Lecture 9

C
Java
C++
PHP
PEARL
Script
Python
ALGOL
Assembly
Language
Pascal
Lisp
VB
Object
JavaScript
shell
Visual
Go
Delphi
DASL
Caml
Prolog
Programming
Ruby
EXEC
REXX
NET
Common
Oxygene
ABC
Ajax
Not
language
IMP
PL/I
DATATRIEVE
Microsoft
Scala
Clojure
COBOL
PL/SQL
Eiffel
Haxe
Turing
Simula
Boomerang
GraphTalk
ChucK
ECMAScript
GameMonkey
Newspeak
JASS
Forth
JavaFX
ObjectScript
EusLisp
Constraint
Cyclone
notation
BLISS
RPL
Arc
Moby
Uniface
ORCA/Modula-2
Coral
WATFOR
Agena
Little
Action
CorVision
RPG
Robot
SP/k
Strand
rex
ISWIM
S/SL
Rust
GDL
JOSS
Bistro
Blue
SenseTalk
Sawzall
Chapel
Rapira
CSKA
SIMPLE
Autolt
code
Lynx
Logo
Microcode
SAM76
Redcode
MicroScript
Squirrel
Kaleidoscope
Windows/Dos
Cg
C/AL
Cesil
ABAP
SYMPL
HAL/S
SIMSCRIPT
Revolution
Genie
Inform
Agora
Trac
Adenine
TACPOL
FP
ELAN
SALSA
ATS
Fjölnir
Fuxi
Fancy
SuperTalk
LC-3
XOTcl
dc
Cayenne
bc
GM

Programming Languages & Program Development

- **Programming** is the creation of software applications.
- **Programmers** are the people who create the software applications.
- A **programming language** is used by programmers to create software that the computer understands.

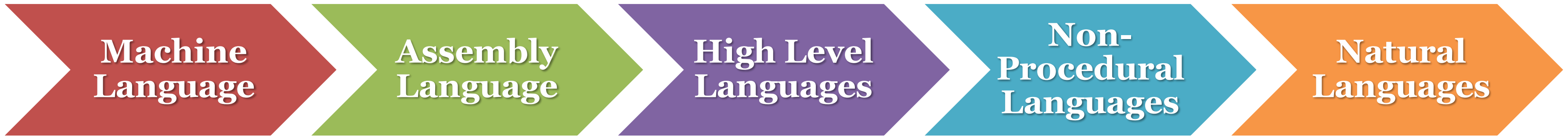
Programming Languages & Program Development

- **Syntax** is the vocabulary and rules of a programming language.

- **Source code**, the programming instructions in their original form, which need to be translated into a form that the computer can understand.

Development of Programming Languages

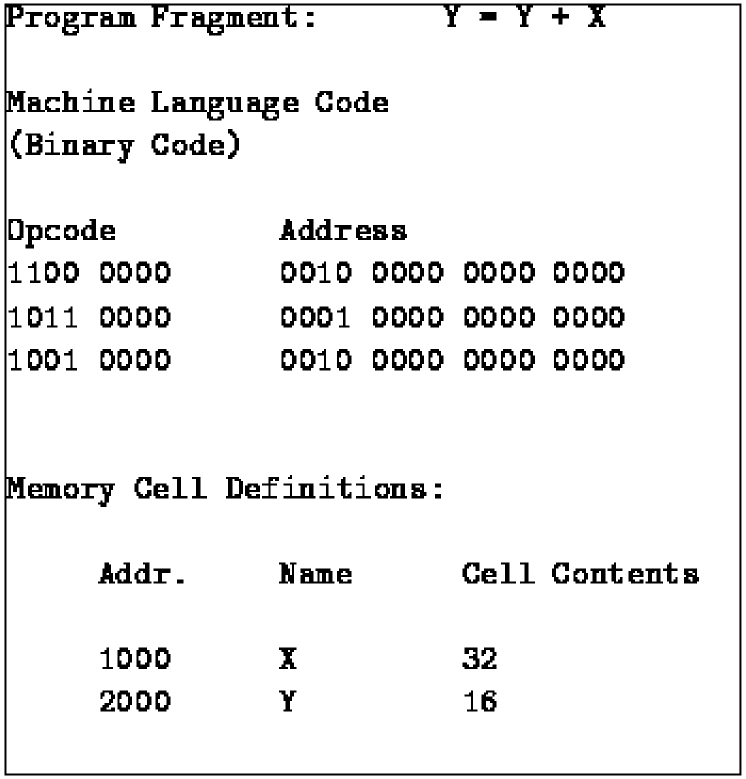
Five distinct programming language generations



1st Generation: Machine Language

- Based on binary numbers
- The only programming language that a computer (CPU) understands directly
- Machine dependent:

