15CSE202 Object Oriented Programming
Lecture 1

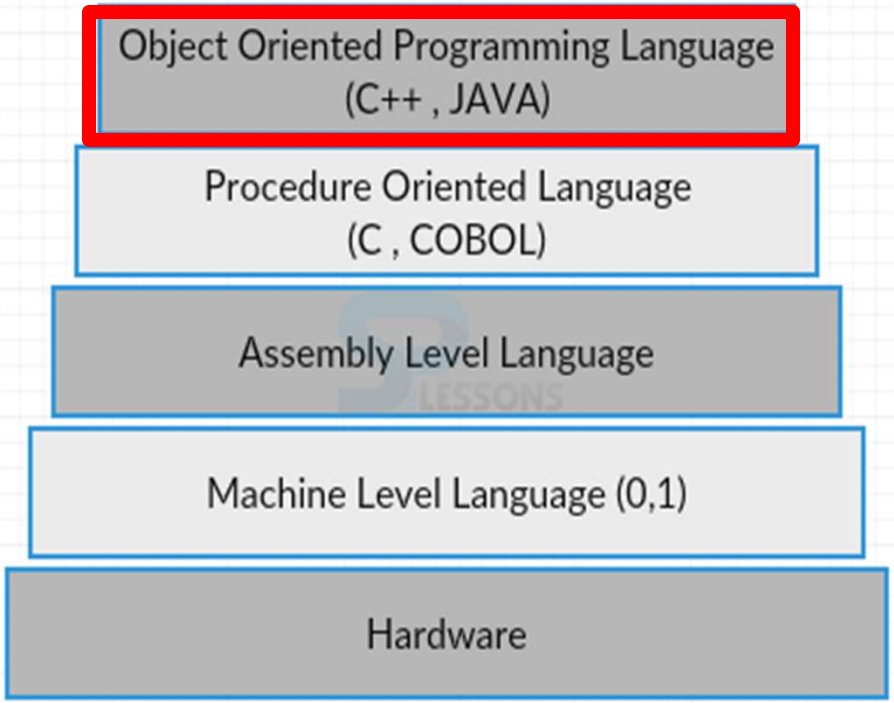
Overview of different programming paradigms

Nalinadevi Kadiresan

CSE Dept.

Amrita School of Engg.

Evolution of Programming Languages



- A computer language is a set of predefined words that are combined into a program according to predefined rules (*syntax*).
- Over the years, computer languages have evolved from *machine language* to *high-level languages*.

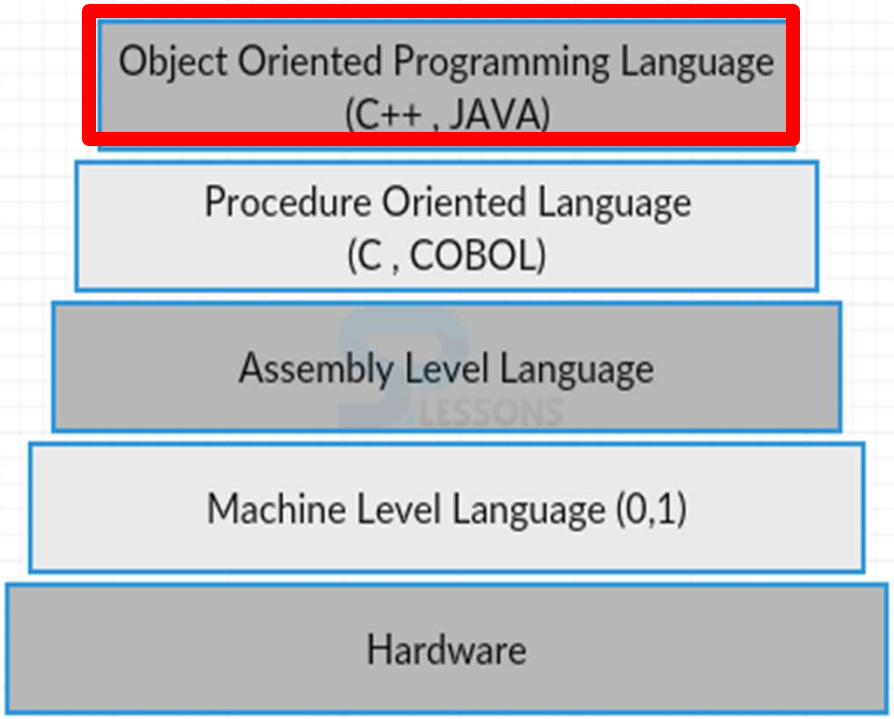
Table 9.1 Code in machine language to add two integers

Hexadecimal	*Code in machine language*			
$(1FEF)_{16}$	0001	1111	1110	1111
$(240F)_{16}$	0010	0100	0000	1111
$(1FEF)_{16}$	0001	1111	1110	1111
$(241F)_{16}$	0010	0100	0001	1111
$(1040)_{16}$	0001	0000	0100	0000
$(1141)_{16}$	0001	0001	0100	0001
$(3201)_{16}$	0011	0010	0000	0001
$(2422)_{16}$	0010	0100	0010	0010
$(1F42)_{16}$	0001	1111	0100	0010
$(2FFF)_{16}$	0010	1111	1111	1111
$(0000)_{16}$	0000	0000	0000	0000

