LECTURER: TAI LE QUY

ALGORITHMS, DATA STRUCTURES, AND PROGRAMMING LANGUAGES

TOPIC OUTLINE

TOPIC OUTLINE

UNIT 6

PROGRAMMING LANGUAGES

STUDY GOALS

- Understand various programming paradigms.
- Understand the process of execution of a program.
- Classify programming languages based on paradigms.
- Analyze programs from the point of view of syntax, semantics, and pragmatics.
- Infer types of variables using type system rules.

EXPLAIN SIMPLY

1. Describe briefly the different paradigms of programming.
2. Explain the steps involved in the execution of a computer program.
3. Explain short-circuit evaluation.

PROGRAMMING PARADIGMS

IMPERATIVE PROGRAMMING
Command or statement driven involving sequence of commands for the computer to perform.

OBJECT-ORIENTED PROGRAMMING
Designed around data items or objects. Language supports classes to create the objects.

LOGIC PROGRAMMING
Goal of the computation is specified rather than the algorithm to reach the goal.

PROGRAMMING PARADIGMS
Different styles of programming

FUNCTIONAL PROGRAMMING
Models computation as a collection of mathematical functions.

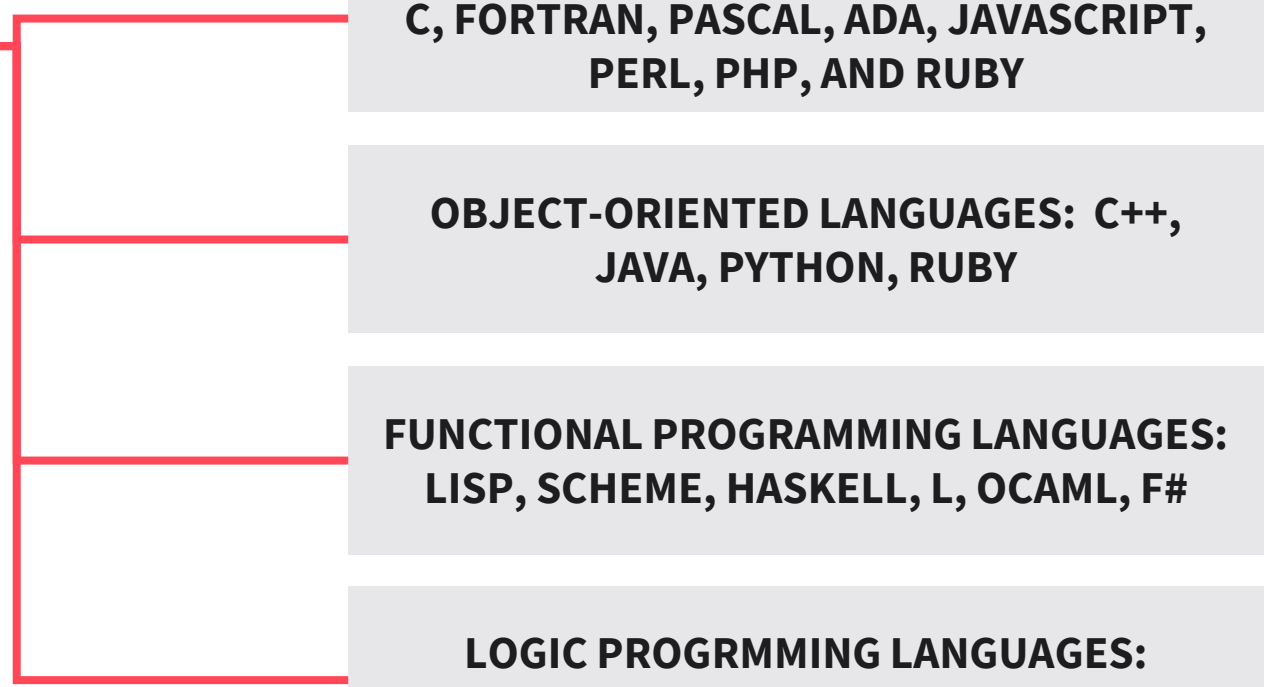
DATA STREAM PROGRAMMING
Based on processing of a data stream which is a sequence of data items available one at a time.

EVENT DRIVEN PROGRAMMING
Control flow is determined by events. Used in GUI programming.

PROGRAMMING LANGUAGES CLASSIFIED BY PARADIGM

TYPES OF PROGRAMMING LANGUAGES

There is no standard classification of programming languages. A paradigm-based classification is common and the most natural. The main paradigms of programming are Imperative Programming, Object-Oriented Programming, Functional Programming, and Logic Programming. Many languages are multi-paradigm, languages often primarily support one paradigm as opposed to others.



