L#6 Problem Solving Tools and Program Design in Computer Programming: The Repetition Structure: Counter, Accumulator, Flag

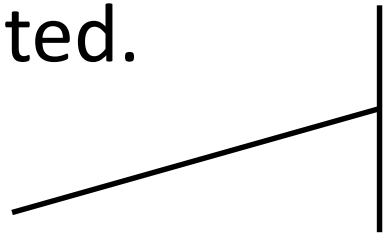
Problems with Solutions Requiring Repetition

PART 4

Aug 2019

Counter

A variable with the function of counting the number of times a given task or event is executed.



- write counter @ both sides of '=' sign
- place statement within a loop
- initialize counter before loop

counter=counter + value

E.g.:
counter=counter+1;
counter=counter+3;
counter=counter-1;

The counter can be increment by any value, e.g., 1, 2, -1, 3, 0.1, etc., positive or negative. There may be situations in which the increment take different values depending on the logic of the program.

Compute the frequency of letter grades A, B, C, D and F of a class group composed by N students

```
INPUT N              % e.g., the class group has N=30 students
SET countA← 0, countB← 0, countC←0, countD←0, countF←0, K←1

WHILE K<=N           % Loops if condition is TRUE
    INPUT score

    IF score>=90
       countA←countA + 1
    ELSEIF score>=80
       countB←countB + 1
    ELSEIF  score>=70
       countC←countC + 1
    ELSEIF score>=60
       countD←countD + 1
    ELSE
       countF←countF + 1


    K←K+1

PRINT  countA+“ students got A”+ countB+” students got B”
PRINT  countC+” student got C”+countD+” students got D”
PRINT  countF+” student got F”
```