Table 9.2 Code in assembly language to add two integers

Code in assembly language	*Description*
LOAD RF Keyboard	Load from keyboard controller to register F
STORE Number1 RF	Store register F into Number1
LOAD RF Keyboard	Load from keyboard controller to register F
STORE Number2 RF	Store register F into Number2
LOAD R0 Number1	Load Number1 into register 0
LOAD R1 Number2	Load Number2 into register 1
ADDI R2 R0 R1	Add registers 0 and 1 with result in register 2
STORE Result R2	Store register 2 into Result
LOAD RF Result	Load Result into register F
STORE Monitor RF	Store register F into monitor controller
HALT	Stop

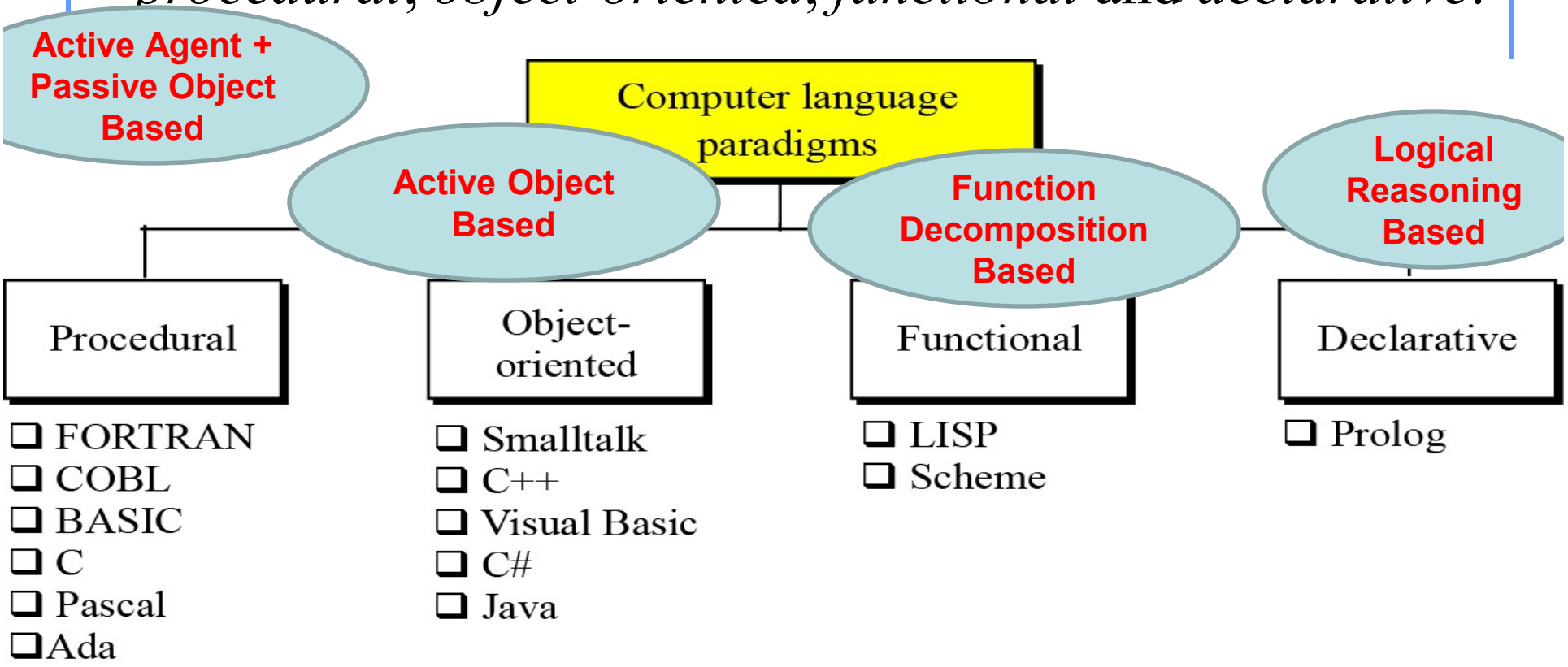
Evolution of Programming Languages - Paradigm



- Today, computer languages are categorized according to the approach they use to solve a problem.
- A paradigm, therefore, is a way in which a computer language looks at the problem to be solved.

Evolution of Programming Languages -contd

- We divide computer languages into four paradigms: *procedural*, *object-oriented*, *functional* and *declarative*.



