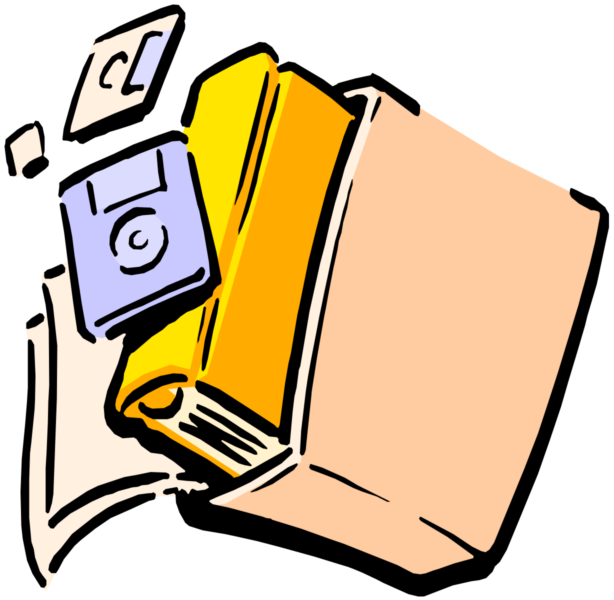
Introduction to Programming Tools

Programs

- A program is a set of step-by-step instructions that directs the computer to do the tasks you want it to do and produce the results you want.



What Is a Programming Language

- A natural language is designed to communicate between human
- A programming language is designed to communicate between human and computers

Programming Languages

- A programming language is a set of rules that provides a way of telling a computer what operations to perform.

Programming Language : Definition

- A vocabulary and set of grammatical rules for instructing a computer to perform specific tasks.

What does the computer understand?

- Computer only understands machine language instructions.

Computer Language

- Digital devices have two stable states, which are referred to as zero and one by convention
- The binary number system has two digits, 0 and 1. A single digit (0 or 1) is called a *bit*, short for *bi*nary digi*t*. A byte is made up of 8 bits.
- Binary Language: Data and instructions (numbers, characters, strings, etc.) are encoded as binary numbers - a series of bits (one or more bytes made up of zeros and ones)

Computer Language (cont.)

- Encoding and decoding of data into binary is performed automatically by the system based on the encoding scheme
- Encoding schemes
  - Numeric Data: Encoded as binary numbers
  - Non-Numeric Data: Encoded as binary numbers using representative code
    - ASCII – 1 byte per character
    - Unicode – 2 bytes per character

Programming Languages

- Computers cannot use human languages, and programming in the binary language of computers is a very difficult, tedious process
- Therefore, most programs are written using a programming language and are converted to the binary language used by the computer
- Three major categories of prog languages:
  - Machine Language
  - Assembly Language
  - High level Language

Programming Language Categories

- Machine Language
  - Binary coded instructions
- Assembly Language
  - Symbolic coded instructions
- Procedural Languages
  - procedural statements or arithmetic notation
- Four-generation Languages
  - Natural language and nonprocedural statements
- Object-oriented Languages
  - Combination of objects and procedures

Evolution of Programming languages

- First Generation : Machine languages
  - Strings of numbers giving machine specific instructions
  - Example:

