

System Programming

Prof. Awadhesh Dixit and Prof. Trilok Suthar

Assistant Professor

Information Technology





CHAPTER-2

Overview of Language Processors

Topics to be covered

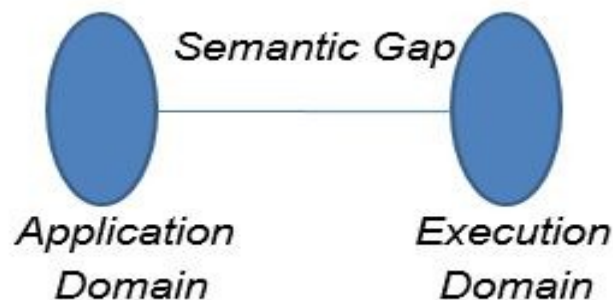
- Overview of Language Processors
- Programming Languages and Language Processors
- Language Processing Activities
- Program Execution
- Fundamental of Language Processing
- Symbol Tables
- Data Structures for Language Processing: Search Data structures, Allocation Data Structures.

Introduction

- Language processor is a system program that bridges the gap between how a user describes a computation (specification of computation) and how a computer executes a program.
- Language processing activities arise due to the difference between the manner in which a software designer describes ideas concerning the behavior of a software and the manner in which these ideas are implemented in a computer system.

Some definitions...

- *Semantic*: It represents the rules of the meaning of the domain.
- *Semantic gap*: It represents the difference between the semantic of two domains.
- *Application domain*: The designer expresses the ideas in terms related to application domain of the software.
- *Execution domain*: To implement the ideas of designer, their description has to be interpreted in terms related to the execution domain of computer system.



Consequences of Semantic Gap

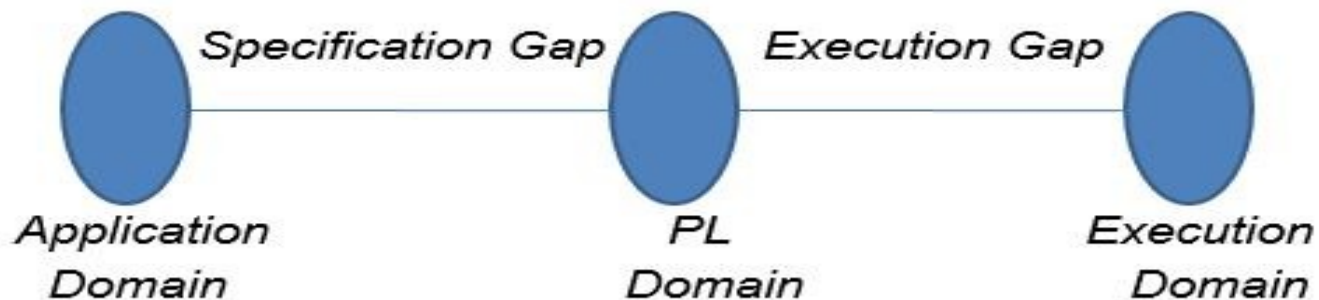
- ☐ Large Development Time
- ☐ Large Development Efforts
- ☐ Poor Quality of Software

Semantic Gap

- ❑ These issues are tackled through the use of programming language (PL).
- ❑ Software implementation using PL introduces a new domain as PL domain.

Semantic Gap

- Now the semantic gap is bridged by the software engineering steps.
- The first step bridges gap between application domain and PL domain known as specification gap.
- While the second step bridges the gap between PL domain and execution domain as execution gap.





- ❑ The specification gap is bridge by the software development team.
- ❑ While the execution gap is bridged by the designer of the programming *language processor*. Like compiler or interpreter.
- ❑ PL domain reduces the difficulties of semantic gap mentioned earlier.
- ❑ The language processor also provides a diagnostics capability which detects and indicates errors in its input. This helps in improving the quality of the software.

Some definitions...

- *Specification gap*: The gap between application and Programming Language domains is called specification and design gap or simply specification gap. Specification gap is the semantic gap between two specifications of the same task.
- *Execution gap*: The gap between programming language and execution domains is called execution gap. Execution gap is the semantic gap between the semantic of programs that perform the same task but written in different programming language.



Example – Application

- Designer of airline reservation application would like to think of how the reservation data would be used by the query, book and cancel operations, rather than how the data and operations should be implemented in the machine language.
- The description of the reservation data and the query, book and cancel operations forms a specification of the application.

Example – Execution Domain

- The system software of the computer has to help in implementing this specification on the computer by using its machine language; we say that the system software has to implement this specification in the execution domain of the computer system.
- The entities in this domain are cells in memory and registers in the CPU, and the operations are the instructions of the CPU

Example – Semantic Gap

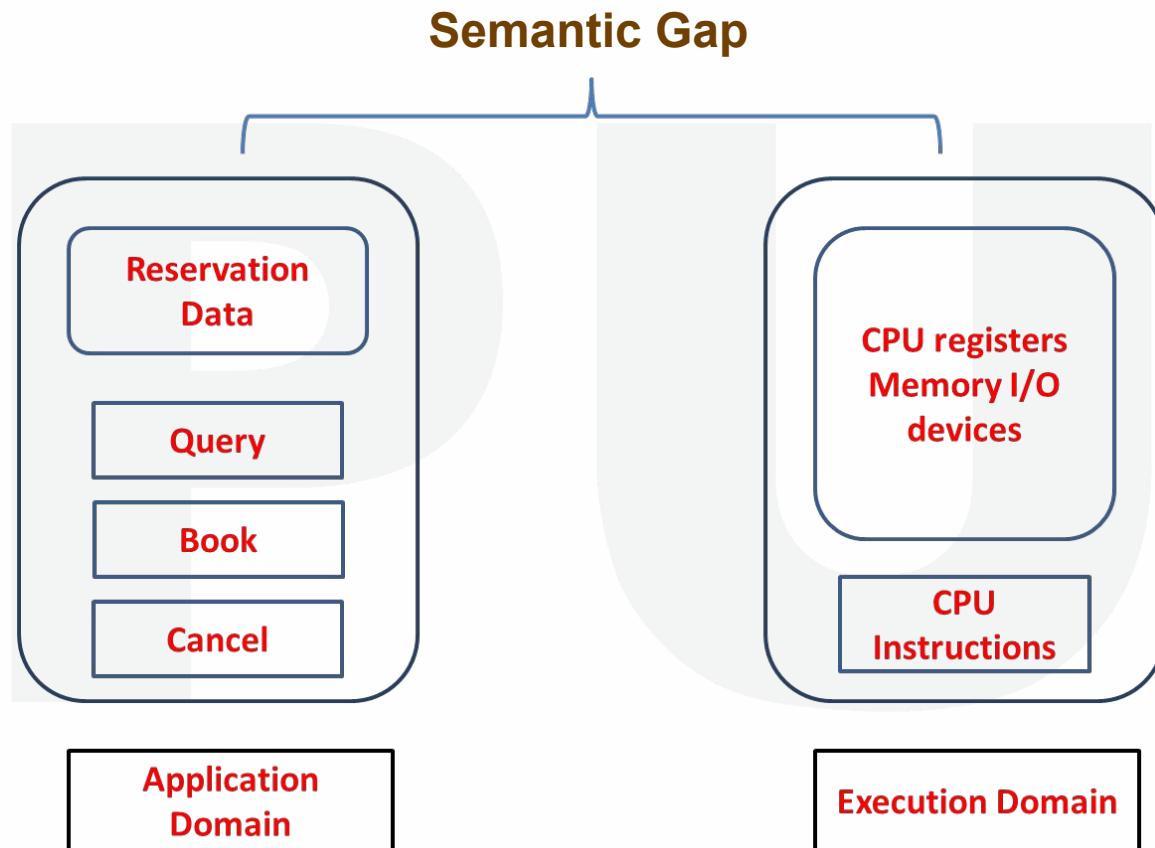
- Semantics : to represent the rules of meaning of a domain. For example, the semantics of the applications domain of an airline reservation system would govern the meaning of query, book and cancel operations.
- We use the term semantic gap to represent the difference between the semantics of two domains.



