INTRODUCTION TO COMPUTERS

Objective

- How to approach a given problem.
- To teach programming concepts in the context of solving problems.
- To apply problem solving techniques to a problem.
- To implement the proposed solution using the C language.

Outline

- Introduction to Programming, Processing of Programs
- Programming Solving Techniques (Pseudocode, Algorithms and Flowcharts).
- Structured Programming
- Problem solving and programming:
- Selection Control Flow, Looping control structure, Arrays and Strings, Pointers, Functions, Structures, Unions, Enumerations, Preprocessor
- File I/O
- The C Standard Library

Program and Programming

Program:

- A set of instruction written in a programming language that a computer can execute so that the machine acts in a predetermined way.
- Program solves a problem
- Before writing a program:
  - Have a thorough understanding of the problem
  - Carefully plan an approach for solving it.

Programming:

- The Process of providing instructions to the computer that tells the processor what to do.

Software and Hardware

- Hardware:
  - Various devices comprising a computer
  - Keyboard, screen, mouse, disks, memory, CD-ROM, and processor.
- Software:

  Any application that guides computer to perform a task and obtain the desired output is known as software component.

  Example : A C program

Software Classification

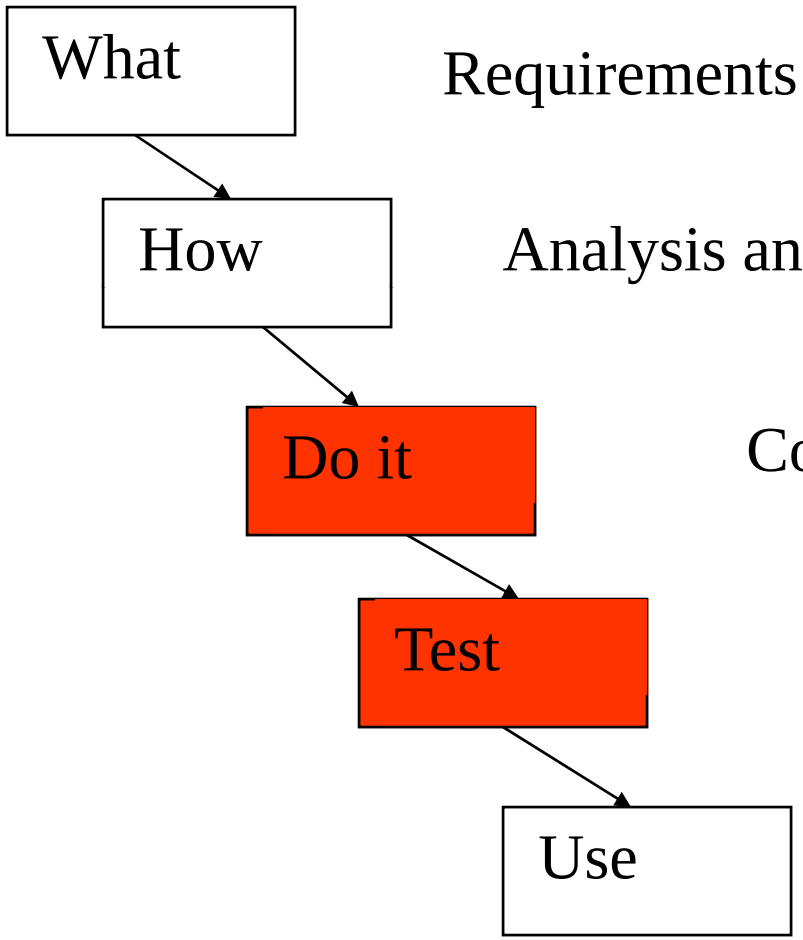
- System Software

- Application Software

➢ System S/W define functioning of any computer irrespective of the area/application e.g. OS, Compiler, Editor etc.

➢ Application S/W are designed for specific task/application.e.g. railway reservation s/w, weather forecasting s/w, online admission/examination s/w etc.

