

**Icon**

Icon is a very-high-level,  
typeless, expression  
language with  
backtracking.  
("Icon" does not mean  
"picture".)

Copyright 1997 Thomas W. Christopher 1

**What's Icon good for?**

Tools  
Prototyping  
Trying out ideas  
Quick programming  
Text processing

Copyright 1997 Thomas W. Christopher 2

**Topics of Discussion**

Program structure  
Data types  
Control constructs

Copyright 1997 Thomas W. Christopher 3

**Program structure**

Files compiled together  
Declarations  
Procedures  
Variables  
Records

Copyright 1997 Thomas W. Christopher 4

**Hello, world.**

```
procedure main()
write("Hello, world")
end
```

Copyright 1997 Thomas W. Christopher 5

**Data types**

"Typeless"  
types attached to values, not to  
variables  
in contrast to "typed" and "type  
ignorant"  
pure and mutable types

Copyright 1997 Thomas W. Christopher 6

## Example 1

```
procedure main()  
x := "Example "  
writes(x)  
x := 1  
write(x)  
end
```

Copyright 1997 Thomas W. Christopher

7

## Pure Data types

integer and real  
string  
cset (character set)  
procedure

Copyright 1997 Thomas W. Christopher

8

## Mutable Data types

list  
record  
table  
set  
co-expression  
file

Copyright 1997 Thomas W. Christopher

9

## Control constructs

Expression language:  
"Statements" are  
expressions  
Expressions are generators:  
Expressions generate  
sequences of values  
Expressions generate their  
sequences of values by  
backtracking

Copyright 1997 Thomas W. Christopher

10

## Backtracking:

expressions can succeed or  
fail  
backtracking into an  
expression generates its next  
value  
backtracking is "cut off"  
between statements and in  
some other contexts

Copyright 1997 Thomas W. Christopher

11

## Simple expressions

Constants and variables  
generate a single value  
 $e1 \mid e2$  generates all values  
generated by  $e1$  followed by  
all values generated by  $e2$   
 $(1 \mid 2) \mid (3 \mid 4)$  generates the  
sequence 1, 2, 3, 4

Copyright 1997 Thomas W. Christopher

12

## Binary operators

**e1 op e2 :**  
for every value x generated by  
e1 take every value y  
generated by e2 and produce  
(x op y) in the resulting  
sequence  
E.g. **(1 | 2) + (10 | 20)** generates  
11, 21, 12, 22

Copyright 1997 Thomas W. Christopher 13

## Relational operators

a relational operator is a binary  
operator  
it fails if the relation does not  
hold  
it returns the right operand if it  
succeeds  
**a < b < c** returns the value of c  
if a is less than b and b is less  
than c

Copyright 1997 Thomas W. Christopher 14

## Conjunction &

**&** has very low precedence  
use it to allow backtracking  
between "statements"  
**&** is just a binary operator that  
returns its right operand  
E.g.  
**writes(" ",**  
**(1 | 2) + (10 | 20) & 1 < 0**  
**writes 11 21 12 22**

Copyright 1997 Thomas W. Christopher 15

## While expressions

**while e1 do e2**  
**while e1**  
at most one value from each (e1  
and e2) per iteration  
e2 may succeed or fail  
restarts e1 from the beginning  
fails when e1 fails

Copyright 1997 Thomas W. Christopher 16

## Example: File copy

```
procedure main()
while write(read())
end
```

Copyright 1997 Thomas W. Christopher 17

## Example: Define Bit Positions

```
procedure main()
i := 1
while x := read() do {
write("#define ",
x, " ", i)
i := i+i
}
end
```

Copyright 1997 Thomas W. Christopher 18

## Every expressions

every e1 do e2  
every e1  
generates all values for e1  
for each value of e1, evaluate e2  
generate at most one value for e2  
e2 may succeed or fail  
fails when e1 fails

Copyright 1997 Thomas W. Christopher 19

## Why file copy couldn't use every

```
while write(read())  
read() returns one line from the  
input  
read() does not generate  
another line when backed  
into
```

Copyright 1997 Thomas W. Christopher 20

## Example: write table of max. values

```
procedure main()  
writes(" "); every writes(" ", 1 to 5); write()  
every i := 1 to 5 do {  
  writes(i)  
  every j := 1 to 5 do  
    writes(" ", max2(i,j))  
  write()  
}  
end
```

Copyright 1997 Thomas W. Christopher 21

## Example: table of max. values

```
  1 2 3 4 5  
1 1 2 3 4 5  
2 2 2 3 4 5  
3 3 3 3 4 5  
4 4 4 4 4 5  
5 5 5 5 5 5
```

Copyright 1997 Thomas W. Christopher 22

## Example: max2(x,y)

```
procedure  
max2(x,y)  
x := x < y  
return x  
end  
  
procedure  
max2(x,y)  
x <:= y  
return x  
end
```

Copyright 1997 Thomas W. Christopher 23

## If expressions

```
if e1 then e2 else e3  
if e1 then e2  
if e1 succeeds, behave like e2;  
if e1 fails, behave like e3  
do not generate more than one  
value from e1
```

Copyright 1997 Thomas W. Christopher 24

## Example: Octal Bit Positions

```
procedure oct(i)
  return if i = 0 then "0" else oct(i / 8) ||
    i % 8
end
procedure main()
  i := 1
  while x:= read() do {
    write("#define ", x, " ", oct(i) )
    i += i
  }
end
```

Copyright 1997 Thomas W. Christopher 25