Programming Paradigm

Introduction

Computer Program

- ❑ A computer program (or just a program or software) is a sequence of instructions, written to perform a specified task with a computer.

- ❑ A program is expected to behave in a predetermined manner. No matter how many times one program is run, the same result should be received for same set of data provided

Programming Language

❑ Machine Language – Strict binary form / byte code

❑ High Level Language – C, Cobol, C++, Java, LISP etc

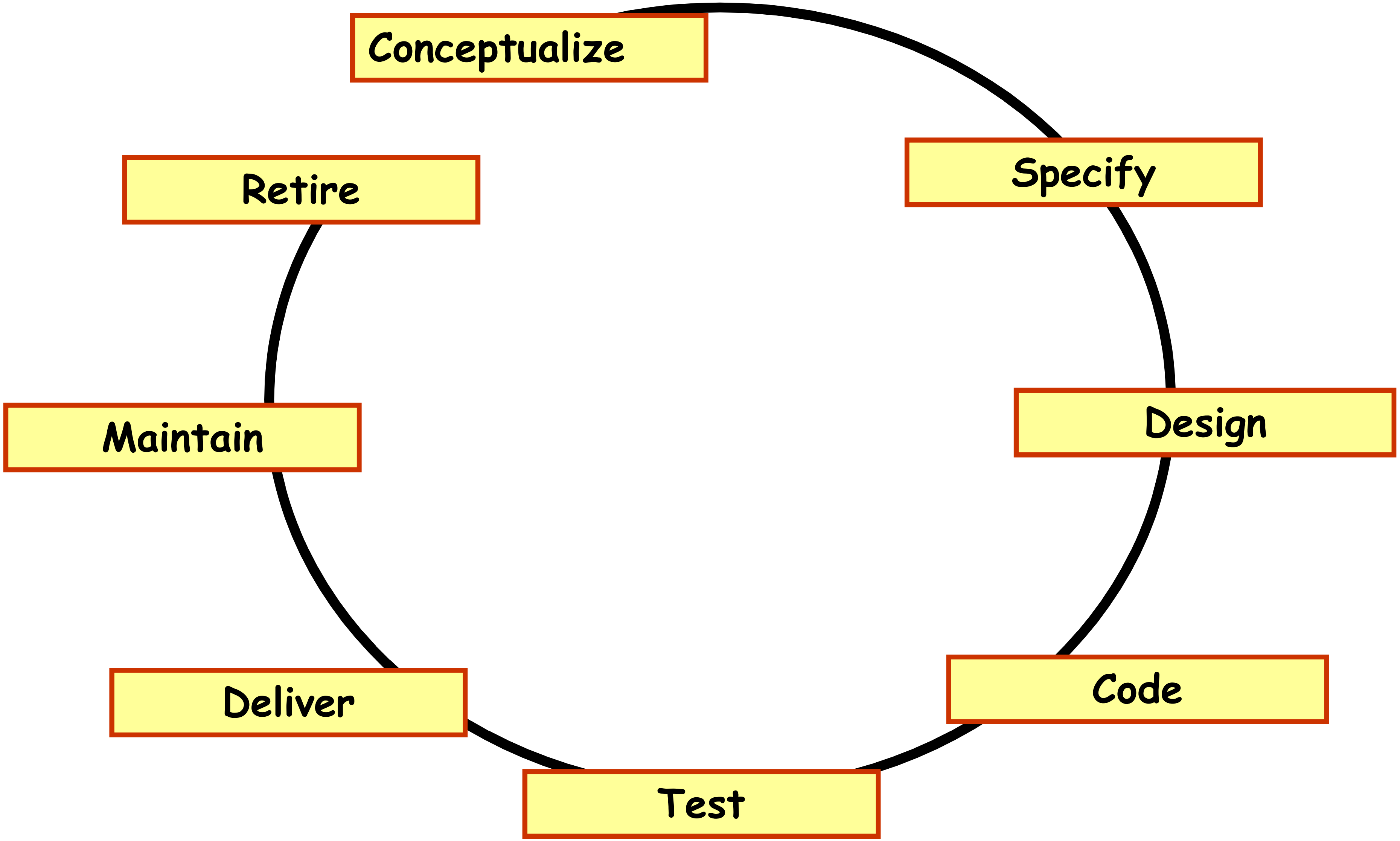
Note : High level languages are compiled or interpreted to Machine language before execution

