

Algorithm Design Tools

An Introduction

What is an Algorithm

Do you like hot sauce? Here is an 'algorithm' for how to make a good one:

Volcanic Hot Sauce (from: <http://recipeland.com/recipe/v/Volcanic-Hot-Sauce-1125>)

10-12 scotch bonnets or Habanero,
serrano, jalapeno
6 cloves Garlic, peeled and chopped
1/3 c Fresh lime juice
1/3 c Distilled white vinegar
2 tbl Dijon style mustard

2 tbl Olive oil
1 tsp Molasses
1/2 tsp Turmeric
1 tbl Salt or to taste

1. Combine the pepper, garlic, lime juice, vinegar, mustard, oil, molasses, turmeric, and salt in a blender and puree until smooth. Correct the seasoning, adding more salt or molasses to taste.
2. Transfer the sauce to a clean bottle. You can use it right away, but the flavor will improve if you let it age for a few days. Volcanic Hot Sauce will keep almost indefinitely, refrigerated or at room temperature. Just give it a good shake before using.

What is an Algorithm

- As you can see, this previous ‘algorithm’ is really a *recipe*, that is, a set of well-defined steps that takes raw ingredients and produces a tasty result. In general, an algorithm can be described as a set of well-defined steps to solve a problem.
- In the context of computer programming, an **algorithm**, is defined as a:
“well-ordered collection of unambiguous and effectively computable operations, that when executed, produces a result and halts in a finite amount of time.”¹

Characteristics of an Algorithm

Well-ordered: the steps are in a clear order

Unambiguous: the operations described are understood by a computing agent without further simplification

Effectively computable: the computing agent can actually carry out the operation

¹ definition from: An Invitation to Computer Science (Gersting/Schneider) via

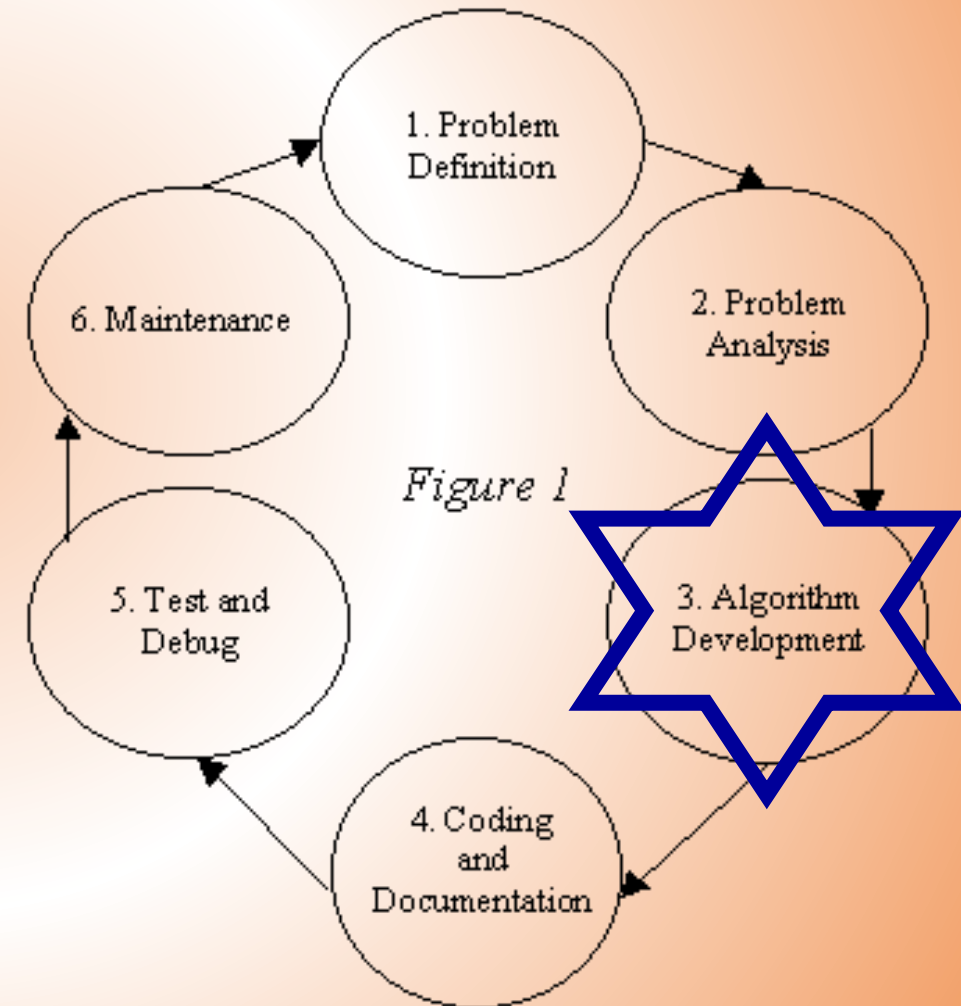
<http://www.cs.xu.edu/csci170/08f/sect01/Overheads/WhatIsAnAlgorithm.html> (visited 19JUN2009)

