IPC144
Session 3
Problem Analysis

Objectives:

By the end of this session, the student will be able to:

- **Explain the purpose of the IPO Chart**
- **List which columns of an IPO Chart contain verbs, which contain nouns**
- **Represent a word problem in an IPO Chart**
- **Explain the purpose of a Hierarchy Chart**
- **Develop a Hierarchy chart for a word problem**
- **Review Module Cohesion**

PDC - Define the Problem

PDC - Define the Problem

From the previous session, the first step in developing a Program was described as:

- Must have a clear understanding of the problem
- There is no room for assumptions
- There is no difference between solving the wrong problem correctly and solving the right problem incorrectly
- To help with analysis, the problem should be broken into:
  - Inputs
  - Outputs
  - Processes
- IPO Charts are a tool used

IPO Charts

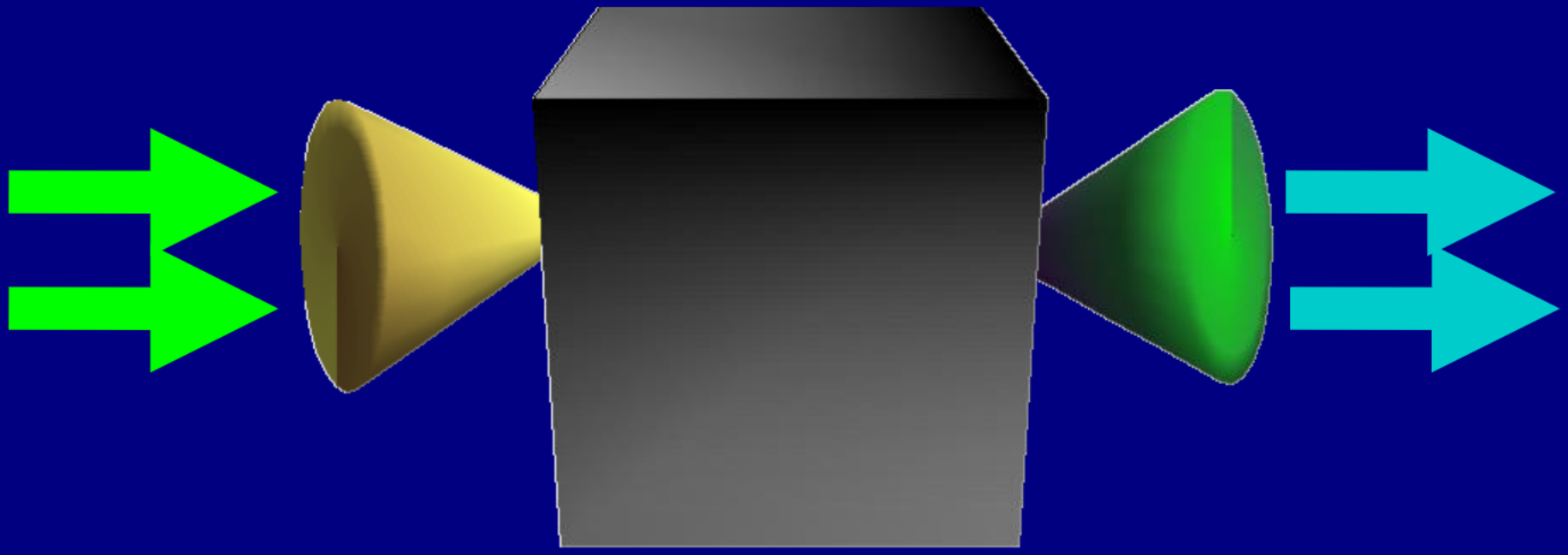
IPO Charts

If we have an understanding of what the input will be, and what the output will be, we will have a better chance of determining what needs to be done to transform input into output.

IPO Charts, continued

IPO Charts, continued

The software that we write is best thought of as a 'Black Box'. There will be data feeding into it, and information flowing out of it.



We can list tasks that must take place in the 'Black Box' in order to facilitate the transformation of input to output.

IPO Charts, continued

IPO Charts, continued

Step 1 of the Program Development Cycle referred to a tool called an IPO Chart. It is also known as a Defining Diagram. IPO is an acronym for Input Process Output. It is a technique to break a problem down into one of these three categories using a chart such as:

Input	Process	Output

The entries in the Input and Output columns are nouns. A distinct, singular item. The entries in the Process column are verbs. A specific, singular action.

IPO Charts, continued

IPO Charts, continued

Example:

Develop a program to accept three numbers, add them together, then print their total and average.

What would an IPO Chart look like?

Input	Process	Output

IPO Charts, continued

IPO Charts, continued

Example:

Develop a program to accept three numbers, add them together, then print their total and average.

What would an IPO Chart look like?

Input	Process	Output
val1	Read val1, val2, val3	total
val2	Calculate total	average
val3	Calculate average	
	Display total	
	Display average	

IPO Charts, continued

IPO Charts, continued

Why is the following chart wrong?

