# اصول کامپیوتر ۱

# مبانی کامپیوتر و برنامه‌سازی

## «جلسه‌ی هجدهم»

**دانشکده‌ی علوم ریاضی - دانشگاه شهید بهشتی**

نیم‌سال اول ۹۰-۱۳۸۹

مدرس: سید علی کتان‌فروش

# پارامترهای مقداری و پارامترهای ارجاعی

**مثال ۱.** پیش زمینه‌ی موضوع.

**برنامه‌ای بنویسید که اعداد صحیح بین a و b را چاپ کند.**

```
int main( ) {
  int a,b;
  cout << "a b ? ";
  cin >> a >> b;
  int i;
  for( i=a+1; i<b; i++ )
    cout << i << endl;
  system("pause");
  return EXIT_SUCCESS;
}
```

# پارامترهای مقداری و پارامترهای ارجاعی

مثال ۱. پیش زمینه‌ی موضوع.

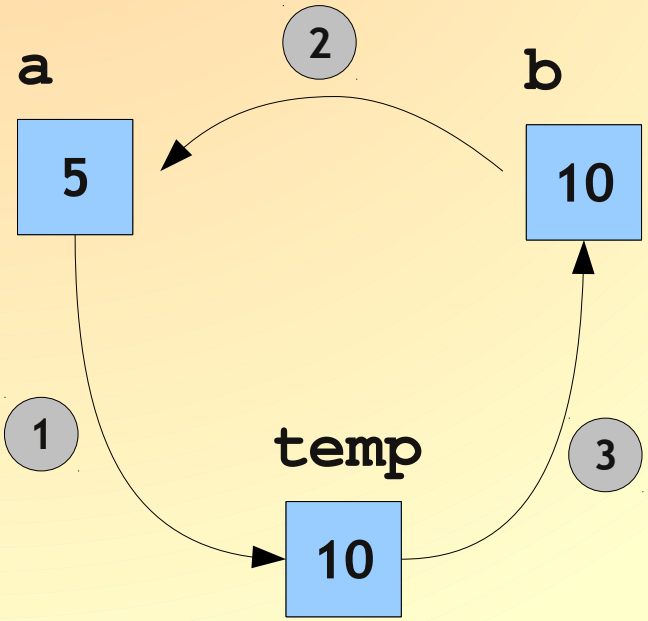
```
int main( ) {
   ....
   cin >> a >> b;
   if ( a > b )

      a ↔ b


   int i;
   for(i=a+1; i<b; i++ )
      cout << i << endl;
   ....
}
```

با حداقل تغییرات در برنامه، ترتیبی دهید تا چنانچه کاربر مقادیر a و b را به ترتیب عکس (a>b) وارد کرد برنامه باز همان خروجی پیشین را چاپ کند.

# پارامترهای مقداری و پارامترهای ارجاعی

**مثال ۱.** پیش زمینه‌ی موضوع: جابجا کردن مقدار دو متغیر.

```
a ↔ b
```




swap(a,b)

```
int temp = a;
a = b;
b = temp;
```

# پارامترهای مقداری و پارامترهای ارجاعی

**مثال ۱.** پیش زمینه‌ی موضوع: جابجا کردن مقدار دو متغیر.

```
int main( ) {
   ....
   if ( a > b ) {
      int temp = a;
      a = b;
      b = temp;
   }

   int i;
   for(i=a+1; i<b; i++ )
     ....
```

# پارامترهای مقداری و پارامترهای ارجاعی

**مثال ۱.**

**الگوریتم جابجا کردن مقدار دو متغیر را در قالب یک تابع بنویسید.**

```
int main( ) {
   ....
   if ( a > b )
     swap(a,b);

   int i;
   for(i=a+1; i<b; i++ )
     ....
```

