SubAlgorithms in General
Data Oriented Programming

DOP or Data Oriented Programming is a combination of
Data Flow Programming and Data Space Programming
DOP = DFP + DSP

WHAT:
Subs, are Sub-Algorithms or Sub-Programs, which are also
called procedures, functions, routines, subroutines and methods.
Subs are ways of packaging many parts into one whole, a component.

WHY: Wholes vs parts, the 3 Rs.
Subs are useful for many reasons:
- ReUsability: the ability to use often, to share and possibly sell,
- ReLiability: the confidence of having safe and sure component to build on,
- Readability: the ability to have clear and clean understanding

HOW:
SubPrograms are of 2 kinds:
1. Functions, also called type methods
2. Routines, also called void methods
Functions return a value; Routines do some action.
Max3 problem:
Create an algorithm to find the maximum value of 3 values a, b, c.
For example, the maximum value of 3, 6, 5 is 6.
Then "grow it" to find the max of more values (say 4, or 7).

Many ways are possible to do this,
(like the 3 pigs building a house)
Using different kinds of parts or "components"

First Way: (worst):
with *Straws* (conditions, actions, etc)

Second Way:
with *Sticks* (Choices or Ifs)

Third way:
with *Bricks* (bigger components)
Create a max3 from max2 components.

First way is at a low level, dealing with details,
Last way is at a high level, dealing with abstractions.
**Max3** components can be created from **max2** components using the data flow diagrams, as shown, where $g$ is the greatest of $a$, $b$, $c$.

Note that a larger component (max3) was created by reusing smaller components (max2s) which have not been created yet!!

It is not necessary to know the details of a component in order to be able to use it. That’s top-down vs bottom up.
Max3 at many levels: high, mid and low

Max2 has been defined once and used or reused twice.
**max4s** can be made from some **max2s** similarly

**max7s** can be made from some **max3s**
(which were made from **max2s**)

So **max2** could be useful as a component, a building block.
Coding components: Defining vs Using & reUsing

**Definition** of maximum in general is \( \text{max2}(p,q) \) where \( p \) and \( q \) are slots which are also called parameters or arguments.

**Use or reUse** of max2 can be done, or invoked or called, in many ways such as the following:

<table>
<thead>
<tr>
<th>Max2 ((p,q))</th>
</tr>
</thead>
<tbody>
<tr>
<td>If ((p &lt; q)) then</td>
</tr>
<tr>
<td>set (\text{max} = q)</td>
</tr>
<tr>
<td>Else</td>
</tr>
<tr>
<td>set (\text{max} = p)</td>
</tr>
<tr>
<td>EndIf</td>
</tr>
<tr>
<td>Set result = max</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{z} &= \text{max2}(x, y) \\
\text{bigest} &= \text{max2}(\text{first}, \text{second}) \\
\text{oldest} &= \text{max2}(\text{age1}, \text{age2}) \\
\text{longer} &= \text{max2}(\text{lenA}, \text{lenB}) \\
\text{faster} &= \text{max2}(65, \text{speed}) \\
\text{smarter} &= \text{max2}(\text{myIQ}, \text{yoIQ})
\end{align*}
\]

**Formal slots** such as \( p, q \) occur in the definition, the internal view.

**Actual slots** such as \( x, y \) etc occur in the use or call of max, the external view

Define once; reUse often.
Slots must match:
the formal slots of the definition, as in \( \text{max2}(p, q) \) must match the actual slots of the use, as in \( \text{max2}(x, y) \).

Matching must be
in number (2 slots each),
in order (first and second), and
in type (both same: ints, reals, etc).

Passing of values, such as 5 and 7 below,
is done from the actual slots \( x \) and \( y \)
to the formal slots \( p \) and \( q \), and
the expression \( \text{max2}(x,y) \) has the resulting value.

\[ \begin{align*}
x & \quad y \\
5 & \quad 7 \\
\hline
p & \quad q \\
\text{max2} & \\
7 & \\
\hline
z &
\end{align*} \]
Connecting components

The graphic interconnection of components is done in pseudoCode by involving some temporary boxes, such as t below, and using the Various boxes within the slots, as show to the right.