Input	Process	Output
3 numbers	Read 3 numbers	(n1 + n2 + n3) / 3
	Calculate total and average	n1 + n2 + n3
	Display total, average	
	Display total	

IPO Charts, continued

IPO Charts, continued

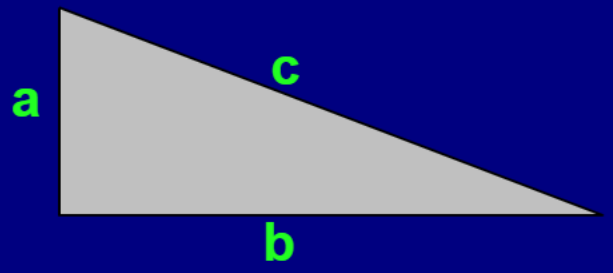
Example:

Develop a program that displays the area and perimeter of a triangle when only two sides are given. The two sides will be 'a' and 'b' from the diagram below.

$c^2 = a^2 + b^2$

area = 0.5 x (a x b)

What would an IPO Chart look like?



IPO Charts, continued

IPO Charts, continued

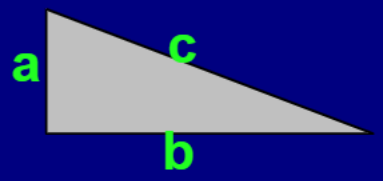
Example:

Develop a program that displays the area and perimeter of a triangle when only two sides are given. The two sides will be 'a' and 'b' from the diagram below.

$c^2 = a^2 + b^2$

area = 0.5 x (a x b)

What would an IPO Chart look like?



Input	Process	Output
sideA	Read sideA, sideB	area
sideB	Calculate sideC	perimeter
	Calculate perimeter	
	Calculate area	
	Display perimeter	
	Display area	

IPO Charts, continued

IPO Charts, continued

In order to determine the maximum range available to an aircraft flying a specific direction, the fuel on board becomes the limiting factor. While flying the assigned heading, the aircraft may experience a headwind or a tailwind. Headwinds will be indicated by negative numbers, tailwinds by positive numbers.

The cruise performance chart for the aircraft provides the airspeed and rate of fuel consumption for various altitudes and power settings. The pilot is responsible for selecting the altitude & power setting from the performance chart. Based on the values selected by the pilot from the cruise performance chart, calculate and display the actual speed of the aircraft over the ground and the maximum range of the aircraft. For maximum range, the aircraft is permitted to expend all but 15% of the starting fuel capacity.

Ground speed = Cruise airspeed + wind
Time aloft = (0.85 * fuel) / rate of consumption
Maximum range = Ground Speed * Time aloft

IPO Charts, continued

In order to determine the maximum range available to an aircraft flying a specific direction, the fuel on board becomes the limiting factor. While flying the assigned heading, the aircraft may experience a headwind or a tailwind. Headwinds will be indicated by negative numbers, tailwinds by positive numbers.

The cruise performance chart for the aircraft provides the airspeed and rate of fuel consumption for various altitudes and power settings. The pilot is responsible for selecting the altitude & power setting from the performance chart. Based on the values selected by the pilot from the cruise performance chart, calculate and display the actual speed of the aircraft over the ground and the maximum range of the aircraft. For maximum range, the aircraft is permitted to expend all but 15% of the starting fuel capacity.

Ground speed = Cruise airspeed + wind
Time aloft = (0.85 * fuel) / rate of consumption
Maximum range = Ground Speed * Time aloft

Input	Process	Output
fuelQty	Read fuelQty, rateFuel, wind, airspeed	maxRange
rateFuel	Calculate groundspeed	groundSpeed
wind	Calculate timeAloft	
airspeed	Calculate maxRange	
	Display groundSpeed	
	Display maxRange	

IPO Charts, continued

IPO Charts, continued

Example:

Calculate the gross pay and net pay for an employee. Display each deduction, total deduction, gross pay and net pay for the employee. Overtime is calculated at 1.5 times the normal rate for hours worked in excess of 44 hours/week. Long term benefits are charged at $2.00 per every full $100.00 of gross weekly earnings to a maximum payment of $100.00 per week. Unemployment insurance is 1.3% of gross pay to a maximum of $11.40 per week. Canada Pension Plan is 1.6% of gross pay on all earnings below $700 per week. Federal tax is calculated at 17% on the first $20,000 earned annually, 23% on the next $20,000 earned annually and 29% on the remainder. Provincial tax which is calculated with the Federal tax is 49% of Federal tax collected.