LANGUAGE SUPORT FOR PROGRAMMING PARADIGMS

OBJECT-ORIENTED PARADIGM
Classes, objects, inheritance, polymorphism, access control rules.

IMPERATIVE PARADIGM
Procedural abstractions, variable declarations, expressions, control structures for selection, iteration, and branching operations.

LANGUAGE SUPPORT
Each programming paradigm requires certain types of features to be present in a language for the language to be able to support programming in that paradigm.

FUNCTIONAL PROGRAMMING PARADIGM
Support for recursion, mechanisms to create complex functions for simple ones, function application operation.

LOGIC PROGRAMMING PARADIGM
Representation of rules and constraint using symbolic logic, means to specify inference mechanism.

EXECUTION OF PROGRAMS

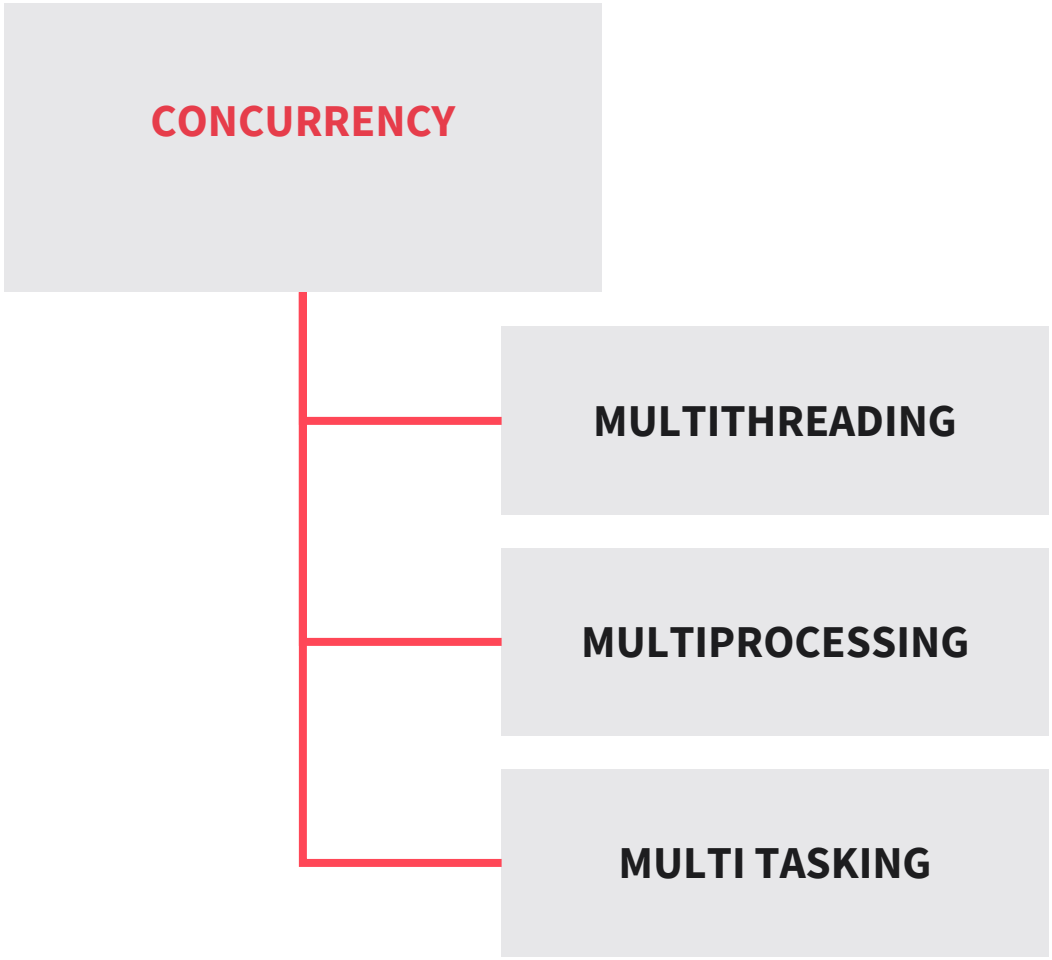
- Execution of program begins from first instruction.
- CPU outputs address of memory location containing next instruction.
- Decoding logic selects RAM chip and location allocated to the address.
- Code for instruction retrieved from the RAM into the CPU.
- Data/operands on which the instruction must act fetched from RAM.
- Operands processed in CPU and partial results stored in data registers.
- These may be required to be written back into the RAM.
- The program counter is then incremented to the address of the next instruction.