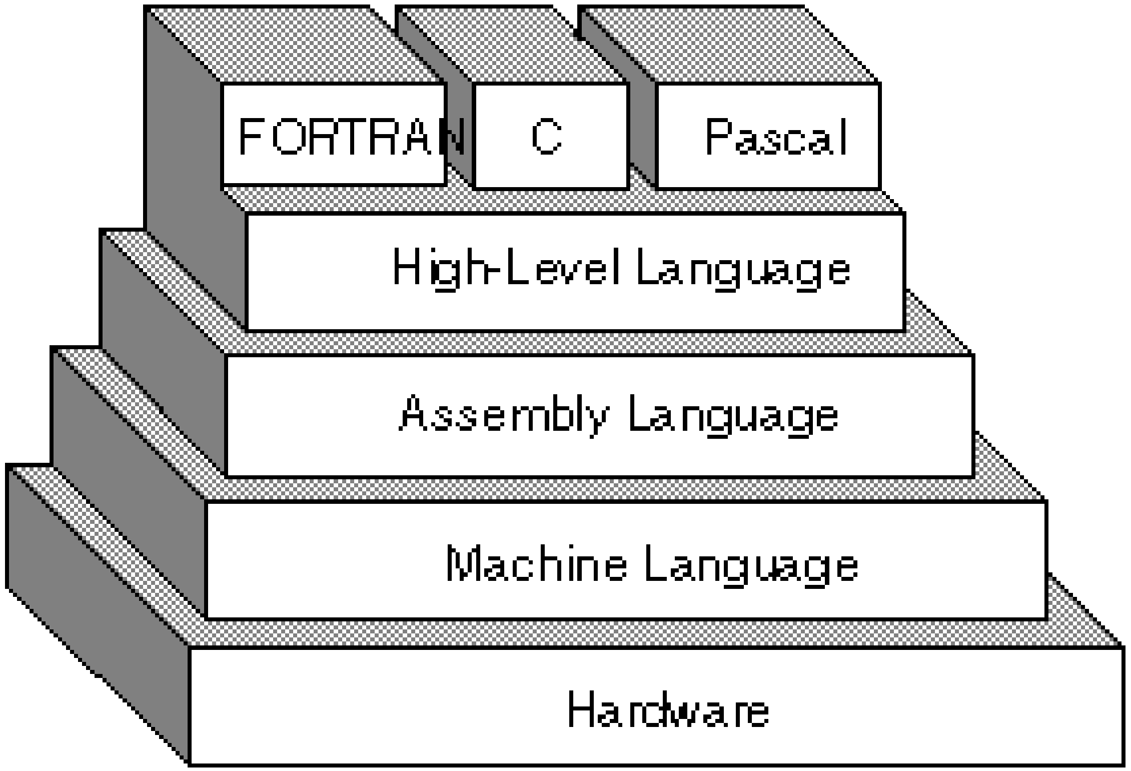
```
+1300042774
+1400593419
+1200274027
```

- Second Generation : Assembly languages
  - English-like abbreviations representing elementary computer operations (translated via assemblers)
    - Example:

```
LOAD    BASEPAY
ADD     OVERPAY
STORE   GROSSPAY
```

- Third Generation : High-level languages
  - Codes similar to everyday English
  - Use mathematical notations (translated via compilers)
  - Example: `grossPay = basePay + overTimePay`

PL hierarchy



Machine Language

- Natural language of a particular computer
- Primitive instructions built into every computer
- The instructions are in the form of binary code
- Any other types of languages must be translated down to this level

Machine Languages

- different for each computer processor

```
0100
001101 100000 001101 110001
00101  10001   10000
01110
111001
. . .
```

Assembly Languages

- English-like Abbreviations used for operations (Load R1, R8)
- Assembly languages were developed to make programming easier
- The computer cannot understand assembly language - a program called assembler is used to convert assembly language programs into machine code

Assembly Languages

- different for each computer processor

```
main   proc pay
       mov ax, dseg
       mov ax, 0b00h
       add ax, dx
       mov al, bl
       mul bl, ax
       mov bl, 04h
```

Assembly Language

- When to use
  - When speed or size of program is critical
  - Hybrid approach
  - Hardware Drivers
  - Can use specialized instructions
- Disadvantages
  - Inherently machine specific
  - Architectures may become obsolete
  - Lack of programming structure

High Level Languages

- English-like and easy to learn and program
- Common mathematical notation
  - Total Cost = Price + Tax;
  - area = 5 * 5 * 3.1415;
- Java, C, C++, FORTRAN, VISUAL BASIC, PASCAL

High-Level Languages

- Higher Level Languages
  - Use traditional programming logic where the programming instructions tell the computer what to do and how to perform the required operations.

Assembler

- Instructions written in assembly language must be translated to machine language instructions :
  - Assembler does this
- One to one translation : One AL instruction is mapped to one ML instruction.
- AL instructions are CPU specific.

Compiler

- Instructions written in high-level language must be translated to machine language instructions :
  - Compiler does this
- Generally one to many translation : One HL instruction is mapped to many ML instruction.
- HL instructions are not CPU specific but compiler is.

