Key Issues in Program Design

- **Choice of data structures:** good data structure makes your life easy!

- **Choice of good algorithm:** this depends on the data structure.

Levels of data structure description

1. Data reality (lots of redundant features)

2. Abstract data structure such as array, list, tree, queue, stack, ...

3. machine encoding (in bits)

An **Abstract Data Type** is defined by

- set of values
- operations on these values

Examples:

Integers (pascal)

- Values: ... -2, -1, 0, 1, 2, ...
- Operations: $+$, -, $\times$, div, $=$, $<$, $>$

Booleans

- Values: true, false

- Operations: $\wedge$, $\vee$, $\oplus$, $=$

Stack

- Values: set of elements of same type

- Operations: push, pop

BinaryTrees

- Values: set of binary trees

- Operations: insert a node, delete a leave, merge two trees, *etc*.

In above definition we were were not precise. And we will not be in the future. But if you want to be precise, here is how.

Definition of Natural Numbers

1. 0 is a natural number. (Sometimes we use empty set $\emptyset$)

2. If n is a natural number, so is succ(n). (This will be called $n+1$. And recursively one can define the $+$ operation.)

3. $= (0, 0)$.

4. If $= (m, n)$, then $= (\text{succ}(m), \text{succ}(n))$.

We won't be this formal usually ... but we could be.

Representation of elementary objects

- **bit** $\oplus$
  It just so happens that 2 valued logic is currently the easiest to build.

- **byte** (addressable unit)

```
|____|____|____|____|____|____|____|____|

                8 bits
```

- **word:** 4 bytes

- **long word:** 8 bytes

Interpretation of a (32 bit) word

- **logical:** array of 32 Boolean values, 0 = false , 1 = true

- **integer:** range $-2^{31} \sim 2^{31} - 1$ (or other choice: $0 \sim 2^{32} - 1$)

- **floating point:** - some reasonable set of real numbers, reasonablly spaced.

sign	exponent	mantissa

- **character:** use 8 bits for 1 character, so this can represent 256 different characters

- *Pascal enumerated types*

With each of above "interpretation", there is also a particular set of operations.

On k Boolean variable, how many Boolean functions are there?
Count number of f: $\{x_1, ..., x_k\} \rightarrow \{0, 1\}$.

- 2^k different inputs
- on each input, f could be "0" or "1"
  $\Rightarrow 2^{2^k}$ different functions
- for $k = 2$ (binary function): $2^{2^2} = 16$
  0 $\}$ constant functions
  1
  $\wedge$
  $\vee$
  xor
  nand
  $\vdots$
  10 more

Array and Search

- Sequential Search: search for x in A.

  $A[n+1] := x$; use of sentinel

  $i := 1$;

  while $A[i] <> x$ do $i := i + 1$

search cost:

worst case: $x \notin A$, n comparisons

Average case: $Prob(x = A[i]) = \frac{1}{n}$

Expected cost: $\sum_{i=1}^{n} i\frac{1}{n} \approx \frac{n}{2} = 0(n)$.
($Prob(x = A[i]) = \frac{1}{n}$ means all elements $A[i]$ are equally likely.)

* Usually the average case is different from worst case.
(This case it happens to be asymptotically same.)

- Binary Search
  - $A[1..n]$ is sorted.

```
low := 1; high := n;
while low < high do
            begin mid:= (low + high) div 2;
                           if A[mid] < x then low:= mid + 1
                                          else high:= mid
            end;
if A[high] = x then return "high"
            else ∉ A
```

<u>Search Cost</u> (worst case)

$$\begin{aligned} T(n) &\leq 1 + T(\lceil \frac{n}{2} \rceil) \\ &= O(\log n) \text{ comparisons} \end{aligned}$$