IPO Charts, continued

IPO Charts, continued

Input	Process	Output
payRate	Read payRate, hoursWorked, earningsToDate	grossPay
hoursWorked	Calculate normalWeek	netPay
earningsToDate	Calculate overtime	benefits
	Calculate grossPay	UI
	Calculate benefits	CPP
	Calculate UI	fedTax
	Calculate CPP	provTax
	Calculate fedTax	ttlDeduct
	Calculate provTax	
	Calculate ttlDeduct	
	Calculate netPay	
	Display grossPay, netPay, benefits, UI, CPP, fedTax, provTax, ttlDeduct	

IPO Charts, continued

IPO Charts, continued

Create a four-function calculator. The program will prompt the user to enter three values. The first value entered is a value that tells the program what operation to perform, the second and third values are the values to be used in the calculation. The possible values that determine the operation to be performed are: 'A' for Add, 'S' for subtract, 'M' for multiply, 'D' for divide, and 'Q' to quit the program. After reading the input values, the program will perform the required operation, and display the output in the form:

x operation y = result

Where *x* and *y* are the two numbers entered, *operation* is a full description of the operation performed to make a proper sentence (i.e. 'plus', 'minus', 'multiplied by' and 'divided by'), and *result* is the result of the calculation.

IPO Charts, continued

IPO Charts, continued

Input	Process	Output
operation	Read operation, val1, val2	val1
val1	Calculate the sum of val1, val2	phrase
val2	Calculate difference of val1 and val2	val2
	Calculate product of val1, val2	result
	Calculate quotient of val1, val2	
	Determine phrase	
	Display val1, phrase, val2, result	

IPO Charts, continued

IPO Charts, continued

The members of the Board of a small university are considering voting for a pay increase for their 25 faculty members. They are considering a pay increase of 8%, but before doing so they want to know how much this pay increase will cost. Design a program which will prompt for and accept the current salary for each of the faculty members, and then calculate and display their individual pay increases. At the end of the program, you are to print the total faculty payroll before and after the pay increase, and the total pay increase involved.

IPO Charts, continued

IPO Charts, continued

The members of the Board of a small university are considering voting for a pay increase for their 25 faculty members. They are considering a pay increase of 8%, but before doing so they want to know how much this pay increase will cost. Design a program which will prompt for and accept the current salary for each of the faculty members, and then calculate and display their individual pay increases. At the end of the program, you are to print the total faculty payroll before and after the pay increase, and the total pay increase involved.

Input	Process	Output
currentPay	Read currentPay	indIncrease
	Calculate indIncrease	ttlCurrPay
	Calculate ttlNewPay	ttlNewPay
	Calculate ttlCurrPay	ttlIncrease
	Calculate ttlIncrease	
	Display indIncrease, ttlCurrPay, ttlNewPay, ttlIncrease	

Modularity and Hierarchy Charts

Program Development Cycle

PDC: Outline the Solution

- Break problem into smaller tasks or steps
- Create a rough draft of the solution, may include
- Major processing steps
  - Major sub-tasks (if any)
  - Major control structures (repetition loops)
  - Major variables and record structures
  - Mainline logic of the program
- Top-down design
- Modularity and Hierarchy charts are tools used

Modularity

Modularity

If programmers start coding at the beginning of a problem and work systematically through each step until reaching the end, it is inevitable that they will get bogged down in the intricacies of a particular part of the problem, and will loose focus on the solution as a whole.

To avoid this problem:

- **Outline a general solution to the problem**
- **Break the general solution down into smaller steps**
- **Repeat until the most detailed levels are complete**

This 'functional decomposition' of a problem is referred to as 'Top-Down Development'

Modularity, continued

Modularity, continued

Referring to the PDC, the break down of a problem would consist of the points:

- **Major processing steps**
- **Major sub-tasks (if any)**
- **Mainline logic of the program**

Each of the Major Processing Steps, Major Subtasks and Mainline logic form Modules

A module is a small program within a larger program that performs a task (a single task if possible).

- **Look at how the menu of an application is laid out – you have general menu choices that are subdivided into smaller menu choices , which may be subdivided again.**
- **Consider a PC – it is made up of modules that perform single functions – keyboard, mouse, monitor, hard drive, memory, CPU…**

There will be one main control module that serves as the starting point of the program.

The Main Control Module is responsible for invoking or calling other modules as needed.

Modularity, continued

Modularity, continued

Each module is a segment of logically related code, a part of the complete program, that gets executed to solve a problem.

As much as possible, each module should be *independent* of all other modules.

It should constitute a logical unit of work, performing one function of the overall problem solving task.

Well designed modularity results in reusable blocks of code that can be used many times within your program – or used by other programs.