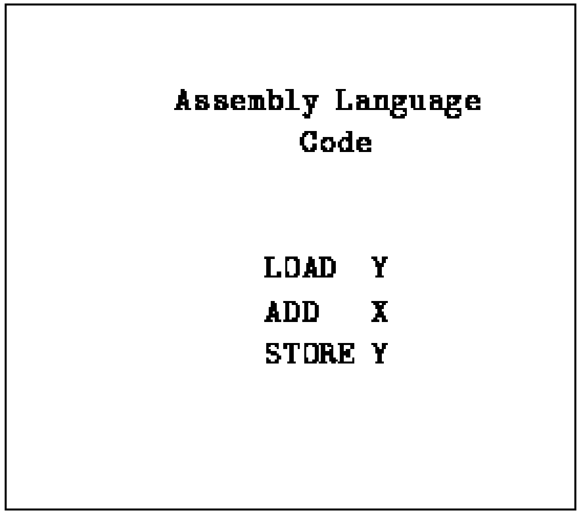
  Each family of processors has its own machine language



```
Program Fragment:        Y = Y + X

Machine Language Code
(Binary Code)

Opcode          Address
1100 0000       0010 0000 0000 0000
1011 0000       0001 0000 0000 0000
1001 0000       0010 0000 0000 0000


Memory Cell Definitions:

        Addr.       Name        Cell Contents

        1000        X           32
        2000        Y           16
```

2nd Generation: Assembly Language

- **Assembly language**
  - Machine dependent
  - Programs use:
    - **Mnemonics**, brief abbreviations for program instructions which makes assembly language easier to use than the machine language
    - Base-10 (decimal) numbers
  - Must be translated into machine language



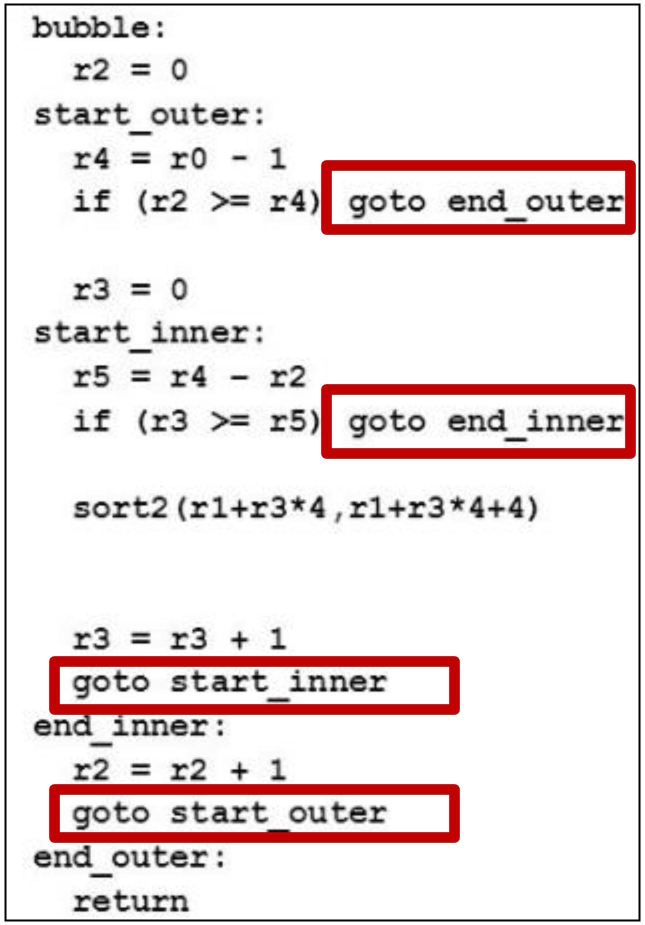
Assembly Language Code

```
LOAD  Y
ADD   X
STORE Y
```

2nd Generation: Assembly Language *cont.*

- The danger of writing **spaghetti code** is especially great caused by **GOTO** statements

- GOTO statements make code difficult to follow and prone to errors



```
bubble:
   r2 = 0
start_outer:
   r4 = r0 - 1
   if (r2 >= r4) goto end_outer

   r3 = 0
start_inner:
   r5 = r4 - r2
   if (r3 >= r5) goto end_inner

   sort2(r1+r3*4,r1+r3*4+4)


   r3 = r3 + 1
   goto start_inner
end_inner:
   r2 = r2 + 1
   goto start_outer
end_outer:
   return
```

3rd Generation: High Level Languages

- Do not require programmers to know details relating to the processing of data
- **Machine independent**
- **Easier** to read, write, and maintain than assembly and machine languages
- Source code must be **translated** by a language translator