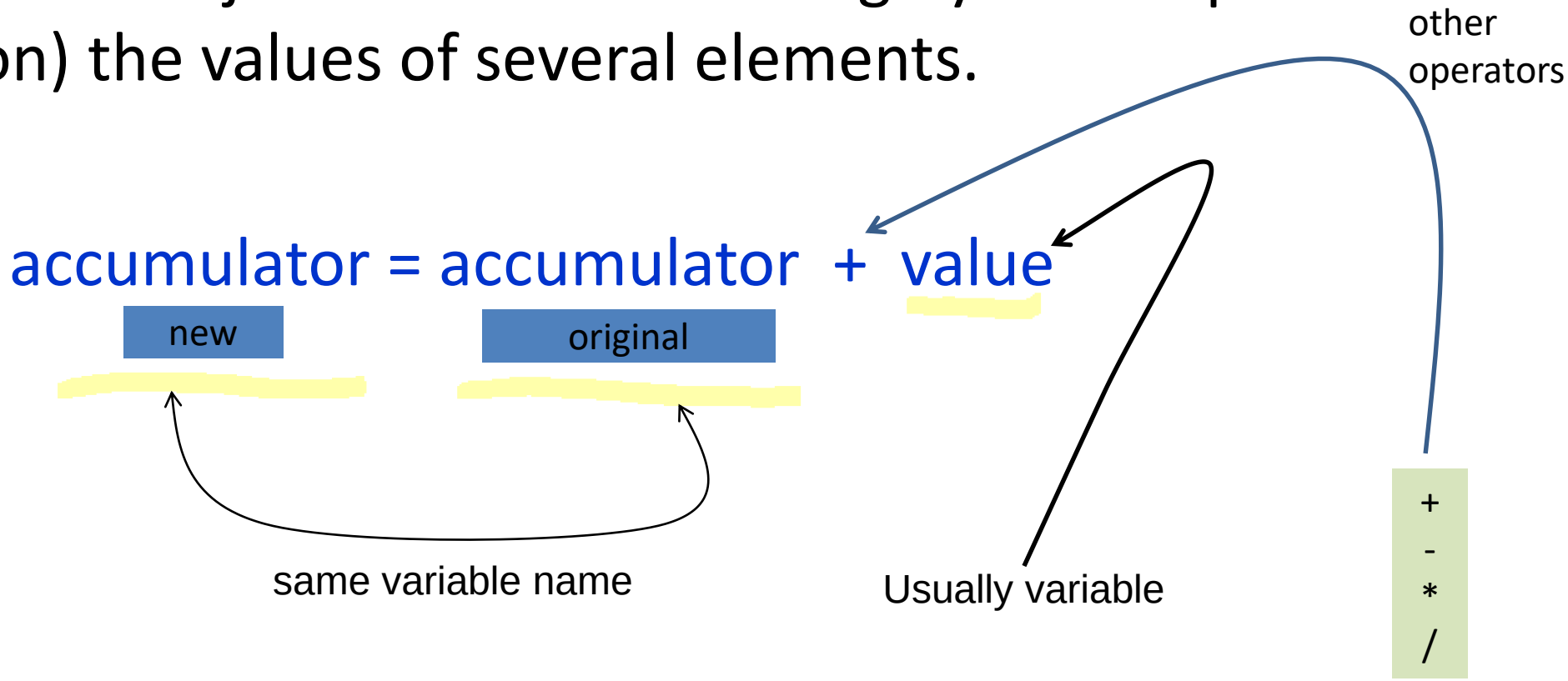
initialize counters

Counter Variable

- How many counters?
- What’s the purpose of K?

Accumulator

A variable with the objective of accumulating by some operation (e.g., addition) the values of several elements.

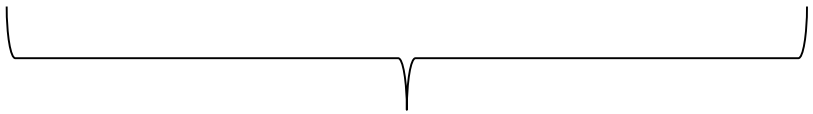


The values accumulated are usually variables.

Example

Write a small program to accumulate the values of the following elements of x:

x=[1,3,5,7,9,11,13,15,17,19]



10 elements

Three different solutions

```
SET c =1, x = 1, s = 0

WHILE c <=10
   s = s + x
   x = x +2
   c = c +1

PRINT “Sum is “ & s

# c = counter
# s = suma
```

```
SET x = 1, s = 0

WHILE x<=19
   s = s + x
   x = x +2

PRINT “Sum is “ & s
```

x is the values generator,
s is the accumulator,
c counts the iterations and controls the looping

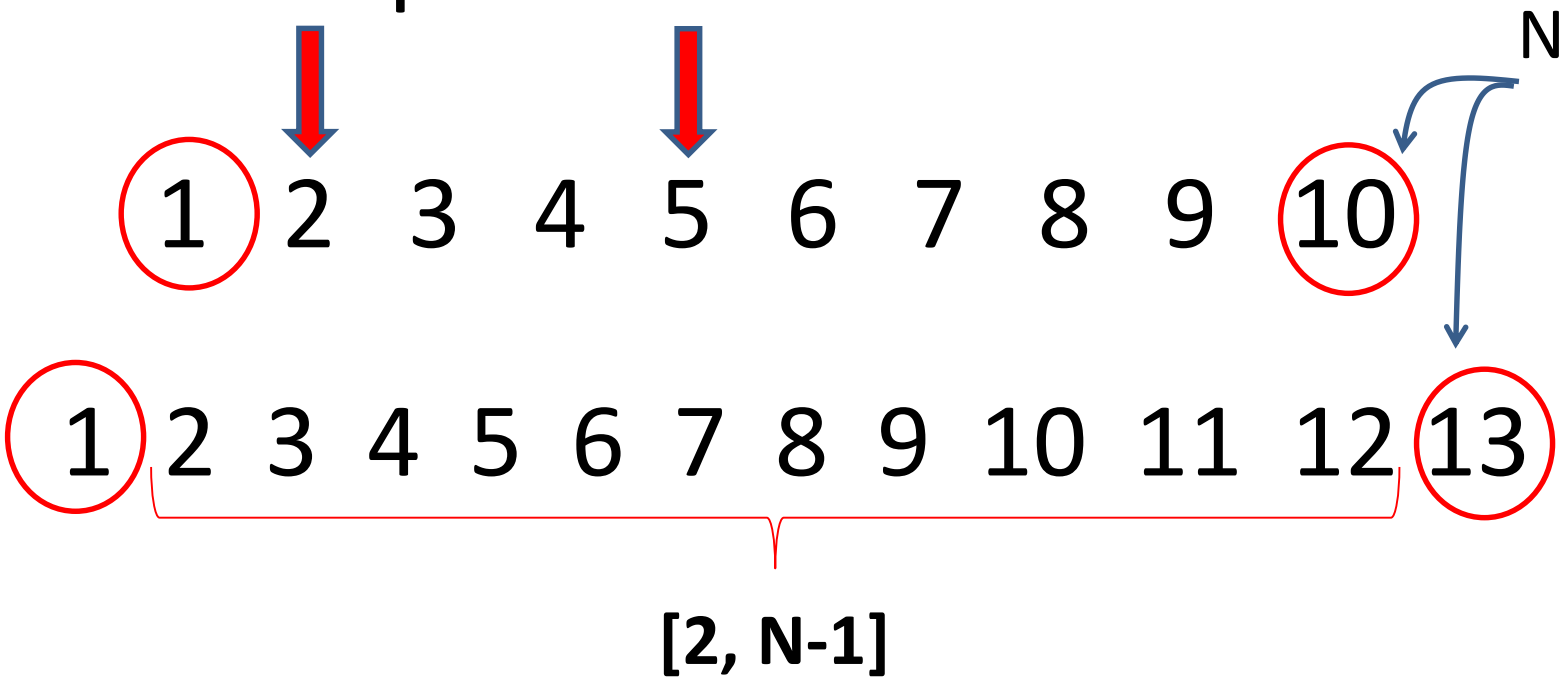
```
SET c =1, s = 0
WHILE c<=10
        INPUT x
        s = s + x
        c = c+1

   PRINT “Sum is “ + s

# Solve a more general problem of a list of 10 elements without any pattern
```