Example

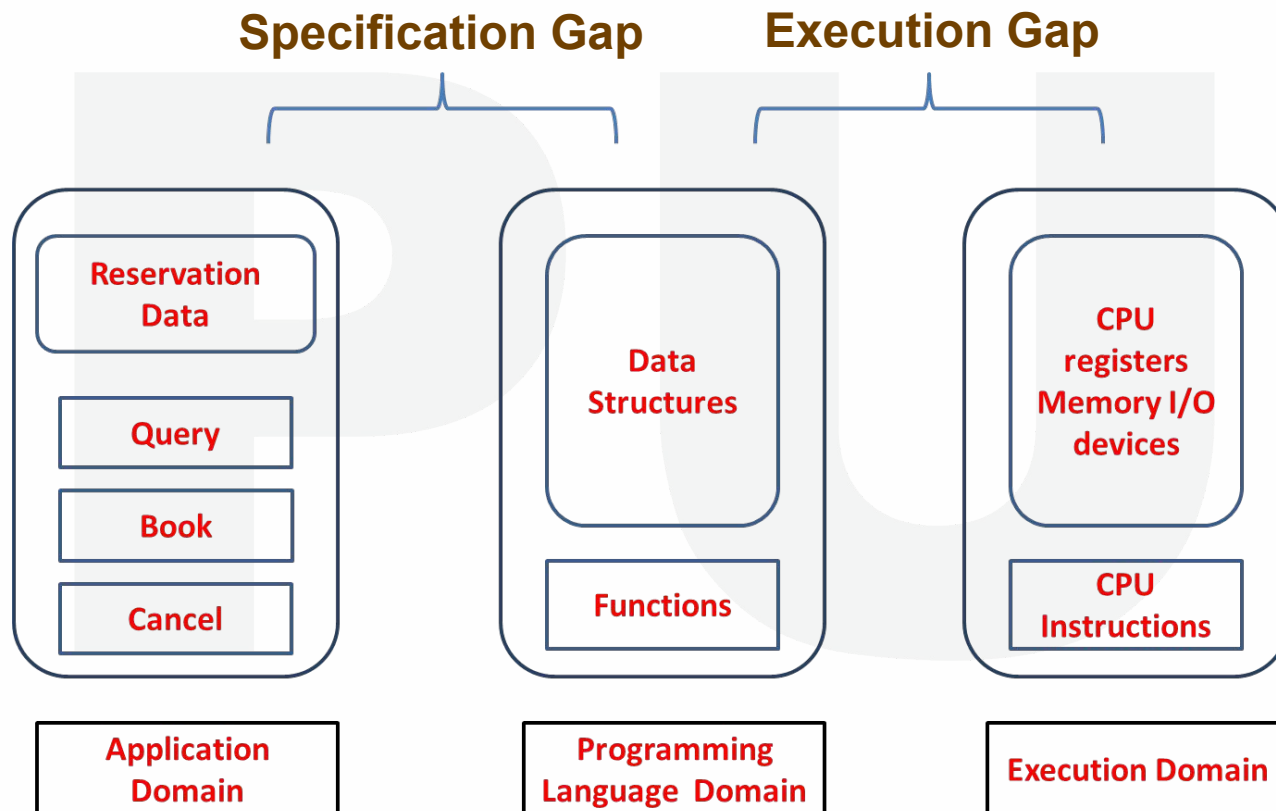
- The above figure is an abstract view of the application and execution domains and semantic gap between them.
- The **data** of domain are represented by an **oval shape** while the **operations** are represented by **rectangular boxes**.
- We say a semantic gap is bridged when a specification in one of the domains is converted into a specification in the other domain.
- Achieved by software engineering- set of methodology for effective software development

Semantic gap between domains





Semantic gap between domains



Language Processors

- *Language processor*: Language processor is a software which bridges a specification or execution gap.
- *Language translator*: Language translator for a programming language PLi bridges an execution gap between PLi and machine language of a computer system.

A language translator is called a *cross translator* if it bridges an execution gap to the machine language of Ck but itself runs on some computer other than Ck.



Language Processors

- *De-translator*: It bridges the same execution gap as language translator, but in the opposite direction.
- *Preprocessor*: It is a language processor which bridges an execution gap but is not a language translator.
- *Language migrator*: It bridges the specification gap between two programming languages.



Language Processors

- *Interpreter*: An interpreter is a language processor which bridges an execution gap without generating a machine language program.
- *Source language*: The program which forms the input to a language processor is a source program. The language in which the source program is written is known source language.
- *Target language*: The output of a language processor is known as the target program. The language, to which the target program belongs to, is called target language.