# پارامترهای مقداری و پارامترهای ارجاعی

**مثال ۱.**

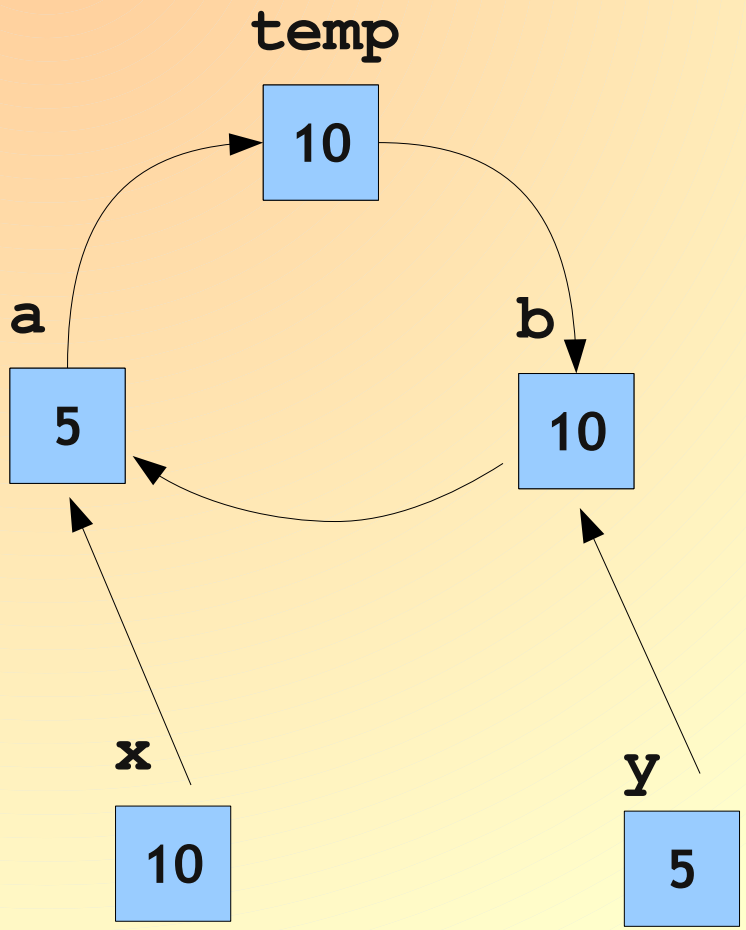
**الگوریتم جابجا کردن مقدار دو متغیر را در قالب یک تابع بنویسید.**

```
void swap( int a, int b ) {
   int temp = a;
   a = b;
   b = temp;
}

int main( ) {
   ....
```

تابع، مقدار متغیرهای a و b (پارامترهای صوری) را جابجا می‌کند اما مقدار متغیرهای x و y (پارامترهای واقعی) بدون تغییر باقی می‌ماند.

```
void swap( int a, int b ) {
   int temp = a;
   a = b;
   b = temp;
   cout <<a<<' '<<b<<endl; // debugging
}

int main( ) {
   int x,y;
   cin >> x >> y;
   swap(x,y);
   cout <<x<<' '<<y<<endl;
   ....
```

# پارامترهای مقداری و پارامترهای ارجاعی




```
void swap(int a,int b){
   int temp = a;
   a = b;
   b = temp;
}
```

تابع درواقع کپی مقادیر متغیرهای واقعی را جابجا می‌کند و مقادیر اصلی بدون تغییر باقی می‌ماند.

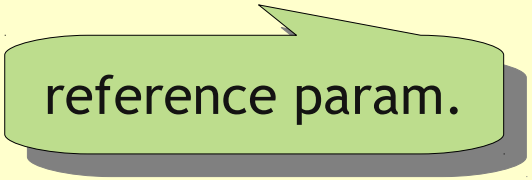
```
int x = 10;
int y = 5;
swap(x,y);
```