EXECUTION OF PROGRAMS

- The Operating System …
  - … controls and monitors various system level activities.
  - … allocates shared resources to programs.
  - … schedules usage of these shared resources.
- Programs are converted to machine language.
- CPU runs program in a sequence of “fetch” and “execute” commands.
  - Control instructions determine order in which CPU executes instruction sequence.
  - CPU may go back to an earlier instruction in the sequence executing loops.
  - It may also skip certain statements, executing conditional instructions.

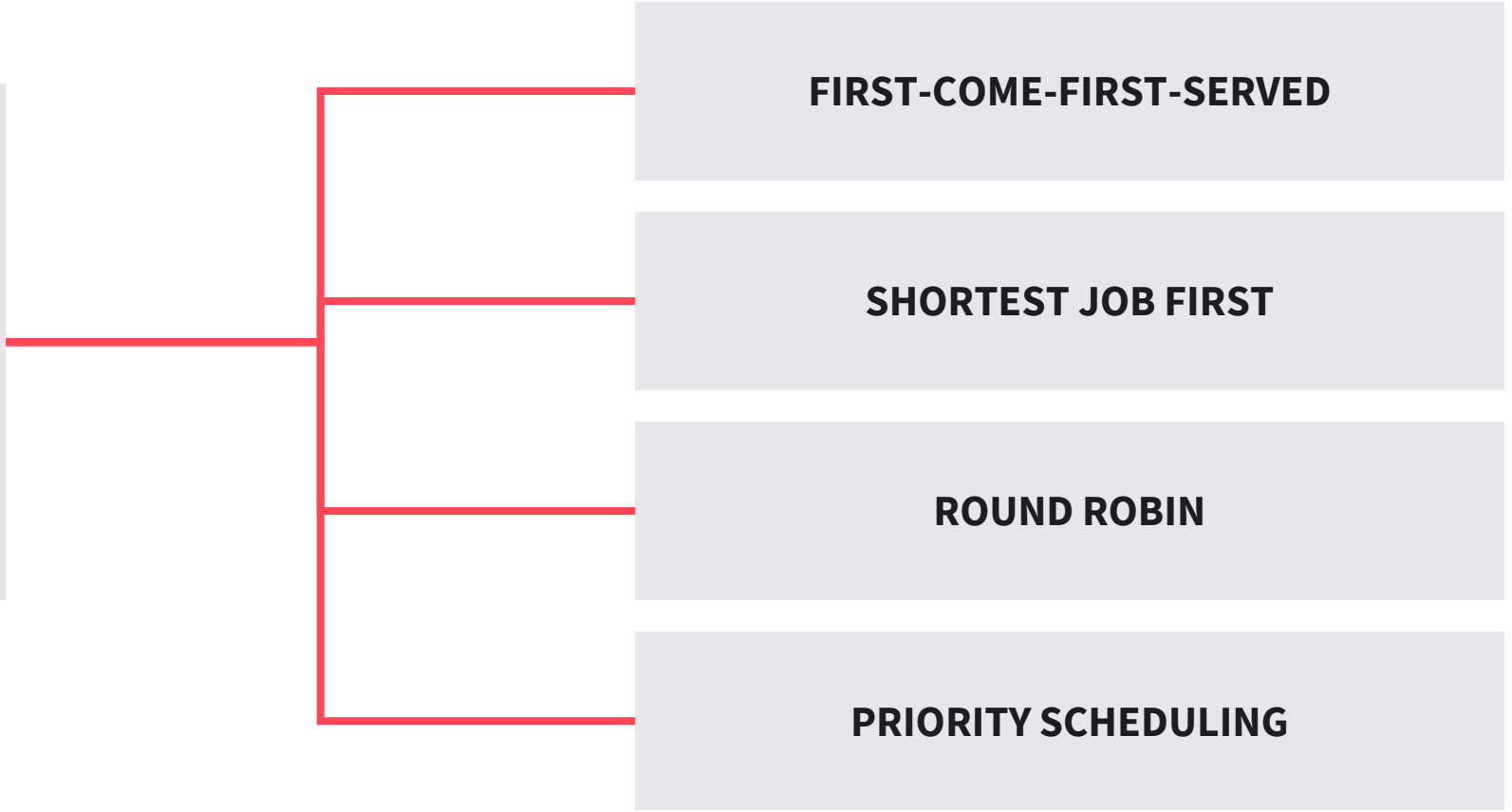
SUPPORT FOR CONCURRENCY

- The Operating System provides support for concurrency.
- This allows multiple programs to run together.
- Multithreading allows multiple tasks belonging to the same program.
- Multiprocessing allows programs to run on multiple processors.
- Multitasking allows multiple programs to work together on the same processor.



