCT۱ / ۱۰۰.۰ +
            (amount - STEP۲) * PRCT۲ / ۱۰۰.۰;
   else                                     /*  پلهٔ سوم محاسبهٔ مالیات  */
      tax = (STEP۲ - STEP۱) * PRCT۱ / ۱۰۰.۰ +
            (STEP۳ - STEP۲) * PRCT۲ / ۱۰۰.۰ +
            (amount - STEP۳) * PRCT۳ / ۱۰۰.۰;
   return tax;                              /*  برگرداندن مالیات محاسبه شده  */
}
```

شکل ۶-۱۰ پ: متن برنامهٔ ۶-۶، تابع محاسبهٔ مالیات.

```
                /* . . .  محل قرار دادن فرمانهای پیش‌ترجمه شکل ۶-۱۰ پ  . . . */

                /* . . .  محل قراردادن تابع محاسبهٔ مالیات شکل ۶-۱۰ پ  . . . */

main()                                          /* برنامهٔ محاسبه و چاپ پرفراژ حقوق کارمندان */
{ long emp_no, base, bonus, extra, overtime, benefit;
  long tax, pension, hlt_insr, lif_insr, saving, net_pay;
  int i, j, no_of_emp, res_no, ovtm_no, ovtm_hr, ovtm_rt;

  scanf("%d", &no_of_emp);
  for (i = 1; i <= no_of_emp; i++)
  { scanf("%5d%7ld%7ld%7ld%2d%2d",      /* خواندن داده‌های هر کارمند */
          &emp_no, &base, &bonus, &extra, &res_no, &ovtm_no);
    overtime = 0;
    for (j = 1; j <= ovtm_no; j++)              /* خواندن داده‌های اضافه‌کاری درصورت لزوم */
    { scanf("%d%d", &ovtm_hr, &ovtm_rt);
      overtime += (long)ovtm_hr * ovtm_rt;
    }
    tax = tax_comp(TAXABLE);                    /* احضار تابع محاسبهٔ مالیات */
    benefit = (long)(res_no - 1) * FML_BNF;/* محاسبهٔ حق عائله‌مندی */
    pension = TAXABLE * PNSN_RT;                /* محاسبهٔ بازنشستگی */
    saving = SAVABLE * SVNG_RT;                 /* محاسبهٔ پس‌انداز مسکن */
    hlt_insr = (long)res_no * HLT_INS;          /* محاسبهٔ بیمهٔ درمان */
    lif_insr = LIF_INS;                         /* محاسبهٔ بیمهٔ عمر */
                                                /* سطر بعدی، محاسبهٔ خالص دریافتی */
    net_pay = TAXABLE + benefit - DEDUCTIONS;
                                                /* چاپ پرفراژ حقوق */
    printf("\f\t\t\t PAYMENT SLIP\n\nEMPLOYEE NO:%5ld\n\n",
            emp_no);
    printf("BASE PAY      :%7ld\t\t TAX              :%7ld\n",
            base, tax);
    printf("BONUS         :%7ld\t\t HEALTH INSURANCE:%7ld\n",
            bonus, hlt_insr);
    printf("EXTRA BONUS   :%7ld\t\t PENSION          :%7ld\n",
            extra, pension);
    printf("FAMILY BENEFIT:%7ld\t\t LIFE INSURANCE   :%7ld\n",
            benefit, lif_insr);
    printf("OVERTIME WORK :%7ld\t\t SAVING           :%7ld\n",
            overtime, saving);
    for (j = 1; j <= 58; printf("-"), j++);
    printf("\nTOTAL PAYMENTS:%7ld\t\t TOTAL DEDUCTIONS:%7ld\n",
              TAXABLE + benefit, DEDUCTIONS);
    printf("\n\t\tNET PAYMENT:%7ld\n\n", net_pay);
  }
  return 0;
```

شکل ۶-۱۰ ت: متن برنامهٔ ۶-۶، تابع اصلی برای محاسبه و چاپ پرفراژ حقوق کارمندان.

داده‌های ورودی