System Development Life Cycle (SDLC)

The algorithm development phase of the life cycle is probably the most challenging and creative.

Development of an algorithm requires the detailed analysis to determine what individual steps or task are necessary to produce an end result and in what order those steps or tasks need to be done.

This phase of the cycle needs a standardized diagramming tool to help in the analysis and in the documentation of the algorithm. So we are going to discuss two such tools in this presentation.



Method for Developing an Algorithm

1. **Define the problem:** State the problem you are trying to solve in *clear* and *concise* terms.
2. **List the *outputs*:** (what the algorithm will produce as a result)
3. **Identify the *inputs*:** (information needed to solve the problem or produce the output)
4. **Describe the steps** needed to convert, manipulate, or calculate with the inputs to produce the outputs. Start at a high level first, and keep refining the steps until they are *effectively computable* operations.
5. **Test the algorithm:** choose data sets and verify that your algorithm works!

It's at this point that your problem analysis work is complete, and you are ready to code the program.

Structured Algorithms / Programming

- Computer scientists through the years have demonstrated that problems (programs) can be solved with 3 control structures
 - **Sequence structure** - It is the construct where one statement is executed after another
(I will study my computer science lesson.
I will complete my review quiz assignment.
I will go play tennis.)
 - **Selection structure** – It is the construct where statements can be executed or skipped depending on whether a conditional decision evaluates to TRUE or FALSE (If *my homework is done* then I will go play tennis else I will study)
 - **Repetition structure** - It is the construct where statements can be executed repeatedly until a conditional decision evaluates to TRUE or FALSE (While ***the sun is still shining***, I will play tennis)

Structured Algorithms / Programming

- Notice that the earlier recipe algorithm exhibits these three control structures
 - **Sequence structure** - Combine the pepper, garlic, lime juice, vinegar, mustard, oil, molasses, turmeric, and salt in a blender
 - **Repetition structure** - puree until smooth
 - **Selection structure** – Correct the seasoning, adding more salt or molasses to taste

Structured Algorithms / Programming






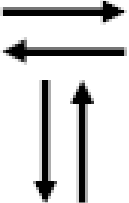
- By opening any cookbook you can observe that recipes have been written with a structured presentation of their steps. A cook can develop a tasty result from reading any recipe.
- The same structured presentation of a computational problem's algorithm has been developed in many different ways through the years, but two forms of algorithm presentation have been most frequently used.
 - Graphic presentation or what is called a **flow chart**
 - Verbal presentation or what is called **pseudocode**

Flowchart Tool

- There are two types of flowcharts
 - System (data) flowcharts
 - Programming flowcharts
- System flowcharts define the major phases of processing for an entire system.
- Programming flowcharts represent the sequence of operations the computer is to perform to solve a specific problem.