It is a little like the creating a program from pre-defined building blocks (like Lego® Blocks)

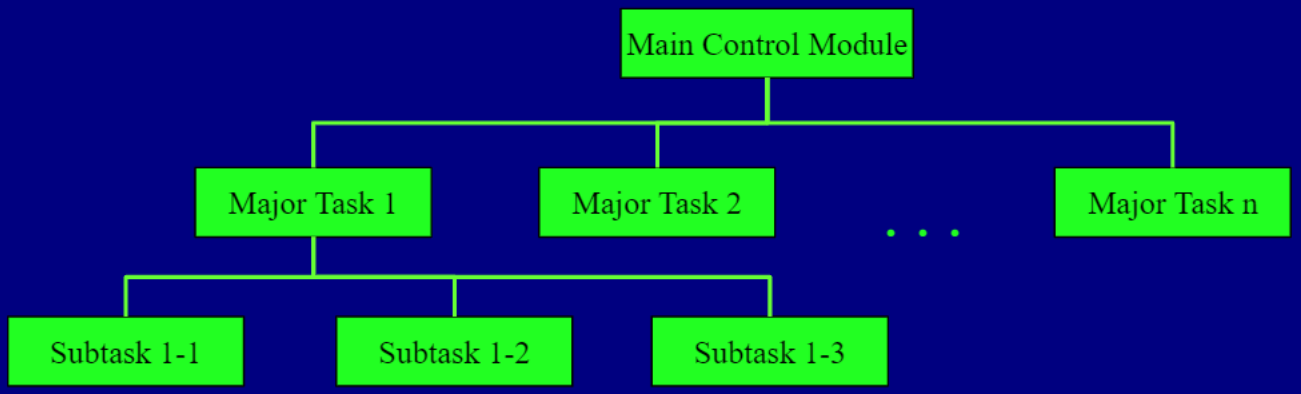
Hierarchy Charts

Hierarchy Charts

The modules and their interrelationships can be diagrammatically represented in a Hierarchy Chart.

Other Hierarchy Charts you may have seen:

- **Family Tree**
- **Organisational Chart of a Company**
- **UNIX / Windows file structure**