3rd Generation: High Level Languages

Fortran Program

```
 A code excerpt implementing an "open loop" for processing several
 consecutive "input blocks" like the one showed above:


100   continue
      read(unit=10, fmt='(/)', err=998, end=999)
      read(unit=10, fmt=*,     err=998, end=999) nstep

      read(unit=10, fmt='(/)', err=998, end=999)
      read(unit=10, fmt=*,     err=998, end=999) x1, x2

      read(unit=10, fmt='(/)', err=998, end=999)
      read(unit=10, fmt=*, err=998, end=999) k

      read(unit=10, fmt='(/)', err=998, end=999)
      read(unit=10, fmt=*,     err=998, end=999) eps

      call compute(nstep, x1, x2, k, eps)
      goto 100

998   stop ' error reading input file '
999   stop ' end of input file '
```

Java Program

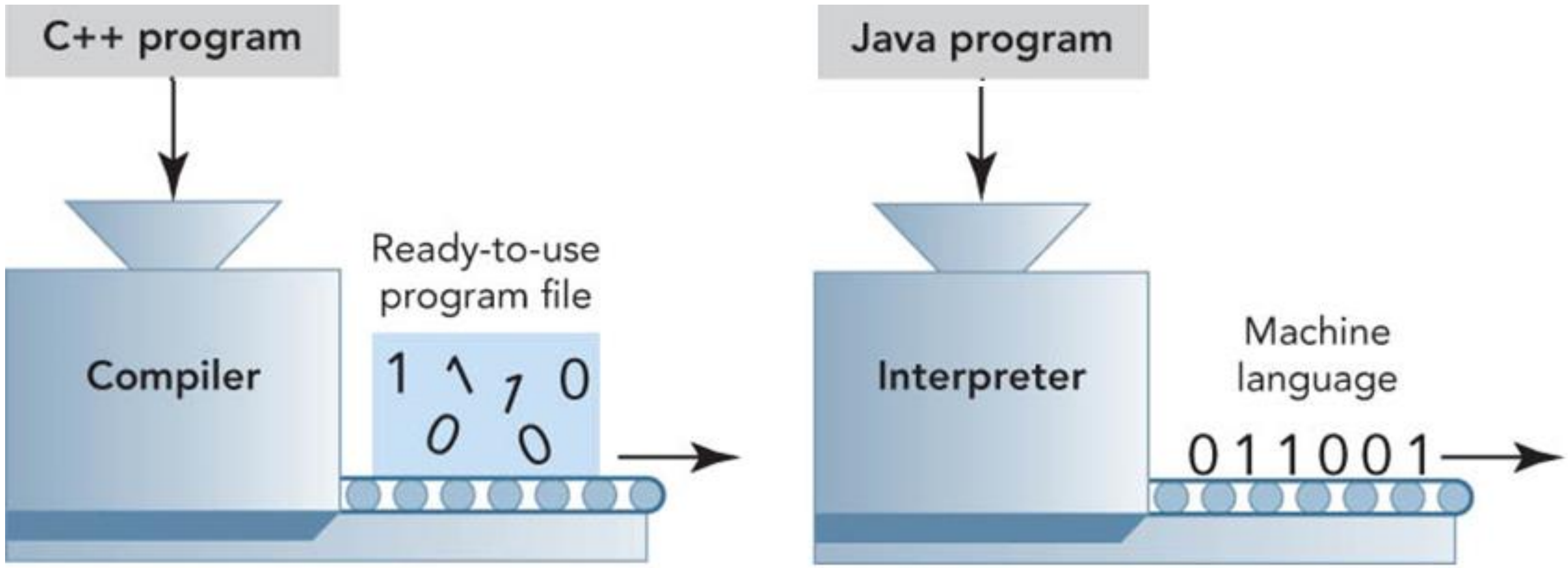
```
public void ejbCreate(String person, String id)
   throws CreateException {

   if (person == null) {
      throw new CreateException("Null person not allowed.");
   }
   else {
      customerName = person;
   }

   IdVerifier idChecker = new IdVerifier();
   if (idChecker.validate(id)) {
      customerId = id;
   }
   else {
      throw new CreateException("Invalid id: "+ id);
   }

   contents = new Vector();
}
```

3rd Generation: High Level Languages

- Translators
  - **Compilers** translate source code into **object code** (object program)
  - **Interpreters** translate source code one line at a time and execute instructions without creating an object code