Procedure v/s Problem Oriented Language

Procedure oriented PL Problem oriented PL

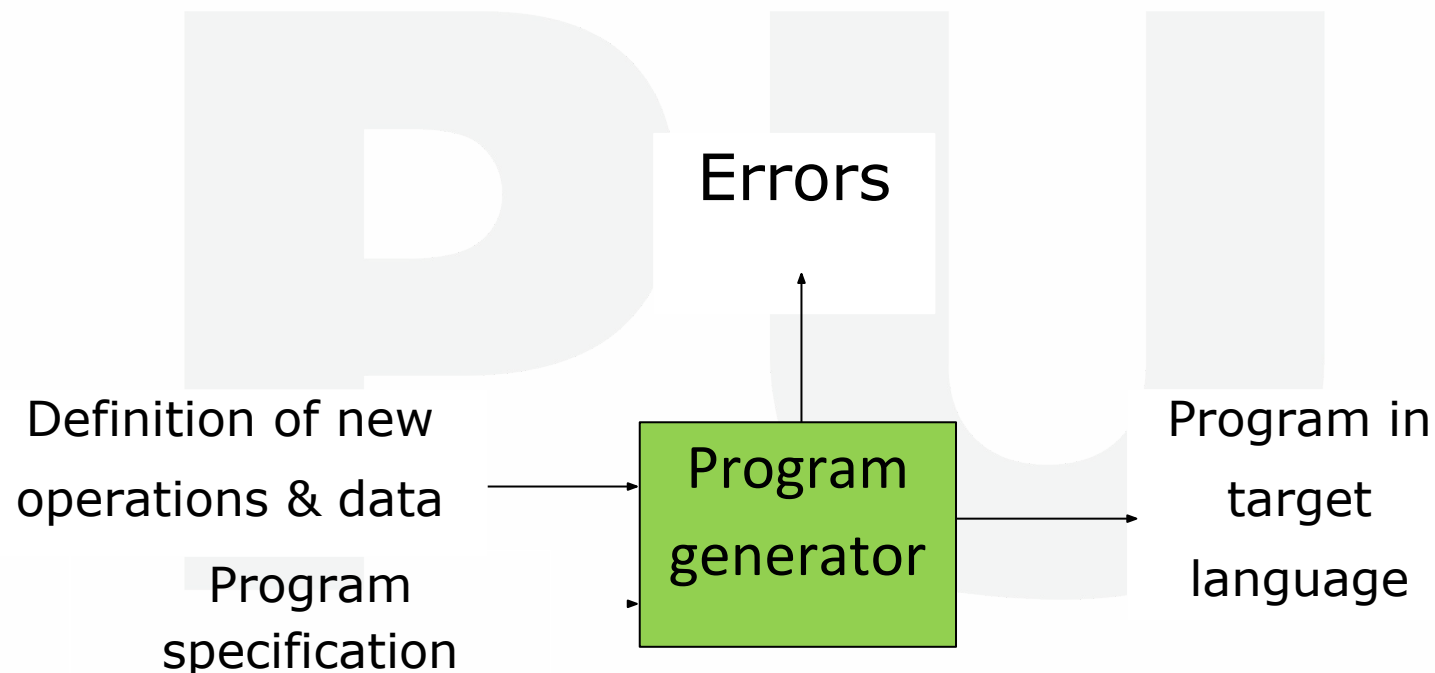
- Provides general purpose facilities for declaring data & performing operations useful in most application domains.
 - Independent of specific application domains and results in a large specification gap which has to be bridged by an application designer.
- Programming language features directly model the aspects of the application domain, which leads to very small specification gap.
 - Such a programming language can only be used for specific application; hence they are called problem oriented languages.

Language processing activity

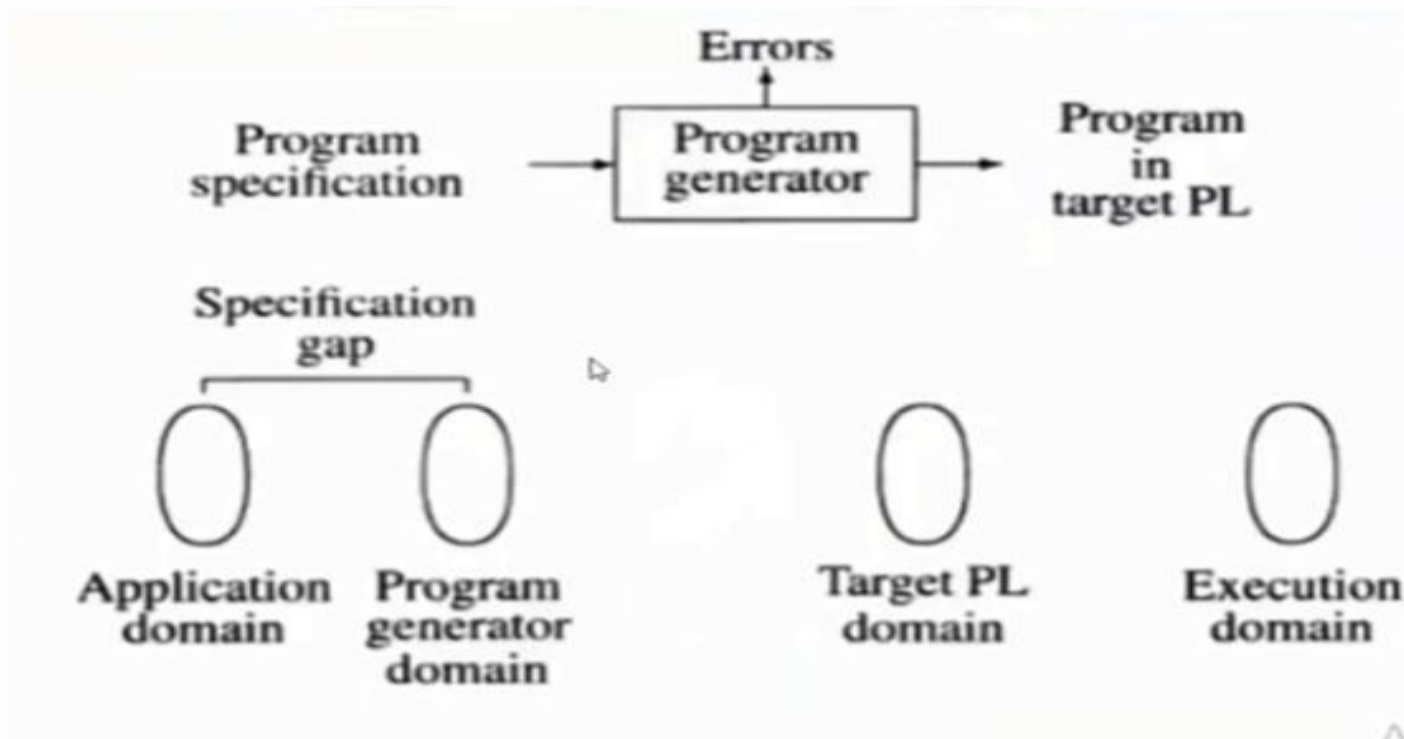
1. Program generation activities
 - A program generation activity aims an automatic generation of a program.
 - Program generator is a software, which accepts the specification of a program and generates a program in target language that fulfills the specification.
 - Program generator introduces a new domain between the application and programming language domain is called program generator domain.