Software Life Cycle



Requirements Gathering, Problem definition

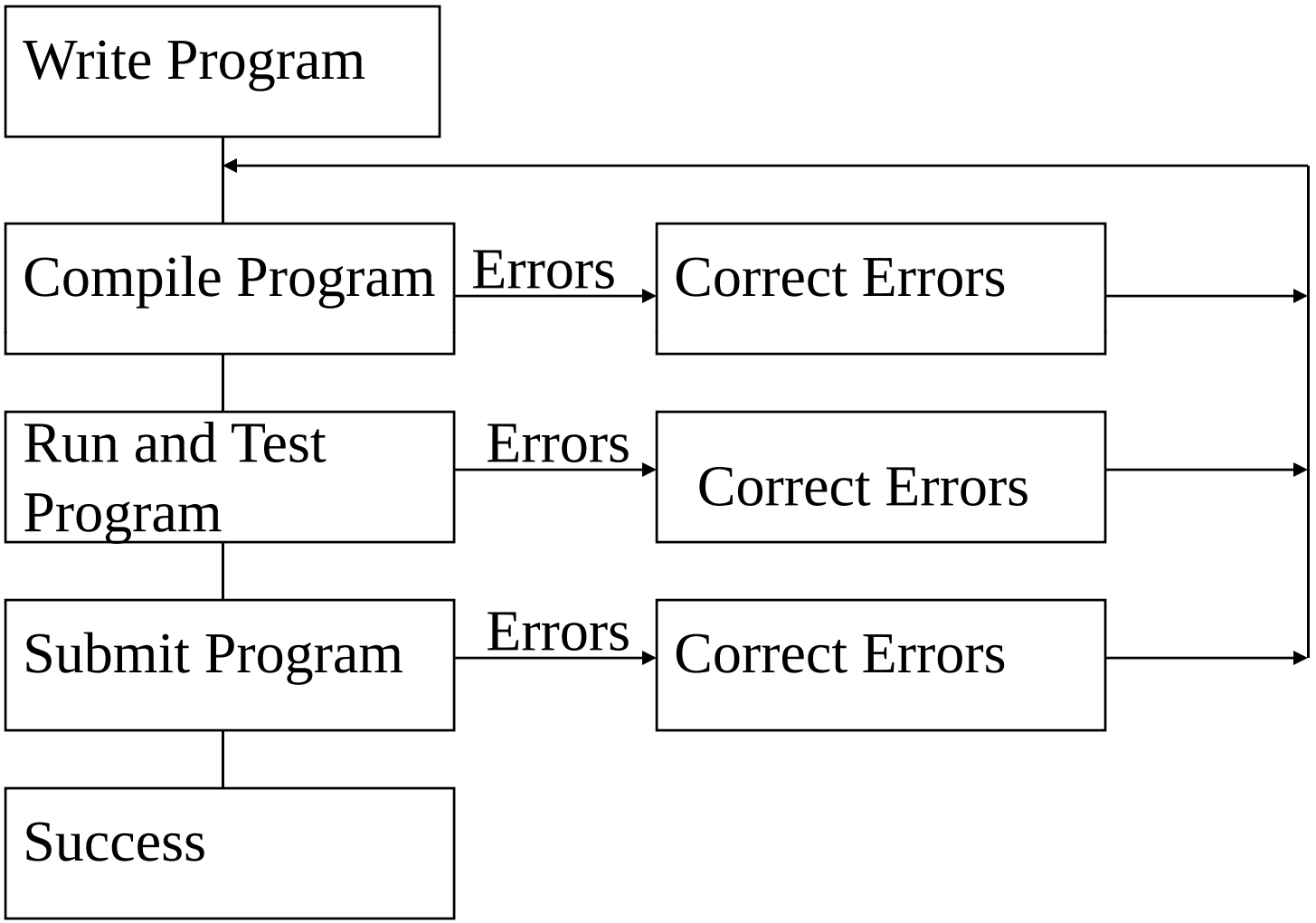
Analysis and Design (Programming techniques)

Coding

Testing

Implementation and Maintenance

Coding and Testing



Algorithms

- An Algorithm is a solution to a problem that is independent of any programming language.
- An algorithm is just steps, written in plain English, that are needed to solve a given problem.
- The general form what an algorithm takes is
  Step 1 : START
  Step 2 : <….>
  Step 3 : <…..>.
  …….
  Step n : STOP
- A flowchart on the other hand is a diagrammatic representation of an algorithm.

Algorithm

- An algorithm is
  - a finite sequence of steps
  - each step shall be clearly stated and understandable
  - for each input, it shall terminate in finite time with output

Example

- Write an algorithm to find the average of three numbers

Solution

Step 1 : START

Step 2 : Accept 3 numbers say num1, num2, num3

Step 3 : Add num1 num2 num3 and store the result in sum

Step 4 : Divide sum by 3 and find the average

Step 5 : Display Average

Step 6 : STOP

Algorithms

- Computing problems
  - All can be solved by executing a series of actions in a specific order
- Algorithm: procedure in terms of
  - Actions to be executed
  - The order in which these actions are to be executed
- Program control
  - Specify order in which statements are to executed

Algorithm

- An algorithm is
  - a finite sequence of steps
  - each step shall be clearly stated and understandable
  - for each input, it shall terminate in finite time with output

Pseudocode

- It is a mixture of English language statement and a programming language(like C) code.
- It is an intermediate step in writing a program.
- Not actually executed on computers.
- Standard mathematical notations like exponents,square-root symbols etc. is allowed in pseudocode.

Example

Addition of two numbers

Input : Number1, Number2

Output : Sum of Number1 and Number2

Computational Procedure :

- Read Number1
- Read Number2
- Sum = N1+N2
- Output/Print Sum

Example

Find the greater of two numbers

Input : Number1 , Number2

Output : Greater of Number1 and Number2

Computational Procedure :

1. Read Number1
2. Read Number2
3. If Number1>Number2 Greater = Number1, go to step 6
4. If Number2>Number1 Greater = Number2 , go to step 6
5. Both are equal, so Greater = Number1=Number2
6. Output/Print Greater

Algorithms & Programs

- An Algorithm is a solution to a problem that is independent of any programming language.

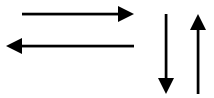
While

- A program is an algorithm expressed in a specific programming language.

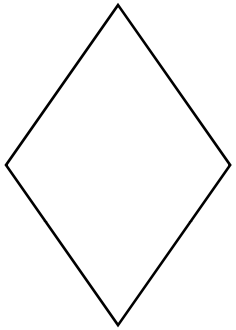
Flowchart

- Graphical representation of an algorithm
- Components:
  - Arrows/lines :Flow of control
  - Parallelogram: Indicates input and output operations
  - Rectangle symbol (action symbol): Indicates any type of action/computational step
  - Oval symbol:Indicates the beginning or end of a program or a section of code
  - Diamond: Decision