CPU SCHEDULING ALGORITHMS

CPU uses a scheduling algorithm to select a process to run. Various algorithms are selected based on optimization of throughput, CPU utilization, turnaround time, and waiting time. The common algorithms are shown here.



SYNTAX, SEMANTICS AND PRAGMATICS

- Programming languages are characterized by having form (Syntax), meaning (semantics) and what the statements in the language achieve in practice (pragmatics).
- The syntax of the while loop in Python is:
- ```
  while boolean_expression:
          statement
  ```
- The semantics of the while loop is that if the boolean expression is true, the statement will be executed. Multiple statements in the same block would all be executed in order they appear. Then the control returns to the boolean expression for evaluation again. This is repeated until the expression becomes false.
  ```

WHILE LOOP PRAGMATICS

```
n=13
while n < 20:
    n+=1
    print(n)
```

The condition of the while loop is satisfied for integers from n=13 to n=19 and these are printed.

```
n=13
while n > 20:
    n+=1
    print(n)
```

The condition of the while loop is never satisfied and so nothing is printed.

```
n=21
while n > 20:
    n+=1
    print(n)
```

The condition of the while loop is always satisfied and so it runs forever printing n=21, 22, 23, …

SHORT CIRCUIT EVALUATION
PYTHON EXAMPLE

```
x = 20
y = 0
x > 20 and x/y < 5
```

Although `y = 0`, since the condition `x > 20` fails, the division by zero in `x/y` never take place. The expression is successfully evaluated to `False`.

```
x = 21
y = 0
x > 20 and x/y < 5
```

The condition `x > 20` is satisfied and so the evaluation of `x/y` is attempted flagging a `ZeroDivisionError`.

```
x = 21
x > 20 and y!= 0
\\ and x/y < 5
```

The guard clause `y != 0` prevents the division by zero when `y = 0` and allows it otherwise.

Types and Type System

- Type System defines rules for creation and usage of types.
- Type checking involves checking of type system rules violations.
- Compatibility is ensured by explicit or implicit conversion.

Type Compatibility Rules

- Compatibility between operands of an operation
- Compatibility between two sides of an assignment
- Compatibility between actual and formal parameters of a function

REVIEW STUDY GOALS

- Understand various programming paradigms.
- Understand the process of execution of a program.
- Classify programming languages based on paradigms.
- Analyze programs from the point of view of syntax, semantics, and pragmatics.
- Infer types of variables using type system rules.

SESSION 5

TRANSFER TASK

TRANSFER TASKS

1. (a) Compare a program for linear search written in Haskell with that written in C.

   (b) Discuss the advantages of the solution in Haskell over that in C.

TRANSFER TASK

```c
#include <stdio.h>
int search(int arr[], int N, int x)
{
    for (int i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}
// Main program in C
int main(void)
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;
    int N = sizeof(arr) / sizeof(arr[0]);

    // Function call
    int result = search(arr, N, x);
    if (result == -1)
        printf("Element is not present in array")
    else
        printf("Element is present at index %d",
result);
    return 0;
}
```

Source of the code: https://101wiki.softlang.org/Concept:Linear_search https://www.geeksforgeeks.org/linear-search/

```haskell
--  linear search in Haskell
search :: Eq a => [a] -> a -> Bool
search [] _ = False
search (x:xs) y = x==y || search xs y
-- Illustrate linear search
main = do let input = [2,4,3,1,4]
print $ search input 1 -- True
print $ search input 5 -- False
```

TRANSFER TASK
PRESENTATION OF THE RESULTS

Please present your results.

The results will be discussed in plenary.

1. Which style of programming uses constructs such as lambda, map, reduce, and filter?
   a) Functional
   b) Logical
   c) Imperative
   d) Procedural

2. Which of the following is primarily a functional programming language?
   a) Java
   b) LISP
   c) C++
   d) Prolog

3. Which kind of type conversion does coercion involve?
    a) Explicit
    b) Implicit
    c) Both implicit and explicit
    d) Neither implicit nor explicit

LIST OF SOURCES

Pratt, T.W. & Zelkowitz, M.V. (2001). *Programming Languages: Design and Implementation* (4th ed.). Prentice-Hall.
Scott, M.L. (2016). *Programming Language Pragmatics* (4th ed.). Morgan Kaufmann.
Sebesta, R.W. (2016). *Concepts of Programming Languages* (11th ed.). Pearson.