Language processing activity

- A general purpose program generator



Program Generation activity



Advantages in Reduction in Specification

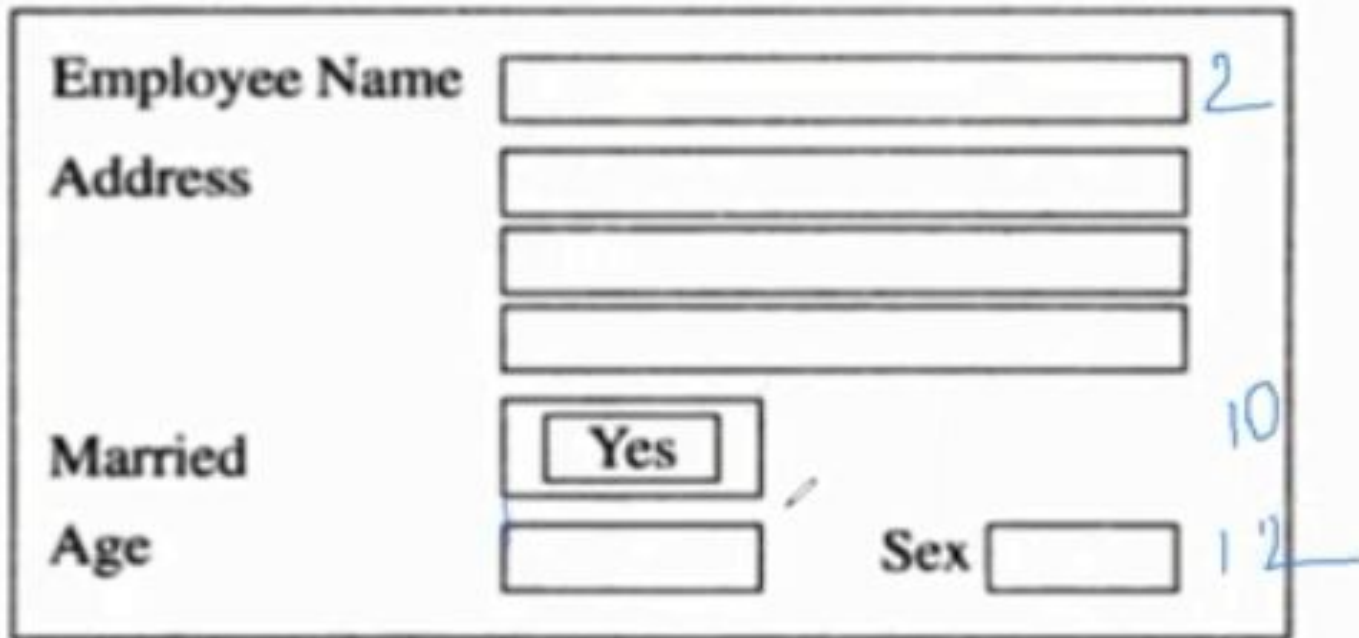
Increase the reliability of the generated program, as generated domain is close to the application domain.

It is easy for the designer to write the specification of the program to be generated.

Specification Gap

```
Employee name   : char : start (line=2,position=25)
                  end (line=2,position=80)
Married         : char : start (line=10,position=25)
                  end (line=10,position=27)
                  value ('yes','No') default ('Yes')
Age             : numeric : start (line=12,position=25)
                  end (line=12,position=26)
Sex             : char : start (line=12,position=35)
                  end (line=12,position=35)
```

Output of Screen Handling Program

A screenshot of a form displayed by a screen handling program. The form has a rectangular border. Inside, there are labels for 'Employee Name', 'Address', 'Married', 'Age', and 'Sex'. 'Employee Name' is followed by a single text box. 'Address' is followed by three stacked text boxes. 'Married' is followed by a button labeled 'Yes'. 'Age' is followed by a text box. 'Sex' is followed by a text box. Handwritten blue annotations are present: a '2' next to the 'Employee Name' box, a '10' next to the 'Married' button, and a '12' next to the 'Sex' box. There is also a small blue mark next to the 'Age' text box.

Employee Name	<input type="text"/>	2
Address	<input type="text"/>	
	<input type="text"/>	
	<input type="text"/>	
Married	<input type="button" value="Yes"/>	10
Age	<input type="text"/>	
Sex	<input type="text"/>	12

Figure 2.7 Screen displayed by a screen handling program



Program Execution Activity

- Program Execution activity reduces the Execution Gap.
- A Program may be executed through **Translation or interpretation.**

Language processing activity

Program Translation

■ Program Translation Characteristics

- A program must be translated before it can be executed.
- Translated program is saved in memory, so can be executed later without the need of translation.
- If source program is modified then it must be re-translated before execution.

A M/c Level Language Program generated by translator is saved in file and loaded in main memory and executed whenever desired.(No need to translate every time.)

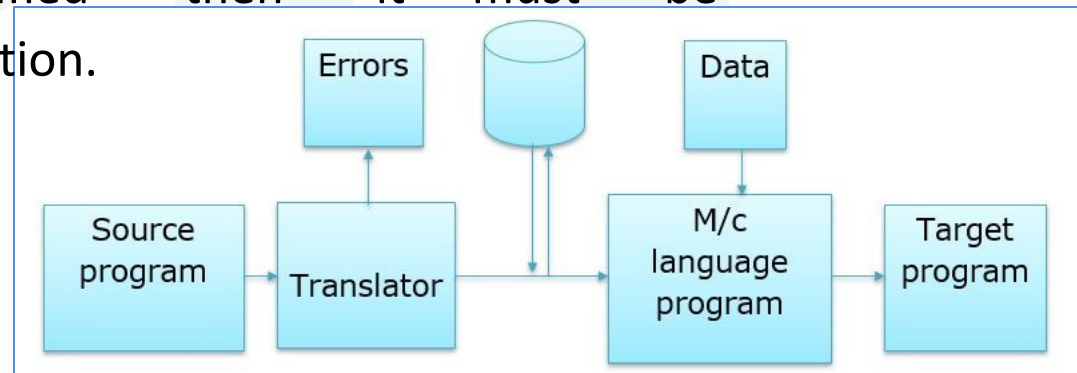


Image source : Google

Language processing activity

2. Interpretation

- The interpreter reads the source program and stores it in its memory.
- The CPU uses the program counter (PC) to note the address of the next instruction to be executed.
- The statement would be subjected to the interpretation cycle, which consists of the following steps:
 - » Fetch the instruction.
 - » Analyze the statement and determine its meaning, the computation to be performed and its operand.
 - » Execute the meaning of the statement.

