PROGRAMMING 'STYLE' [FIG. 3-4]

- USE COMMENTS THROUHOUT !
- GOOD VAR. NAMES / CONST. NAMES / FCT. NAMES
- GOOD STYLE & READABILITY
- USE CONSTANTS

  # define checkBouncingFee 10.00
- GOOD FORMATTING & INDENTATION (of C source files)
  - SEE "PROGRAM LEVELS":

- USE SIMPLICITY, NOT COMPLEXITY
- USER-FRIENDLY < FOR PROGRAMMER / FOR USER

THUS: 1) DESIGN PROGRAM PRIOR TO CODING

2) DESCRIBE PROGRAM IN "PSEUDO-CODE" (= mixture of English & C)

3) WHEN "LOGIC" OF DESIGN AND PSEUDO-CODE IS GOOD / CORRECT,

CHAPTER 4 READ IMPLEMENT IN C

TYPES OF STATEMENTS

1) SIMPLE ⊛ → PERFORM AN OPERATION

2) CONTROL → CONTROL ORDER IN WHICH STATEMENTS ARE EXECUTED

⊛ NOTATION: EXPRESSION ;

ASSIGNMENT

1) EMBEDDED – DONE AS PART OF EXPRESSION

z= " (x = 6) + (y = 7) "

i) assign 6 to x, assign 7 to y

ii) value is 13

2) MULTIPLE :

" n1 = n2 = n3 = 0; "

≙ " (n1 = (n2 = (n3 = 0))); "

/* From RIGHT to LEFT */

3) PRECEDENCE of alignment '=':

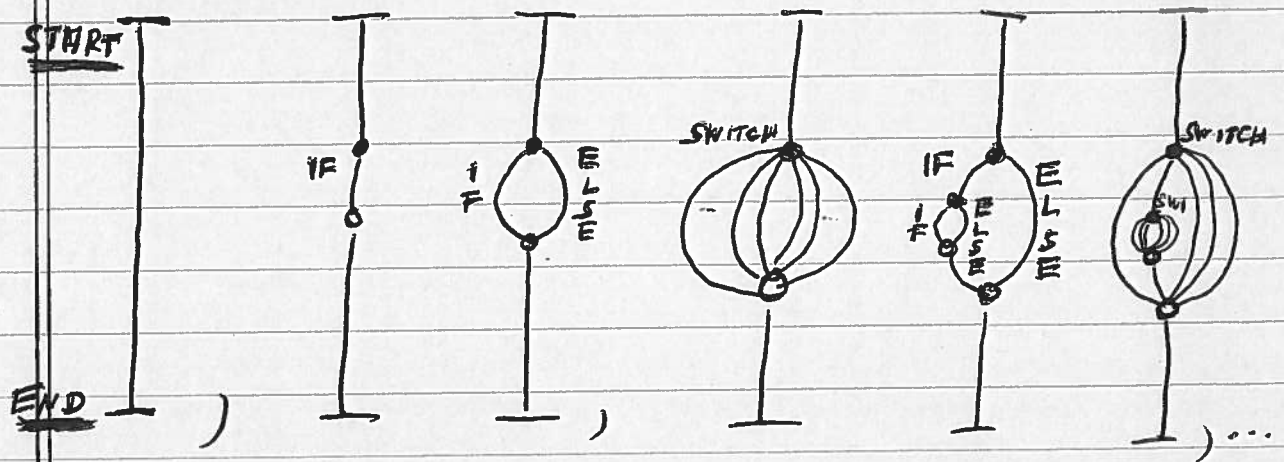
ASSIGNMENT OPERATOR

USED AFTER ARITHMETIC OPERATORS

BLOCK :

{ statement 1;
statementN;
}

PROGRAM FLOW



CONTROL AND BOOLEAN DATA

→ genlib.h has type **bool**

TRUE (=1) or **FALSE** (0)

"Bools = Results of relational comparison"

op's: ==, !=, >, <, >=, <=

• LOGICAL OPERATORS:

		CLANG.	EX int a, b;
¬	NOT	≙ !	
∧	AND	≙ &&	if (a == 4)
∨	OR	≙ ||	if (a != 4)
			if (a==4 || b==5)
			if (a==4 && b==6)

A, B of type **bool**:

①

A	!A
T	F
F	T

, ②

A \ B	T	F
T	T	F
F	F	F

, ③

A \ B	T	F
T	T	T
F	T	F

- PRECEDENCE:

EXP$_1$ relop$_1$ EXP$_2$ relop$_2$ ...