Picture courtesy

CONCEPTS OF OBJECT ORIENTED PROGRAMMING

By Grady Booch

By Ravi P Reddy

Sample C program

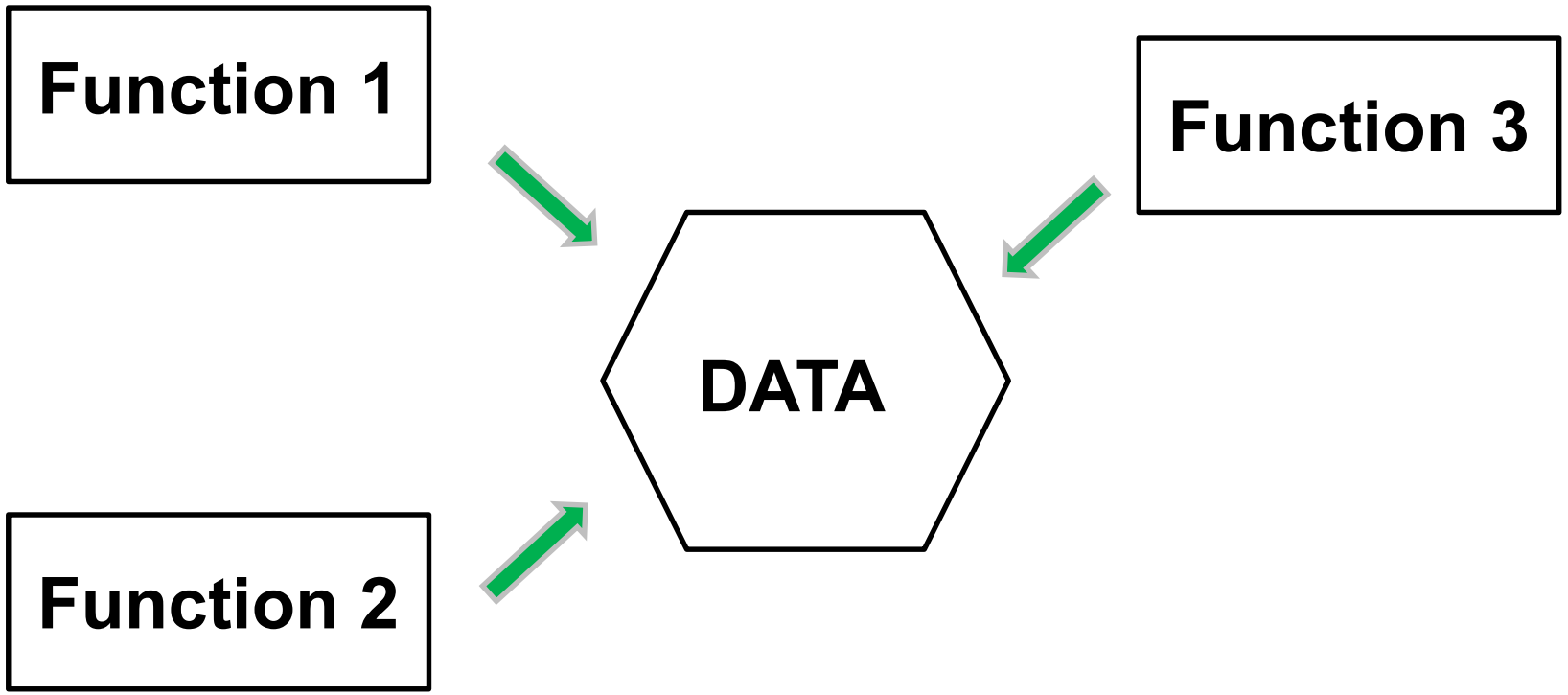
```
#include <stdlib.h>
#include <time.h>
#define SIZE 5

struct bot { int xpos; int ypos; };
struct bot initialize(struct bot b);

int main()
 {
      struct bot robots[SIZE];
      int x;
      srandom((unsigned)time(NULL));
      for(x=0;x<SIZE;x++)
      {
            robots[x] = initialize(robots[x]);
            printf("Robot %d: Coordinates:
            %d,%dn", x+1,robots[x].
            xpos,robots[x].ypos);
      }
      return(0);
}
struct bot initialize(struct bot b)
{
      int x,y;
      x = random();
      y = random();
      x%=20;
      y%=20;
      b.xpos = x;
      b.ypos = y;
      return(b);  }
```

श्रद्धावान् लभते ज्ञानम्

Problems with Procedural Languages

- Data does not have an owner.
- Difficult to maintain data integrity.
- Functions are the building blocks.
- Many functions can modify a given block of data.
- Difficult to pinpoint bug sources when data corrupted.

Problems with Procedural Languages



Problems with Procedural Languages

- Uninitialized variables.

```
int k ;
printf("%d",k) ;
```

Problems with Procedural Languages

- Resource Deallocation Problems

Examples: Memory leaks, open files, open database connections, open network connections.

```
int* kp;
kp = (int*) malloc(1000);
kp = (int*) malloc(1000);
kp = (int*) malloc(1000);
```

Problems with Procedural Languages

- Insufficient support for abstraction

  ADT = data + functions

  data: modelled as structures

  functions: global, not related to structure

Next Session will be

OO Concepts