Computer Programming

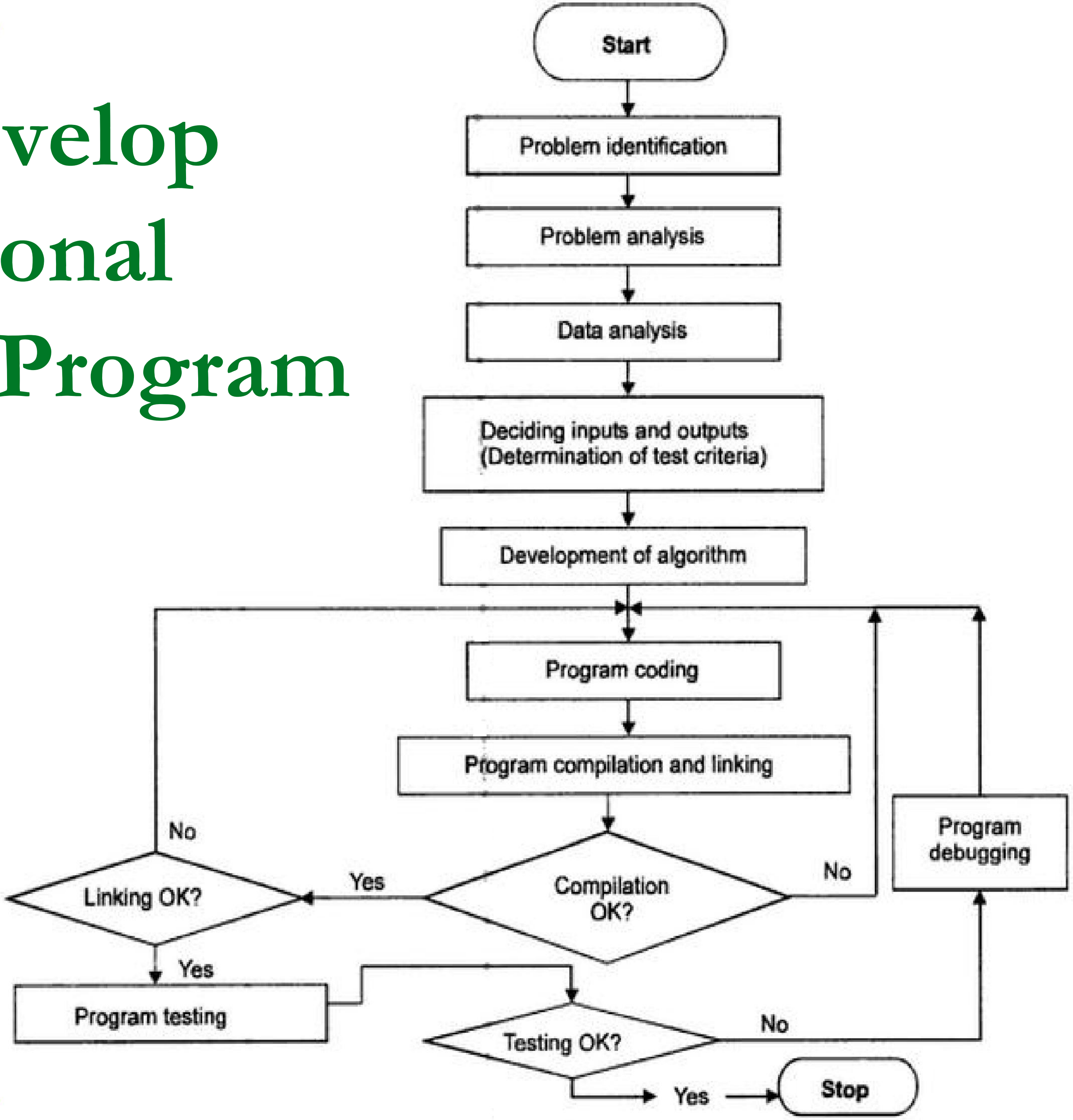
- ❑ Computer programming is the iterative process of writing or editing source code that can be executed in a computer