What is a Flow Chart

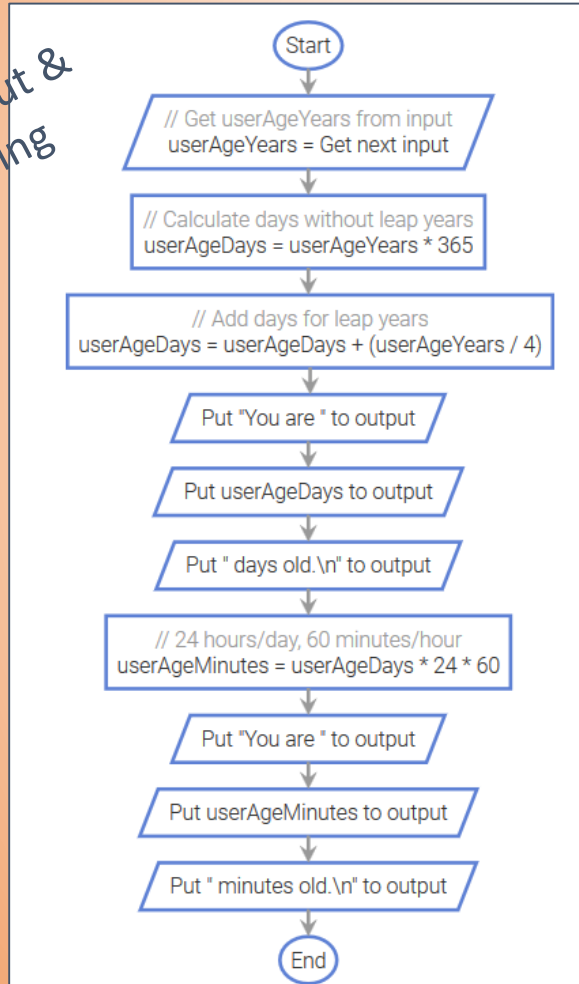
- A flow chart is a graphical tool that ***diagrammatically*** depicts the steps and structure of an algorithm or program
- Flow charts utilize graphic symbols to represent type of actions in the algorithm. Table below provides a list of key symbols used.

Symbol	Name/Meaning	Symbol	Meaning
	<u>Process</u> – Any type of internal operation: data transformation, data movement, logic operation, etc.		<u>Connector</u> – connects sections of the flowchart, so that the diagram can maintain a smooth, linear flow
	<u>Input/Output</u> – input or output of data		<u>Terminal</u> – indicates start or end of the program or algorithm
	<u>Decision</u> – evaluates a condition or statement and branches depending on whether the evaluation is true or false		<u>Flow lines</u> – <i>arrows</i> that indicate the direction of the progression of the program

Flow Chart Examples

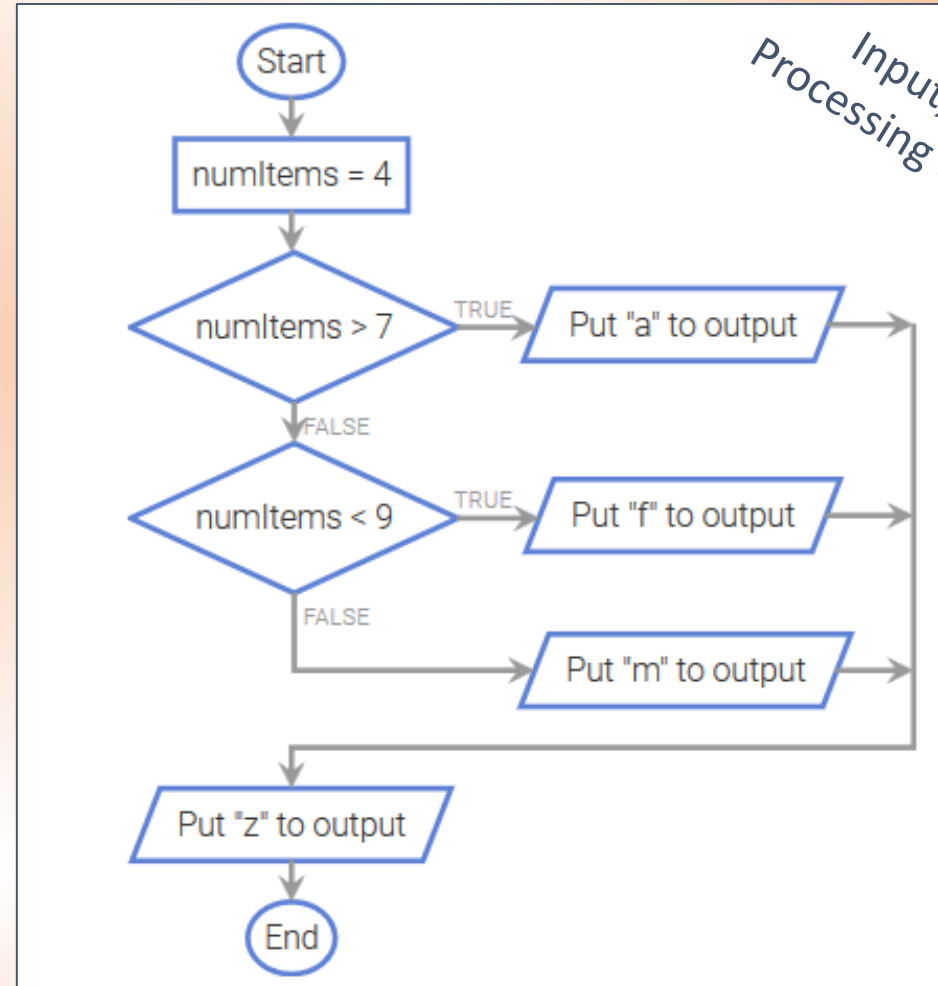
Calculate Person's Age
in Days and Minutes