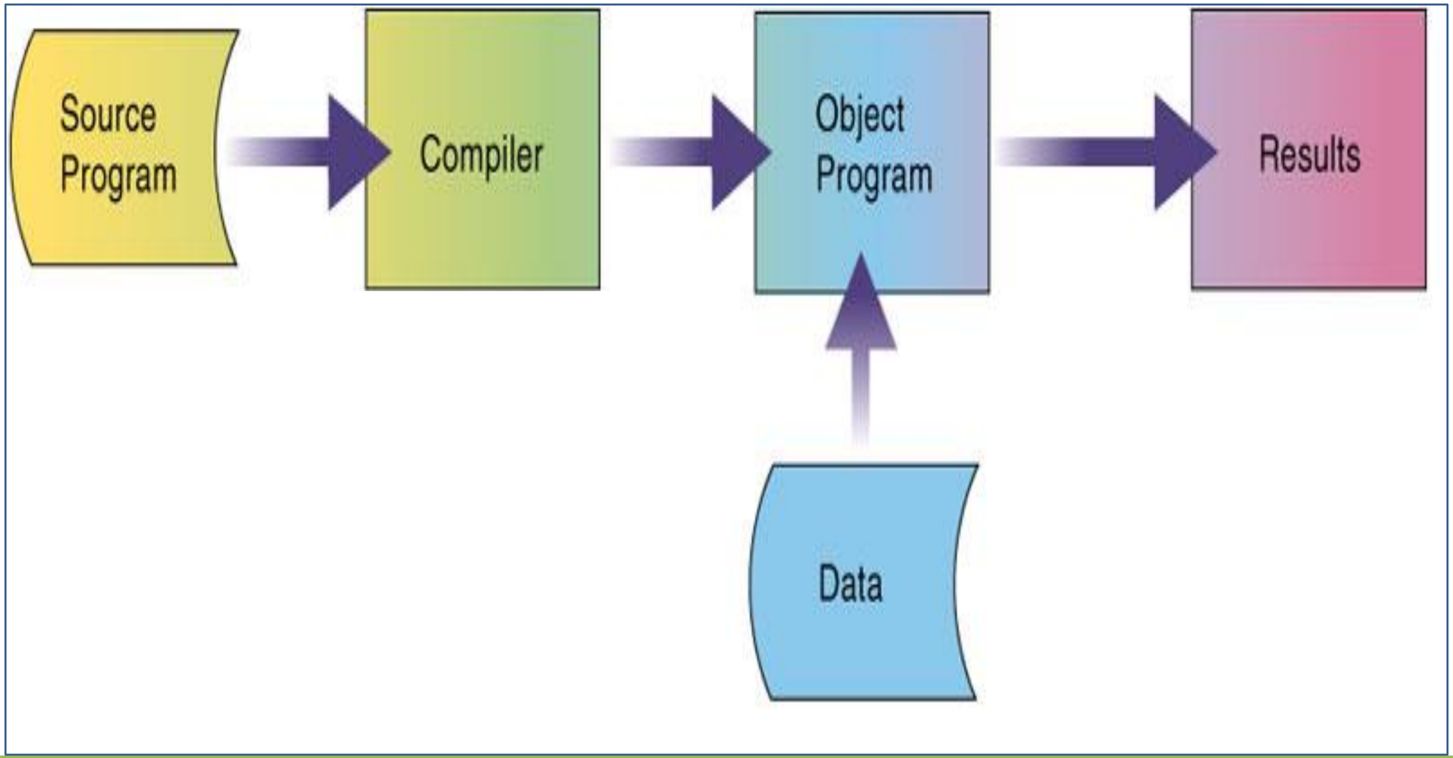
3rd Generation: High Level Languages

- Compilers and Interpreters



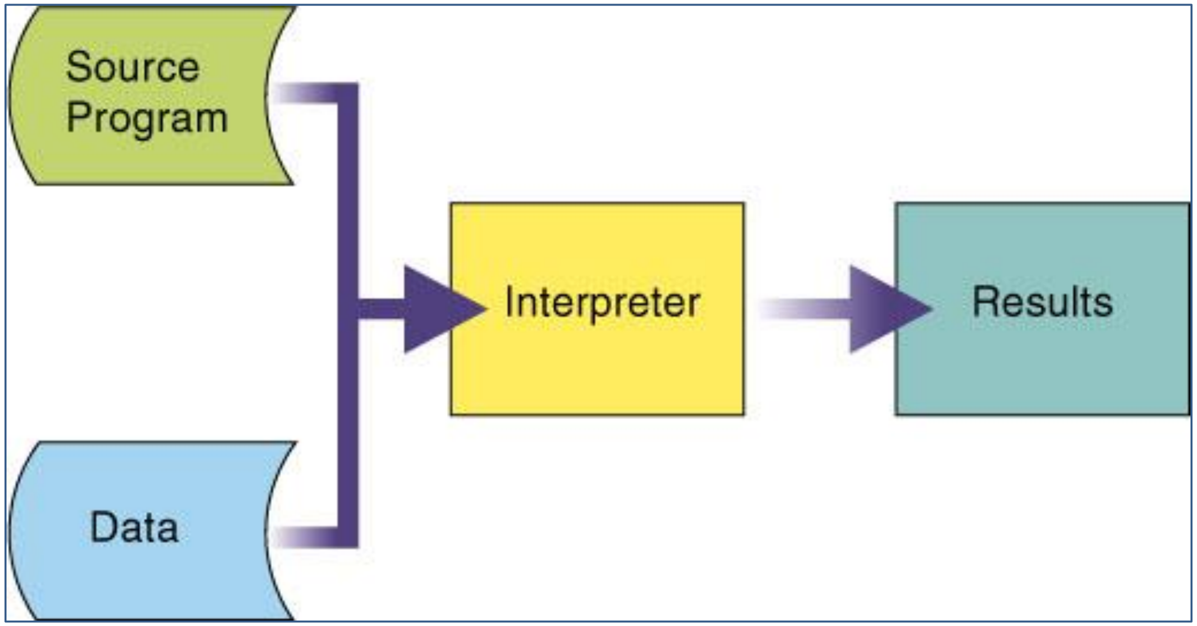
Compilation

- Compilation: Converting source code into object program
- After compilation, programmers have an executable program (object program)



Interpreter

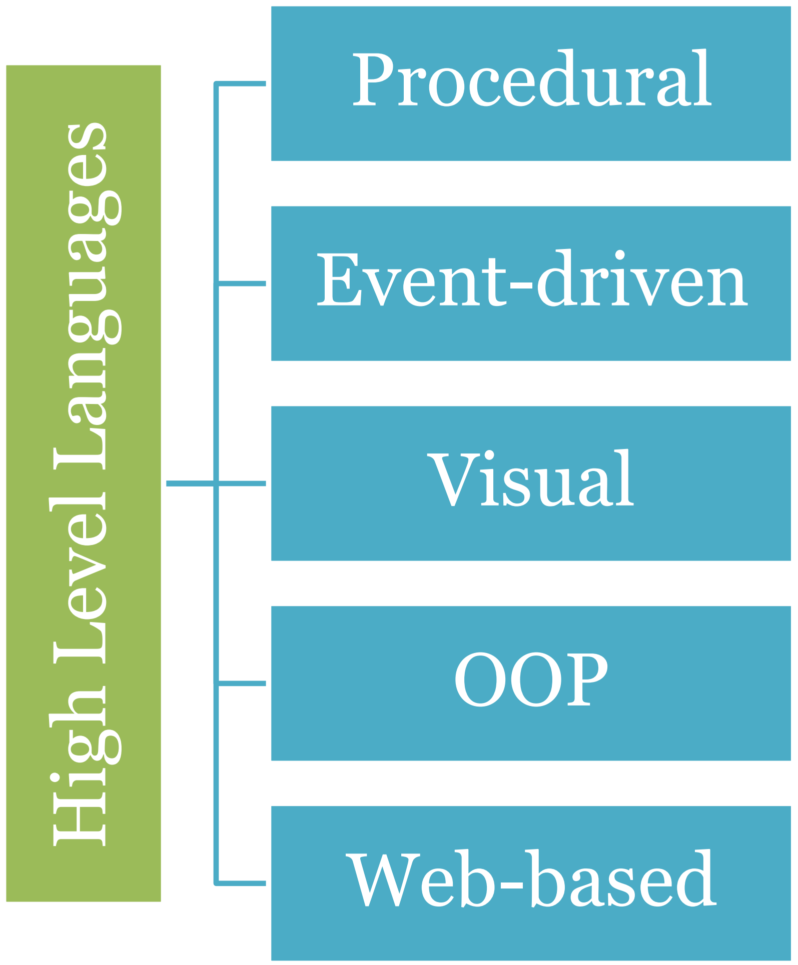
- Interpreter translates source code line by line
- Each line is executed before the next line is processed
- Programmers do not have to wait for the entire program to be reprocessed each time they make a change



Compilation vs. Interpretation

- The compilation process takes longer than the interpretation process because in compilation, *all* of the lines of source code are translated into machine language before any lines are executed.
- The finished compiled program runs faster than an interpreted program because the interpreter is constantly translating and executing as it goes.
- The interpreter immediately displays feedback when it finds a syntax error. Thus, the programmer can correct any errors or debug the code before the interpreter evaluates the next line. They immediately see the results of changes as they are making them in the code.