- ❑ It involves testing, analyzing, refining, and sometimes coordinating with other programmers on a jointly developed program

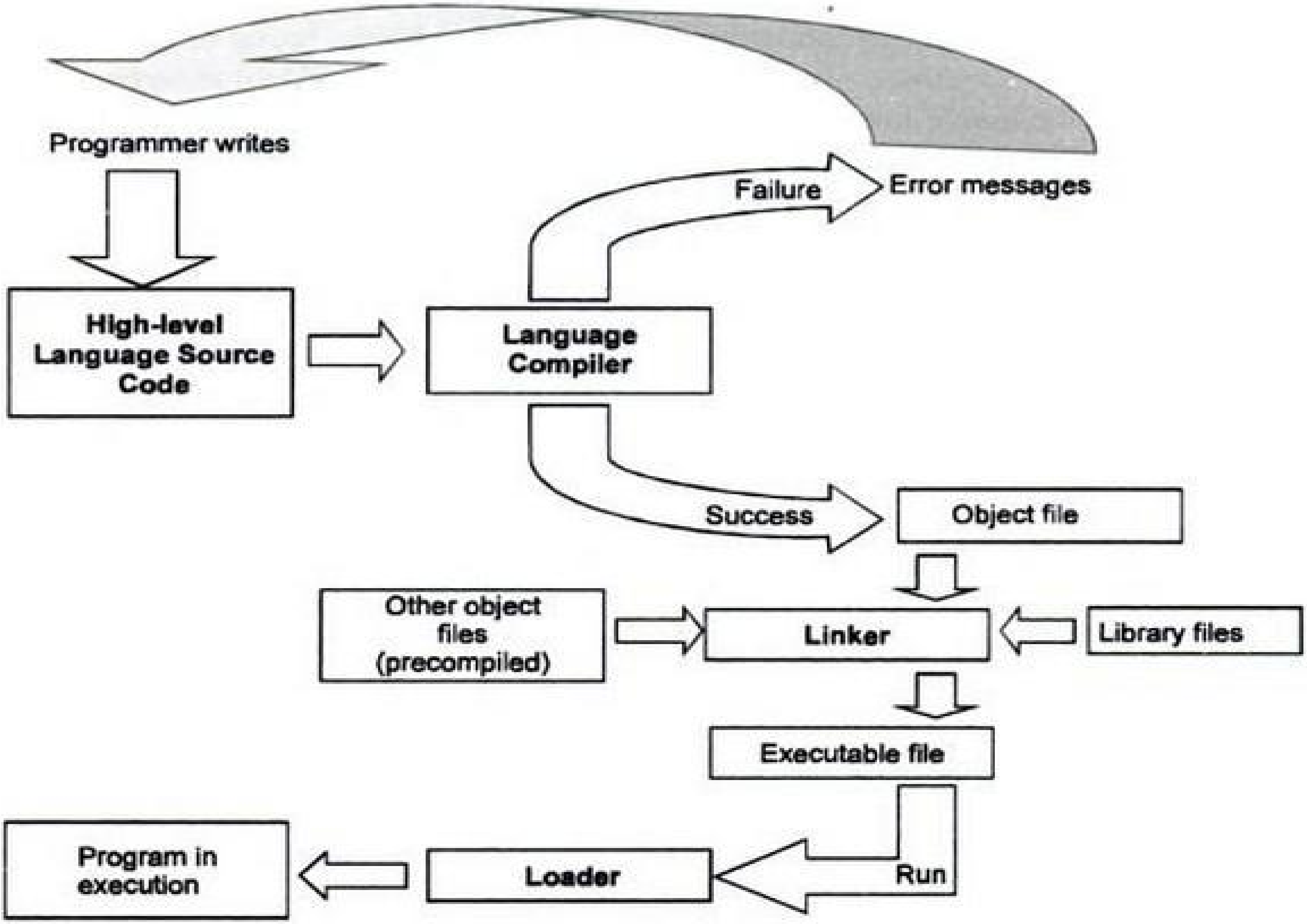
Life cycle of a Software



Steps to develop a Conventional Computer Program



Steps to execute a high level language program



Compilation of high level language program

Compilation is NOT a single step operation !!!