Input, Output &
Processing



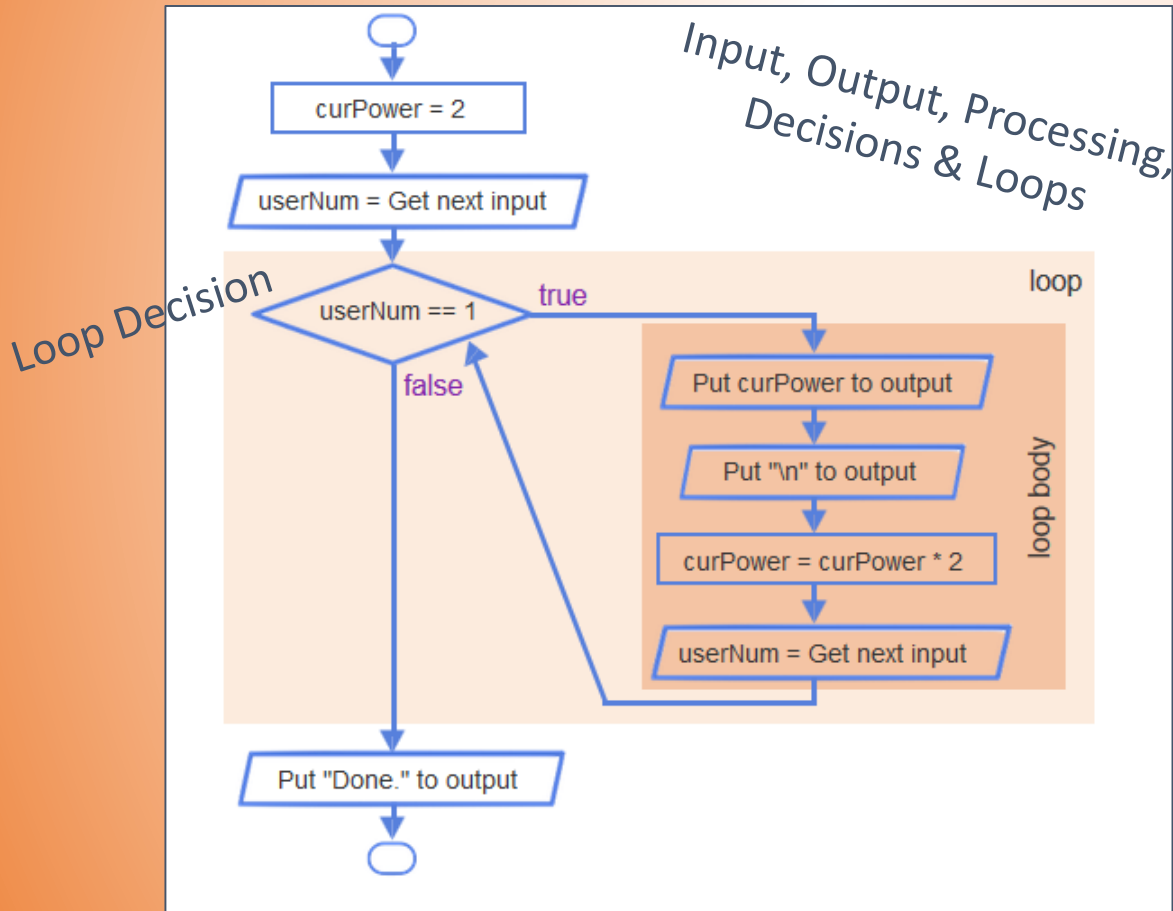
Determine output
based on number of items