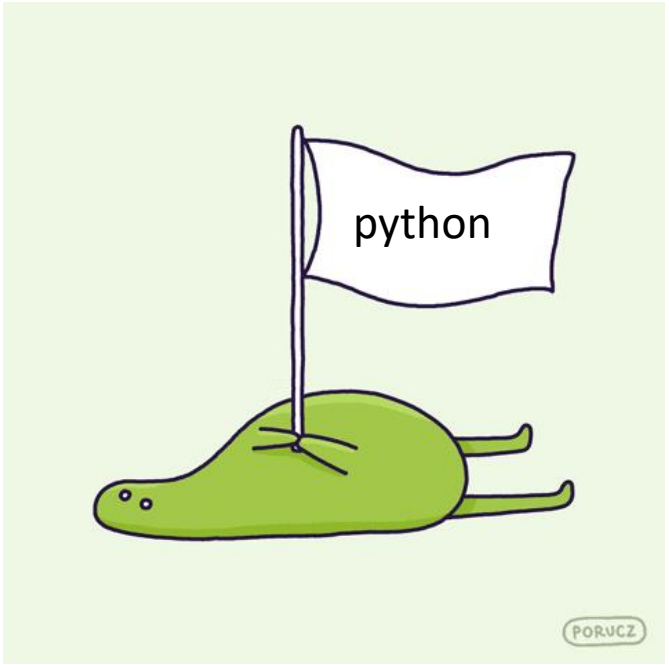
Flag (or Sentinel)

- A Flag is used to know the status of a block of program. It is also used to signal the termination of a loop or recursive algorithm

- It may have two or more predefined states either, e.g.:(Yes or No), (0,1), (True or False) (‘A’ or ‘B’), (1,2,3), etc.

Example-1: is N prime? 10? 13?





Consider a program which test if a number N is prime. Primes are positive integers which have no other divisors except 1 and itself. When a number has more than two factors it is called a composite number (no prime).

Example-1: is N prime?

Flag=1 ➔ prime
Flag=0 ➔ no prime

```
SET flag =1, k=2
INPUT N                    % N is an integer number different than 1 or 2
WHILE k<=(N-1)
      IF N % k ==0
        flag=0

      k=k+1

IF flag==1
      PRINT N & " is prime"
ELSE
      PRINT N & "is not prime"
```

Flag is set initially to one (1), which means we assume the number is prime until we demonstrate otherwise.

For example, initially set flag=1, which assumes the number is prime, if the program finds it is not it will change it to flag=0, otherwise flag stays 1. And finally for the output, the program checks the status of flag if it is one then reports prime otherwise reports not prime.