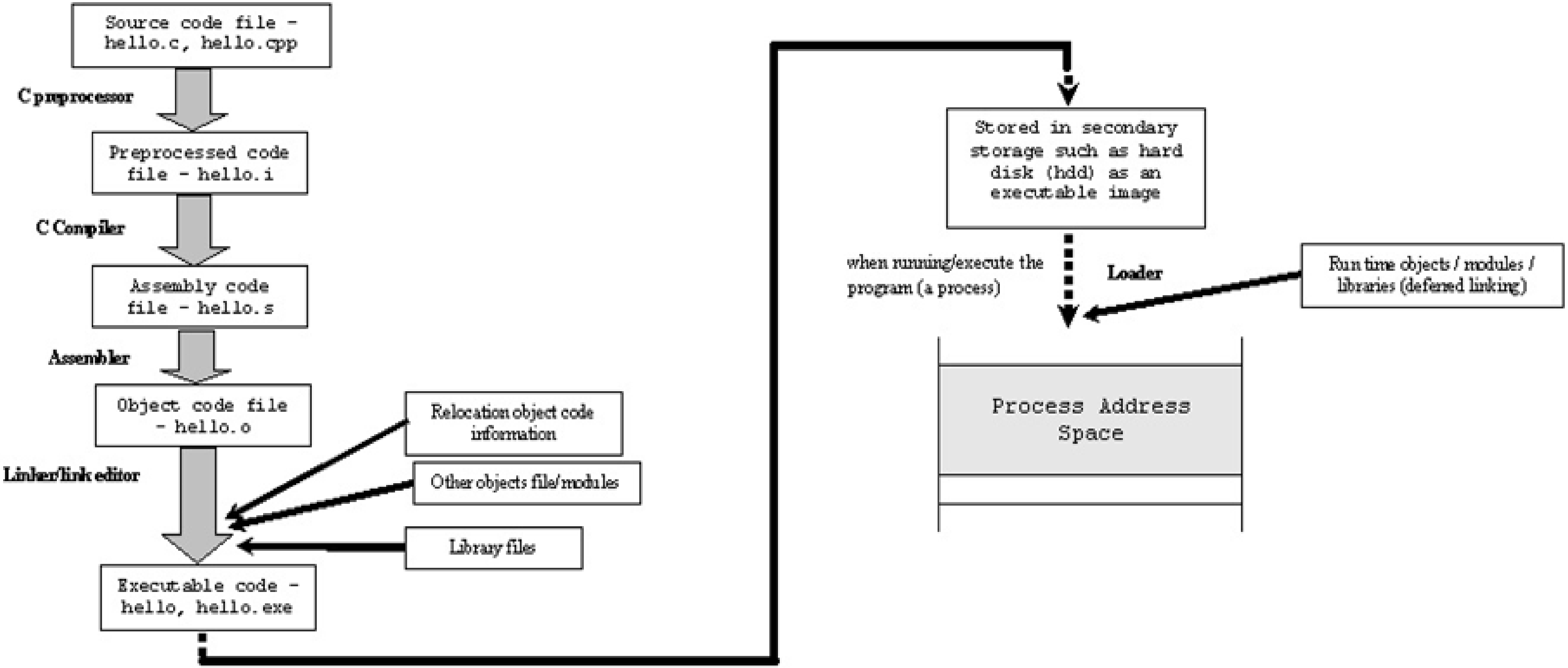

Source code file - hello.c, hello.cpp
C preprocessor
Preprocessed code file - hello.i
C Compiler
Assembly code file - hello.s
Assembler
Object code file - hello.o
Linker/link editor
Relocation object code information
Other objects file/modules
Library files
Executable code - hello, hello.exe
Stored in secondary storage such as hard disk (hdd) as an executable image
when running/execute the program (a process)
Loader
Run time objects / modules / libraries (deferred linking)
Process Address Space

Image source: The Internet

Compilation cont …

`$> gcc hello.c -o hello` : the gcc compiler reads the source file `hello.c` and translates it into an executable `hello`. The compilation is performed in four sequential phases by the compilation system

- a collection of four programs - preprocessor, compiler, assembler, and linker.