Language processing activity

Interpretation

- The interpreter reads the source program and stores it in memory.
- The CPU uses the program counter (PC) to note the address of the next instruction to execute
- Execution of program is performed by repeating the execution cycle.

Steps of Execution cycle

- Fetch the instructions
- Decode the instruction
- Perform the operation

Language processing activity

Schematics of Interpretation

The statement would be subjected to the interpretation cycle, which consists of the following steps:

- » Fetch the instruction.
- » the statement and
Analyze its meaning,
computation to be performed and
determine the
its operand.
- » Execute the meaning of the
statement.

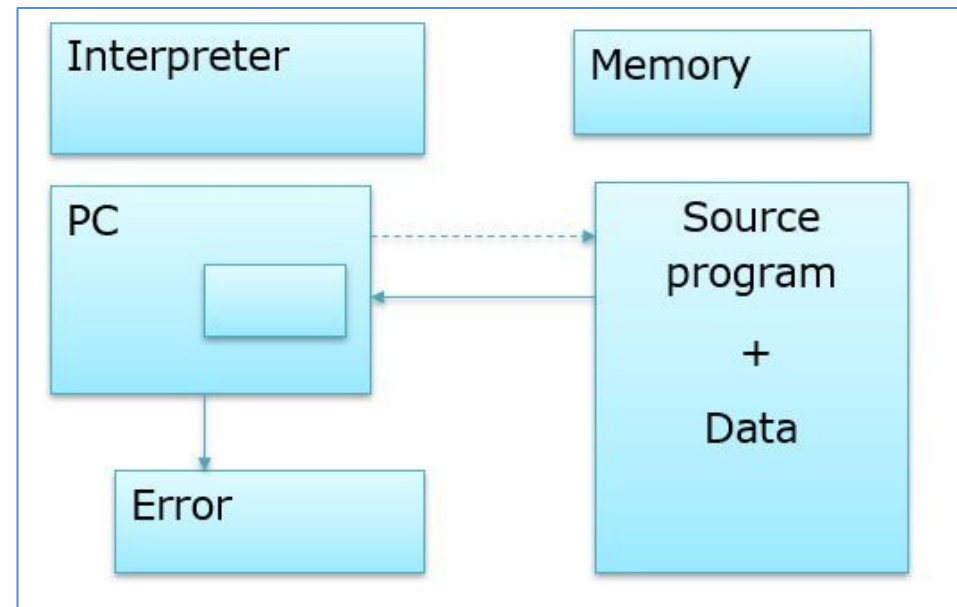


Image source : Google

Fundamental of language

*Language processing = Analysis of Source Program +
Synthesis of Target Program*

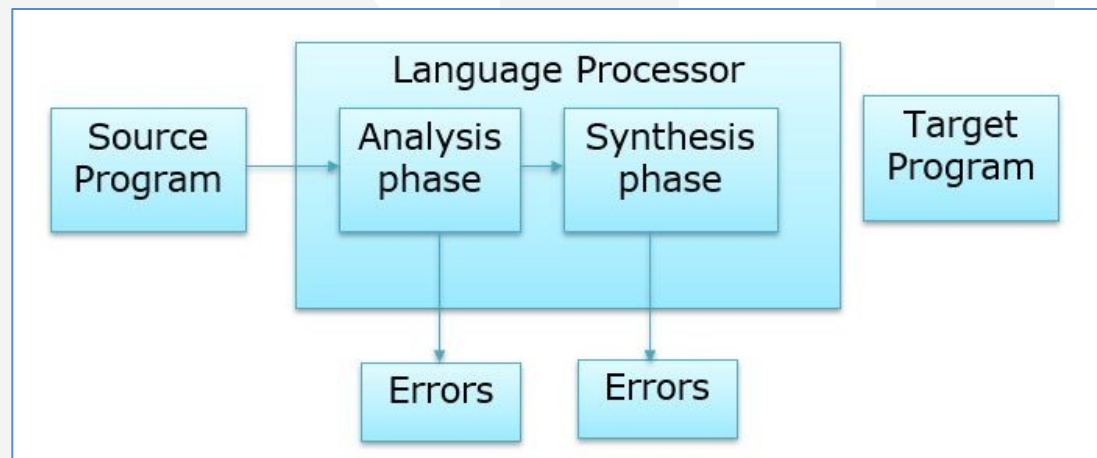
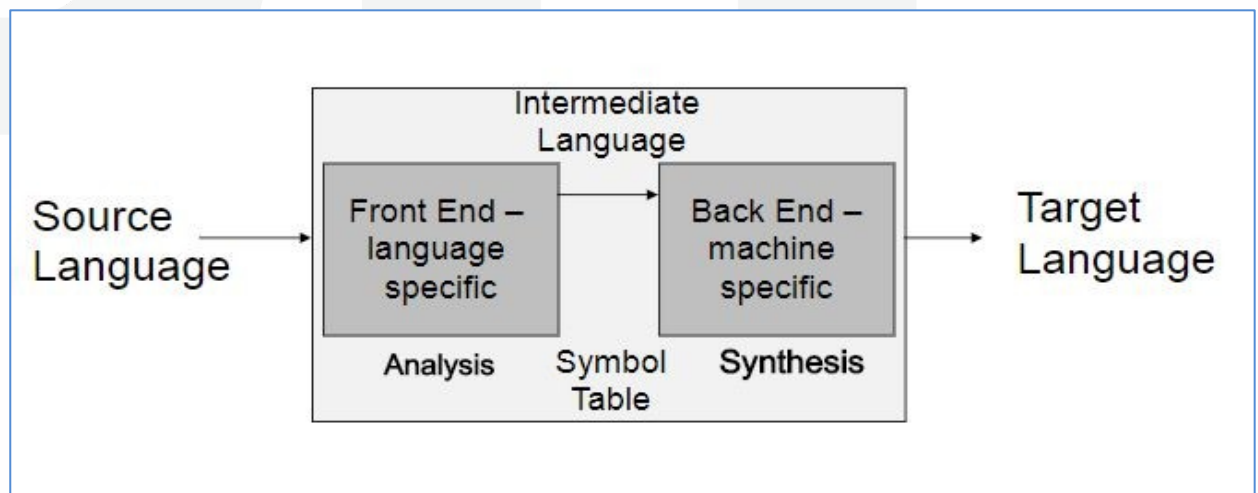


Image source : Google

Fundamental of language

- A specification of the source language forms the basis of source program
 1. Lexical rules govern formation of valid lexical units in the source language.
 2. Syntax rules govern formation of valid statements in the source language.
 3. Semantic rules associate meanings with valid statements of the source language.