Example-2: is N prime?

```
SET flag =0
INPUT N     # N is an integer number different than 1 or 2
        IF (N % 1 ==0)   AND    (N % N==0)
          flag=1


IF flag==1
        PRINT N & " is prime"
ELSE
        PRINT N & "is not prime"
```

This algorithm is wrong and was proposed by someone. It is not enough to test if N is divisible by 1 and N to claim is prime as all number are.

Example-2: Is Juan del Pueblo in the list?

```
SET flag =0, k=0
INPUT N, name                    # N [names on list], name [first name from list]
WHILE k<=N
        IF name ==“Juan del Pueblo”
          flag=1
        ELSE
          INPUT name    # input the next name, and the next name, and the next name...
        k=k+1



IF flag==1
        PRINT “Juan del Pueblo is in the list”
ELSE
        PRINT “Sorry, Juan del Pueblo is not in the list”
```

Program searches a name of a person from a list. Initially flag=0, if the user enters the name of the person and it is present in the list, program changes flag=1, otherwise flag stays (flag=0). And finally we just check the status of flag if it is 1 then the name is present and if it is zero means the name was not found.

Suggested Exercise

- Now, on your own, work the recommended exercises: A, B, C in L#5 through the steps:
- Design the program, constructing an algorithm using decomposition (identify the need of a loop and work few iterations by paper and pencil), flowcharting, pseudocode and trace table for the following example.
- Next 7 (seven) slides present solutions for B and C.

B. Summing inputted integers until the user enters -1

Controlling the loop from outside the program, i.e., by user

Solve it by paper and pencil:

