CHAP. 5

W3
FUNCTIONS

- **"DEF":** "FUNCTION = Named Collection of Statements"

EX: → printf ("...", ...);
fct. name ↑ ↑ arguments / argument list

→ n1 = GetInteger();
name ↑ ↑ no arguments
returns an int

→ in <math.h> : sin(x), sqrt(x), ...
(see "man" pages)

GENERAL:

"result_type fct_name (argument_specification);"
↑ types & number of arguments

EX: → "double sine (double angle);" ← ; for prototype def.

→ "int GetInteger (void);"
↑ no arguments

→ Celsius to Fahrenheit: $F(C) = \frac{9}{5}C + 32$

↓

```
#include <stdio.h>
"genlib.h"
```

↓ C program

```
#define Freezing    0
        Boiling   100
        Step        5

/*
 * Prototype of Fcts.
 */

double CelsiusToFahrenheit
       (double c);
```

W3

```
main ()
{ int cel;
  printf ("Celsius to Fahrenheit \n");
  for (cel = Freezing; cel <= Boiling; cel += Step)
  { printf ("%7d %10d \n",
            cel, (int) CelsiusToFahrenheit (cel));
  }
}
```

EX.

```
/*
* CelsiusToFahrenheit
*/
double CelsiusToFahrenheit (double c)   NO ;
{ return ( 9.0/5.0 * c + 32);
}
```

INT.

```
int AbsValueInt (int n)
{ if (n<0) { return (-n); }   /* and EXIT */
  else     { return (n); }    /* and EXIT */
```

STRING

```
string MonthName (int mo)
{ switch (mo)
  { case 1: return ("January");
    :
    case 12: return ("December");
  } default: ...
}
```

→ CALL: printf ("%s \n", MonthName (month));

BOOL

```
bool IsEven (int n)
{ return (n % 2 == 0);
}
```

- SEE FIG. 5-4 / READ PP. 150–160

→ Parameter-passing, call-by-value, local variables, ...

EX: "Calling FCTS from within other FCTS."

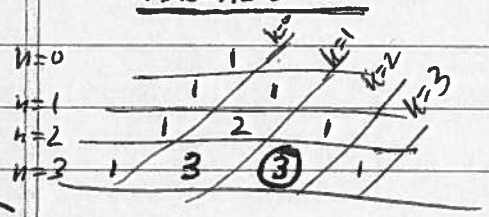
→ No. of different subsets of cardinality k of a set of n elements: $\binom{n}{k}$

{A, B, C}, n = 3, k = 2

⇒ {A, B}, {A, C}, {B, C} → $\binom{3}{2} = 3$

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}, \text{ where } i! = \begin{cases} 1 & \text{if } i = 0 \\ i * (i-1)! & \text{if } i > 0 \end{cases}$$

PASCAL'S Δ



$$\binom{3}{2} = \frac{3!}{2!\;1!} = \frac{6}{2 \cdot 1} = \frac{6}{2} = 3$$

```
int NoSubsets (int n, int k);
int Factorial (int n);
...
main ()
{ int n, k;
  printf ("Total no of elems.: n ");
  n = GetInteger ();
  printf ("No of elems in subset: n ");
  k = GetInteger ();
  printf ("There exist %d distinct
           subsets with %d elems. \n",
           NoSubsets (n, k), k);
}
int NoSubsets (int n, int k)
{ return ( Factorial (n) /
         ( Factorial (k) * Factorial (n-k) ) );
}
```

```
int Factorial (int n)
{ int prod, i;
  prod = 1;
  for (i=1; i<=n; i++)
  { prod *= i;
  }
  return (prod);
}
```

{Rem: "Procedure" = Function that does not return a value}

STEPWISE REFINEMENT

→ Divide BIG problem into many smaller sub-problems

→ Top-down design / Stepwise refinement / Decomposition

```
main ()
{
   P1;  <  P1.1 <
        <  P1.2 <
   P2; ...
   P3;  <<  P3.1 <<
}
```

SUB-PROBLEM SHOULD BE SIMPLE!