```
۲
۱۲۳۴۵۳۰۰۰۰۰۲۰۰۰۰۰۴۰۰۰۰۰۵  ۲
۱۰  ۱۰۰۰۰
۲۰  ۲۰۰۰۰
۵۴۳۲۱۳۰۰۰۰۰  ۲۰۰۰۰۰  ۴۰۰۰۰۰  ۱  ۰
```

نتیجهٔ خروجی

```
                    PAYMENT SLIP

EMPLOYEE NO:۱۲۳۴۵

BASE PAY        :۳۰۰۰۰۰۰      TAX                 :۱۸۱۵۰۰۰
BONUS           :۲۰۰۰۰۰۰      HEALTH INSURANCE:    ۲۵۰۰۰
EXTRA BONUS     :۴۰۰۰۰۰۰      PENSION             :  ۷۶۰۰۰۰
FAMILY BENEFIT:    ۴۰۰۰۰      LIFE INSURANCE      :   ۱۵۰۰۰
OVERTIME WORK  :   ۵۰۰۰۰۰      SAVING              :   ۶۰۰۰۰۰
-------------------------------------------------------------
TOTAL PAYMENTS:۹۵۴۰۰۰۰      TOTAL DEDUCTIONS:۳۲۱۵۰۰۰

            NET PAYMENT:۶۳۲۵۰۰۰


                    PAYMENT SLIP

EMPLOYEE NO:۵۴۳۲۱

BASE PAY        :  ۳۰۰۰۰۰      TAX                 :  ۱۰۰۰۰۰
BONUS           :  ۲۰۰۰۰۰      HEALTH INSURANCE:      ۵۰۰۰
EXTRA BONUS     :  ۴۰۰۰۰۰      PENSION             :  ۷۲۰۰۰
FAMILY BENEFIT:          ۰      LIFE INSURANCE      :   ۱۵۰۰۰
OVERTIME WORK  :          ۰      SAVING              :   ۶۰۰۰۰
-------------------------------------------------------------
```

شکل ۶-۱۰ ث: نمونهٔ اجرای برنامهٔ ۶-۶، محاسبه و چاپ پرفراژ حقوق کارمندان.

۶-۵ اشتباهات متداول برنامه‌نویسی

۶-۶ پرسشها

توصیه اکید در مورد انجام پرسشهای ۶-۱۳ و ۶-۱۴

تکلیف پنجم: مهلت دو هفته

انجام پرسش۶-۱۲

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

***۶-۱۲- برای هر دانشجو داده‌هایی با قالب کلی ارائه شده در شکل ۶-۱۲ الف آماده شده است. برنامه‌ای بنویسید که داده‌های مربوط به هر دانشجو را خوانده، برای او یک کارنامه با الگوی داده شده در شکل ۶-۱۲ ب چاپ نماید. علاوه بر اصل کارنامه‌ها، در پایان کارنامه‌های هر رشته، معدل متوسط آن رشته چاپ گردد و در پایان همهٔ کارنامه‌ها، معدل متوسط دانشگاه، شماره و معدل دانشجو با بالاترین معدل، شماره و معدل دانشجو با پایین‌ترین معدل، تعداد کل دانشجویان ممتاز (معدل بالای ۱۷) و تعداد کل دانشجویان سه ترم مشروط همراه با توضیحات مناسب، روی یک صفحهٔ مجزا چاپ شود. فرض کنید فقط یک دانشجو با بالاترین و همچنین فقط یک دانشجو با پایین‌ترین معدل وجود دارد. برنامه را به سلیقهٔ خود به چند تابع تقسیم کرده، برای آنها یک دیاگرام ساختاری نیز رسم نمایید.**

<table>
<tr><td colspan="3">قالب کلی داده‌های ورودی</td><td>تعداد رشته‌های دانشگاه (دو رقم)</td></tr>