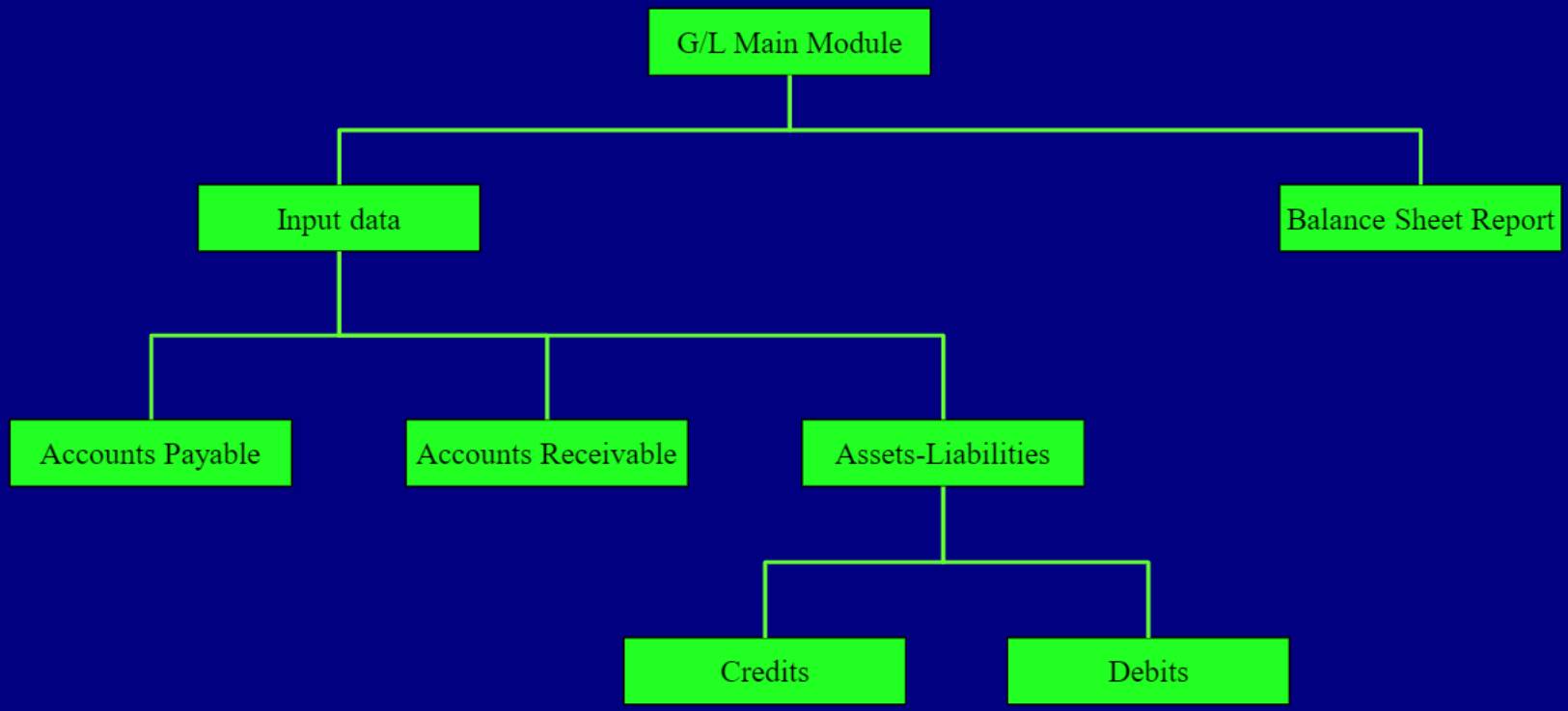
Procedural cohesion

Functional cohesion

Hierarchy Charts, continued

Hierarchy Charts, continued

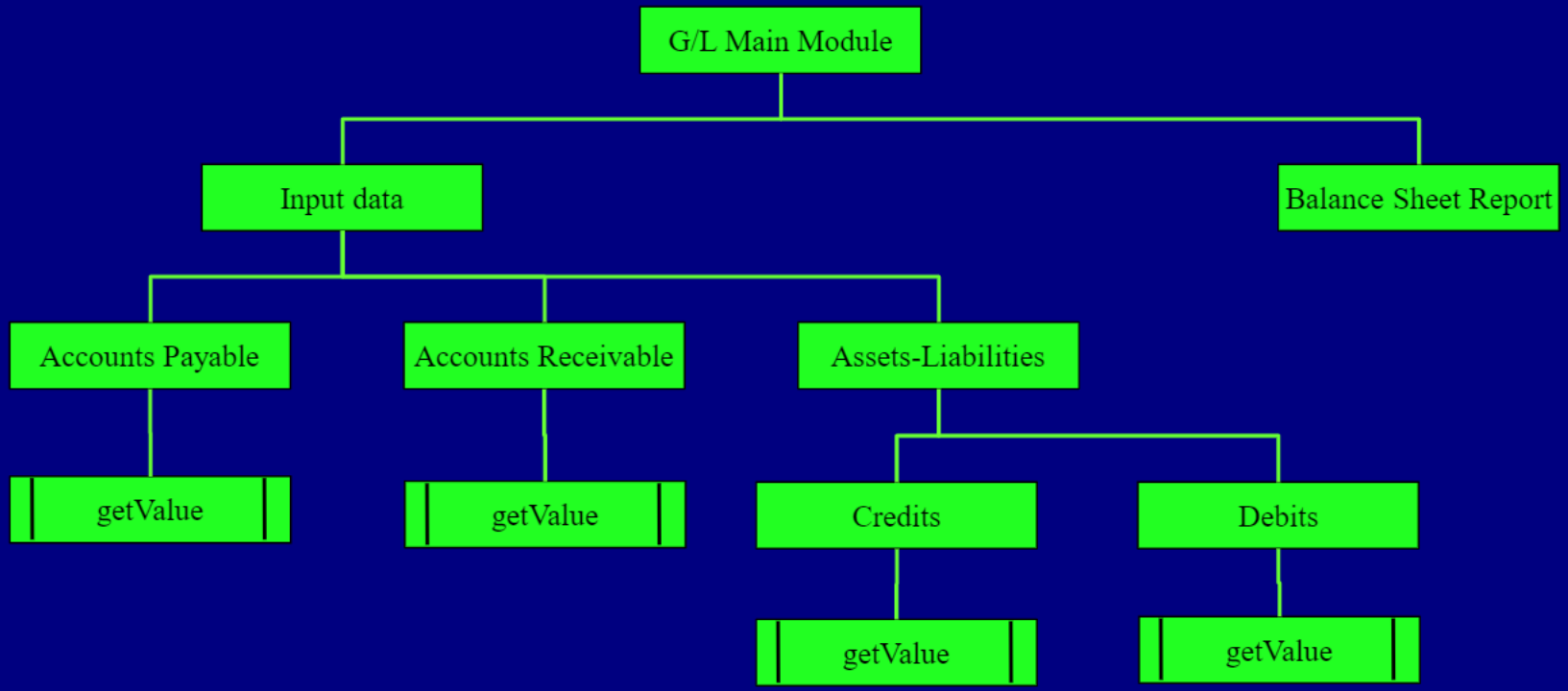
An example: A General Ledger Program



Hierarchy Charts, continued

Hierarchy Charts, continued

There are times when a module can be reused in more than one location within the Hierarchy Chart. One of the goals of structured programming is to identify and use common modules. These modules will typically be Functionally Cohesive. To show a module that is reused in more than one place, identify it with sidebars in addition to maintaining a consistent name throughout the Hierarchy Chart.