# پارامترهای مقداری و پارامترهای ارجاعی

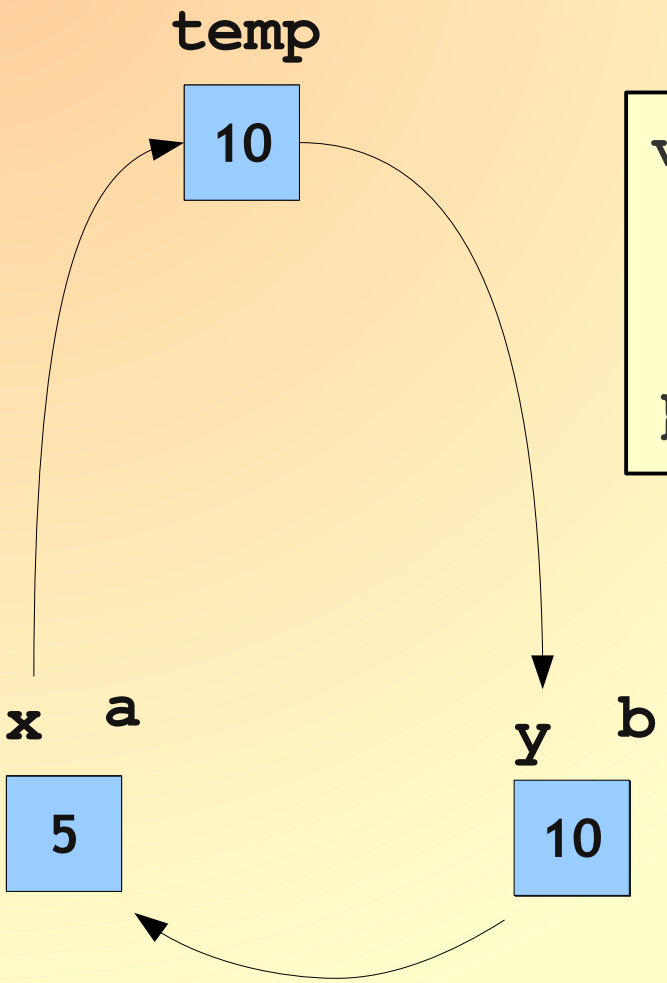
- در تعریف تابع، پارامترهای صوری را به دو صورت می‌توان تعریف کرد.
  - value parameterپارامتر مقداری. همان نوعی است که تا کنون پارامترهای تابع را تعریف می‌کردیم. با فراخوانی تابع، مقدار پارامتر واقعی در پارامتر مقداری کپی می‌شود
  - reference parameterپارامتر ارجاعی. علامت & جلوی نوع داده‌ای پارامتر، یک پارامتر ارجاعی تعریف می‌کند. با فراخوانی تابع، پارامتر ارجاعی متغیری می‌شود برای همان حافظه‌ای که پارامتر واقعی آنرا نامگذاری کرده است. از این رو، هر تغییراتی که توسط تابع بر روی پارامتر ارجاعی صورت می‌گیرد در پارامتر واقعی نیز دیده می‌شود

# پارامترهای مقداری و پارامترهای ارجاعی

```
void swap( int &a, int &b ) {
   int swap = a;
   a = b;
   b = temp;
}

int main( ) {
   int x,y;
   x = 10;
   y = 5;
   swap(x,y);
   ....
```



# پارامترهای مقداری و پارامترهای ارجاعی




```
void swap(int &a,int &b){
   int temp = a;
   a = b;
   b = temp;
}
```

```
int x = 10;
int y = 5;
swap(x,y);
```

# پارامترهای مقداری و پارامترهای ارجاعی

- توجه کنید که برای فراخوانی یک تابع، استفاده از مقادیر ثابت یا نتیجه‌ی محاسبات برای ارسال به عنوان پارامترهای ارجاعی، از نظر قراردادهای زبان برنامه‌نویسی C/C++ خطا و از نظر منطقی نیز بی‌معنی است.

```
void swap( int &a, int &b ) {
   int temp = ....
   ....
}

int main( ) {
   ....
   swap(10,5);   // Error
   ....
```

# پارامترهای مقداری و پارامترهای ارجاعی

مثال ۲. تابعی که کاراکتر را به شکل بزرگ حرف تبدیل می‌کند.

```
void toUpper ( char &c ) {
  if ( 'a'<=c && c<='z' )
    c = c + 'A' - 'a';
}
```

مثال ۲. تابعی که کاراکتر را به شکل بزرگ حرف تبدیل می‌کند.