<tr><td colspan="3">با داده‌های دانشجویان برای هر رشته تکرار می‌شود تعداد دانشجو در این رشته (چهار رقم)</td><td>کد رشته (دو رقم)</td></tr>
<tr><td colspan="3">با داده‌های ترمهای مختلف برای هر دانشجو تکرار می‌شود تعداد ترم این دانشجو (دورقم)</td><td>شمارهٔ دانشجویی (هفت رقم)</td></tr>
<tr><td colspan="3">با سطرهای بعدی به تعداد ترم تکرار می‌شود تعداد درس در ترم (دو رقم)</td><td>شمارهٔ ترم (سه رقم)</td></tr>
<tr><td>نمرهٔ درس (از ۲۰ با یک رقم اعشار)</td><td>واحد درس (یک رقم صحیح و یک رقم اعشار)</td><td>شمارهٔ درس ۱ (پنج رقم)</td></tr>
<tr><td>نمرهٔ درس (از ۲۰ با یک رقم اعشار)</td><td>واحد درس (یک رقم صحیح و یک رقم اعشار)</td><td>شمارهٔ درس ۲ (پنج رقم)</td></tr>
<tr><td>به تعداد درس در ترم تکرار می‌شود</td><td>.
.
،</td><td>.
.
،</td></tr>
</table>

شکل ۶-۱۲ الف: قالب کلی داده‌های ورودی پرسش ۶-۱۲.

ISFAHAN UNIVERSITY OF TECHNOLOGY

STUDENT TRANSCRIPT

Major Code: کد رشتهٔ تحصیلی Student No: شمارهٔ دانشجویی

Term No: شمارهٔ ترم اول

Course No	Units	Mark
شمارهٔ درس	واحد درس	نمرهٔ درس
.	.	.
.	.	.
.	.	.

Registered: تعداد واحد ثبت نام شده در ترم Passed: تعداد واحد گذرانده شده در ترم Term GPA: معدل ترم

Term No: شمارهٔ ترم دوم

Course No	Units	Mark
شمارهٔ درس	واحد درس	نمرهٔ درس
.	.	.
.	.	.
.	.	.

Registered: تعداد واحد ثبت نام شده در ترم Passed: تعداد واحد گذرانده شده در ترم Term GPA: معدل ترم

. . .

. . برای هر ترم تکرار می‌شود.

. . .

Total Registered: کل واحد ثبت نام شده Total Passed: کل واحد گذرانده شده GPA: معدل کل

Conditioned terms: تعداد ترمهای مشروطی

شکل ۶-۱۲ ب: قالب کلی کارنامه، خروجی پرسش ۶-۱۲.

***۶-۱۳- برنامهٔ ارائه شده در شکل ۶-۱۳ را دنبال کرده، نتیجهٔ اجرای برنامه را شامل تغییراتی که در مقادیر متغیرهای برنامه انجام می‌گیرد در قالب یک جدول و همچنین حاصل چاپ آن را در قالب جدولی دیگر مشخص نمایید. سپس برنامه را روی کامپیوتر اجرا کرده، پاسخ خود را با حاصل اجرا توسط کامپیوتر مقایسه کنید و در صورت وجود تفاوت، اشتباهات خود را مورد بررسی قرار دهید.**

***۶-۱۴- برنامهٔ ارائه شده در شکل ۶-۱۴ را دنبال کرده، نتیجهٔ اجرا و حاصل چاپ آن را در جدولهای مناسبی مشابه پرسش قبل مشخص نمایید. سپس برنامه را روی کامپیوتر اجرا کنید و پاسخ خود را با حاصل اجرا توسط کامپیوتر مقایسه کرده اشتباهات خود را درصورت وجود، مورد بررسی قرار دهید.**