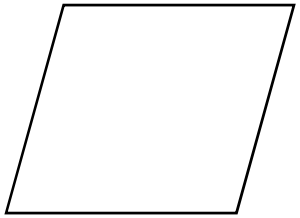
Flowchart Notations



Arrows

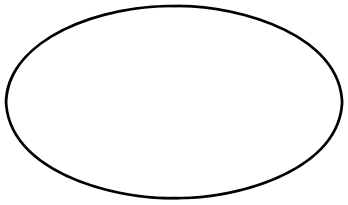


Diamond



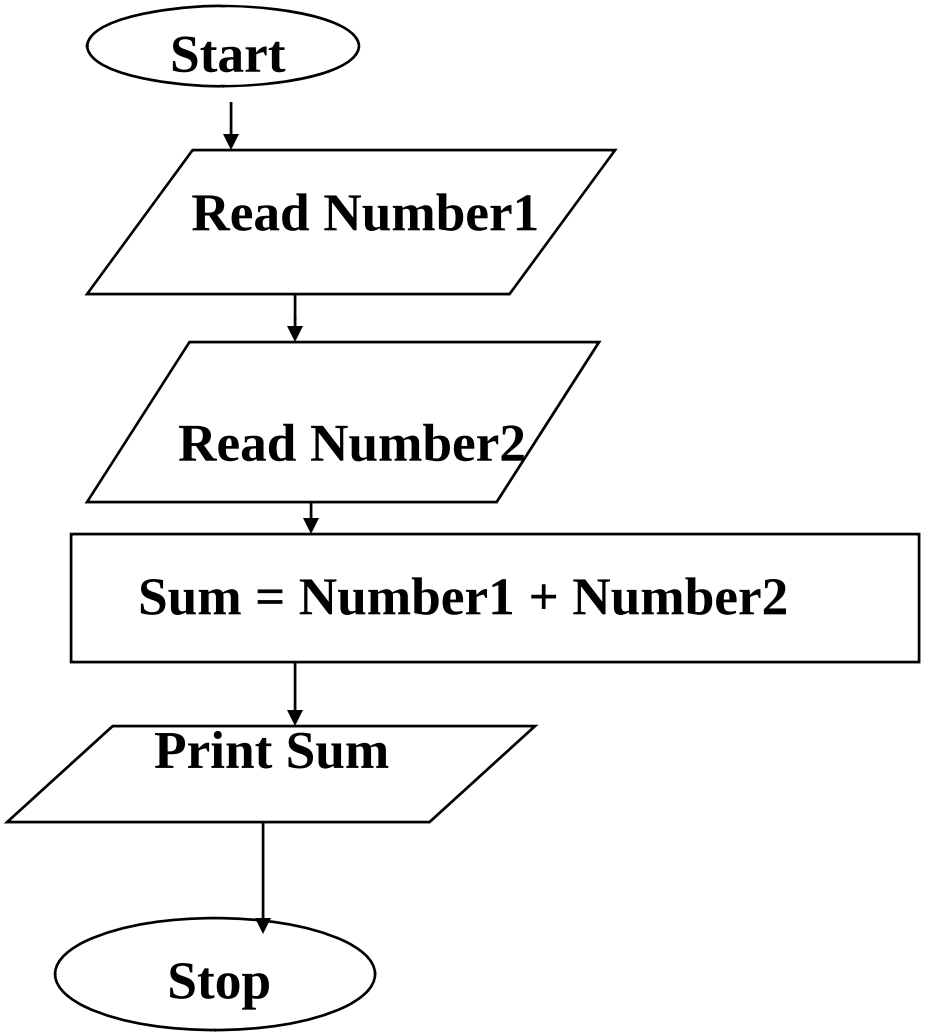
Parallelogram

Rectangle

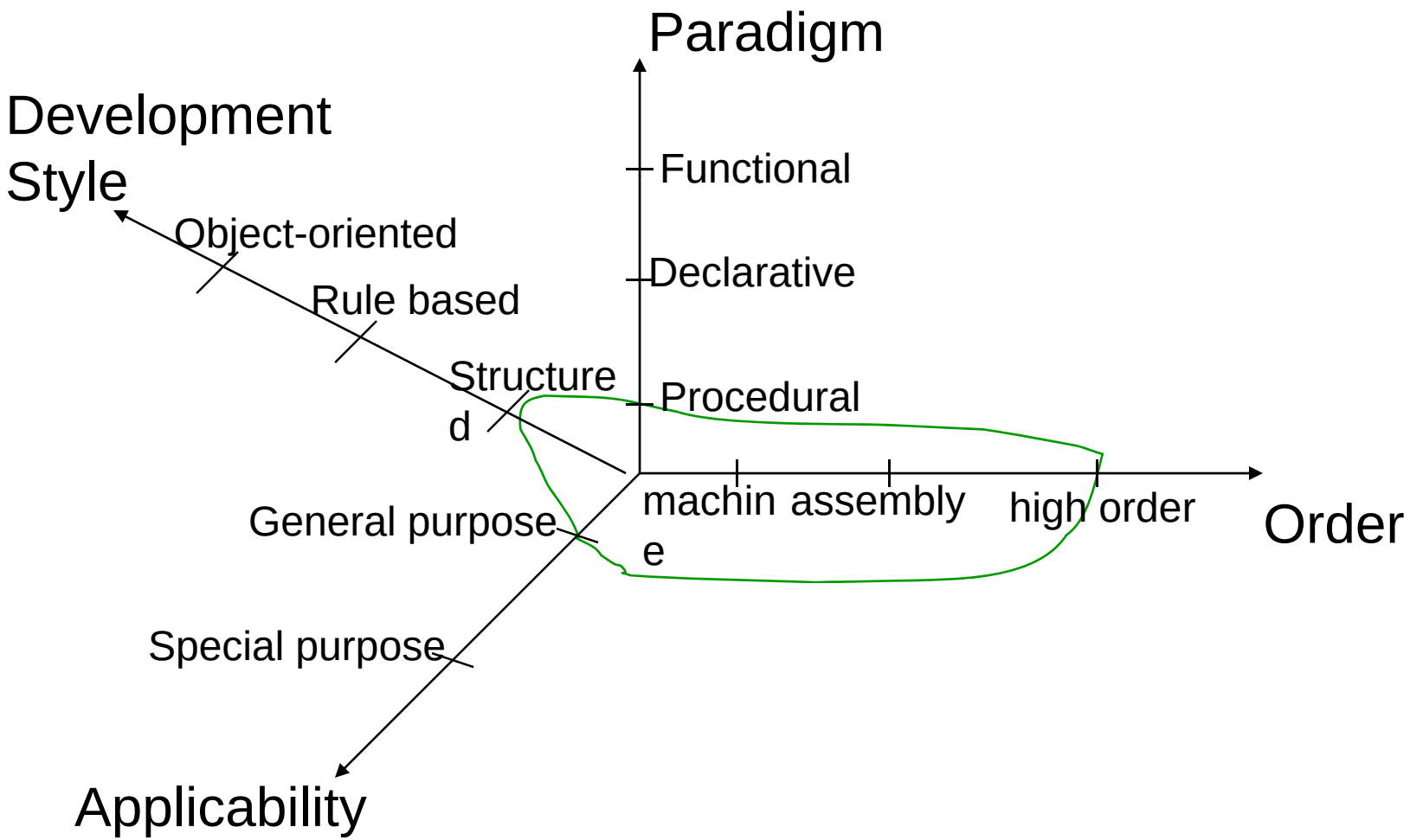


Oval

Example: Add two Numbers



Types of Programming Languages



Order of Languages

Machine Language

- machine/hardware dependent
- Too hard to program in
- zeros and ones

Assembly Language

- English-like operations (e.g., load, store, add)

High-level language

- single statement can accomplish substantial tasks
