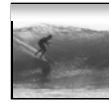




Algorithms

Algorithms are a familiar idea. Our goal is to learn to specify them right so someone or something else does the work



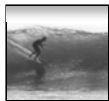
Previous Algorithms

Algorithm, a precise, systematic method to produce a specified result

- We have seen algorithms already...
 - Placeholder technique is an algorithm with an easy specification:


```
longStringWithShortStringInt ← placeholder
ShortString ← ε
placeholder ← longStringWithShortStringInt
```

Not every process is an algorithm -- debugging



Properties of Algorithms

For an algorithm to be well specified it must have ...

- Inputs specified
- Outputs specified
- Definiteness
- Effectiveness
- Finiteness



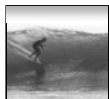
Programs vs Algorithms

A program is an algorithm specialized to a particular situation

- * Algorithm:


```
longStringWithShortStringInt ← placeholder
ShortString ← ε
placeholder ← longStringWithShortStringInt
```
- * Program:

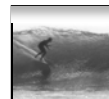

```
↵↵ ← #
↵ ← ε
# ← ↵↵
```



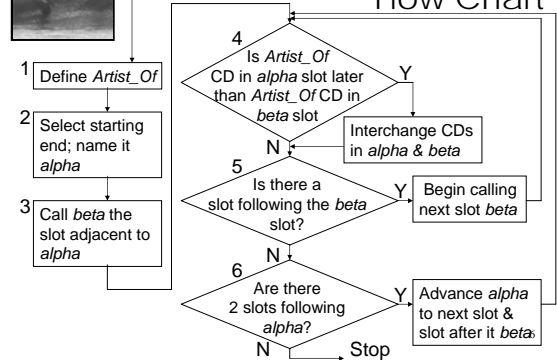
Alphabetize CDs


1. Def *Artist_of/Use Artist_of* to refer to the group name
2. Pick *Alpha* Decide which end of rack is to be start of alphabetic sequence, and call the first slot *alpha*
3. Pick Beta Call the slot next to *alpha*, *beta*
4. Exchange If *Artist_of* of the CD in the *alpha* slot is later in the alphabet than the *Artist_of* of the CD in the *beta* slot, interchange the CDs, otherwise continue on
5. More Betas? If a slot follows *beta* slot, begin calling it the *beta* slot and go to step 4, otherwise continue on
6. More Alphas? If two slots follow the *alpha* slot, begin calling the next one the *alpha* slot and the one following it the *beta* slot; go to step 4; otherwise stop

Spoon
Beethoven
Hampton
Wynette
Pearl Jam



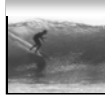
Flow Chart





Demonstration

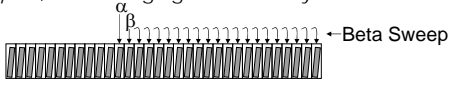
7



Abstraction


Abstraction means removing an idea or process from a situation

Beta sweep -- while *alpha* points to a fixed slot, *beta* sweeps through slots following *alpha*, interchanging as necessary

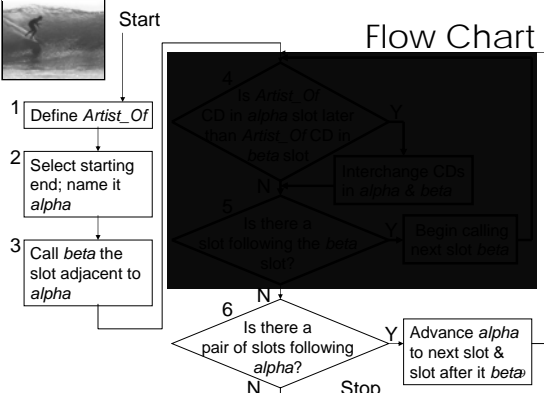


The beta sweep is a concept removed based on our understanding of the operation of the algorithm

8



Flow Chart



Start

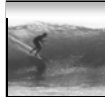
- 1 Define Artist_Of
- 2 Select starting end; name it *alpha*
- 3 Call *beta* the slot adjacent to *alpha*

Is there a pair of slots following *alpha*?

Y Advance *alpha* to next slot & slot after it *beta*

N Stop

6




The Beta Sweep

By abstracting we can analyze parts of an algorithm ...

* The beta sweep has 4 properties:

- *Exhaustive* -- it considers all CDs after *alpha*
- *Non-redundant* -- no slot pair is checked twice
- *Progressive* -- the alphabetically earliest CD considered so far is always in the *alpha* slot
- *Effective* -- at completion, the alphabetically earliest CD from *alpha* to end is in *alpha* slot

these properties apply only to Alphabetize CDs¹⁰



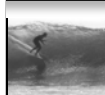
Alpha Sweep

The alpha sweep...

Process of sweeping through all of the CDs (but the last) performing the beta sweep

- *Exhaustive* -- considers all but last CD
- *Non-redundant* -- a slot is *alpha* only once
- *Progressive* -- when *beta* sweep completes the alphabetically next CD in *alpha*
- *Complete* -- when last *beta* sweep is done the last slot's CD is later than next to last slot
- *Effective* -- the *alpha* sweep alphabetizes

11



Summary

We figure out most algorithms on our own, abstracting from specific cases

Also we abstract parts of an algorithm or program to understand them

* Thinking of how the program works and reasoning about its properties allows us to know *why* an algorithm works ... and then we can let the computer do it

12