به تفاوت نحوه‌ی تعریف و فراخوانی این تابع و مثال جلسه‌ی قبل توجه کنید.

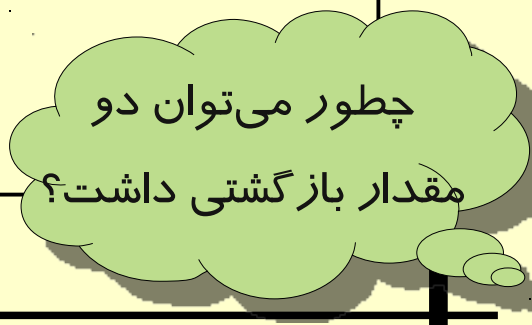
```
void toUpper ( char &c ) {
  if ( 'a'<=c && c<='z' )
    c = c + 'A' - 'a';
}

int main( ) {
  char c = 't';
  toUpper(c);
  cout << c << endl;
  ....
}
```

```
char upperCase ( char c ) {
  if ( 'a'<=c && c<='z' )
    c = c + 'A' - 'a';
  return c;
}

int main( ) {
  cout << upperCase('t')
       << endl;
  ....
}
```

# پارامترهای مقداری و پارامترهای ارجاعی

**مثال ۳. تابعی که محیط و مساحت مثلث را محاسبه می‌کند.**

```
??? measureTriangle
      ( double a, double b, double c )
{
  double p = (a+b+c)/2;
  double s = sqrt(p*(p-a)*(p-b)*(p-c));
  return (2*p,s); // Wrong!
}
```



# پارامترهای مقداری و پارامترهای ارجاعی

در زبان C/C++، نمی‌توان توسط یک تابع، بیش از یک مقدار را گزارش کرد.

- رویکرد جایگزین، استفاده از پارامترهای ارجاعی است.

راه حل ۱

```
double measureTriangle
       ( double a, double b, double c,
         double &p )
{
  p = a+b+c;
  double z = p/2;
  return sqrt(z*(z-a)*(z-b)*(z-c));
}
```

# پارامترهای مقداری و پارامترهای ارجاعی

مثال ۳. تابعی که محیط و مساحت مثلث را محاسبه می‌کند.

راه حل ۲

```
void measureTriangle
     ( double a, double b, double c,
       double &p, double &s )
{
  p = a+b+c;
  double z = p/2;
  s = sqrt(z*(z-a)*(z-b)*(z-c));
}
```

# پارامترهای مقداری و پارامترهای ارجاعی

مثال ۳. تابعی که محیط و مساحت مثلث را محاسبه می‌کند.

به تفاوت نحوه‌ی فراخوانی توابعی که در بالا تعریف کردیم توجه کنید.

راه حل ۱

```
double p;
cout << "Area = "
     << measureTriangle(3,4,5,p) << endl;
cout << "Perimeter = " << p << endl;
```

راه حل ۲

```
double s,p;
measureTriangle(3,4,5,p,s);
cout << "Area = " << s << endl;
cout << "Perimeter = " << p << endl;
```

# پارامترهای مقداری و پارامترهای ارجاعی

- توجه کنید که در چنین کاربردهایی، پارامترهای ارجاعی نقش ظرفی را ایفا می‌کنند که نتایج از طریق آن‌ها به برنامه‌ی فراخواننده گزارش می‌شوند.

```
void measureTriangle
     ( double a, double b, double c,
       double &p, double &s )
{
  p = a+b+c;
  double z = p/2;
  s = sqrt(z*(z-a)*(z-b)*(z-c));
}
```

input arguments

output arguments

# پارامترهای ارجاعی

**مثال۴. تابعی که (روز،ماه) را یک روز به جلو برد.**

```
void nextDay( int &d, int &m ) {
  d++;
  if( m < 7 ){
    if(d==32) { d=1; m++; }
  } else if( m < 12 ) {
    if(d==31) { d=1; m++; }
  } else { // m == 12
    if(d==30) { d=1; m=1; }
  }
}
```