Types of High Level Languages (Not mutually exclusive)



Types of High Level Languages

- Procedural Languages

A sequence of instructions to run, which uses program control structures (IF, Loops, Functions…)

Examples: PASCAL, BASIC, COBOL, FORTRAN

- Event-driven Languages

Waits for events (mouse click, button press…) to process a defined set of instructions (event handler)

Examples: C++, Javascript

Types of High Level Languages

- Visual Languages

Allow programmer to manipulate items visually on a form setting their layout and properties.

Creating Graphical User Interface (GUI) applications.

Examples: Visual Basic, Visual C++, Delphi

- Web-based Languages

Uses special coding instructions to indicate style and layout of text and other elements of web sites.

Mark-up and scripting languages.

Examples: HTML, XML, ASP

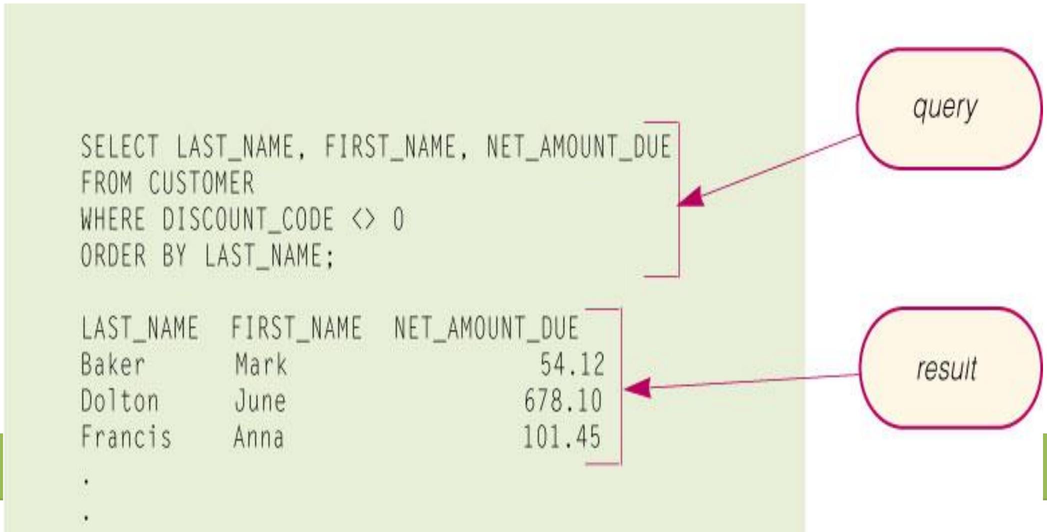
Types of High Level Languages

- OOP (Object Oriented Languages)

  Coding is attached to basic prebuilt items called **objects**, which include:
  - **Data**
  - **Attributes** that define the data
  - Procedures or operations called **methods**
  - An **interface** to exchange messages with other objects
- Relies on **reusability**
- Makes information hiding **(encapsulation)** a reality
- Examples: C#, Java, Python

4th Generation: Non-Procedural Languages

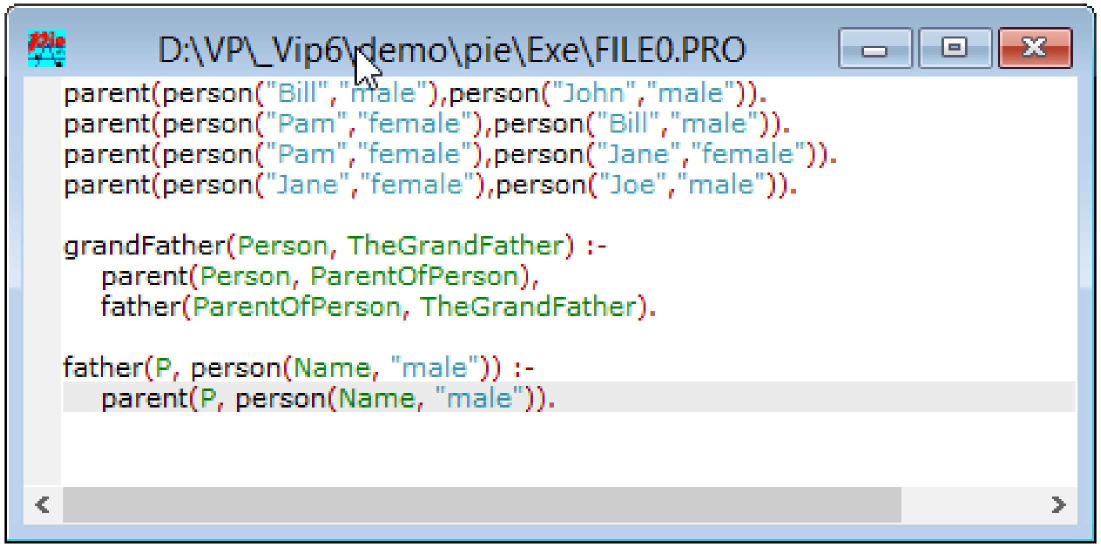
- **Non-procedural languages**
  - Do not require step-by-step procedures to achieve the appropriate programming outcome
  - Allows complex operations to be processed in one statement
  - Examples
    - Database report generators
    - Query languages: Structured Query Language (SQL)



```
SELECT LAST_NAME, FIRST_NAME, NET_AMOUNT_DUE
FROM CUSTOMER
WHERE DISCOUNT_CODE <> 0
ORDER BY LAST_NAME;

LAST_NAME   FIRST_NAME   NET_AMOUNT_DUE
Baker       Mark                  54.12
Dolton      June                 678.10
Francis     Anna                 101.45
.
.
```