```
#include <stdio.h>
#include <math.h>
#define DAH ۶
#define MGM(a) ((a) > ۰ ? (a) : (-(a)))

main()
{  int f, l, k, kl, m;
   float b;

   l = ۱.۲, k = ۳, b = ۵;
   m = k - MGM(b) *
       pow(pow(۲, k - l / ۲ * ۳ + DAH / ۶), k - (int)b / ۲) + ۰.۷۵;
   scanf("%i%d%*d%f%d",&l, &k, &b, &m);
   scanf("%*۲d٪۳f%*۴c٪li%*۳f٪ld", &b, &l, &m, &k);
   if (l = l - k > ۰)
   {  f = ۵;
      l = ۲;
      k = DAH + ۱;
   }
   else if (l < ۰)
   {  f = -۵;
      l = -۲;
      k = -۷;
   }
   else if (l == ۰)
   {  f = ۰;
      l = -۱;
      k = ۰;
   }
   for (kl = k;f <= k + l * ۳ - ۱; f += l * l)
      for (l = k; l <= l + ۷; l += b)
         kl++;
   b += ۰.۱۵;
   l -= ۳۰;
   printf("٪۳f٪۳i\n٪-۵d\n٪+۴d***% ۵i", b, l, m, k, f);
   printf("٪۵.۱f٪li٪۳f", b, f, b);
   switch (m + kl * (int) b - MGM(kl))
   {  case ۲:
      case ۳:
         k *= DAH / ۳;
         printf("٪۵d cases ۲ and ۳.\n", k);
      case ۵:
      {  l /= ۲;
         printf("٪۵d case ۵.\n", l);
         break;
      }
      default:
```

شکل ۶-۱۳: متن برنامهٔ مربوط به پرسش ۶-۱۳.

```
#include <stdio.h>
#include <math.h>
#define HAFT ۵
#define TRS(a, b) ((a) > (b) ? (a) : (b))

main()
{  int f = ۹, l, k, kا, m;
   float a, b;
   long n;

   b = m = ۴.۹۵, k = ۳۳, a = ۴.۵;
   l = k - TRS(a, b) *
       pow(m, k - (int)b / ۲ * ۱۵) + (float)HAFT / ۱۰;
   scanf("٪۴f%*f٪۴f%*d٪۳d٪۴d", &a, &b, &l, &m);
   switch (m / ۲ + (int)b)
   {  case ۲:
      case ۳:
         a *= HAFT / ۲;
         printf("Entered into cases ۲ and ۳ where a is:٪۴.۱f\n",a);
      case ۵:
      {  b += b <= ۲;
         printf("Entered into case ۵ where b is:%f\n", b);
         break;
      }
      default:
         printf("Default encountered.\n");
   }
   printf("٪۳f٪۳i\n٪-۵d\n٪+۴d***% ۲i and bee:٪-۵.۱f",
          a, l, m, -k, f, b);
   if (kا = l - k + ۲ا)
   {  f = ۵;
      k += HAFT / ۲;
   }
   else if (m + kا)
   {  f = -۵;
      k -= HAFT * ۲;
   }
   else if (l == ۰)
   {  f = ۰;
      l = -۱ + (k = ۰);
   }
   printf("٪۳d\n-٪۵d--٪-۵d---٪+۴d\n", k, kا, f, l);
   for (kا = k;f <= k - ۱; f += k * ۲)
      while (m + k >= b && l > f)
      {  l += b;
         k++;
         break;
      }
   b += ۰.۱۵;
   l -= ۳۰;
   printf("٪۳.۱f٪۳i\n\n٪+۴d***% ۵i\n", b, l, k, f);
   scanf("%*۲d٪۶e%*۴c٪۴li%*۳f٪۴G", &b, &n, &a);
   printf("٪۵.۱f٪۵lX%E", a, n, b * ۱۰);
   printf("\n EEHH!! ENGAR TAMOOM SHOD!!!\n");
   return ۰;
}
```

داده‌های ورودی:

```
-۲.۳  -۵۴.۴۵  ۱.۲  -۱+۱۲۳
-۱+۲۳E-۱DIGEXIA۱.۲۱e+۲
```


شکل ۶-۱۴: متن برنامهٔ مربوط به پرسش ۶-۱۴.