`$> gcc -save-temps hello.c -o hello` : The "`-save-temps`" option will preserve and save all temporary files created during the C compilation. It will generate four files in the same directory namely

- hello.i (Generated by pre-processor)
- hello.s (Generated by compiler)
- hello.o (Generated by assembler)
- hello (Generated by linker)

Compilation cont …

Also can be observed as follows:

Step 1 – Preprocessing : `cpp hello.c > hello.i`

Step 2 – Compilation : `gcc -S hello.i`

Step 3 – Assembly : `as hello.s -o hello.o`

Step 4 – Linking :

```
ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 /usr/lib64/crt1.o /usr/lib64/crti.o /usr/lib64/crtn.o helloworld.o /usr/lib/gcc/x86_64-redhat-linux/4.1.2/crtbegin.o -L /usr/lib/gcc/x86_64-redhat-linux/4.1.2/ -lgcc -lgcc_eh -lc -lgcc -lgcc_eh /usr/lib/gcc/x86_64-redhat-linux/4.1.2/crtend.o -o helloworld
```

(This can be different from platform to platform)

Programming Language Timeline

- **FlowMatic -** 1955 Grace Hopper UNIVAC
- **ForTran -** 1956 John Backus IBM
- **AlgOL -** 1958 ACM Language Committee
- **LISP -** 1958 John McCarthy MIT
- **CoBOL -** 1960 Committee on Data Systems Languages
- **BASIC -** 1964 John Kemeny & Thomas Kurtz Dartmouth
- **PL/I -** 1964 IBM Committee
- **Simula -** 1967 Norwegian Computing Center Kristen Nygaard & Ole-Johan Dahl
- **Logo -** 1968 Seymour Papert MIT
- **Pascal -** 1970 Nicklaus Wirth Switzerland
- **C -** 1972 Dennis Ritchie & Kenneth Thompson Bell Labs
- **Smalltalk** - 1972 Alan Kay Xerox PARC
- **ADA -** 1981 DOD
- **Objective C -** 1985 Brad Cox Stepstone Systems
- **C++** - 1986 Bjarne Stroustrup Bell Labs
- **Eiffel -** 1989 Bertrand Meyer France
- **Visual BASIC -** 1990 Microsoft
- **Delphi -** 1995 Borland
- **Object CoBOL -** 1995 MicroFocus
- **Java -** 1995 Sun Microsystems

Five Generations of Programming Languages

- ❑ First - Machine Languages
  - ➢ machine codes
- ❑ Second - Assembly Languages
  - ➢ symbolic assemblers
- ❑ Third - High Level Procedural Languages
  - ➢ (machine independent) imperative languages
- ❑ Fourth - Non-procedural Languages
  - ➢ domain specific application generators
- ❑ Fifth Natural Languages

Each generation is at a higher level of abstraction

How do Programming Languages Differ?

Common Constructs:

- basic data types (numbers, etc.);
- variables;
- expressions;
- statements;
- keywords;
- control constructs;
- procedures;
- comments;
- errors ...

Uncommon Constructs:

- type declarations;
- special types (strings,
- arrays, matrices,...);
- sequential execution;
- concurrency constructs;
- packages/modules;
- objects;
- general functions;
- generics;
- modifiable state;...