- English-like statements with mathematical notations

Paradigm

Functional

- It is programming paradigm that treats computation as the function of mathematical function

- A program is a set of nested functions

A(B(C(x), y), z) Lisp

Paradigm

Declarative

- Write what has to be done, not how it is to be done
- It describes a problem rather than defining solution
- select Name from Employee SQL

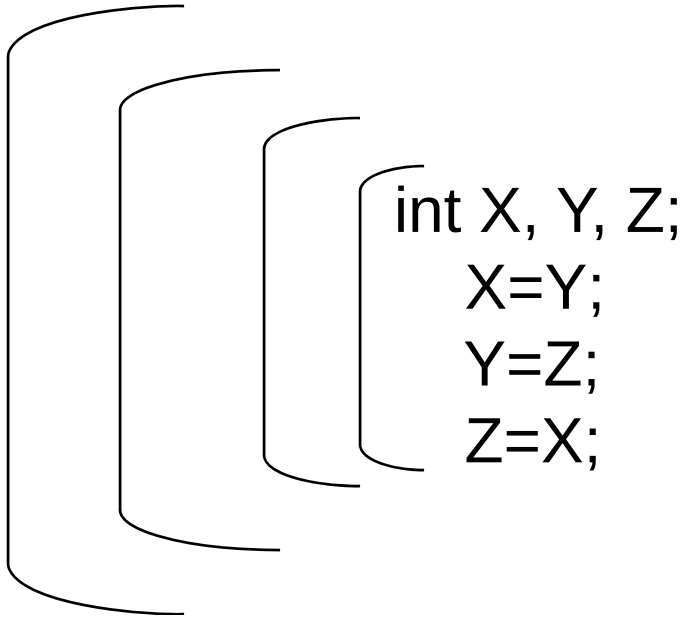
Procedural

- Detailed description of the steps to solve a problem
- Xchange(Y,Z) : X=Y; Y=Z; Z=X C, Pascal