Hierarchy Charts, continued

Hierarchy Charts, continued

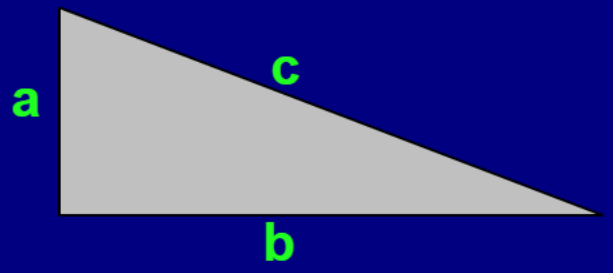
Example:

Develop a program that displays the area and perimeter of a triangle when only two sides are given. The two sides will be 'a' and 'b' from the diagram below.

$c^2 = a^2 + b^2$

area = 0.5 x (a x b)

What could the Hierarchy Chart look like?



Hierarchy Charts, continued

Hierarchy Charts, continued

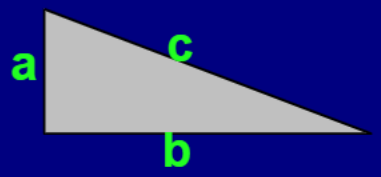
Example:

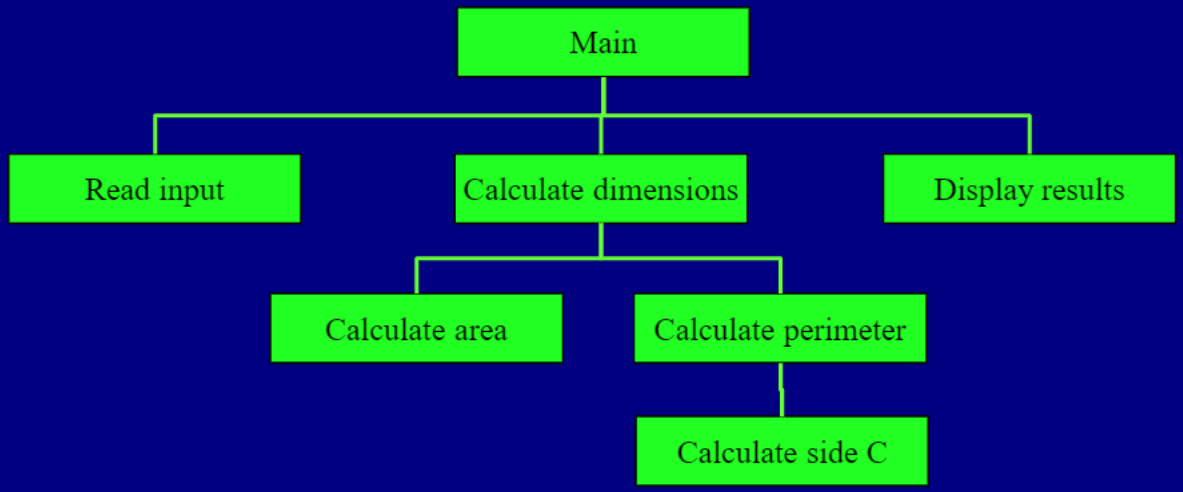
Develop a program that displays the area and perimeter of a triangle when only two sides are given. The two sides will be 'a' and 'b' from the diagram below.

$c^2 = a^2 + b^2$

area = 0.5 x (a x b)

What could the Hierarchy Chart look like?





Hierarchy Charts, continued

Hierarchy Charts, continued

In order to determine the maximum range available to an aircraft flying a specific direction, the fuel on board becomes the limiting factor. While flying the assigned heading, the aircraft may experience a headwind or a tailwind. Headwinds will be indicated by negative numbers, tailwinds by positive numbers.

The cruise performance chart for the aircraft provides the airspeed and rate of fuel consumption for various altitudes and power settings. The pilot is responsible for selecting the altitude & power setting from the performance chart. Based on the values selected by the pilot from the cruise performance chart, calculate and display the actual speed of the aircraft over the ground and the maximum range of the aircraft. For maximum range, the aircraft is permitted to expend all but 15% of the starting fuel capacity.

Ground speed = Cruise airspeed + wind
Time aloft = (0.85 * fuel) / rate of consumption
Maximum range = Ground Speed * Time aloft