Analysis Phase

Example

- We can say analysis of source statement consists of lexical, syntax and semantic analysis.
- Example:- **`percent_profit = (profit * 100) / cost_price;`**



Analysis Phase

Lexical analysis

- identifies following things
- =, * and / as operators
- 100 as constant
- Remaining strings as identifiers.

Analysis Phase

Syntax analysis

- Identifies the statement as the assignment statement.
- LHS and RHS is checked
- $a+b=c;$ will it work?,
- Its assignment statement only

Analysis Phase

Semantic analysis

- Determines the meaning of the statement.
- Assigns

$(\text{profit} \times 100) / \text{cost_price}$ to **percent_profit**

RHS assigned to LHS

Synthesis Phase

This consists of two main Activities

1. Creation of Data Structure in the Target Program (Memory Allocation)
2. Generation of Target Code (Code generation)



Synthesis Phase

- A Language Processor generates following assembly language statements for

percent_profit = (profit * 100) / cost_price

	MOVER	AREG, PROFIT
	MULT	AREG, 100
	DIV	AREG, COST_PRICE
	MOVEM	AREG, PERCENT_PROFIT
	...	
PERCENT_PROFIT	DW	1
PROFIT	DW	1
COST_PRICE	DW	1

Phases of language Processor

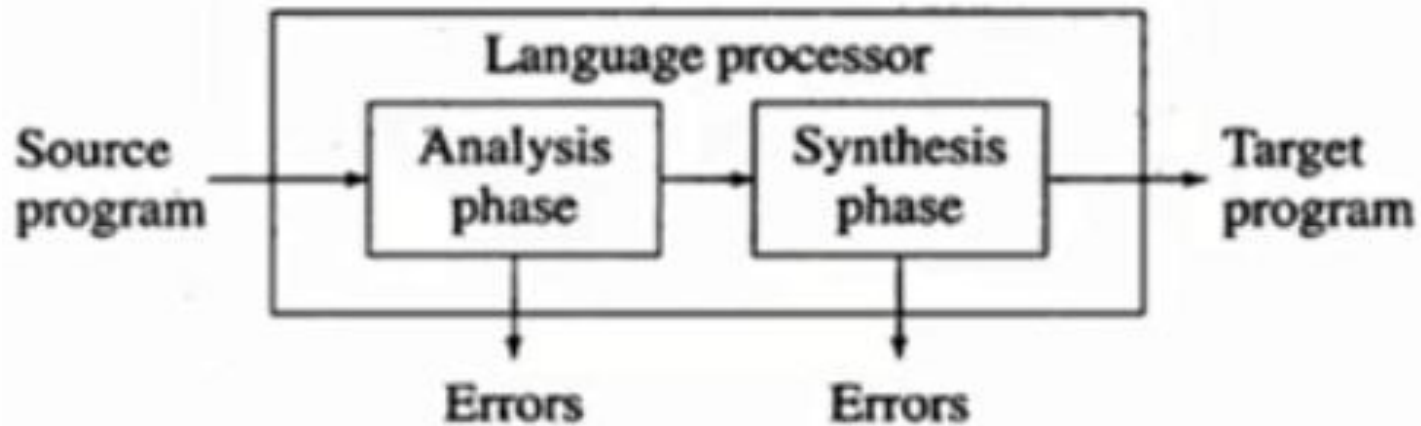
Analysis Phase

- Analysis phase uses each component of source language to determine relevant information concerning a statement in the source statement
- Analysis of source statement consists of lexical, syntax and semantic analysis.
(Front end)

Synthesis phase

- synthesis phase is concerned with the construction of target language.
- It includes mainly two activities memory allocation and code generation.
(Back end)

Phases of language Processor



Phases of language Processor

- The language processor diagram gives the impression that language processing can be performed on statement by statement basis, that is a analysis of source statement is immediately followed by synthesis of equivalent target statements.
- This may not be feasible due to following two issues.

Phases of language Processor

- Forward reference
- Issues concerning memory requirements and organization of language processor.

Phases of language Processor

Forward Reference

“A forward reference of a program entity is reference to the entity which precedes its definition in the program”

```
percent_profit := (profit * 100) / cost_price;
```

.....

.....

```
long profit;
```



Phases of language Processor

Issues concerning memory requirements and organization of language processor.

- `int a;`
- `int abcdefghi;`
- `int a[5];`
- `int abcdefghi[5];`
- Size of a language processor.



Pass of Language Processor

- ❖ A Language Processor Pass is processing of every statement in the source program to perform a Language Processing Activities.
 - Pass I
 - Perform analysis of the source program and note relevant information.
 - The first pass performs analysis of the source program and reflects its results in the intermediate representation.
 - Pass II
 - Perform synthesis of target program.
 - The second pass reads and analyzes the intermediate representation to perform synthesis of the target program.

Pass of Language Processor

Example(Solving
Problem of forward
reference)

```
percent_profit := (profit * 100) / cost_price;
```

```
.....
```

```
.....
```

```
long profit;
```

- Here information about profit is unknown and noted in pass 1
- In pass 2 information about profit is noted and translation is performed in two passes

Pass of Language Processor

Two Pass Translation



In pass I language processor analyses the source program to note the type of the information.