Solution of B

set sum=0

x = 5

sum=sum+x=0+5=5

---

x = 9

sum=sum+x=5+9=13

---

x = 3

sum=sum+x=13+3=16

---

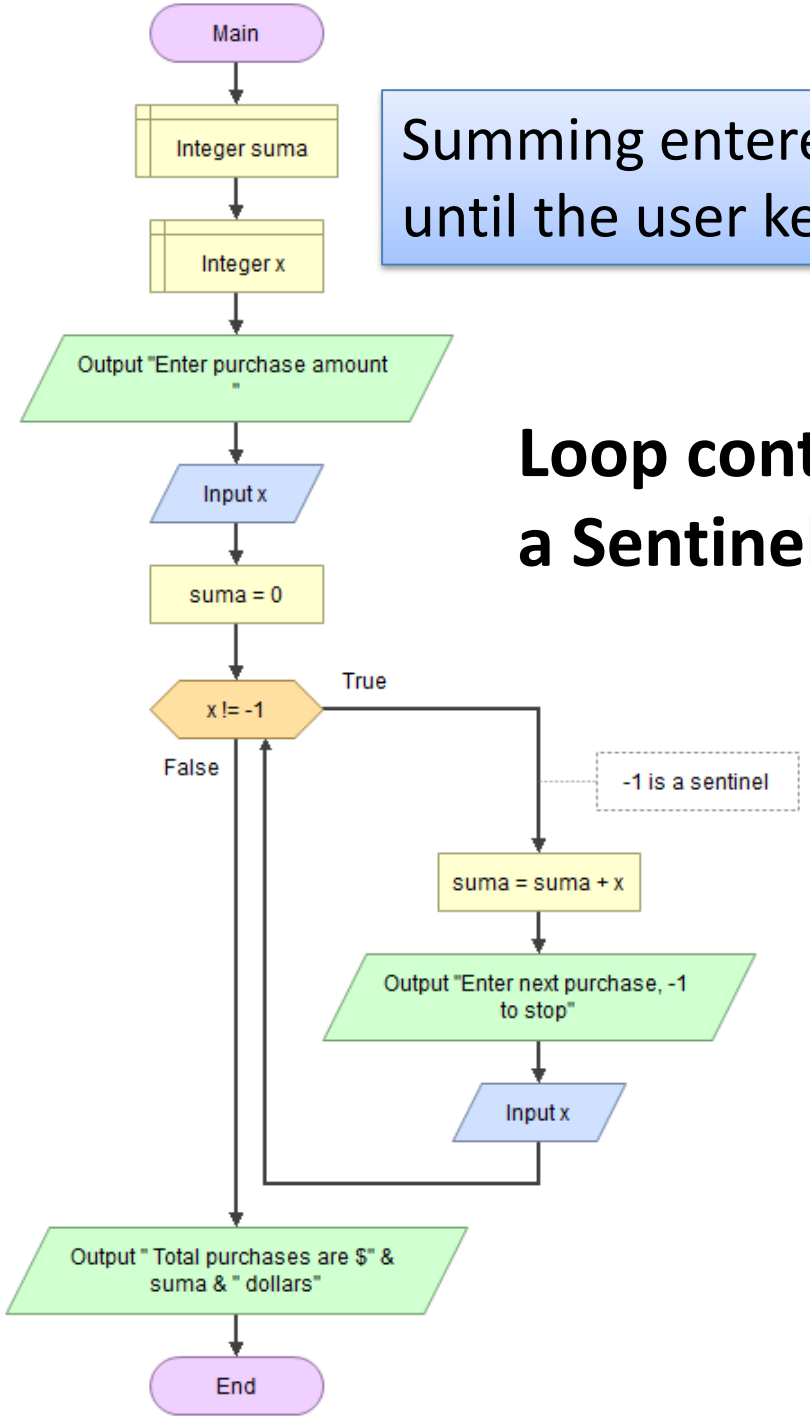
x = 6

sum=sum+x=16+6=21

---

x = -1

loop STOPS !!!



Summing entered integers until the user keys -1

Loop controlled by a Sentinel

```
SET suma=0
INPUT x
WHILE  x != -1
      suma = suma + x
      INPUT  next x, “enter -1 to stop”

PRINT suma
```

```
\\ Program written with LARP
START
   suma=0
   READ x
   WHILE  x != -1  DO
     suma = suma + x
     QUERY 'Enter a new x or -1 to stop ', x
   ENDWHILE
   WRITE 'The suma is ', suma
END
```

```
SET suma=0
INPUT x
WHILE  x != -1
      suma = suma + x
      INPUT  next x, “enter -1 to stop”

PRINT suma
```

B

Trace Table

Assume the user entered x values: 5, 3, 7, 10, 13, -1

SET	suma=	0				
INPUT	x	5	% assumed value			
LOOP	iter	x = -1	suma	x		
	1	TRUE	5	3	% assumed value	
	2	TRUE	8	7	% assumed value	
	3	TRUE	15	10	% assumed value	
	4	TRUE	25	13	% assumed value	
	5	TRUE	38	-1	% assumed value	
	EXIT	FALSE				
PRINT	suma=	38				

Controlling the loop from outside the program, i.e., by user

C

Next four slides present the solution of problem C in Part 3

Construct or Decompose the looping by reviewing the iteration sequence

```
INPUT Year
SET Decision = "noquit"
WHILE Decision != "quit"
      INPUT BirthYear
      Age = Year – BirthYear
      PRINT Age
      PRINT "Write quit or noquit"
      INPUT Decision
```

Computation for some friends

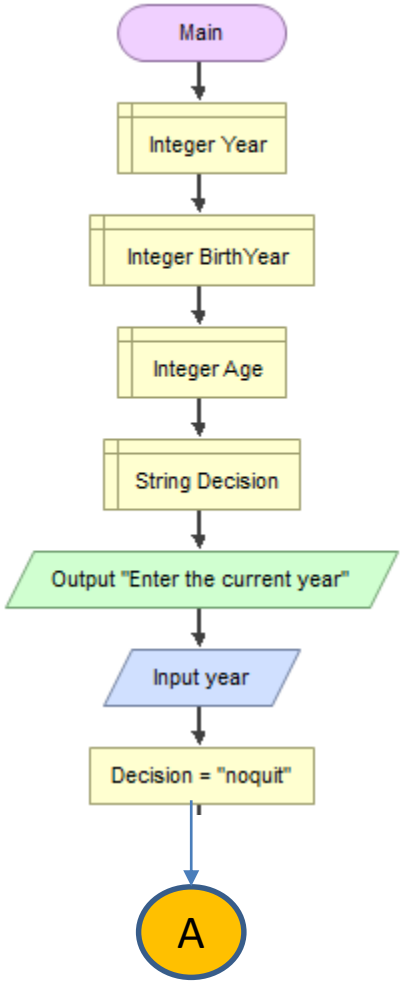
```
Year=2017
BirthYear = 1998
Age = 2017-1998=19
PRINT Age=19
----------------------------
BirthYear = 1997
Age = 2017-1997=20
PRINT Age=20
----------------------------
BirthYear = 1999
Age = 2017-1999=18
PRINT Age=18
```