Input, Output,
Processing & Decisions



Flow Chart Examples

Calculate Powers of 2

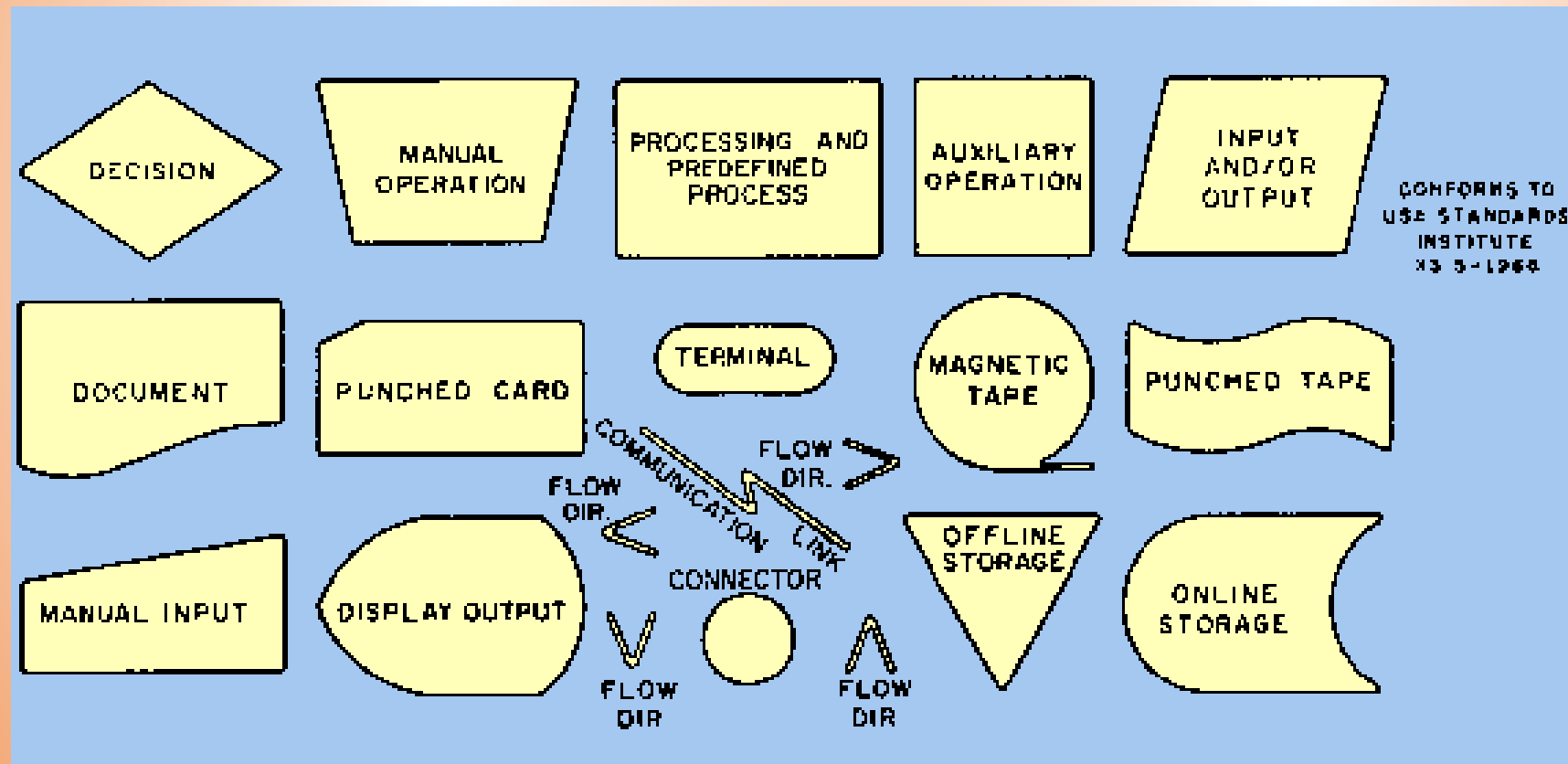


General rules for flowcharts

- All symbols of the flowchart are connected by flow lines (note *arrows*, not lines)
- Flow lines enter the top of the symbol and exit out the bottom, except for the Decision symbol, which can have flow lines exiting from the bottom or the sides
- Flowcharts are drawn so flow generally goes from top to bottom
- The beginning and the end of the flowchart is indicated using the Terminal symbol

Flowchart Templates

Flowchart Templates were used up until the late 1990's to aid in the drawing of flowcharts.



Flowcharting Software

- Flowcharting software has replaced the template and the hand drawn flowcharts
 - MS PowerPoint
 - MS Visio
 - SmartDraw
- Internet Cloud - Flowcharting
 - DrawAnywhere - www.drawanywhere.com
 - Lucidchart– www.lucidchart.com
 - Draw.io – www.draw.io

Flowchart Advantages / Disadvantages

- Advantages

- Standardized: all pretty much agree on the symbols and their meaning
- Visual (but this does bring some limitations)

- Disadvantages

- Hard to modify
- Need special graphing or drawing software
- Does not graphically depict all complex control structure designs

What is Pseudocode

- Consists of English-like statements that precisely describe the steps of an algorithm or program.
- Often called “fake code”. It is a cross between human language and a programming language that builds a model of the algorithm.
- Focuses on the *logic (what to be done)* of the algorithm or program
- Avoids programming language-specific elements
- Written with a level of detail so that the desired programming code can be generated almost automatically from each statement
- Algorithm steps are in sequence. Indentation of steps are used for dependent statements in selection and repetition structures

General Rules for Pseudocode and Examples

- Write only 1 statement or action per line
- Capitalize keywords
- We will often have to represent an expression like the one that computes *grossPay*. To symbolize the arithmetic operators we use these symbols:

grouping ()

exponent ** or ^

multiply *

divide /