The Procedural Style

Procedural

A program is a collection of nested procedures : C, Pascal



Each procedure has its own declaration and executable part

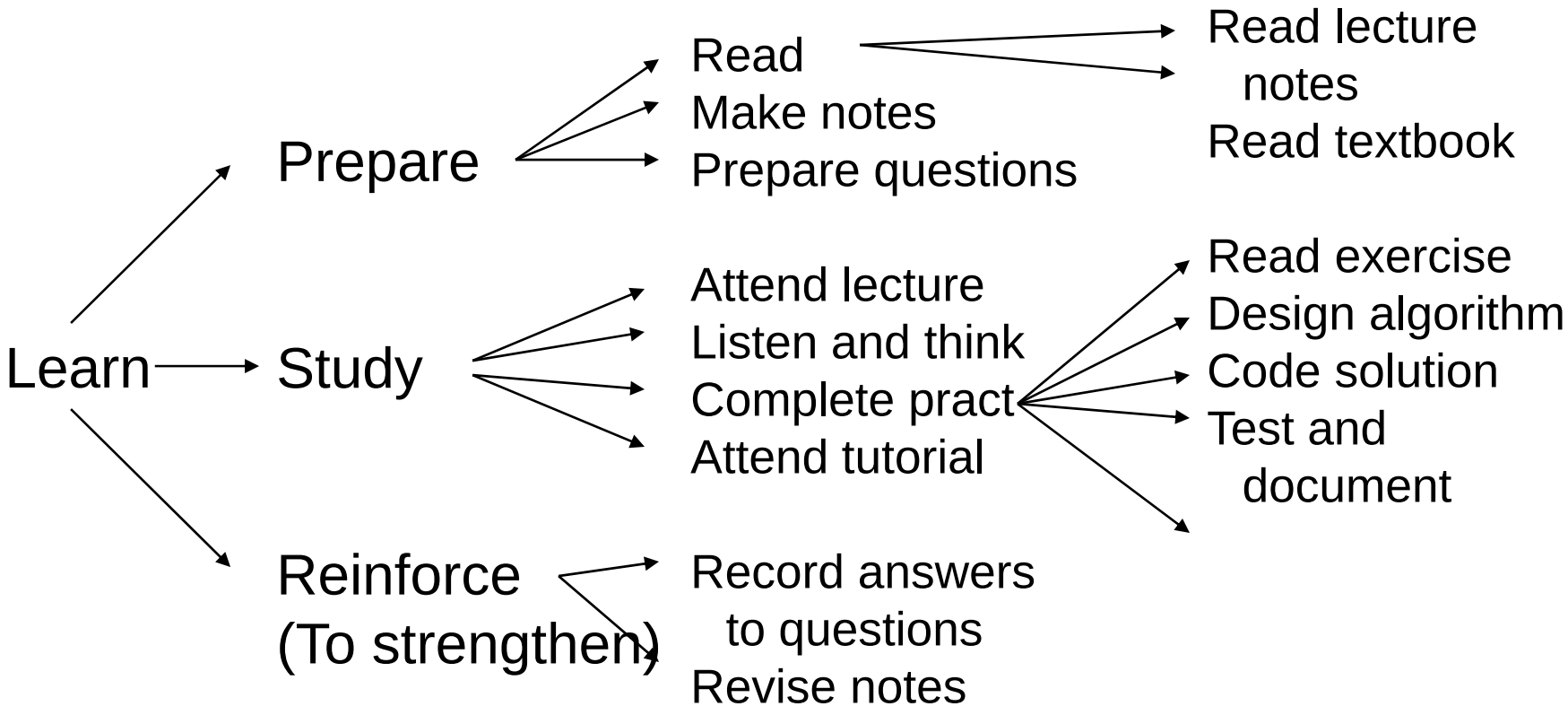
Development Style

- Structured Programming: Program is subdivided into modules.
- Rule based programming:The rules are defined in the programming language.
- Object-Oriented programming: Problem is broken into object and classes.

Top-down Design

- Write down what you have to do
- Break that into 3-7 smaller steps
- Break each step into 3-7 smaller steps
- Subdividing until each individual step is easy enough to do
- Example:
  - Learning

Top-down Design -- Example



Structured Programming

- Program is subdivided into modules (one logical thought, approx. one page long)
- Each page is organized into recognizable paragraphs, using indentation to show nesting and subordinate clauses.
- Embedded comments describe the function of each variable and purpose of each module.

Structured Programming

- Final program is created via top-down design.
- Main code with empty modules, slowly refined to lowest level
- Allows easy division of work (modules assigned to different programmers)

Benefits of Structured Programming:

- More readable
- Easier to maintain
- More likely to be correct on first run
- Easier to "prove" by systematic program verification

High Level Languages

- Are portable
- User writes program in language similar to natural language
- Compiler translates high-level language program into machine language
- Examples of high level languages

  FORTRAN, COBOL, Pascal, C, C++

High Level Languages

C Program:

```
#include<stdio.h>
  main( )
  {       int k;
          scanf(“%d”, &k);
          if (k%4 = = 0)
                  printf(“Year %d”,k, “is a leap year”);
          else
                  printf(“Year %d”,k, “is not a leap year”);
 }
```

Processing a Program

- Programs are written in a programming Language
- Processor understands only machine language.
- Programs written in other languages are to be translated.

Why there are so many programming languages

- Many different machines
- Many different Applications
- Need for better tools yielding software that are trustworthy,less expensive,faster

How should Programming languages be judged?

Some criteria:

- Extent of support for program correctness
- Cost of Program Development and modification
- Cost of program use(time,space)
- Total Cost:always combination of 1 and 2

Editor

- **Used to create the program and store it on disk (secondary storage)**
- **C programs should be saved with a *.c* extension**

Preprocessor

- Handles various manipulations before compiling, including
  - Inclusion of additional specified files (e.g. stdio.h,conio.h)
  - Handles textual(macro) substitution
- Results in more efficient, clearer and less unwieldy code.

Compiler

- **Translates the saved program file to machine language (*object code*) and saves it to a file**

Linker

- **Links object code from any additional specified files into appropriate places in your code**

OR

- **A program that combines one or more files containing {object code} from separately compiled program {modules} into a single file containing loadable or executable code**

- **Produces a file that is an executable image of the linked object code and stores it on the disk**