Language Styles …

Procedural Languages

- Individual statements
- FORTRAN, ALGOL60, ALGOL68, Cobol, Pascal, C, Ada

Functional Languages

- When you tell the computer to do something it does it
- LISP, Scheme, CLOS, ML, Haskell

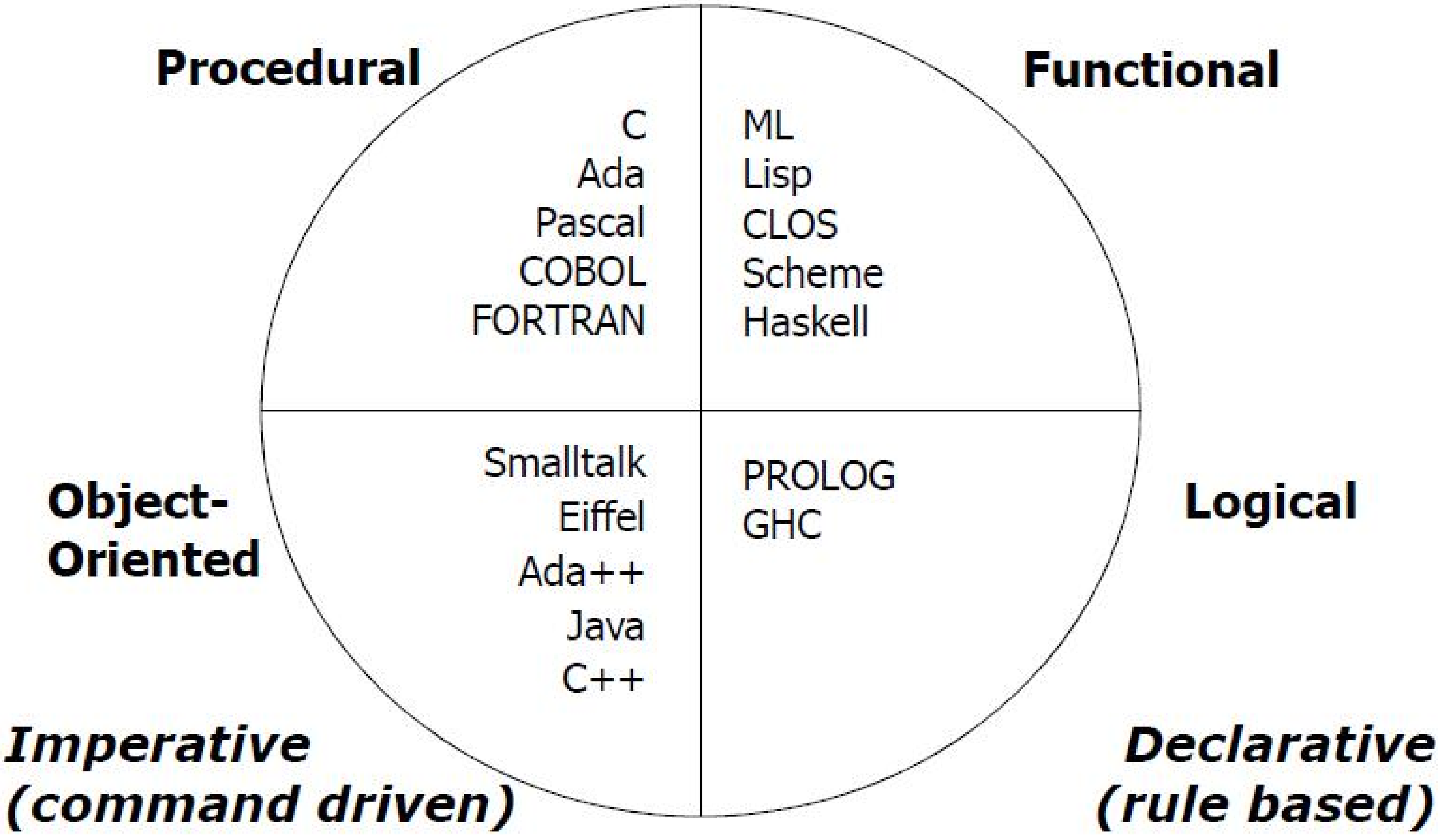
Logic Languages

- Inference engine that drives things
- Prolog, GHC

Object-oriented Languages

- Bring together data and operations
- Smalltalk, C++, Eiffel, Sather, Python, Ada95, Java, OCAML

… and Programming Paradigms



Procedural

C
Ada
Pascal
COBOL
FORTRAN

Functional

ML
Lisp
CLOS
Scheme
Haskell

Object-Oriented

Smalltalk
Eiffel
Ada++
Java
C++

Logical

PROLOG
GHC

Imperative (command driven)