add +

subtract -

Sequence – Calculate and print typical payroll amounts

name = Get next input

hourlyRate = Get next input

hoursWorked = Get next input

deductionRate = Get next input

grossPay = hourlyRate * hoursWorked

deduction = grossPay * deductionRate

netPay = grossPay – deduction

Put name to ouput

Put grossPay to output

Put deduction to output

Put netPay to output

General Rules for Pseudocode and Examples

- Indent to show hierarchy or subordination or dependency
- For a **SELECTION** structure indent the statements that fall inside the selection structure, but not the keywords that form the selection
- When we have to make a choice between actions, we almost always base that choice on a relational comparison test
 - > (greater than)
 - < (less than)
 - >= (greater than or equal to)
 - <= (less than or equal to)
 - == (equal to)
 - <> (not equal to)

Selection – Calculate and print typical payroll amounts

name = Get next input

hourlyRate = Get next input

hoursWorked = Get next input

grossPay = hourlyRate * hoursWorked

If grossPay >= 100

deduction = grossPay * deductionRate

Else

deduction = 0

netPay = grossPay – deduction

Put name to output

Put grossPay to output

Put deduction to output

Put netPay to output

General Rules for Pseudocode and Examples

- Indent to show hierarchy or subordination or the body of statements for a decision or a repetition loop.
- Indent the statements that fall inside the loop, but not the keywords that form the loop.
- Keep statements programming language independent – that is resist the urge to write in whatever programming language you are most comfortable with. In the long run, you will save time!

Repetition – Calculate powers of 2

curPower = 1

Put “Enter a 0 to end the calculations” to output

While userNum != 0

curPower = curPower * 2

Put curPower to output

Put newline to output

userNum = Get next input

Put “The end” to output

Pseudocode Advantages / Disadvantages

- Advantages

- Can be done easily on a word processor
- Easily modified
- Implements structured concepts well

- Disadvantages

- It's not visual
- There is no accepted standard, so it varies widely from company to company