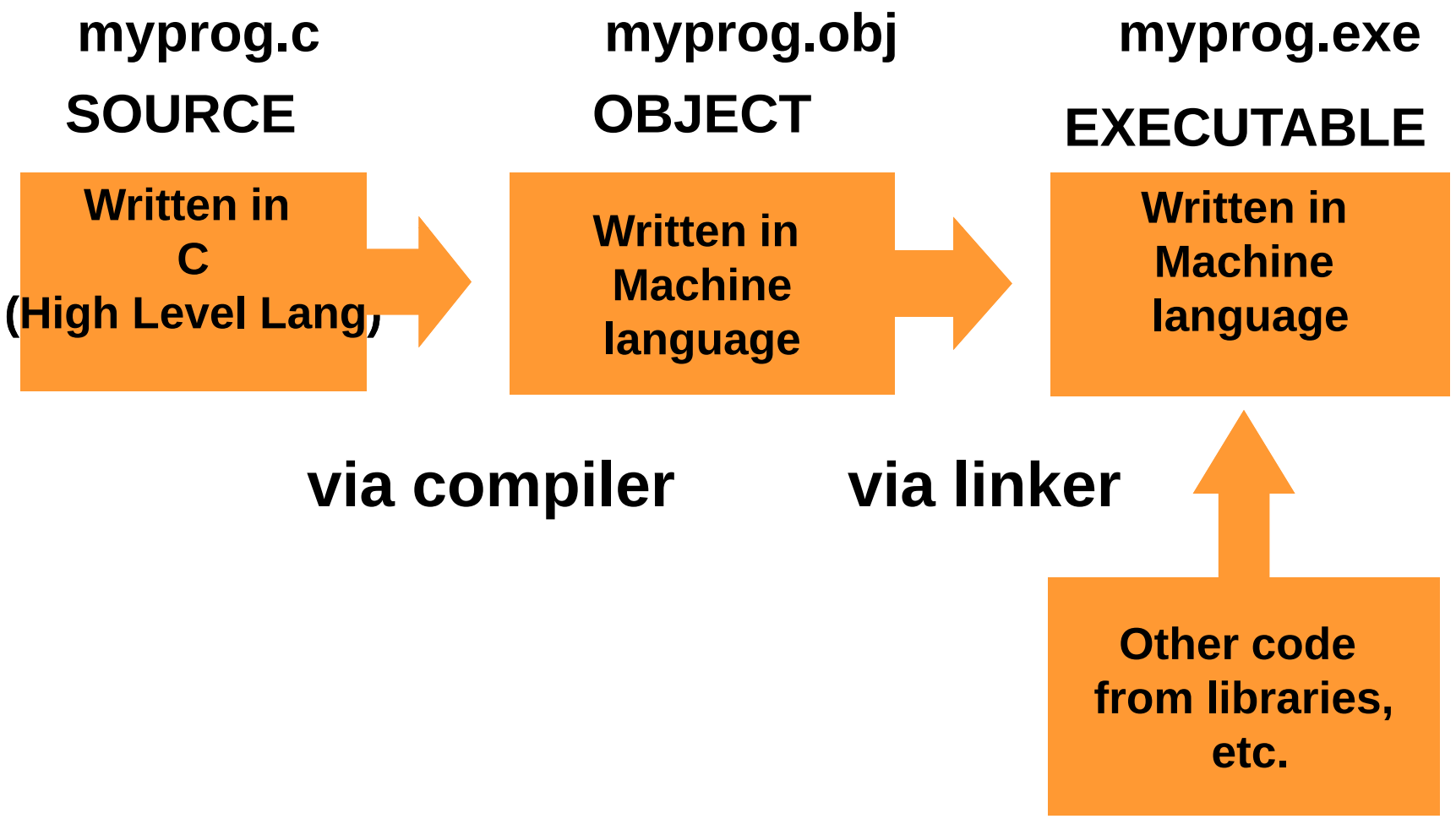
Loader

- **Puts the executable image (instructions) from the disk into primary memory**

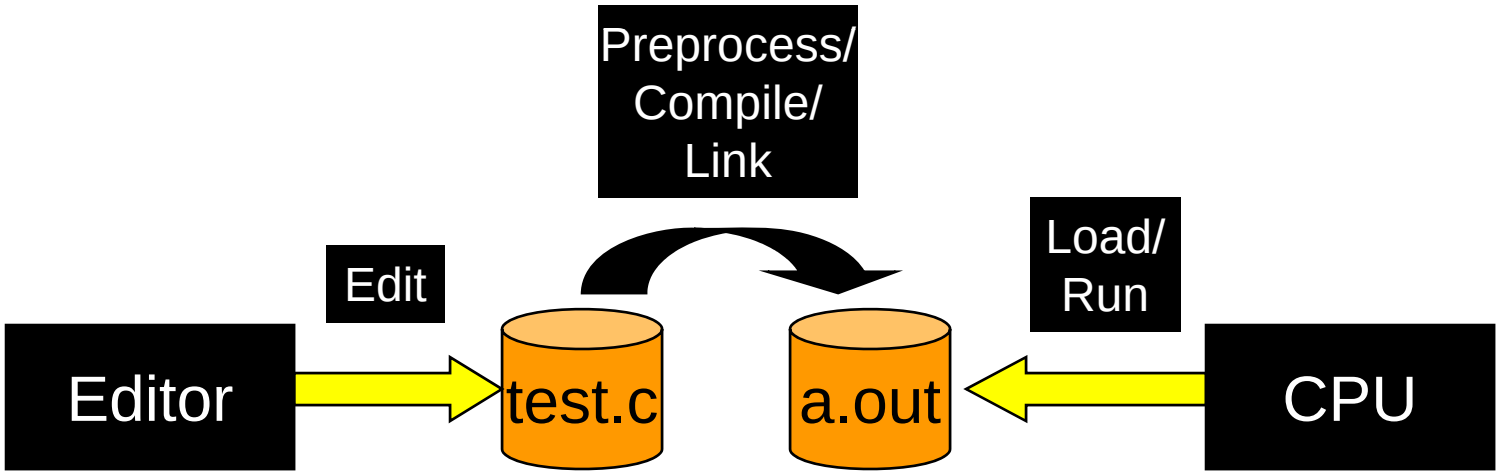
Execute

- **CPU takes each instruction in primary memory and executes it**
- **CPU may also store new data values as program executes**