Programming Tools Overview

- Editors
- Assemblers
- Debuggers
- Compilers
- Linkers
- Loaders
- Interpreters

Integrated Development Environments (IDEs) combine several of the above programming tools

Example of an HLL Program

makes input and output available to us

```
#include     <stdio.h>

int          main(void)
{
        printf("This is my first C program.\n");
        return(0);
}
```

header

statements

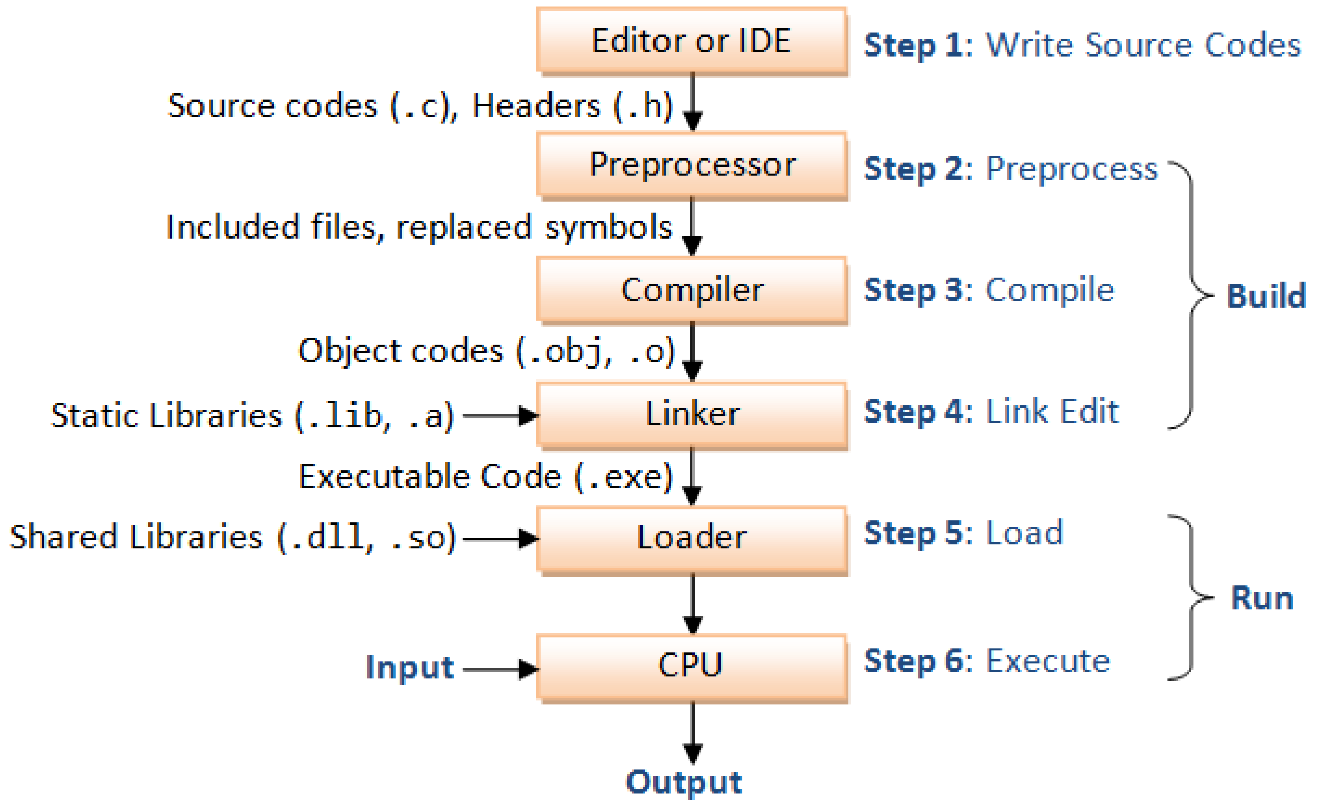
open and close braces mark the beginning and end

From source to execution

  - A programmer writes a
- Source file (helloworld.c file)
  - A compiler then translates it into an
- Object module (helloworld.obj file)
  - The linker combines various object modules it an
- Executable image (helloworld.exe file)
  - The loader does the final work in getting the image executing on the system

- But what does a ".obj" or a ".exe" file really contain?