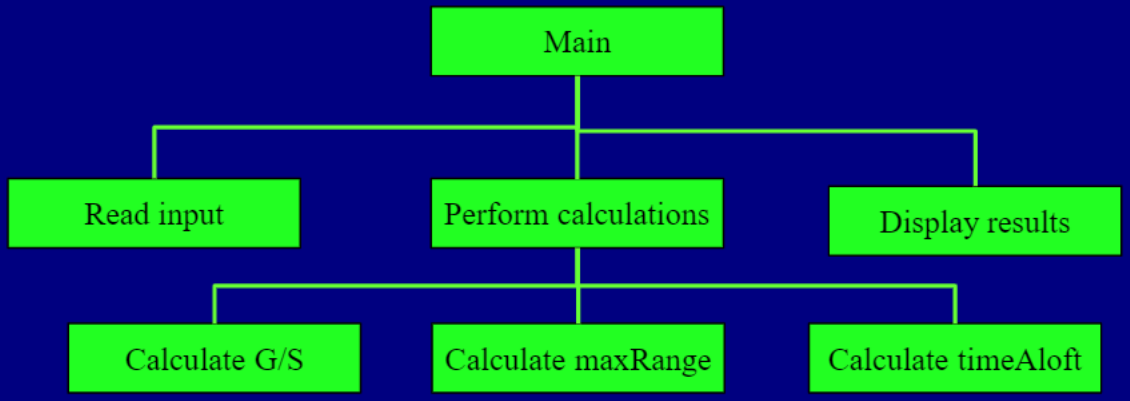
Hierarchy Charts, continued

Hierarchy Charts, continued

In order to determine the maximum range available to an aircraft flying a specific direction, the fuel on board becomes the limiting factor. While flying the assigned heading, the aircraft may experience a headwind or a tailwind. Headwinds will be indicated by negative numbers, tailwinds by positive numbers.

The cruise performance chart for the aircraft provides the airspeed and rate of fuel consumption for various altitudes and power settings. The pilot is responsible for selecting the altitude & power setting from the performance chart. Based on the values selected by the pilot from the cruise performance chart, calculate and display the actual speed of the aircraft over the ground and the maximum range of the aircraft. For maximum range, the aircraft is permitted to expend all but 15% of the starting fuel capacity.

Ground speed = Cruise airspeed + wind
Time aloft = (0.85 * fuel) / rate of consumption
Maximum range = Ground Speed * Time aloft



Hierarchy Charts, continued

Hierarchy Charts, continued

Create a four-function calculator. The program will prompt the user to enter three values. The first value entered is a value that tells the program what operation to perform, the second and third values are the values to be used in the calculation. The possible values that determine the operation to be performed are: 'A' for Add, 'S' for subtract, 'M' for multiply, 'D' for divide, and 'Q' to quit the program. After reading the input values, the program will perform the required operation, and display the output in the form:

x operation y = result

Where *x* and *y* are the two numbers entered, *operation* is a full description of the operation performed to make a proper sentence (i.e. 'plus', 'minus', 'multiplied by' and 'divided by'), and *result* is the result of the calculation.

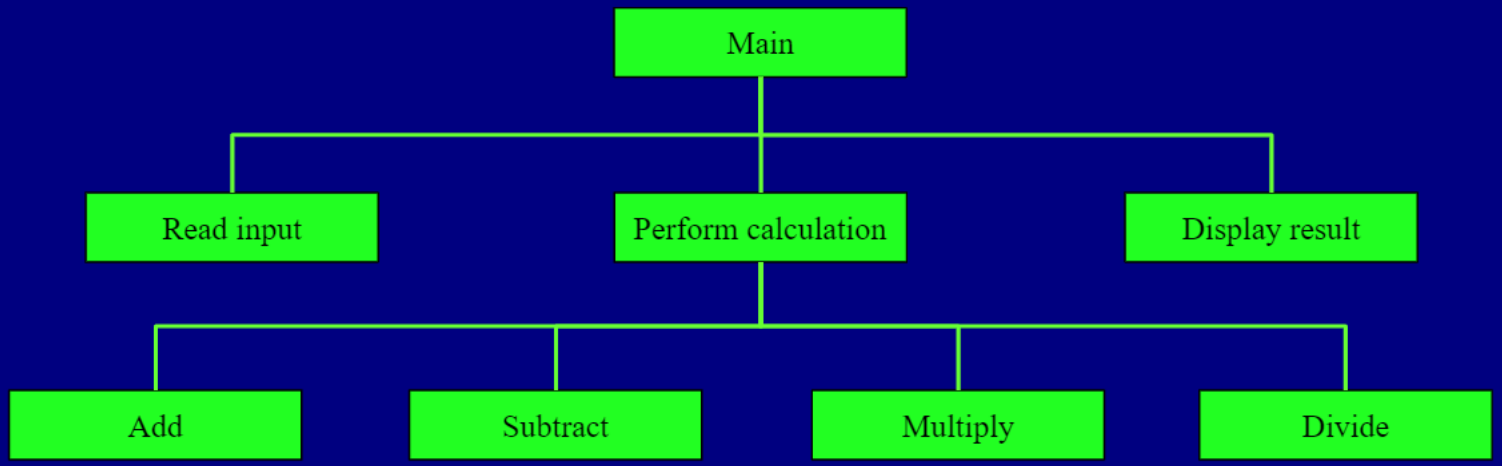
Hierarchy Charts, continued

Hierarchy Charts, continued

Create a four-function calculator. The program will prompt the user to enter three values. The first value entered is a value that tells the program what operation to perform, the second and third values are the values to be used in the calculation. The possible values that determine the operation to be performed are: 'A' for Add, 'S' for subtract, 'M' for multiply, 'D' for divide, and 'Q' to quit the program. After reading the input values, the program will perform the required operation, and display the output in the form:

x operation y = result

Where *x* and *y* are the two numbers entered, *operation* is a full description of the operation performed to make a proper sentence (i.e. 'plus', 'minus', 'multiplied by' and 'divided by'), and *result* is the result of the calculation.