Three Program Stages

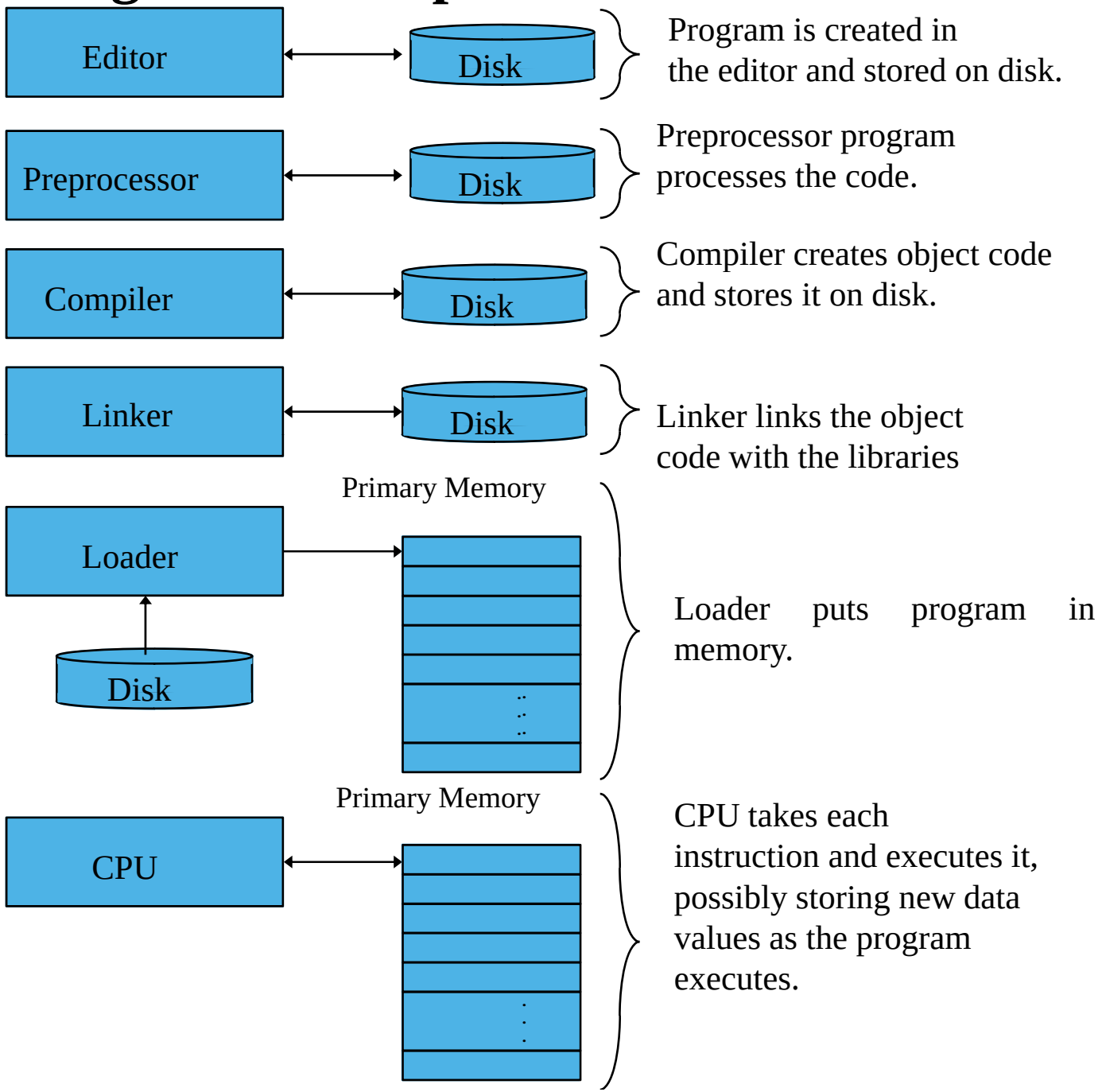


C Programming Environment in UNIX



Typical C Program Development Environment

1. *Edit*
2. *Preprocess*
3. *Compile*
4. *Link*
5. *Load*
6. *Execute*



Pseudocode for factorial of number

```
START
INITIALIZE   product =1 ;
ACCEPT num
WHILE  num>0
   product = product * num;
   num = num -1;
DISPLAY   " the factorial of num is is : ",
   product ;
END
```

Multiply 2 Matrices

```
START
INITIALIZE  I=0, j=0, k=0, prod[ ][ ];
ACCEPT  r1,c1,mat1;
ACCEPT  r2,c2,mat2;
IF   c1=r2
     REPEAT
          REPEAT
     prod[I][j] = 0;
                    REPEAT
          prod[I][j] = prod[I][j] +  mat1[I][k] * mat2[k][j]
                    UNTIL  k < c1
          UNTIL   j < c2
     UNTIL  I < r1
ELSE
          DISPLAY "Multiplication not possible";
ENDIF
END
```

Exercise

- Develop an algo to compute 1/n!
- Develop an algo to print the elements of upper-left Triangular matrix

Characteristics of Programs

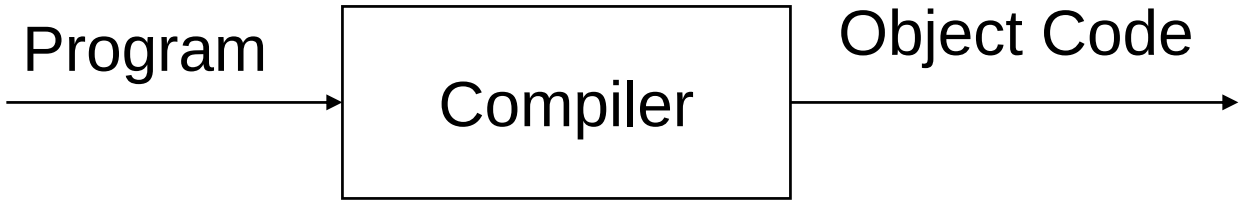
Genericity

1. The same procedure can be used for different input
2. The procedure is not to be written repeatedly
3. The procedure has fixed characteristics
    - Input: only three numbers can be added
    - Output: the sum of the input numbers