EX: "CALENDAR": Print the 12 months of a year!

■ Known: 1 Jan. 1900: a Monday

SU	MO	TU	WE	TH	FR	SA
		1	2	3	4	5	6
7	8	9	10	11	12	13
14						20
21						27
28	29	30	31			

INP: year

OUT: each month printed

ALGO: ? difficult ?

• Facts: (1) 1/1/1900 was Monday.

(2) Jan, Mar, ..., Dec have 31 days.

(3) Apr, Jun, ..., Nov " 30 ".

(4) Feb has 28 days, generally, and 29 days in LEAP years.

(5) LEAP years: {..., 1988, 92, 96, ...}
{..., 1200, 1600, 2000, 2400, ...}

¬LEAP years: {..., 1700, 1800, 1900, 2100, 2200, ...}
{..., 1989, 90, 91, 93, ...}

ALGO ⇓ ⇕

BOOL fct to determine LEAP vs. ¬LEAP year

W3

```
main ()
{...
 yr = GetIntegerYr ();
 PrintCal (yr);
}
```

```
void PrintCal (int yr)
{...
 "for all 12 mo"
 {
  PrintMo (mo, yr);
 }
```

```
void PrintMo (int mo, int yr)
{...
 "Print line: Month & year";
 "Print Line SU MO TU ... SA";
 "Determine No of month's days"; ➡ noDays = MoLength (mo, yr); (1)
 "Determine weekday of the 1st"; ➡ Weekday = FirstDayOfMo (mo, yr); (2)
 "Indent first line properly"; ➡ IndentFirstLine (weekday); (3)
 "For all days print digit in proper place";
}...
```

(1)
```
int MoLength (int mo, int yr)
{...
 switch (mo)
 { "return 31 for Jan, ..., Dec";
   "return 30 for Apr, ..., Nov";
 }"return 28 for Feb - or 29 for Feb...";
}
```

(2)
```
int FirstDayOfMo (int mo, int yr)
{... /* 1-1-1900 was MO ➡ COUNT FORWARD! */
 "for every day since 1-1-1900 move
     weekday counter (0...6) up by 1"; /* in a modulo fashion */
 "return weekday";
}
```

(3)
```
void IndentFirstLine (int weekday)
{...
 "Indent first line properly based on weekday";
}
```

→ SEE FIG. 5-6: Detailed Implementation

Example:

JANUARY 1900

SU	MO	TU	WE	TH	FR	SA
(6)	(1)	(2)	(3)	(4)	(5)	(6)
	1	2	3	4	5	6
7	8	9	10	11	12	13
14						20
21						27
28	29	30	31			

APRIL 2014

SU	MO	TU	WE	TH	FR	SA
		1	2	3	4	5
6		Indnt by 2				
13						
20						
27	28	29	30			

→ Leap-year test:

```
bool IsLeapYr (int yr)
{ return (     (yr%.400 == 0)
          || ((yr%4 == 0) && (yr%100 != 0))
        );
}
```

CHAP. 6 ALGORITHMS "Recipe how to solve problem computationally"

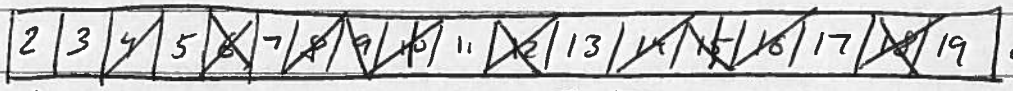
DEF.
- (1) UNAMBIGUOUS – must consider all cases
- (2) EFFECTIVE – all cases can effectively carried out computationally
- (3) FINITE (TIME) – must terminate

(REM: Also: Want to design more and more efficient algorithms!)

EX: "IS N A PRIME NUMBER?"

(DEF: "N > 1 ∧ CAN DIVIDE N ONLY BY 1 AND ITSELF")

→ 2, 3, 5, 7, 11, 13, 17, 19, ... are PRIME.



1) ELIMINATE MULTIPLES OF 2
2) " " " OF 3