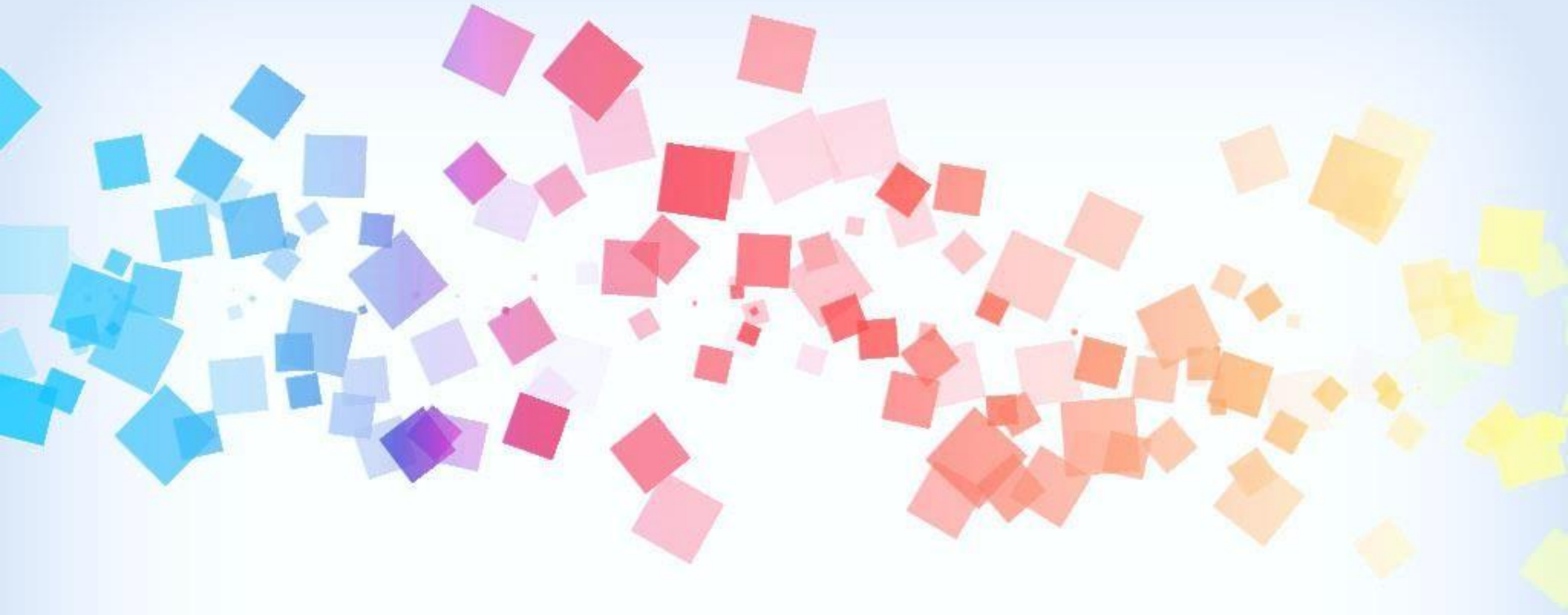


Compiler 1

Lecture 2: Compiler structure

BY: Assist Prof. Dr. Ielaf O Abdul Majjed

Compiler Architecture

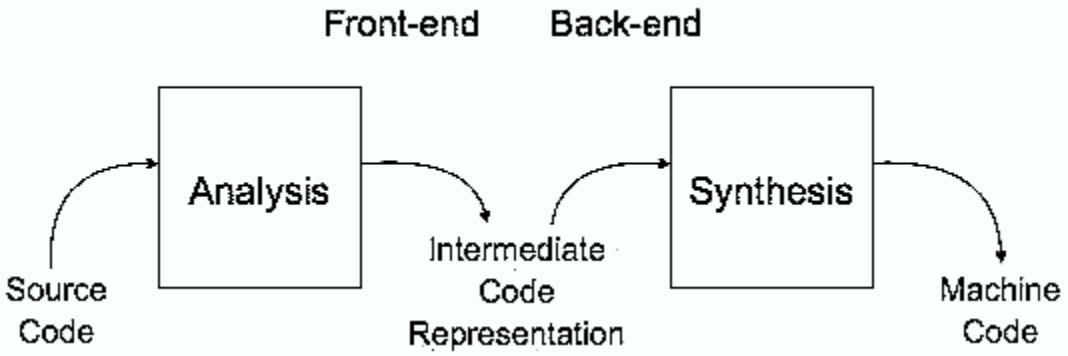
A compiler can broadly be divided into two phases based on the way they compile.