Quality: The procedure must be guaranteed to be correct

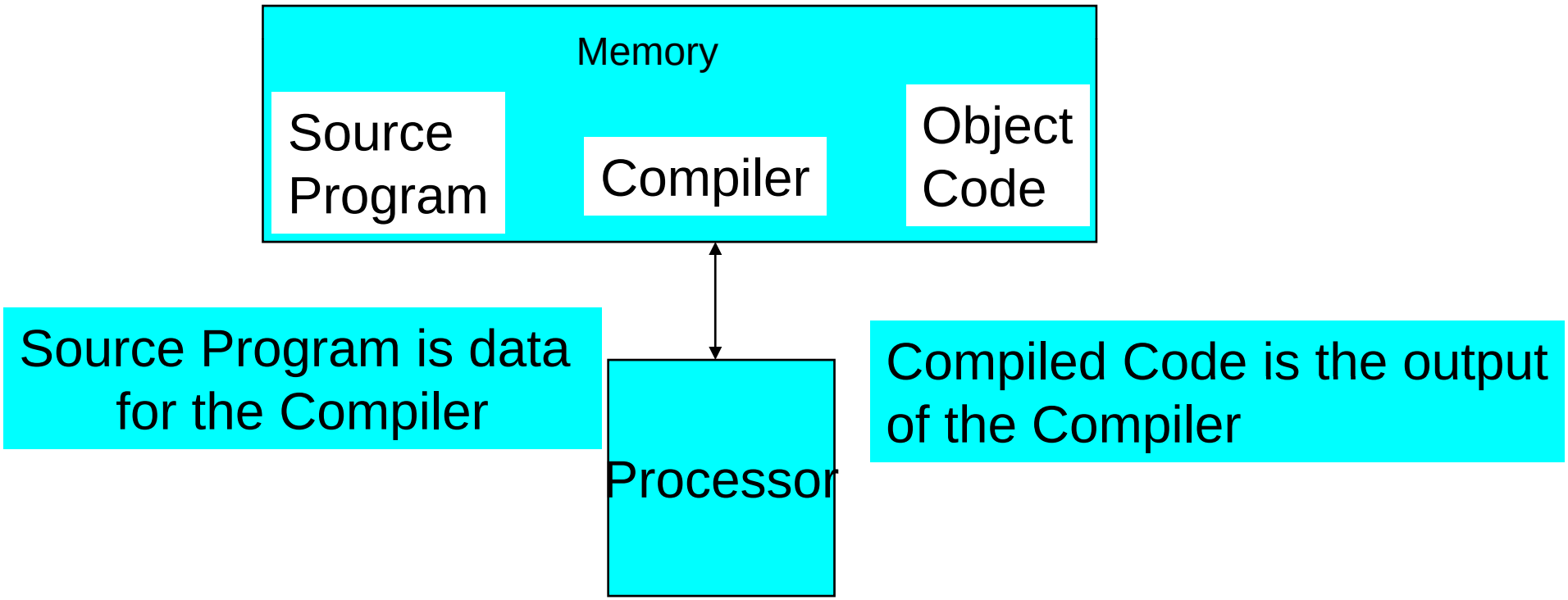
Design: The procedure should be efficient

Processing a Program



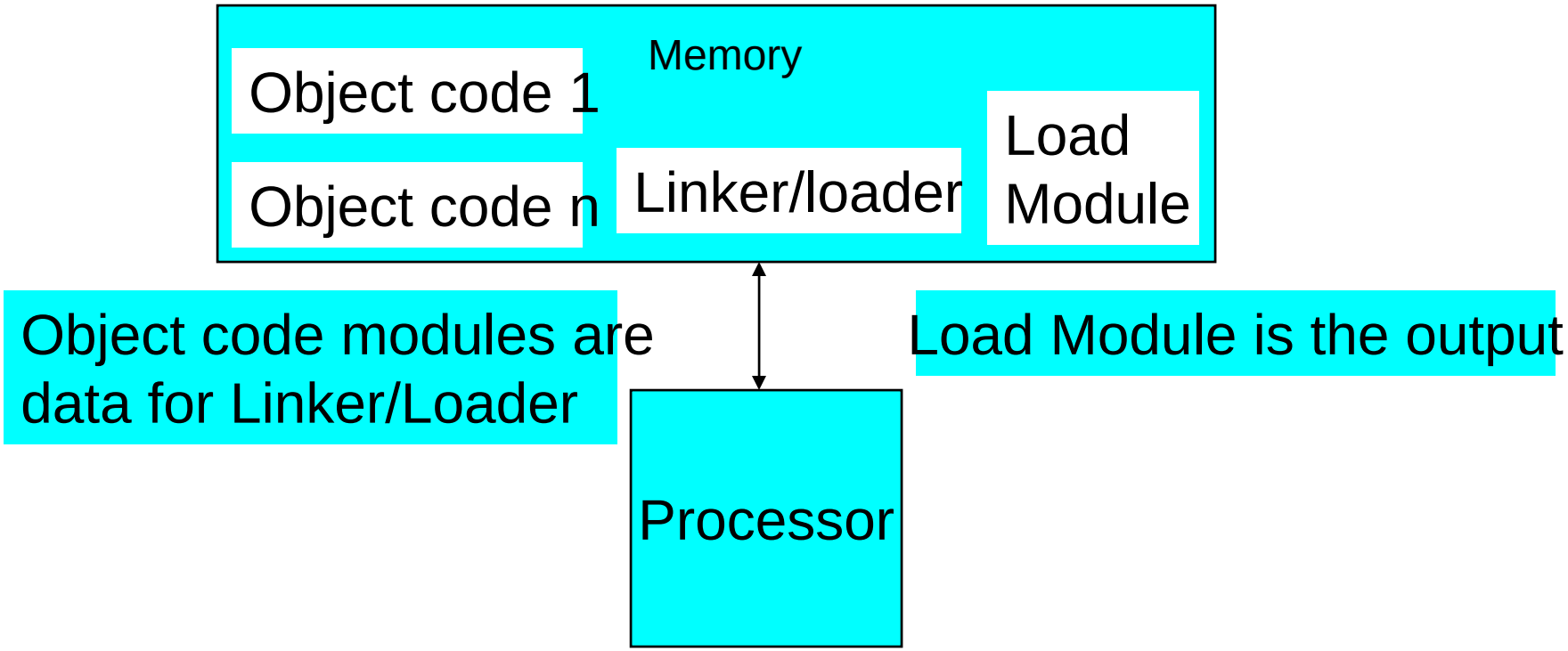
Processing a Program

Compiling a Program: Compile time



Processing a Program

Linking Programs: Link time



Processing a Program

Executing Programs: Run time