## مثال۴. تابعی که (روزِ هفته، روز، ماه) را یک روز به جلو برد.

```
void nextDay( int &wd, int &d, int &m ) {
  wd = (wd+1) % 7;
  d++;
  if( m < 7 ){
    if(d==32) { d=1; m++; }
  } else if( m < 12 ) {
    if(d==31) { d=1; m++; }
  } else { // m == 12
    if(d==30) { d=1; m=1; }
  }
}
```

## چرا برنامه را با استفاده از تعریف توابع جدید توسعه می‌دهیم؟

- توسعه‌ی نرم‌افزارهای بزرگ نیازمند پیاده‌سازی الگوریتم‌های متعدد است.
- یکی از رویکردهای رایج برای این کار، شکستن برنامه به زیربرنامه‌های مستقل و پیاده‌سازی الگوریتم اصلی برنامه بر پایه‌ی فراخوانی توابع است.
- مستقل بودن یک تابع از دیگر اجزاء برنامه بدین معنی است که می‌توان مسأله‌ی مشخصی نام برد که الگوریتم حل آن مسأله، در این تابع پیاده‌سازی شده باشد.
- برنامه با استفاده از فراخوانی توابع و رد و بدل کردن پارامترها و مقادیر بازگشتی بین آن‌ها ساخته می‌شود.

## چرا برنامه را با استفاده از تعریف توابع جدید توسعه می‌دهیم؟

- رویکرد استفاده از توابع، این امکان را فراهم می‌کند که بتوان برنامه‌نویسی یک یا تعدادی چند از توابع را به طور مستقل بر عهده‌ی یک فرد از یک تیم از برنامه‌نویسان قرار داد.
- برنامه‌نویس بدون نگرانی درباره‌ی دیگر بخش‌های برنامه، تمرکز بیشتری برای پیاده‌سازی هر یک از توابع دارد؛ یعنی کاهش خطا در برنامه‌نویسی.
- با این روش، بخش عمده‌ای از مراحل تولید نرم‌افزار را می‌توان بطور موازی پیش برد و در عین حال، با پایان مراحل برنامه‌نویسی هر گروه از تابع، بخش قابل استفاده‌ای از برنامه تکمیل شده است.

## چرا برنامه را با استفاده از تعریف توابع جدید توسعه می‌دهیم؟

- پیش از بکارگیری توابع در برنامه‌ی اصلی، هر تابع را می‌توان به طور مستقل، تست و خطایابی کرد. بطور کلی، جستجوی منشاء خطا در برنامه‌های مبتنی بر طراحی و استفاده از توابع، ساده‌تر است.
- الگوریتم حل یک مسأله را می‌توان یکبار در قالب یک تابع پیاده‌سازی کرد و آنرا بارها در برنامه‌ی متفاوت بکار گرفت.

# توسعه‌ی نرم‌افزار به روش طراحی از بالا به پایین

# top-down scheme

- طرح کلی برنامه را بدون وارد شدن به جزئیات الگوریتم و با فرض اینکه توابعی برای اجرای دستورات سطح بالا وجود دارند در تابع main پیاده‌سازی کنید.
- به نوبت، هر یک از توابعی را که در برنامه‌ی main مفروض انگاشته‌اید برنامه‌نویسی کنید البته، رویکردی را که برای طراحی تابع main بکار برده‌اید در اینجا نیز بکار گیرید؛ یعنی بدون نگرانی درباره‌ی جزئیات الگوریتم، تابع را بر پایه‌ی استفاده از فراخوانی دیگر توابعِ ساده‌تر پیاده‌سازی کنید.