Hierarchy Charts, continued

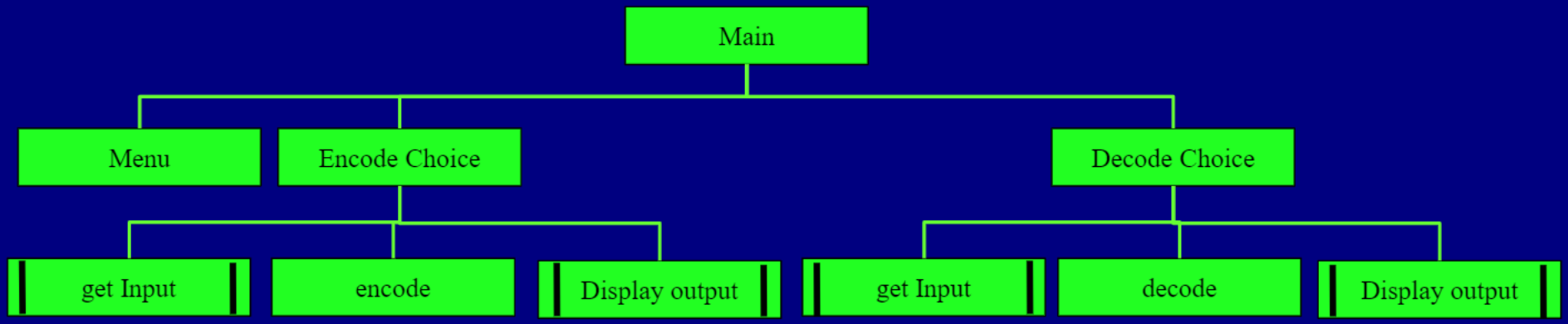
Hierarchy Charts, continued

Create a program that displays a menu for the user that allows them to 1) Encode a line of text into Morse code; 2) Decode a line of Morse code into text; 3) quit the program. After accepting their choice, the program gets a line of data (English or Morse) from the user and performs a conversion on it.

Hierarchy Charts, continued

Hierarchy Charts, continued

Create a program that displays a menu for the user that allows them to 1) Encode a line of text into Morse code; 2) Decode a line of Morse code into text; 3) quit the program. After accepting their choice, the program gets a line of data (English or Morse) from the user and performs a conversion on it.



Module Cohesion

Module Cohesion

Can we measure the quality of the modules we design?

Module Cohesion is defined as a measure of the internal strength of a module.

There are seven levels of Module Cohesion (from weakest to strongest):

- **Coincidental cohesion**
- **Logical cohesion**
- **Temporal cohesion**
- **Procedural cohesion**
- **Communicational cohesion**
- **Sequential cohesion**
- **Functional cohesion**

The following sections on Module Cohesion:
Computer Science: A Structured Programming Approach Using C.
B.A. Forouzan, R.F. Gilberg

Module Cohesion, continued

Coincidental Cohesion

Statements are grouped together simply because they happen to fall together - there is no meaningful relationship between the elements.

Exists in theory. The authors of the source material have not seen this in production software

Module Cohesion, continued

Logical Cohesion

Combines processes that are related only by the entity (module) that controls them.

For example, a module that conditionally opens several files based on a flag.

Do I open FileA?
Do I open FileB?
Do I open FileC?
..

Module Cohesion, continued

Temporal Cohesion

Combines several unrelated processes that always occur together.

Two examples are modules that programmers use to set-up or initialize a program, and shut-down a program upon its completion.

Module Cohesion, continued

Procedural Cohesion

Combines unrelated processes that are linked by a control flow. This is different than Sequential Cohesion, in that in Sequential Cohesion, data flows from one process to the next (no control flow).

Elements are grouped together because they operate according to a particular procedure.

They are executed in a particular sequence so that the objectives of the program are met.

Typical of Mainline modules, however not best as they span functional boundaries.

Module Cohesion, continued

Communicational Cohesion

Combines processes that work on the same data. Higher level modules commonly have Communicational Cohesion, however lower level modules typically do not.

Example: a module that reads an inventory file, prints current status of parts, then checks to see if parts need to be ordered.

Module Cohesion, continued

Sequential Cohesion

A module, that contains two or more related tasks that are closely tied together, usually with the output of one flowing into the input of the other.

For example, the tasks might be:

- **Calculate extended item price**
- **Sum items**
- **Calculate sales tax**
- **Calculate total**

Module Cohesion, continued

Functional Cohesion

The module contains only one process.

All the elements contribute to the performance of a single specific task.