Translation from HLL to ML (C program)



Summary of Program Development Phases

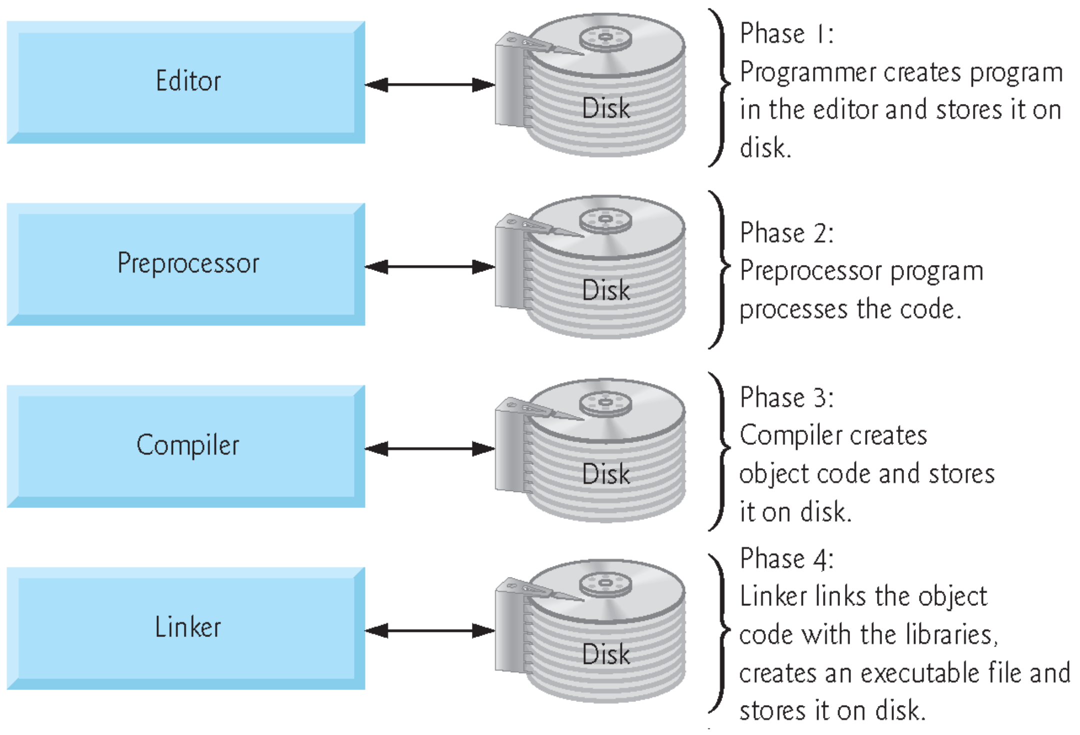

Editor
Disk
Phase 1:
Programmer creates program in the editor and stores it on disk.
Preprocessor
Disk
Phase 2:
Preprocessor program processes the code.
Compiler
Disk
Phase 3:
Compiler creates object code and stores it on disk.
Linker
Disk
Phase 4:
Linker links the object code with the libraries, creates an executable file and stores it on disk.

Fig. 1.7 | Typical C development environment. (Part 1 of 3.)