Also, it is possible to combine both into one as:

\[ g = \text{max2} (\text{max2}(a,b), c) \]
**max4s** can be make from **max2s** in two different ways, as shown

First: Top right **max4a**

\[
\begin{align*}
e &= \text{max2}(a, b) \\
f &= \text{max2}(c, d) \\
g &= \text{max2}(e, f)
\end{align*}
\]

Second: Bottom left **max4b**:

\[
\begin{align*}
i &= \text{max2}(a, b) \\
j &= \text{max2}(i, c) \\
h &= \text{max2}(j, d)
\end{align*}
\]

Each of these can be written as a single line.

\[
\begin{align*}
g &= \text{max2}(\text{max2}(a, b), \text{max2}(c, d)) \\
h &= \text{max2}(\text{max2}(\text{max2}(a, b), c), d)
\end{align*}
\]
Max5 can be made from smaller maxes: many ways

Now you write the code for these, and then do max6, max7, max8 and max9 in many ways.
**Mid3**, the mid value of any 3 different values can be made from maxes and mins as shown below.

This result is not obvious, so it is proved by the table at the right.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>a &lt; b &lt; c</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>a &lt; c &lt; b</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>b &lt; a &lt; c</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>b &lt; c &lt; a</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>c &lt; a &lt; b</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>c &lt; b &lt; a</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Show that mid3 can be done in another way with maxes and mins.
**DSP: Data Space Programming, an internal view**

Sub-programs, in general, may have many internal parts as shown.

SubProgram `max2` is shown below with all its detail.
**General View:** at 3 levels from high to low levels

1. **DFD:**
   Data Flow Diagram

2. **DSD:**
   Data Space Diagram

3. **FBD:**
   Flow Block Diagram

Defined as \( r = \text{name}(x,y,z) \) formally (internal)
and used as \( d = \text{name}(a,b,c) \) actually (external)
where \text{name} could be max3, min3, mid3, maj3, mean3, etc
SubPrograms may be of two types: functions and routines.

Functions, also known as type methods, return a value,

Routines, also known as void methods, do some action.

Functions often have a name which is a noun such as:

max2(x, y) which returns the maximum value of x, y

circleArea(d) which returns the area of a circle
  given its diameter, not the radius

isIsosceles(a, b, c) which returns true or false
  depending on whether angles a, b, c make a triangle

Routines have a name which is a verb such as:

FormatDollar(amount) which prints money amounts properly

SpellOut(digit) which outputs a digit in English (pass in 2, get out “two”)

OutRow(num, str) which prints a row of n occurrences of string str

DrawBox(len, wid, chr) which draws a box of String characters chr
**triArea** is a function, or type method, that uses Hero’s formulas to find the area of any triangle, given the three sides. Note that the boxes are drawn as dotted lines since values are reals.

Call, or invoking or using can be done for example as:

```plaintext
p = triArea (a,b,c)  
q = triArea (3.0, 4.0, 5.0)  
r = triArea (x, 7.0, (x+y)/2.0)
```

The slots may have boxes (variables), constants, or even other formulas, which are evaluated before passed.
**outRow**(n, S) is a routine, or **void method**, that prints out a row consisting of n repetitions of a string S. Note that no value is passed out; only the row of characters is printed.

The slots may have boxes (variables), constants, or even other formulas, which are evaluated before passed.
Problems on Function SubPrograms in General

1. Worth
Create a subProgram `worth(p,n,d,q,h)` which computes the total worth of any number of coins (p pennies, n nickels, etc, h half dollars).

2. Mid3
Show in another way how the mid value of 3 values can be computed using only `max2` and `min2` components, and various arithmetic operations.

3. More Max2
The maximum value of two numbers `a, b` can be determined from the absolute value function `abs(n)` by adding and subtracting various differences. Show this as a data flow diagram.

4. Quadrant
Create a subAlgorithm `quad(x,y)` that indicates which quadrant the point `(x,y)` falls into (1,2,3,4). If the point falls on an axis the value is 0.
Problems on Routine SubPrograms in General

1. SpellOut
Create a subProgram spellOut(d) which spells out whatever digit is passed in. For example spellOut(2) prints out “two”, spellout(6) prints “six”, but spellout(13) prints out 13.

2. FormatDollars
Create a routine to convert a given real amount of money into a proper amount, rounding it off to the nearest cent.

3. MoneyChanger
Create a routine to accept an integer number of cents, and outputs the corresponding number of pennies, nickels, and quarters.

4. RomanNums
Create a routine to accept an integer and print out the corresponding Roman number.

5. RomanStrings
Create a function to accept an integer and to return a string of the corresponding Roman number.