1. Analysis Phase

Known as the **Front-End** of the compiler, the analysis phase of the compiler reads the source program, divides it into core parts and then checks for lexical, grammar and syntax errors. The analysis phase generates an intermediate representation of the source program and symbol table, which should be fed to the Synthesis phase as input.

2.Synthesis Phase

Known as the **Back-End** of the compiler, the synthesis phase generates the target program with the help of intermediate source code representation and symbol table.



The Phases of a Compiler

The compilation process is a sequence of various phases. Each phase takes input from its previous stage, has its own representation of source program, and feeds its output to the next phase of the compiler. Let us understand the phases of a compiler.

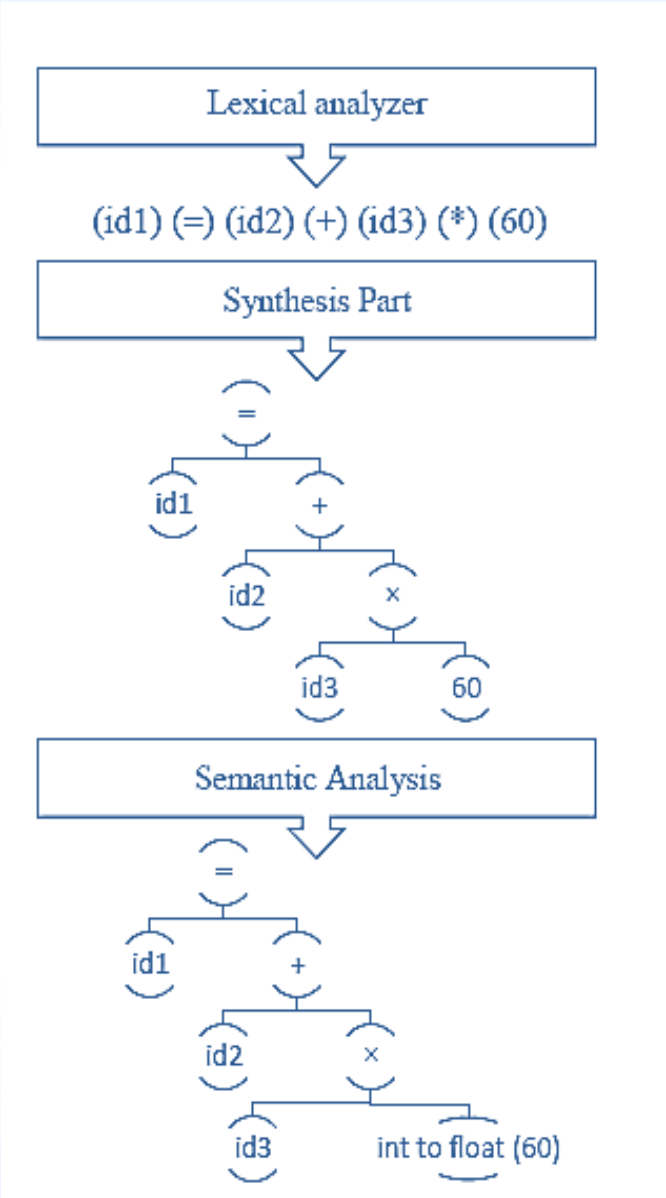
1. Lexical Analyzer.
2. Syntax Analyzer.
3. Semantic Analyzer.
4. Intermediate Code Generator.
5. Code Optimizer.
6. Code Generator.