Summary of Program Development Phases

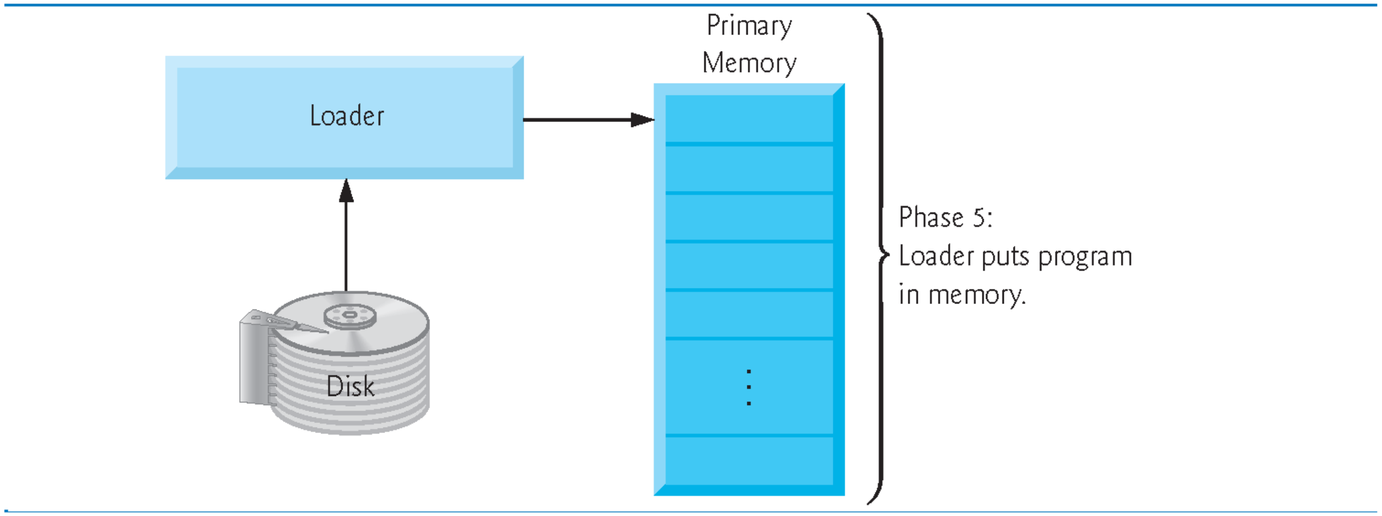


Fig. 1.7 | Typical C development environment. (Part 2 of 3.)

Editors

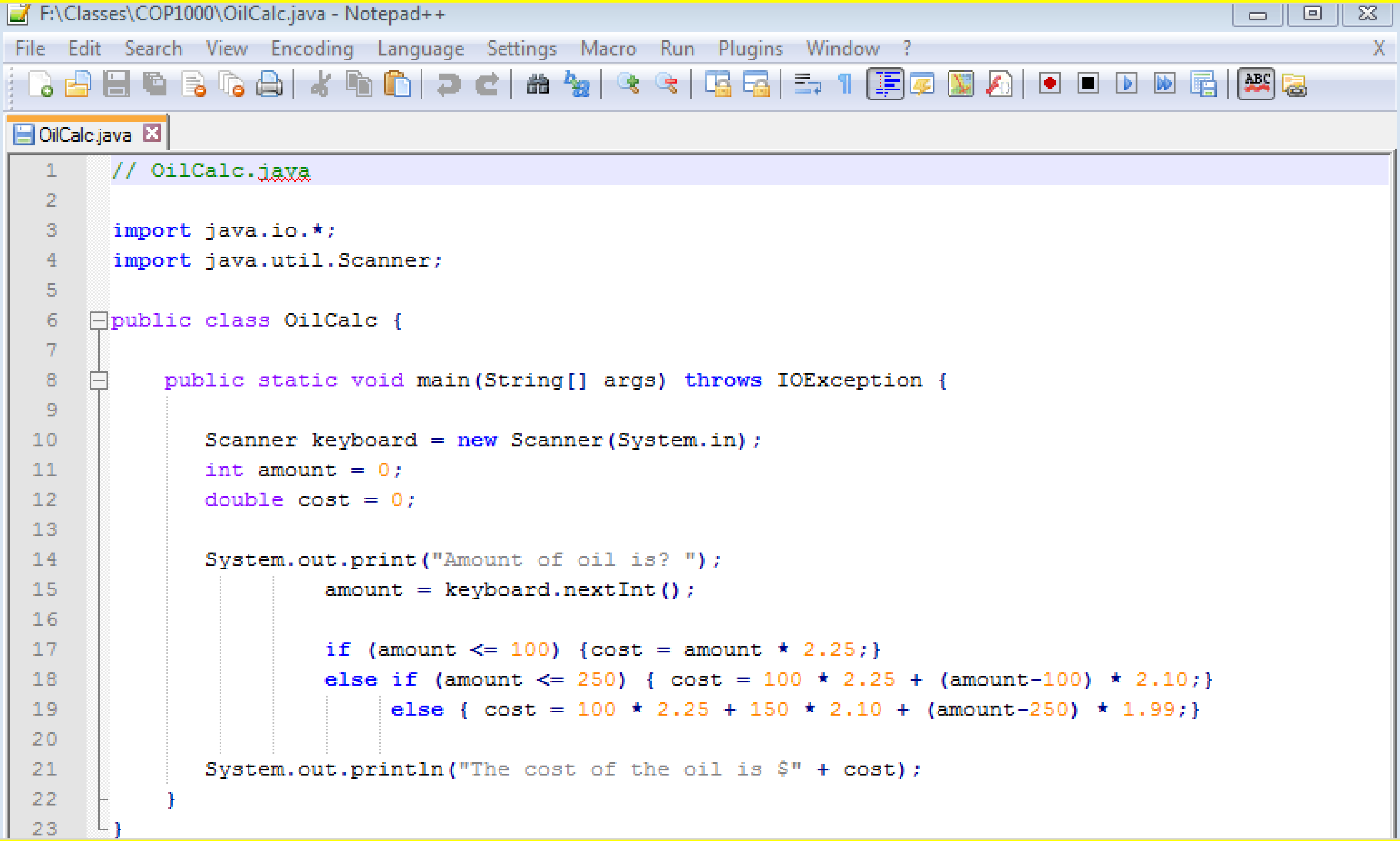
Programming Environment

- When designing, use tools to generate flowcharts and pseudo code
  - SFC, Flowgorithm, and Raptor for flowcharts
- When programming use an IDE
  - At a minimum an IDE will have a source code editor
    - A source code editor is to a program what word processing software is to a document