Declarative (rule based)

Programming Paradigm

The basic structuring of thought underlying the programming activity

- ❑ Programming paradigm is a pattern or model of programming that derives the process of programming
- ❑ Every high level language has a paradigm that guides in a problem solving within a framework and gives a solution
    - ➢ It does not mean that all high level language are strictly following one particular programming paradigm
- ❑ Every Programming paradigm is a collection of conceptual patterns that control human thinking process to formulate the solution to a problem
- ❑ Different programming paradigms lead to different programming techniques

Programming Paradigm Contd…

Four main programming paradigms –

- ❑ Imperative - Program as a collection of statements and procedures affecting data (variables). FORTRAN, BASIC, COBOL, Pascal, C
- ❑ Functional - Program as a collection of mathematical functions. LISP, ML, Haskell
- ❑ Logic – Program as a set of logical sentences. Prolog
- ❑ Object Oriented - Program as a collection of classes for interacting objects. SmallTalk, C++, Java

Brief on various Programming Paradigms

Imperative

- Idea is - “First do this and next do that”, i.e. a step-by-step execution model - based on the stored program concept of Von Neumann
- Latin word "imperare" means “to command” - based on commands that update variables in storage
- It describes computation in terms of statements that change a program state.
- Similar to descriptions of everyday routines, such as food recipes
- Natural abstraction is function, procedures or subroutines

Imperative Programming (Example)

Sum of N positive numbers

```
Procedure sum(n)
Begin
    Define variable x with initial value of 1
    While the variable is greater than 1 do
    Begin
        Add x by n to store   result in x
        Decrement n
    End while
    Return value of variable x
end
```

Equivalent program in C

```c
int sum(unsigned int n)
{
    int x = 1;
    while(n>1)
    {
        x += n;
        n--;
    }
    return x;
}
```

Functional Programming

- Idea is – evaluation an expression and using the resulting value for something else
- Functions are the fundamental building blocks (first class value) of a program. Functions in this sense (not to be confused with C Language functions which are just procedures) are analogous to mathematical equations: they declare a relationship between two or more entities.
- Based on mathematical model of function composition – Lamda calculas.
- The values produced are non-mutable
- No step by step execution model, result of one computation is input to the next and so on until some computation yields the desired result

Functional programming (Example)

Sum of N positive numbers

```
Sum n   = 1 (if n = 1)
          N + sum(n-1)
```

Equivalent program in LISP

```
(defun sum(n)
    (cond ((eq n 1) 1)
          (t (+ n (sum (- n 1))))
    )
)
```

Example

Summing the integers 1 to 10 in imperative language C:

```
int total = 0, i;
for (i = 1; i <= 10; ++i){
    total = total+i;
}
```

Values change for both `total` and `i` during program execution

Summing integers 1 to 10 in a pure functional language

```
sum (m, n) : if (m > n) 0
     else m + sum (m+1, n)

sum (1, 10) // main function
```

No side effect => No assignments to variables!

Logic Programming

- It is based on the idea of answering a question through search for solution from a knowledge base
- Based on
  - Axioms/Facts
  - Inferences rules
  - Queries / Goals
- Program execution becomes a systematic search in a set of facts, making use of a set of inference rules – Set of know facts and set of rules results in deduction of other facts.
- Evaluation starts with a goal and attempts to prove it with a known fact or by deducing it from some rules.

Logic Programming (Example)

Axioms/Facts

```
F1. father(dasarath, ram).
F2. father(ram, lav).
F3. mother(kaushalya, ram).
F4. mother(sita, lav).
```

Inferences rules

```
R1. parent (X, Y) :- father (X, Y)
R2. parent (X, Y) :- mother (X, Y)
R3. grandfather(X, Y) :- father(X, Z),
      parent (Z, Y)
R4. grandmother(X, Y) :- mother(X, Z),
      parent (Z, Y)
```

Queries / Goals

```
G1. ? father(X, ram). X = dasaratha.
G2. ? father (dasaratha, X). X = ram
G3. ? grandmother(X, lav). X = kaushalya.
G4. ? parent (X, ram). X = ???
```