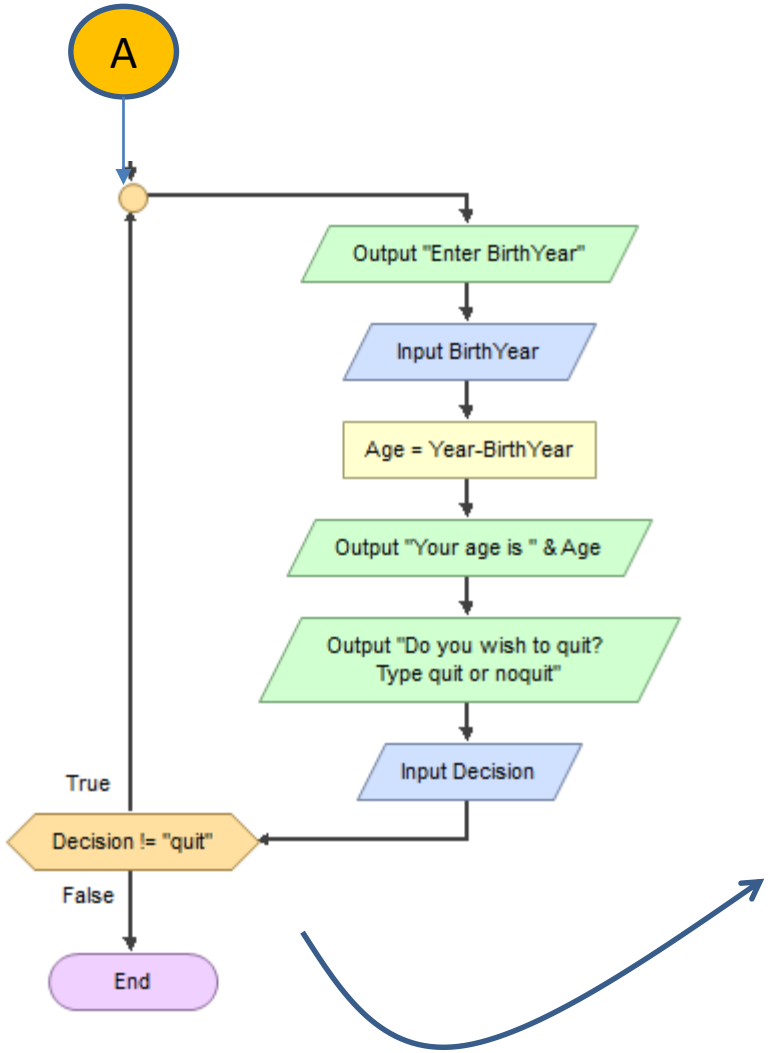
C. The user enters in the current year and then his/her birth year. Your program computes the user's age. Perform this task again for all her/his friends.

C

The user enters in the current year and then his/her birth year. Your program computes the users age. Perform this task again for all her/his friends.

See **Ex-14 Age Calculator in the Loop section of SampleExercises**






```
INPUT Year
SET Decision to “noquit”
LOOP
      INPUT BirthYear
      Age = Year – BirthYear
      PRINT Age
      PRINT “Write quit or to stop”
      INPUT Decision
DO WHILE Decision != “quit”
```

OUTPUT
Enter the current year 2017
Enter BirthYear 1998
Your age is 19
Do you wish to quit? Type quit or noquit
noquit
Enter BirthYear 1960
Your age is 57
Do you wish to quit? Type quit or noquit
quit