Program Text Editors

- Word processors format the appearance of the text
- Text editors
  - Format the spacing between words for legibility
  - Ideal for structured languages
  - Text is the same font size
- Examples
  - DOS – Edit
  - Windows – Notepad, Wordpad
  - Unix / Linux – ed, vi, emacs
- IDEs
  - MS Visual C++, Symantec Visual Cafe

Notepad++ is an example of a basic editor


F:\Classes\COP1000\OilCalc.java - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
OilCalc.java

```
// OilCalc.java

import java.io.*;
import java.util.Scanner;

public class OilCalc {

    public static void main(String[] args) throws IOException {

        Scanner keyboard = new Scanner(System.in);
        int amount = 0;
        double cost = 0;

        System.out.print("Amount of oil is? ");
                amount = keyboard.nextInt();

                if (amount <= 100) {cost = amount * 2.25;}
                else if (amount <= 250) { cost = 100 * 2.25 + (amount-100) * 2.10;}
                    else { cost = 100 * 2.25 + 150 * 2.10 + (amount-250) * 1.99;}

        System.out.println("The cost of the oil is $" + cost);
    }
}
```

Let's a programmer enter, change and save source code

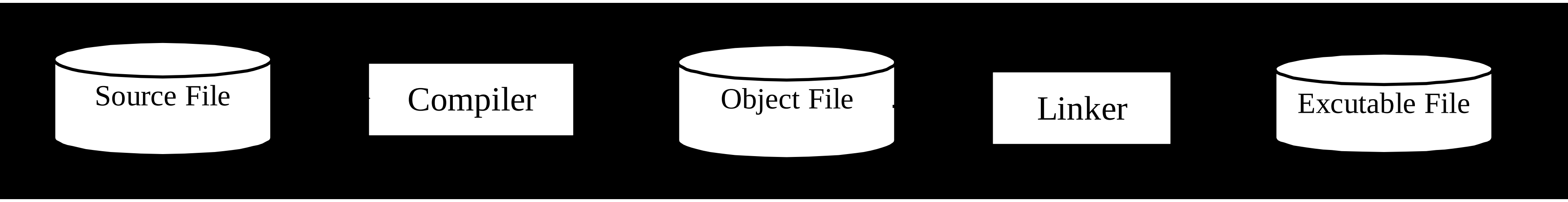
Compilers

Compilation

- Translates high-level language into low-level instructions
- High-level language: Source code
- Machine-level: Object code
- Changes, including bug fixes, require recompiling

Compiling the HL program

A program written in a high-level language is called a *source program (or source code)*. Since a computer cannot understand a source program. Program called a *compiler* is used to translate the source program into a machine language program called an *object program*. The object program is often then linked with other supporting library code before the object can be executed on the machine.



Language Components

- Lexicon
  - All legal words in the language
  - Meaning and type
- Syntax
  - grammar rules
- Semantics
  - meaning of command

Definition of Programming Languages

- The Syntax of a programming language specifies the structure of programs
- The Semantics of a programming language specifies the meaning of programs

Syntax

An if-statement consists of the word “if” followed by an expression inside parentheses, followed by a statement, followed by an optional else part consisting of the word “else” and another statement.