# توسعه‌ی نرم‌افزار به روش طراحی از بالا به پایین

## top-down scheme

**مثال ۱. برنامه‌ای که سه‌تائی‌های فیثاغورثی تحویل ناپذیر را چاپ کند.**

- **سه‌تائی (a,b,c) را تحویل‌پذیر می‌گوئیم اگر $k>1$ وجود داشته باشد که هر سه عدد بر آن بخشپذیر باشد.**
- **می‌توان نشان داد سه‌تائی (a,b,c) تحویل ناپذیر است اگر $gcd(a,b,c)=1$.**

## مثال ۱. برنامه‌ای که سه‌تائی‌های فیثاغورثی تحویل ناپذیر را چاپ کند.

```
int main ( ) {
  int n;
  cout << "n ? "; cin >> n;
  int a,b,c;
  for ( a=1; a<=n; a++ )
    for ( b=1; b<a; b++ )
      for ( c=1; c<b; c++ )
        if ( gcd(a,b,c)==1
             && pythagorean(a,b,c) )
          printTriple(a,b,c);
  system("pause");
  return EXIT_SUCCESS;
}
```

## مثال ۱. برنامه‌ای که سه‌تائی‌های فیثاغورثی تحویل ناپذیر را چاپ کند.

```
int gcd2 ( int a, int b ) {
  int r;
  do {
    r = a%b;
    a = b;
    b = r;
  } while (r>0);
  retur a;
}

int gcd ( int a, int b, int c ) {
  return gcd2( a, gcd2(b,c) );
}
```

تابع محاسبه‌ی بزرگترین مقسوم علیه مشترک.

## مثال ۱. برنامه‌ای که سه‌تائی‌های فیثاغورثی تحویل ناپذیر را چاپ کند.

```
bool pythagorean ( int a, int b, int c) {
  retur a*a==b*b+c*c;
}
```

تابع تست فیثاغورثی بودن سه‌تائی.

```
void printTriple ( int a, int b, int c) {
  cout << '(' << a << ','
       << b << ',' << c
       << ')' << endl;
}
```

تابع چاپ سه‌تائی.

## مثال ۱. برنامه‌ای که سه‌تائی‌های فیثاغورثی تحویل ناپذیر را چاپ کند.

به سادگی می‌توان اثبات کرد سه‌تائی فیثاغورثی تحویل‌ناپذیر است اگر و تنها اگر gcd(b,c)=1.

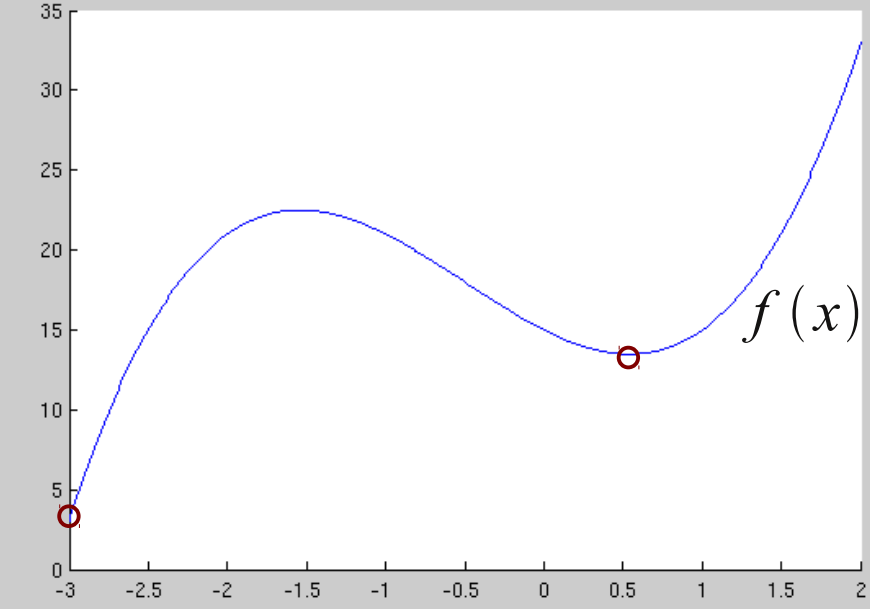
```
int main ( ) {
   ....
   for ( a=1; a<=n; a++ )
      for ( b=1; b<a; b++ )
         for ( c=1; c<b; c++ )
            if ( gcd2(b,c)==1
                  && pythagorean(a,b,c) )
               printTriple(a,b,c);
   ....
}
```