≙ (EXP$_1$ relop$_1$ EXP$_2$) relop$_2$ ...

WHY? EFFICIENCY: stop evaluating once result is defined/clear

- (FALSE) && 'whatever' is **FALSE**
- if (A || B || C || D || ...)

↑ if A||B is **TRUE**, the result is **TRUE**

EX: "LEAP YEAR"

(→ DEF: LEAP YEAR (=) FEB HAS 29 DAYS.)

- Leap years: ..., 1200, 1600, 2000, 2400, ...
- Leap years: ..., 1988, 1992, 1996, 2004, ...
- NOT Leap yrs: ..., 1300, 1400, 1500, 1700, 1800, 1900, 2100, ...
  " " " ..., 2001, 2002, 2003, 2005, ...

→ bool isLeapYear;
int yr;

isLeapYear = ("if one can divide by 400"
OR
"if one can divide by 4 but not by 100")

(*) { = ((yr % 400 == 0)
|| (yr % 4 == 0 && yr % 100 != 0))

FIG. 4-1

```
main ()
{ int yr;
  bool isLeapYear;
  printf ("Type a year:");
  yr = GetInteger();
  isLeapYear = (*);
  if (isLeapYear)
  { printf ("A leap year\n");
  }
  else
  { printf (...);
  }
}
```

• NOTE

i) SINGLE-LINE IF:

if (condition) statement;

ii) MULTI-LINE IF

if (condition)
{ many statements;
}

IF - ELSE

```
if (con)
{ ...
}
else
{ ...
}
```

"CASCADE"

```
if (c1)
{ ...
}
else if (c2)
{ ...
}
else if (c3)
{ ...
}
```

if ≡ else if

else if

The '? :' operator

```
if (x > y)
{ max = x;
}
else
{ max = y;
}
```

≙ `max = (x > y)? x : y;`

- "SWITCH" statement

```
switch ("control expr.")
{
case c1 : statements;
          break;
  :
case cN : statements;
          break;
default: /* none of cases c1-cN */
          statements;
          break;
}
```

- EX:

```
int age;
age = ...;
switch (age / 10)
{ case 0 : printf ("Between
                    0 and 9.\n");
           break;
  :
  case 10 : printf ("Over
                     100.\n");
  default: ...
}
```

W2

- **WHILE** – "while (cond) {...}"
  1) COND test BEFORE every cycle
  2) Complete cycles executed even when COND becomes FALSE ...

"Loop-and-a-half problem"

→ Mechanism for exiting a while loop!

(1)

```
while (TRUE)
{ "READ INPUT";
  if ("INP == SPECIAL") break;
  statements;
}
```

DOING ALL OF THEM OR NOT A SINGLE ONE.

(2)

```
"READ INPUT";
while ("INPUT != SPECIAL")
{ statements;
  "READ INPUT";
}
```

equiv.

- **FOR LOOP**

General form: for ("init"; "test"; "step") {...}

"init" / "step" must/can use ONLY ints (0.999 ≠ 1.000)

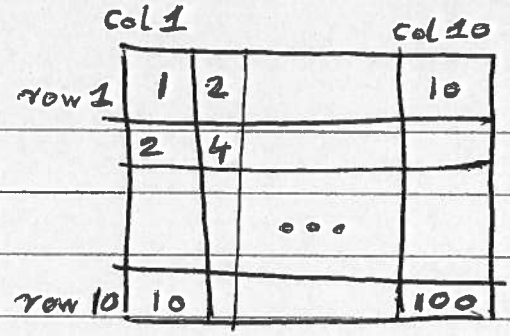
"test": loop executed as long as test yields TRUE.

→ **NOTE:** for (;;) ≙ while (TRUE)

NESTED LOOPS

→ Nested for loop

Ex.: "Multiplication Table"




```
#include <stdio.h>
#   "   "genlib.h"
#define From 1
#   "    To  10
main()
{ int row, col;
  for (row = From; row <= To; row ++)
  {  for (col = From; col <= To; col ++)
     {  printf ("␣%.4d", row * col);
     }
     printf("\n");
  }
```

• NOTE: `for (i=0; i<10; i++)` ≙ `i = 0;`

```
while (i<10)
{ ...
  i++;
}
```

• NOTE: NO FLOATS AS LOOP VARIABLES (COUNTERS):

```
for (x = 1.0; x <= 2.0; x += 0.1)
{ ...
```

ISSUE: $0.999999 \neq 1.000001$

but should probably be used as "being equal" ≈