CS Teaching Academy

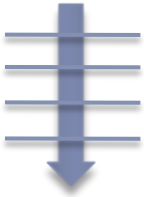
Chapter 5, Module 1: Program Structure

Objectives

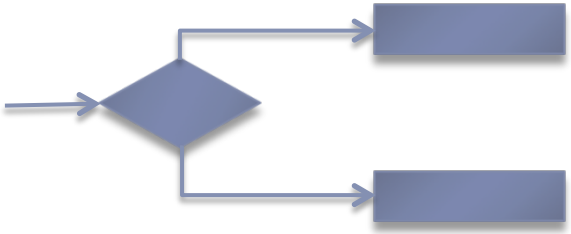
- Understand the essential elements of program structure
- Be able to interpret recursive procedures

Essential Elements of Program Structure

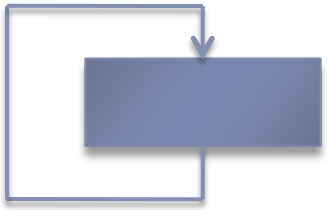
1. Sequence



2. Selection



3. Iteration
   (or Recursion)



***Böhm-Jacopini Program Structure theorem**, 1966*

Sequence

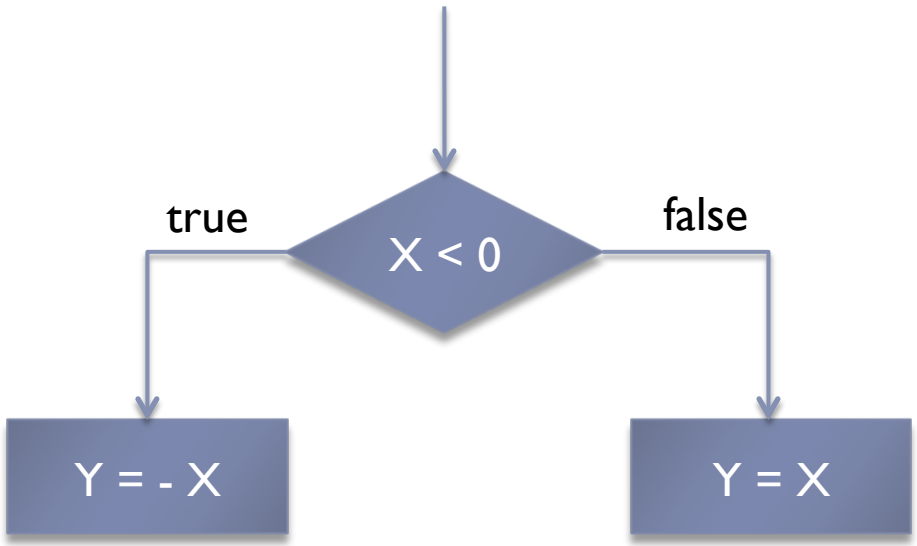
- Sequential Execution of Instructions

Step	Instruction
1.	fetch X to R1
2.	fetch Y to R2
3.	add R2 to R1
4.	store R1 to X

- Sequential execution reflects the *von Neumann* architecture

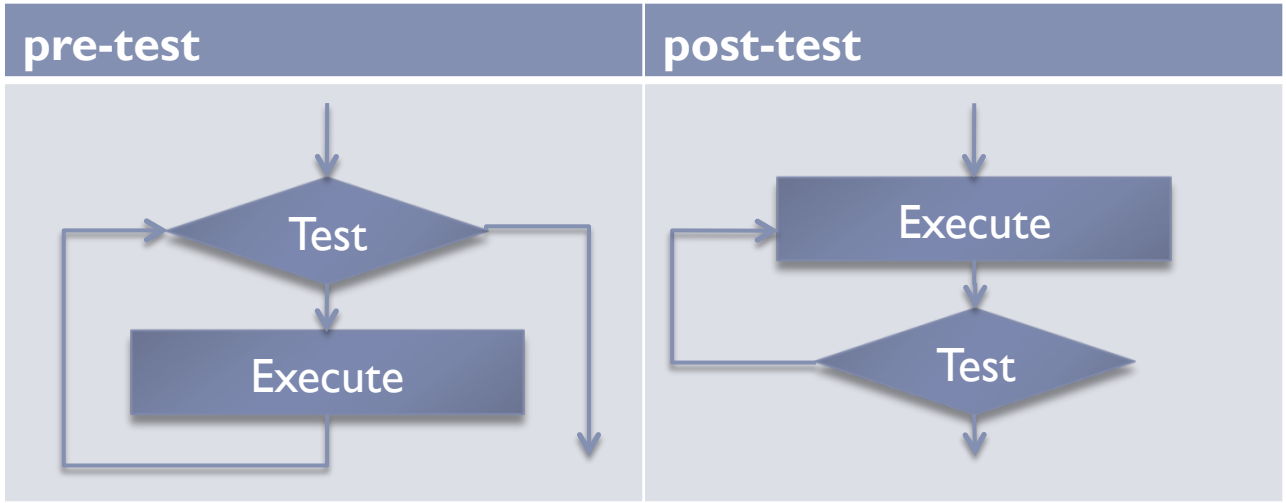
Selection

- Choose a control flow, depending upon a variable value



Iteration

- Repeated execution of a set of instructions



- Test conditions can be stated in the form
  - while-true while X < 0
  - until-true (while false) until X < 0

Example Program

```
double averageArray(int[] array) {
  int sum = 0, count = 0;
  for (int i=0; i<array.length; i++) {
    sum += array[i];
    count++;
  }
  if (count > 0) {
    return ((double)sum) / count;
  }
  else {
    return 0.0;
  }
}
```

sequence

iteration

selection

Recursion

- A computationally equivalent alternative to iteration
- Example: indexOf (search for a character in a string)
  - indexOf('a', "dcba") = 3, indexOf('a', "xyz") = -1

Iterative	Recursive

int indexOf(char c, String s) {
for (int i=0; i<s.length(); i++) {
if (c == s.charAt(i)) {
return i;
}
}
return -1;
}
	int indexOf(char c, String s) {
	if (s.length == 0) {
	return -1;
	}
	if (c == s.charAt(0)) {
	return 1;
	}
	return 1 + indexOf(c, s.subString(1));
	}

Anatomy of a Recursive Procedure

- Every recursive procedure must have two parts:

1. Base case(s): the conditions for terminating recursion
2. Recursive case: a call to itself with reduced parameters

```
int factorial(int n) {
  if (n <= 0) {
    return 1;
  }
  return 1 * factorial(n-1);
}
```

Base case

Recursive case

Anatomy of a Recursive Procedure

- Every recursive procedure must have two parts:

1. Base case(s): the conditions for terminating recursion
2. Recursive case: a call to itself, with reduced parameters

```
int indexOf(char c, String s) {
  if (s.length == 0) {
    return -1;
  }
  if (c == s.charAt(0)) {
    return 1;
  }
  return 1 + indexOf(c, s.subString(1));
}
```

Base cases

Recursive case

Recursion and Functional Languages

- Recursion is more natural in functional languages, such as LISP

```
(defun sum (list)
  (cond
    ( (null list) 0 )
    ( else (+ (car list) (sum (cdr list))))
  )
)
```

Summary

- Programming languages must support three types of control structure
  - sequence
  - selection
  - iteration (or recursion)
- Recursive procedures must have
  - a base case
  - a recursive case
- Recursion is more natural in functional languages