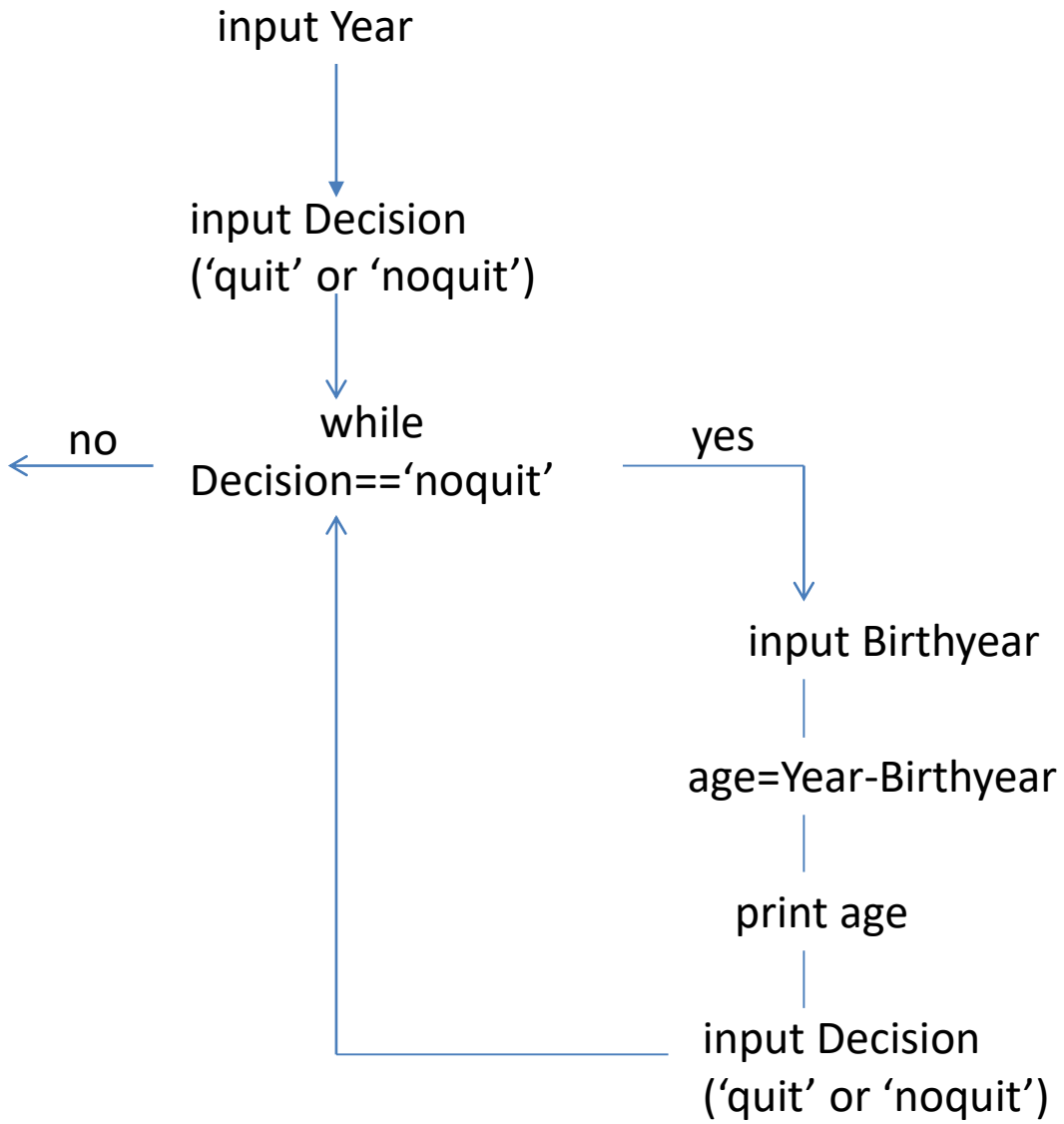
```
INPUT Year
SET Decision to “noquit”
WHILE Decision != “quit”
      INPUT BirthYear
      Age = Year – BirthYear
      PRINT Age
      PRINT “Write quit or noquit”
      INPUT Decision
```