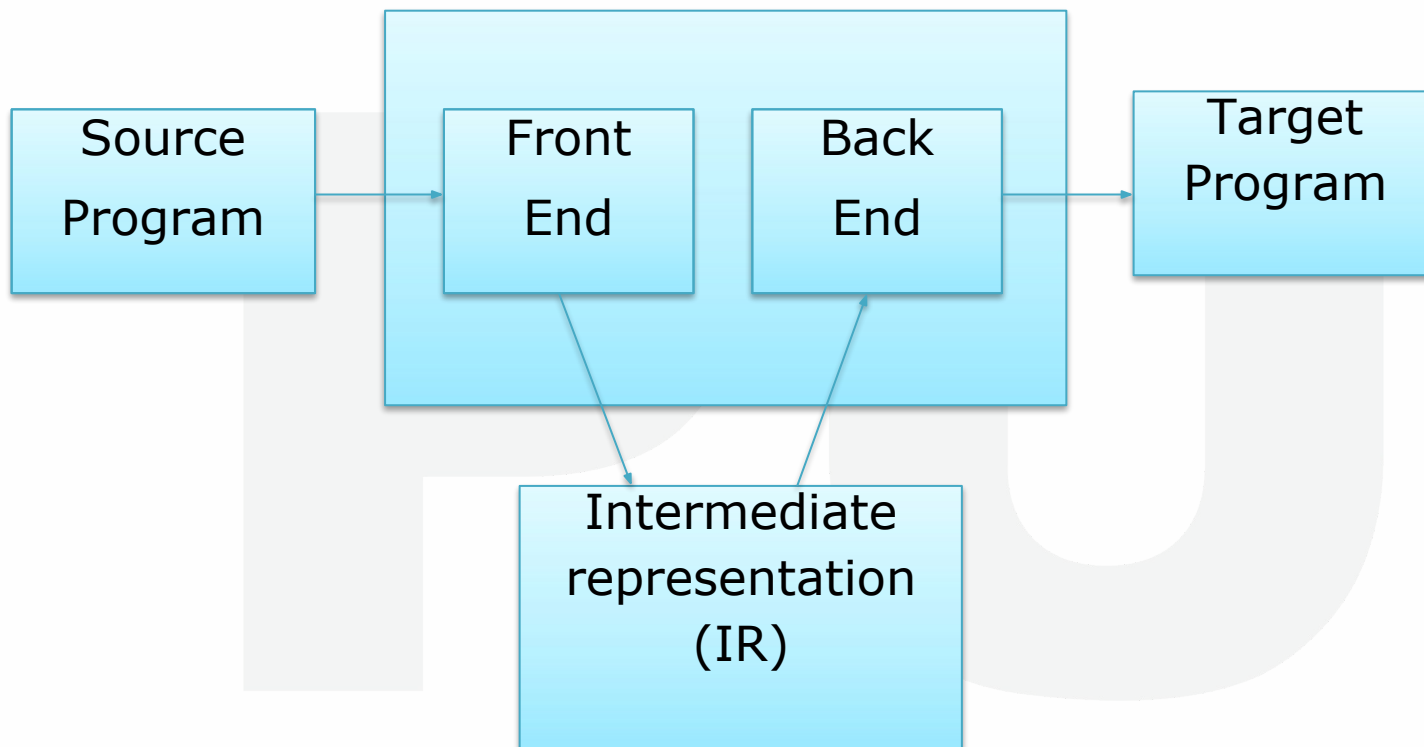


In pass II it once again analyses the source program to generate the target code using the type of information noted in pass I.



So this repeated visit to the source program can be avoided using an IR, intermediate representation of the source program.

Two-pass schematic of language





Passes? Front End? Back End?



The first pass analyses the source program and reflects its result in IR. This is called and **front end**.

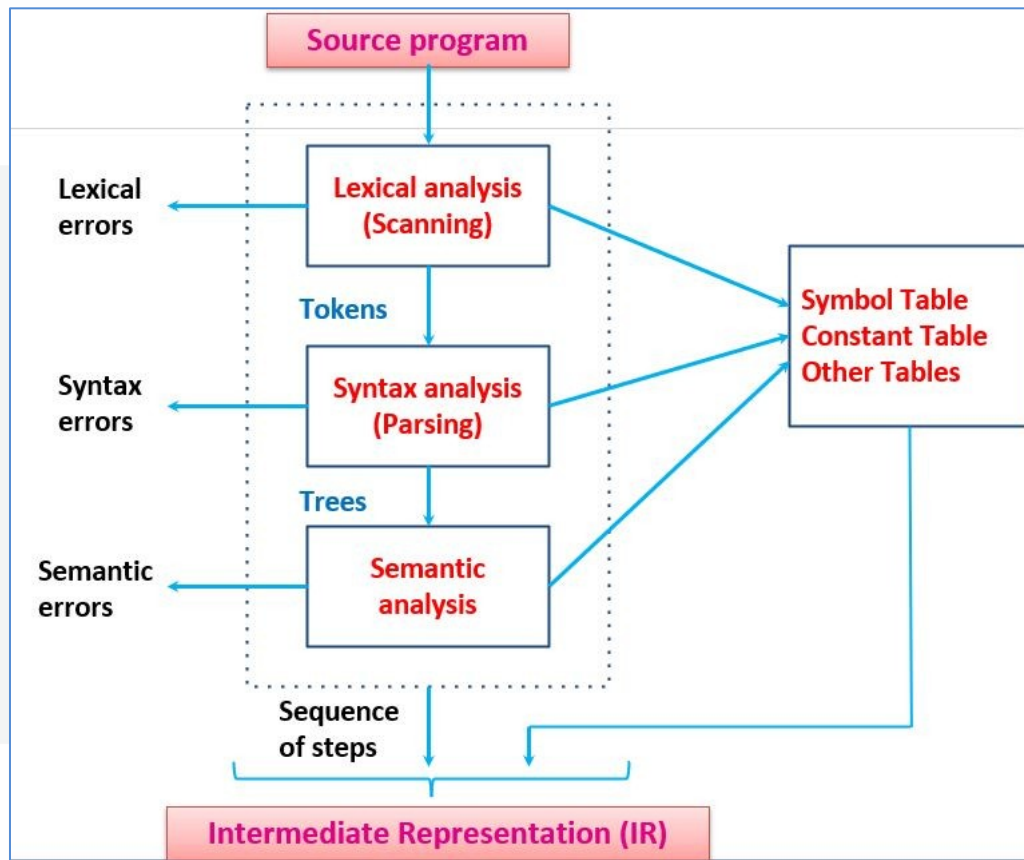


The second pass reads and analyses the IR, instead of source program, and perform the synthesis of the target program. This is called as **back end**.

Intermediate representation (IR)

- An intermediate representation is a representation of a source program which reflects the effect of some, but not all analysis and synthesis functions performed during language processing.
- An IR properties,
 - Ease of use
 - Processing efficiency
 - Memory efficiency

Front end of Language Processor



Front end of Language Processor

Intermediate representation (IR)

- Output of front end – IR
- IR contains
 1. Intermediate Code (which is the description of the source program).
 2. Tables –Symbol Table

Symbol Table

Most important table is symbol table

Contains information about all identifiers used in the program

It is built during lexical analysis

Semantic analysis adds information in the symbol table



Intermediate representation (IR)

Intermediate Code

IC is a sequence of IC units.