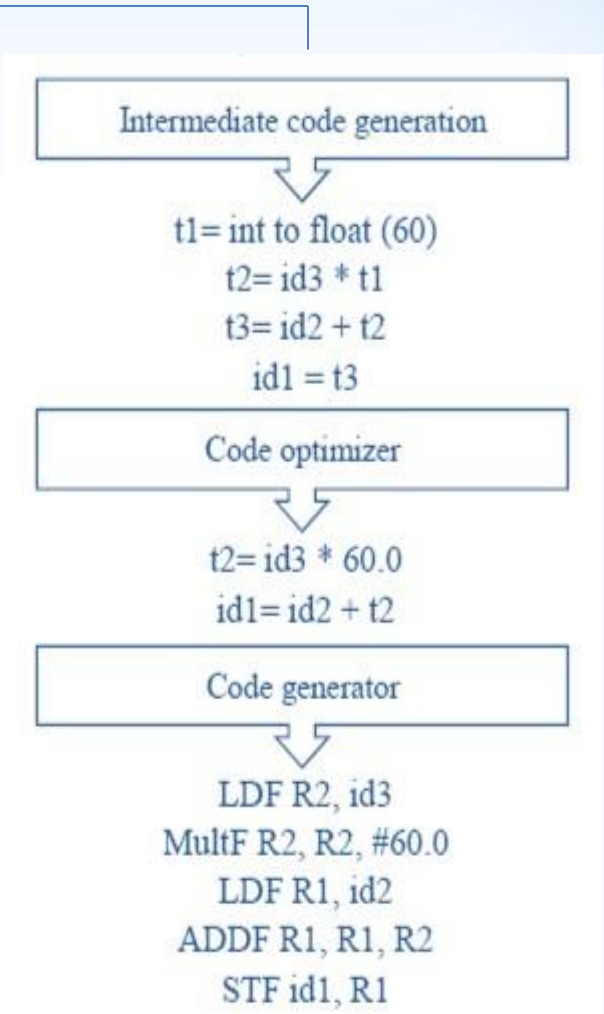
In each phase we need variables that can be obtained from a table called **Symbol Table** manager, and in each phase some errors may be generated so we must have a program used to handle these errors , this program called **Error Handler.**

Symbol Table:- It is a data-structure maintained throughout all the phases of a compiler. All the identifiers' names along with their types are stored here. The symbol table makes it easier for the compiler to quickly search the identifier record and retrieve it.

Error Handler:- Each phase can produce errors. However, after detecting an error, a phase must deal with that error, so that the compilation can proceed. So dealing with that error is done by a program known as Error Handler which is software used to handle any error that may be produced from any phase and it is needed in all phases of the compliers.

Example of the Compilation Process Consider the following sentence is segment source program: X: = a + b *60.0



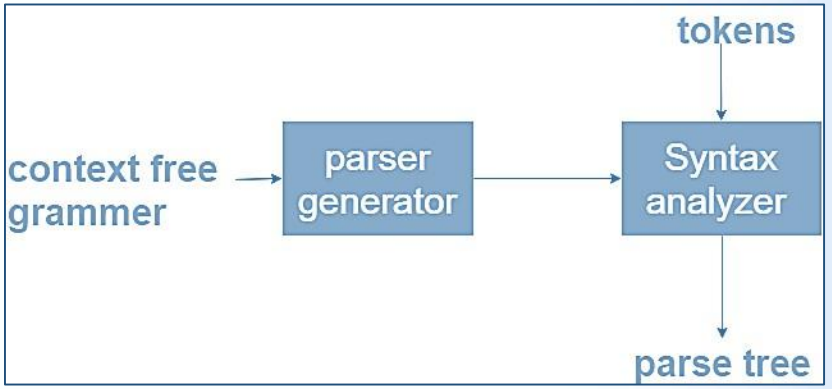


Compiler construction tools

The compiler writer can use some specialized tools that help in implementing various phases of a compiler. These tools assist in the creation of an entire compiler or its parts. Some commonly used compiler construction tools include:

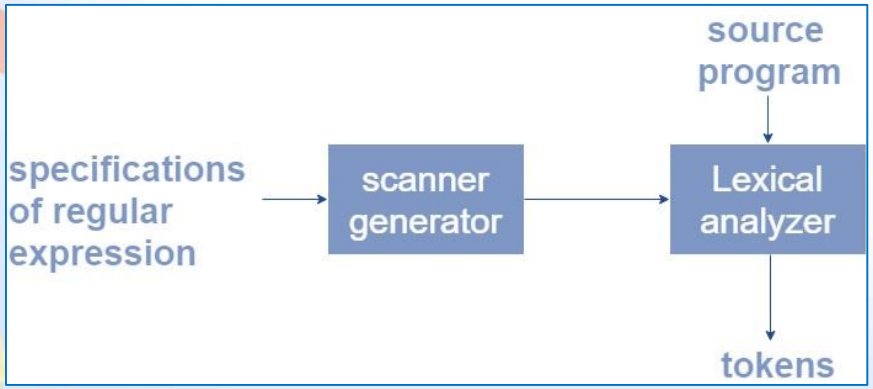
1- Parser Generator:

It produces syntax analyzers (parsers) from the input that is based on a grammatical description of programming language or on a context-free grammar. It is useful as the syntax analysis phase is highly complex and consumes more manual and compilation time.



2. Scanner Generator:

It generates lexical analyzers from the input that consists of regular expression description based on tokens of a language. It generates a finite automaton to recognize the regular expression.



3.Syntax directed translation engines:

It generates intermediate code with three address format from the input that consists of a parse tree. These engines have routines to traverse the parse tree and then produces the intermediate code. In this, each node of the parse tree is associated with one or more translations.

4.Automatic code generators:

It generates the machine language for a target machine. Each operation of the intermediate language is translated using a collection of rules and then is taken as an input by the code generator. A template matching process is used. An intermediate language statement is replaced by its equivalent machine language statement using templates.

5.Data-flow analysis engines:

It is used in code optimization. Data flow analysis is a key part of the code optimization that gathers the information, that is the values that flow from one part of a program to another.

6.Compiler construction toolkits:

It provides an integrated set of routines that aids in building compiler components or in the construction of various phases of compiler.

Error Handling in Compiler Design

The tasks of the Error Handling process are to detect each error, report it to the user, and then make some recovery strategy and implement them to handle the error. During this whole process processing time of the program should not be slow.

Functions of Error Handler:

- Error Detection
- Error Report
- Error Recovery

Errors in the program should be detected and reported by the parser. Whenever an error occurs, the parser can handle it and continue to parse the rest of the input. Although the parser is mostly responsible for checking for errors, errors may occur at various stages of the compilation process.

So, there are many types of errors and some of these are:

There are three types of error: logic, run-time and compile-time error:

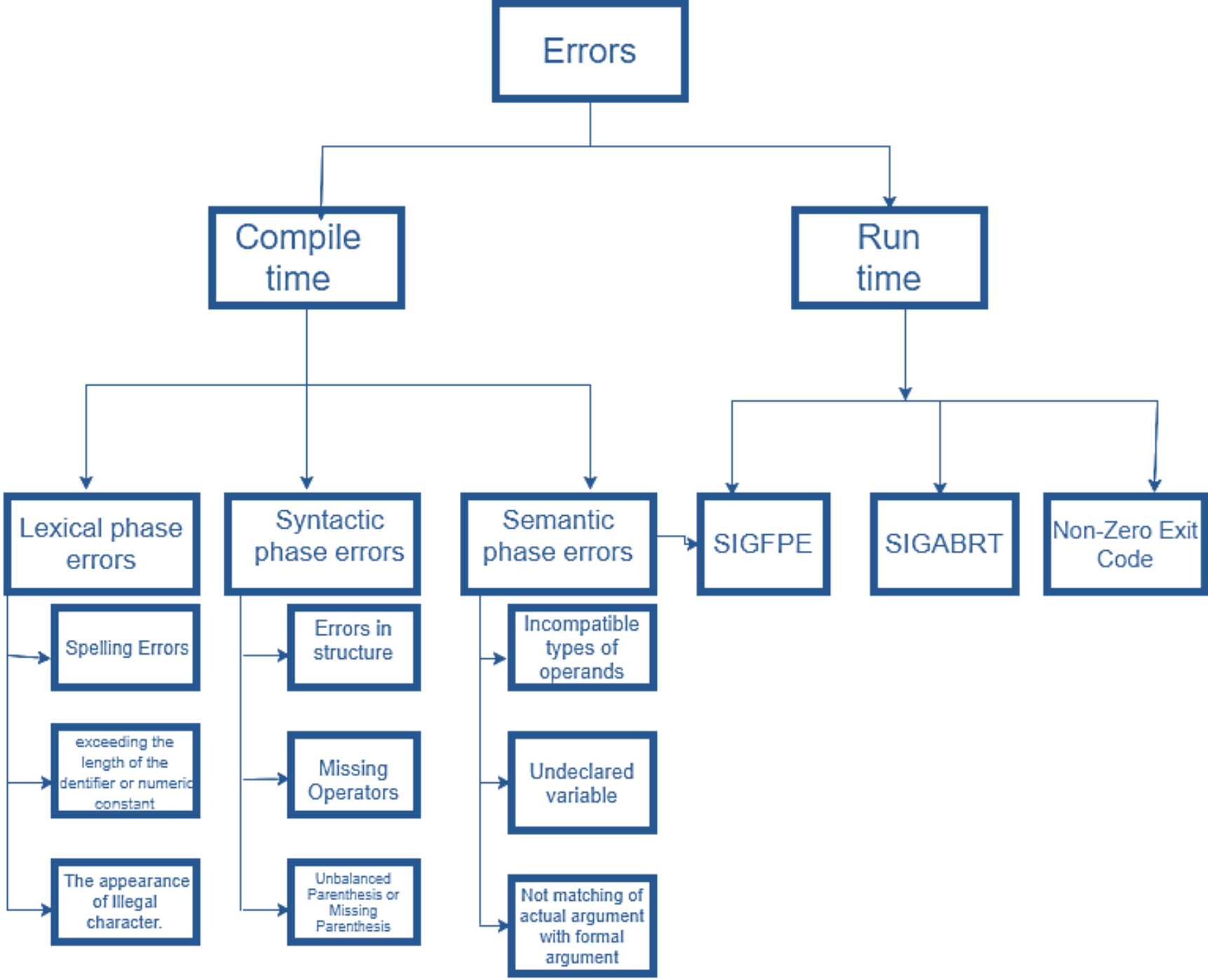
1.Logic errors occur when programs operate incorrectly but do not terminate abnormally (or crash). Unexpected or undesired outputs or other behaviour may result from a logic error, even if it is not immediately recognized as such.

2.Run-time error is an error that takes place during the execution of a program and usually happens because of adverse system parameters or invalid input data. The lack of sufficient memory to run an application or a memory conflict with another program and logical error is an example of this. Logic errors occur when executed code does not produce the expected result. Logic errors are best handled by meticulous program debugging.

3.Compile-time errors rise at compile-time, before the execution of the program. Syntax error or missing file reference that prevents the program from successfully compiling is an example of this.

Classification of Compile-time error

- Lexical : This includes misspellings of identifiers, keywords or operators
- Syntactical : a missing semicolon or unbalanced parenthesis
- Semantical : incompatible value assignment or type mismatches between operator and operand
- Logical : code not reachable, infinite loop.


Errors
Compile time
Run time
Lexical phase errors
Syntactic phase errors
Semantic phase errors
SIGFPE
SIGABRT
Non-Zero Exit Code
Spelling Errors
exceeding the length of the dentifier or numeric constant
The appearance of Illegal character.
Errors in structure
Missing Operators
Unbalanced Parenthesis or Missing Parenthesis
Incompatible types of operands
Undeclared variable
Not matching of actual argument with formal argument