# توسعه‌ی نرم‌افزار به روش طراحی از بالا به پایین

## top-down scheme

**مثال ۲. برنامه‌ای که مینیمم چندجمله‌ای درجه سوم را بدست آورد.**

**می‌خواهیم مینیمم تابع را در بازه‌ی [l,u] بدست آوریم.**




$$f(x) = a\,x^3 + b\,x^2 + c\,x + d$$

$$l = -3$$
$$u = 2$$

## مثال ۲. برنامه‌ای که مینیممِ چندجمله‌ای درجه سوم را بدست آورد.

```
int main ( ) {
  double a,b,c,d;
  readCoefficient(a,b,c,d);
  double l,u;
  readLimits(l,u);
  double xMin, yMin;
  findMin(a,b,c,d,l,u,xMin,yMin);
  cout << "xMin = " << xMin << endl;
  cout << "yMin = " << yMin << endl;
  system("pause");
  return EXIT_SUCCESS;
}
```

## تابع تعیین مینیمم چندجمله‌ای درجه سوم در بازه‌ی [l,u].

```
void findMin(double a, double b, double c,
             double d, double l, double u,
             double &xMin, double &yMin )
{
  double x1,x2;
  if( solveQuadEq(3*a,2*b,c,x1,x2) ) {
    if( 3*a*x1+b > 0 )
      xMin = x1;
    else
      xMin = x2;
  }
  else // there is not any real root
    xMin = l;
  ....
```

## ... تعیین مینیمم چندجمله‌ای درجه سوم در بازه‌ی [l,u] (ادامه).

```
  ....

  if ( xMin<l || u<xMin )
    xMin = l;
  yMin = cubic(a,b,c,d,xMin);

  double y;
  if ( xMin != l ) {
    y = cubic(a,b,c,d,l);
    if ( y<yMin ) { xMin=l; yMin=y; }
  }
  y = cubic(a,b,c,d,u);
  if ( y<yMin ) { xMin=u; yMin=y; }
}
```

# توسعه‌ی نرم‌افزار به روش طراحی از بالا به پایین

## top-down scheme

**تمرین. برای تکمیل برنامه‌ی مینیممِ چندجمله‌ای درجه سه، توابع زیر را توسعه دهید.**

```
readCoefficient(a,b,c,d);
readLimits(l,u);
solveQuadEq(a,b,c,x1,x2);
cubic(a,b,c,d,x);
```

# توسعه‌ی نرم‌افزار به روش طراحی از بالا به پایین

# top-down scheme

**مثال ۳. برنامه‌ای که تقویم سال را چاپ کند.**

برنامه‌ای که تقویم سال را چاپ کند.

```
int main( ) {
  int wd,d,m;
  wd = firstDayOfYear(1389);
  d = m = 1;
  do{
    if(d==1){
      cout << monthName(m) << endl;
      printSpace(3*wd);
    }
    cout << setw(2) << d;
    if(wd==6)  cout << endl;
    else       cout << ' ';
    if(nextDay(wd,d,m)) // if(end of month)
      cout << endl << endl;
  }while(d!=1 || m!=1);

  system("pause");
  return EXIT_SUCCESS;
}
```

# توسعه‌ی نرم‌افزار به روش طراحی از بالا به پایین

## top-down scheme

تمرین. برای تکمیل برنامه‌ی چاپ تقویم سال، توابع زیر را توسعه دهید.

```
firstDayOfYear(y);
monthName(m);
printSpace(n);
nextDay(wd,d,m);
```