Each IC unit is representing the meaning of one action in source program.

IC units contains reference to the information in the various tables.

Example 1.7 Figure 1.13 shows the IR produced by the analysis phase for the program

```
i : integer;
a,b : real;
a := b+i;
```

Symbol table

	<i>symbol</i>	<i>type</i>	<i>length</i>	<i>address</i>
1	i	int		
2	a	real		
3	b	real		
4	i*	real		
5	temp	real		

Intermediate code

1. Convert (Id, #1) to real, giving (Id, #4)
2. Add (Id, #4) to (Id, #3), giving (Id, #5)
3. Store (Id, #5) in (Id, #2)

Memory Allocation and Code Generation

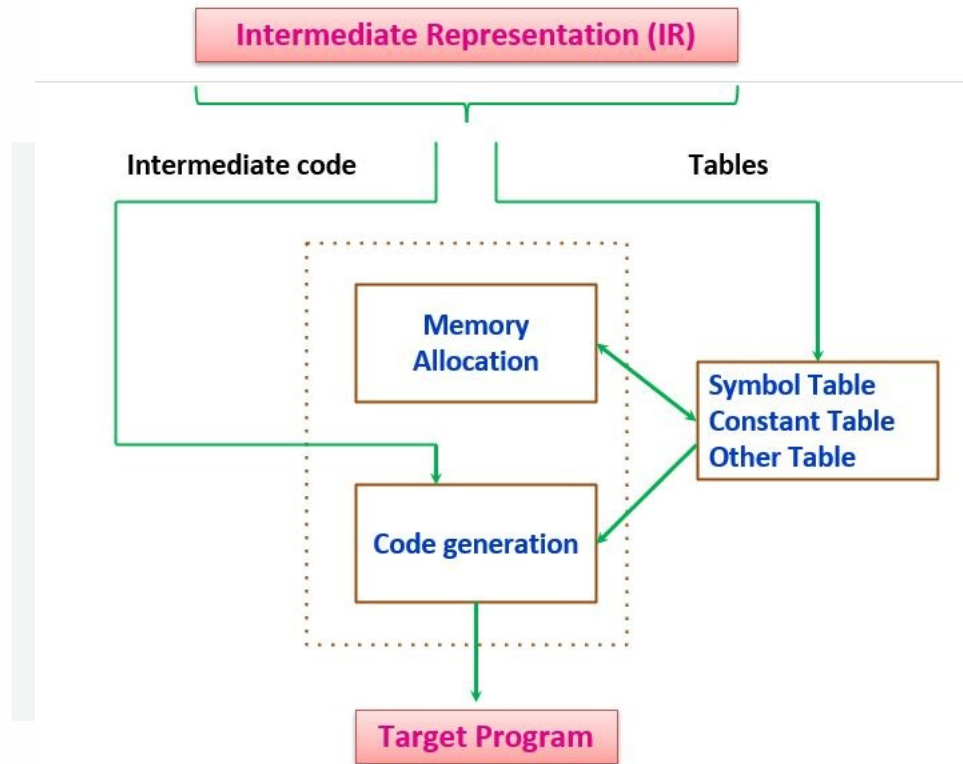
Memory Allocation

- The back end computes the memory requirement of a variable from its type, length and dimensionality information found in the symbol table and allocates memory to it.
- The address of the allocated memory area is entered in the symbol table.

Code Generation

- Two key decisions involved in generating good quality target code are:
 1. What instructions should be used for each of the actions in the intermediate code?
 2. Which CPU registers should be used for evaluating expressions ?

Back end of Language Processor



Back end of Language Processor

Symbol Table

- An identifier used in the source program is called a symbol.
- Names of variables, functions and procedures are symbols.
- A language processor uses the symbol table to maintain the information about attributes of symbols used in a source program

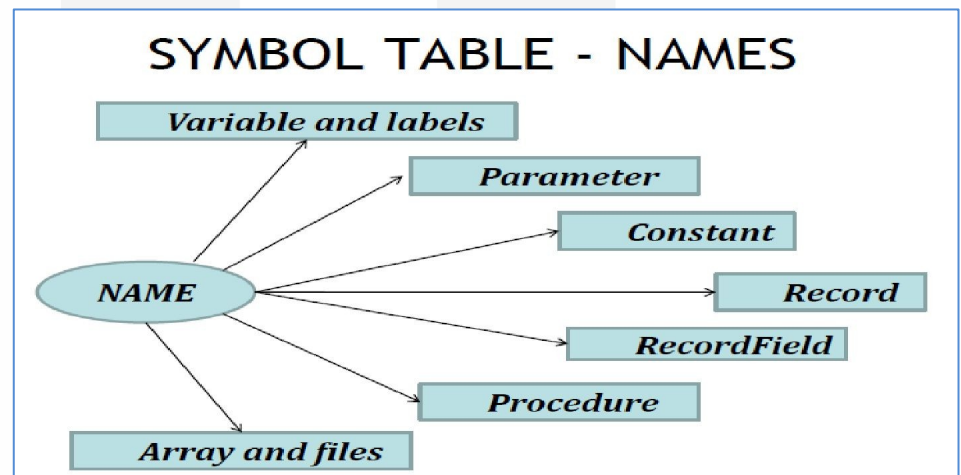
Symbol Table

- Language processor performs the following four kinds of operations on the symbol table:
 1. **Add a symbol and its attributes:** Make a new entry in the symbol table.
 2. **Locate a symbol's entry:** Find a symbol's entry in the symbol table.
 3. **Delete a symbol's entry:** Remove the symbol's information from the table.
 4. **Access a symbol's entry:** Access the entry and set, modify or copy its attribute information.

Symbol Table

Use of Symbol Table

- Symbol table information is used by the analysis and synthesis phases
- To verify that used identifiers have been defined (declared)
- To verify that expressions and assignments are semantically correct – type checking
- To generate intermediate or target code



References

1. System Programming by D M Dhamdhare McGraw Hill Publication
2. System Programming by Srimanta Pal OXFORD Publication
3. System Programming and Compiler Construction by R.K. Maurya & A. Godbole.

× ○ DIGITAL LEARNING CONTENT



Parul[®] University



www.paruluniversity.ac.in