Trace Table

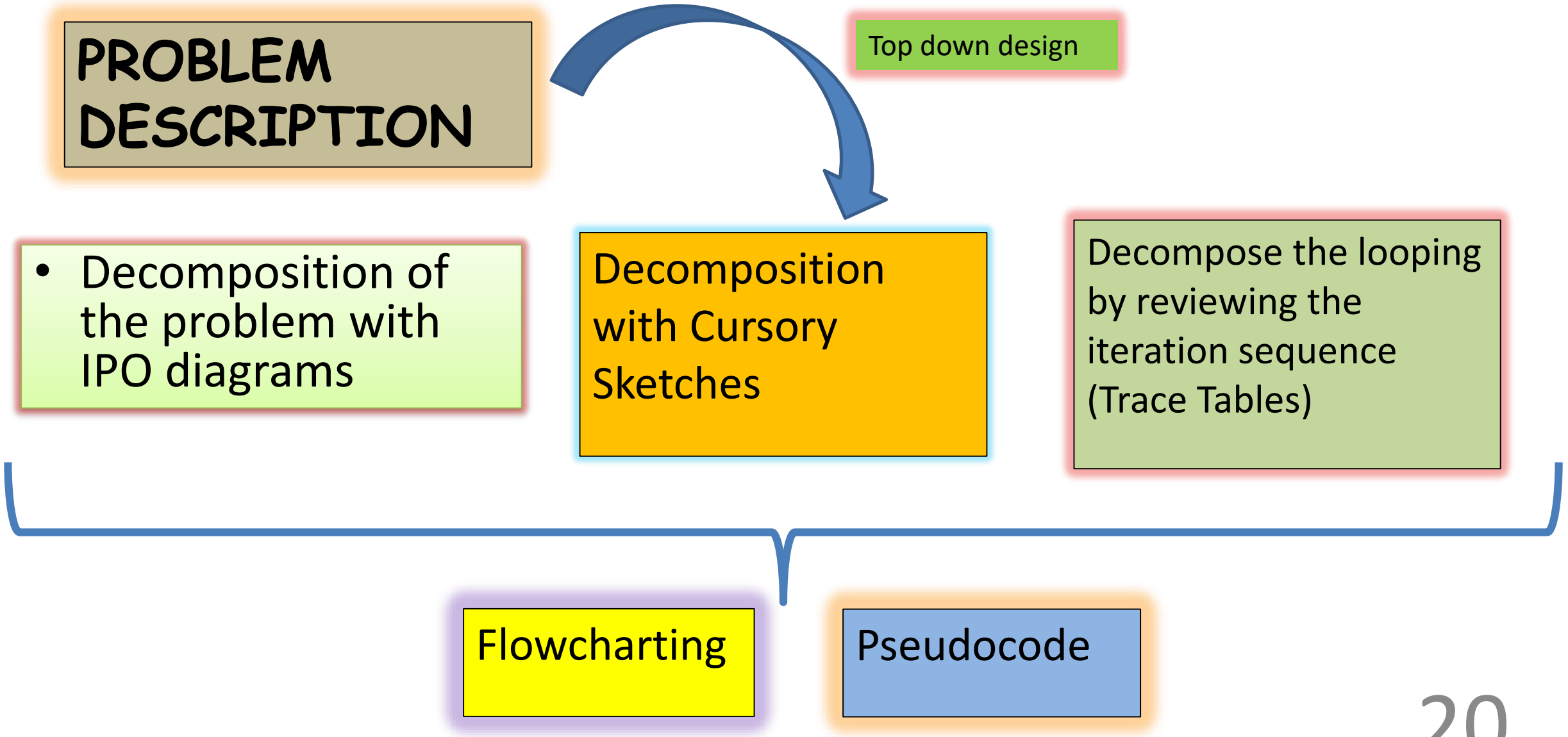
Decision != “quit”

iter	YEAR	Decision	Decision = "quit" ?	BirthYear	Age	print Age	Decision
none	2017	"noquit"	FALSE TRUE	1997	20	20	"noquit"
1		"noquit"	FALSE TRUE	1998	19	19	"noquit"
2		"noquit"	FALSE TRUE	1999	18	18	"noquit"
		"quit"	TRUE FALSE				

Cursory Sketch



Synthesis of Tools for (Programming) Problem Solving



Programming Structures

- Sequence
- Selection (Decision)
- Repetition (Loops)

Suggested Class Examples

- Eucliden GCD: Euclidean GCD.fprg;