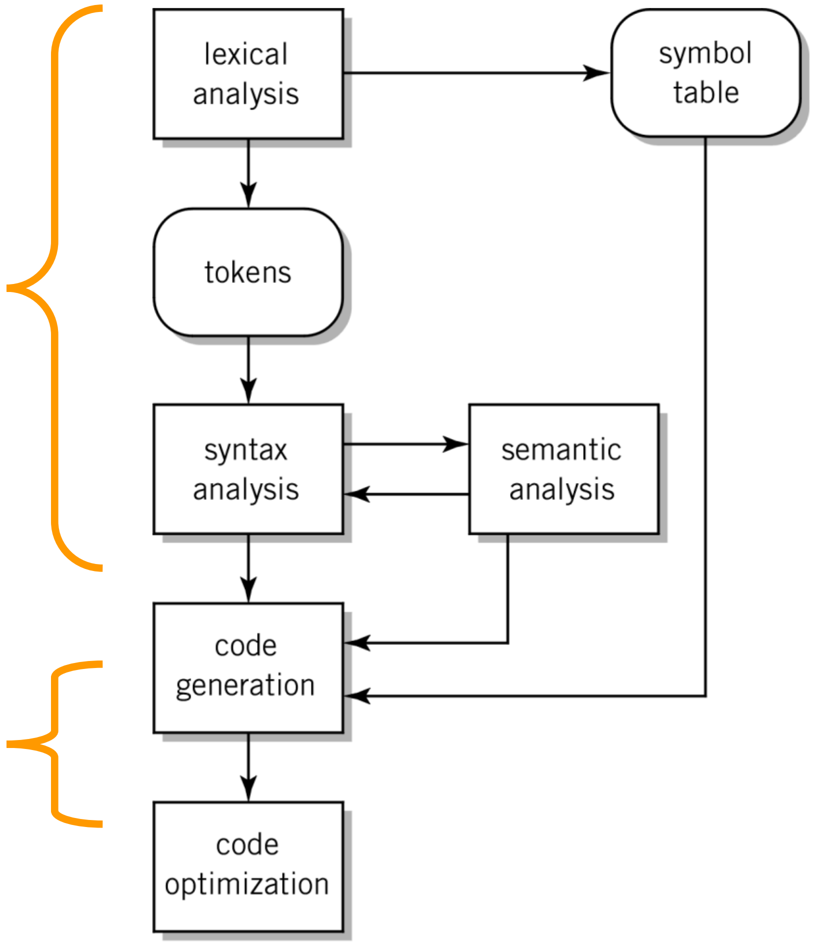
Semantics

An if-statement is executed by first evaluating its expression, which must have arithmetic or pointer type, including all side effects, and if it compares unequal to 0, the statement following the expression is executed. If there is an else part, and the expression is 0, the statement following the "else" is executed.

The Compilation Process

Checks for errors

Generates CPU instructions or library calls



Updates internal tables

Process of Parsing

- Lexical analysis
  - Also known as scanning
  - Divides the string of input characters into single elements, tokens, based on strict computer punctuation
- Syntactic analysis
  - Checks for errors in grammar rules
- Semantic parsing
  - Determines the meaning of the string

Interpreter

- An interpreter translates high-level instructions into an intermediate form, which it then executes. In contrast, a compiler translates high-level instructions directly into machine language.
- Compiled programs generally run faster than interpreted programs.
- The advantage of an interpreter, however, is that it does not need to go through the compilation stage during which machine instructions are generated. This process can be time-consuming if the program is long. The interpreter, on the other hand, can immediately execute high-level programs. For this reason, interpreters are sometimes used during the development of a program, when a programmer wants to add small sections at a time and test them quickly.

Interpreters

- Translates source code instructions into machine language and executes it one statement at a time
- Disadvantages
  - Longer to execute, particularly bad for loops
  - Uses more memory
- Advantage
  - Faster testing and code modification
- Examples of interpreted languages
  - Java, BASIC, LISP