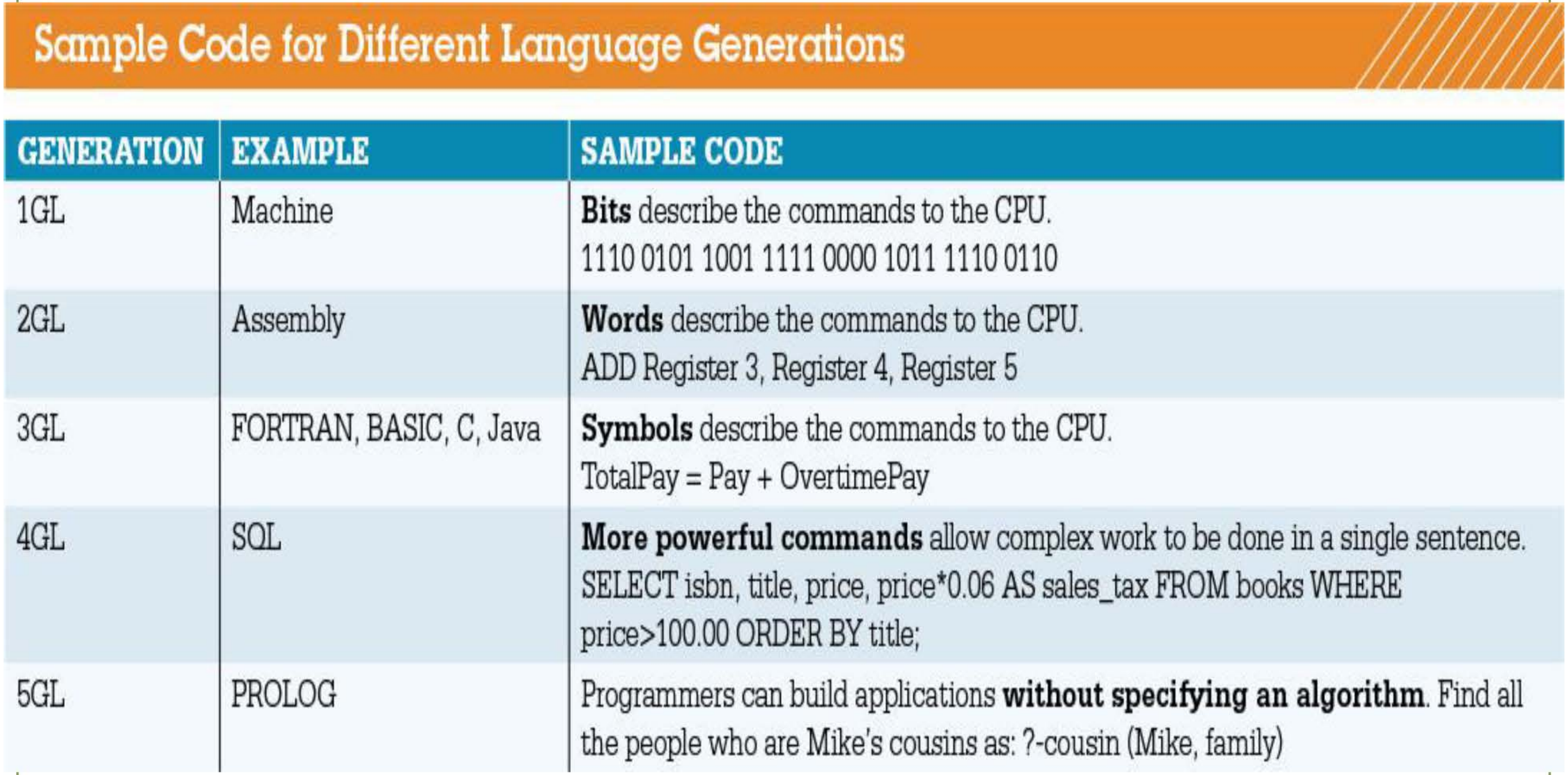
5th Generation: Natural Languages

- **Natural language**
  - Non-procedural
  - Uses everyday language to program
  - Example: Prolog



```
parent(person("Bill","male"),person("John","male")).
parent(person("Pam","female"),person("Bill","male")).
parent(person("Pam","female"),person("Jane","female")).