Interpreter vs Compiler

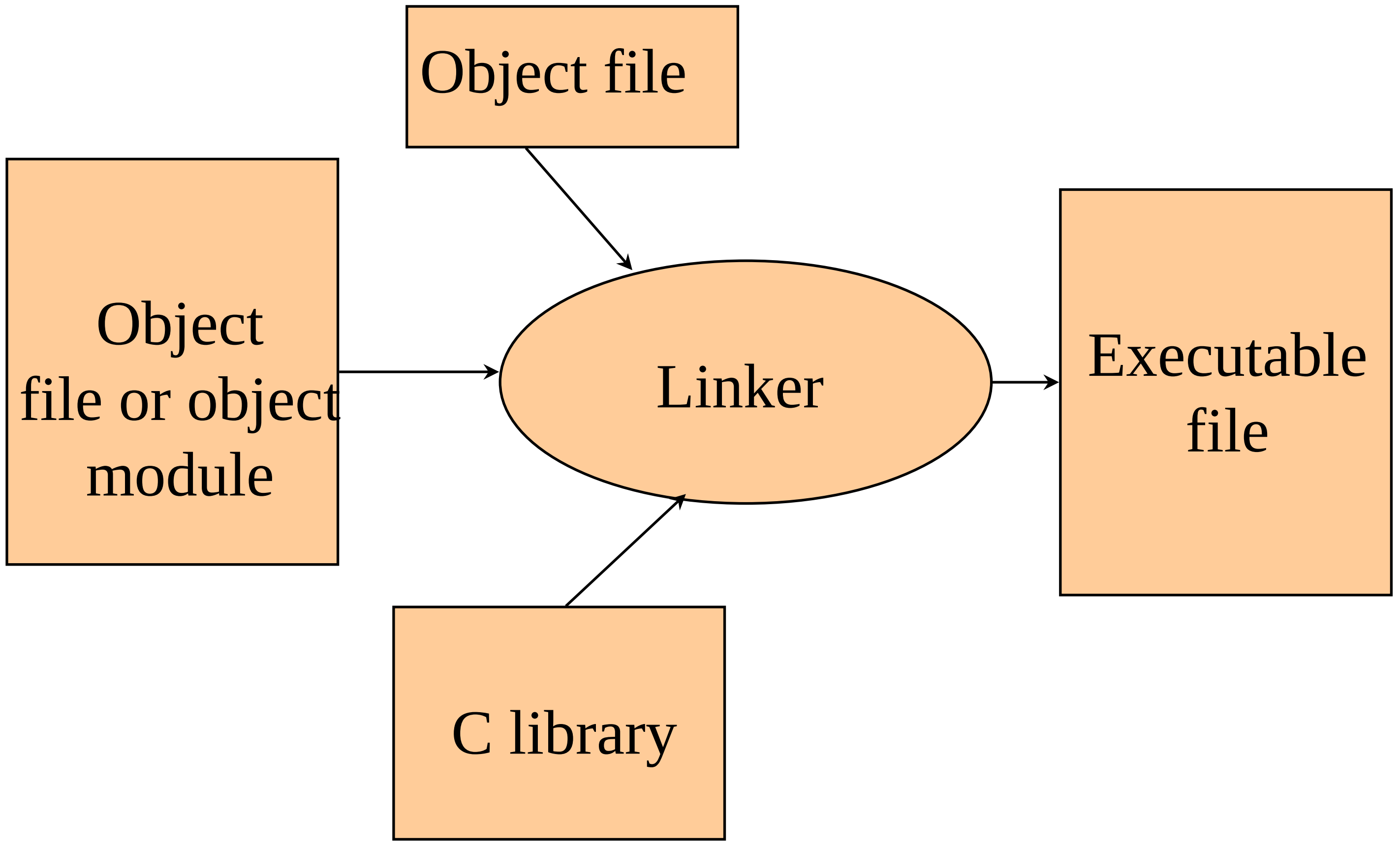
- Interpreter
  - Translates instructions to machine code line-by-line.
- Compiler
  - Translates the entire program to machine code before running it.

Interpreter vs. Compiler

Resources during execution	**Interpreter**	**Compiler**
Contents in memory		
Interpreter/compiler	Yes	No
Source code	Partial	No
Executable code	Yes	Yes
CPU cycles		
Translation operations	Yes	No
Library linking	Yes	No
Application program	Yes	Yes

Linkers

Linking


Object file
Object file or object module
Linker
Executable file
C library

Linkers

- Searches program libraries to find library routines used by the program
  - Library: collection of pre-written functions and subroutines made available to perform commonly required activities
- Determines the memory locations that code from each module will occupy and relocates instructions by adjusting absolute references
- Resolves references among files

Why Link?

- Construct single executable program from multiple object code files compiled at different times
- Program can be subdivided into components and parceled out to different developers
- Example
  - Main program and multiple subroutines written and compiled by different programmers at different times

Loaders

Loader

- Loads binary files that have been linked into main memory
- Program is ready for execution

Loading a Program

- Three ways a program can get loaded
  - Absolute loading – Load program at the same address (virtual and/or physical) every time
  - Relocatable loading – Load program at different addresses based on what is available
  - Dynamic run-time loading – Load and reload the program at different addresses while the program is running
- Address Binding
  - Where a symbolic label/name is translated (bound) to an actual address
  - The actual binding can be specified in the program, or resolved at compile time, link time, load time, or run time.

Debuggers

Debuggers

- Assembly language debuggers
- Source code debuggers
- Step through programs
- Check variable values