parent(person("Jane","female"),person("Joe","male")).

grandFather(Person, TheGrandFather) :-
    parent(Person, ParentOfPerson),
    father(ParentOfPerson, TheGrandFather).

father(P, person(Name, "male")) :-
    parent(P, person(Name, "male")).
```

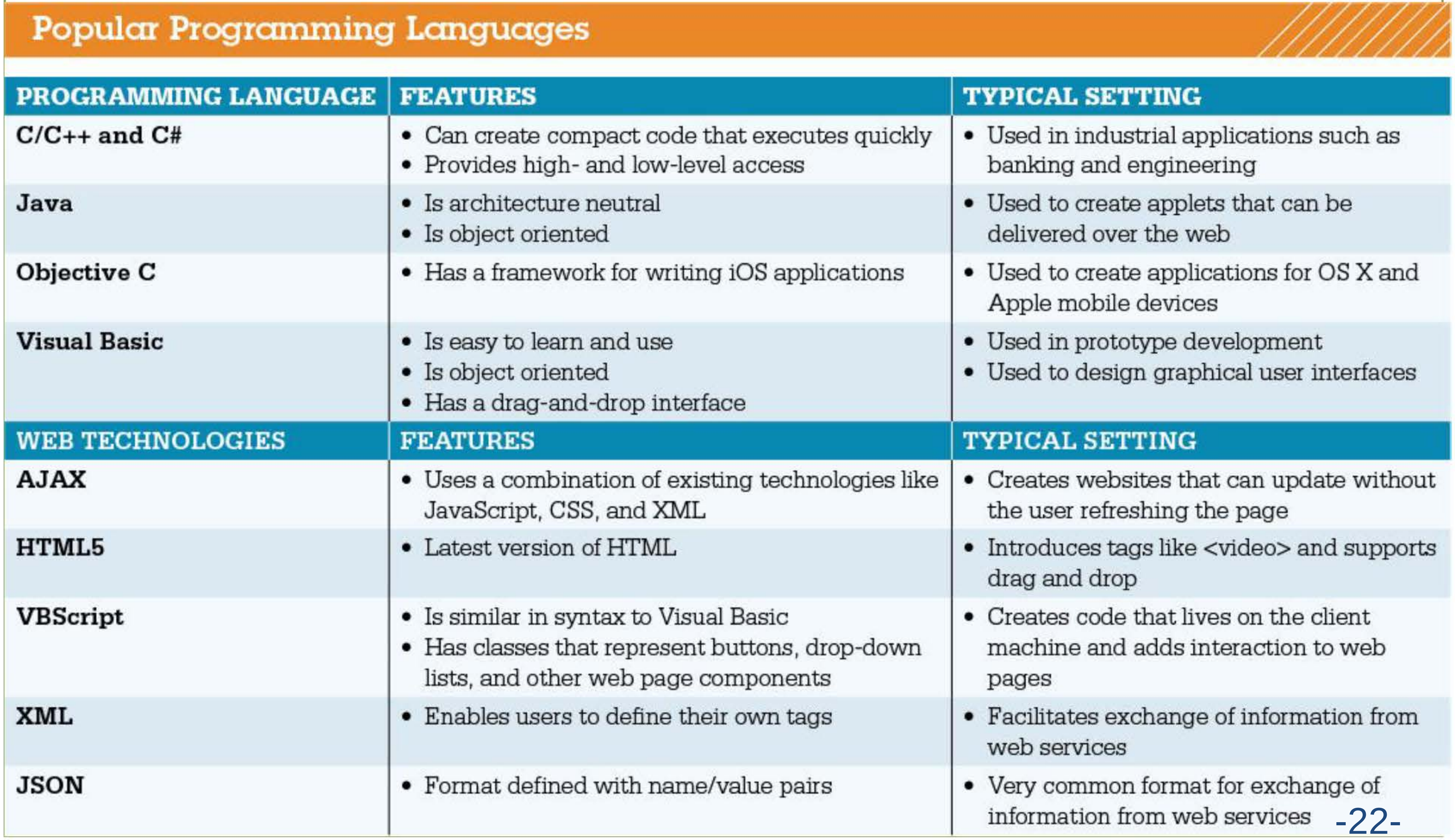
Sample Code for Language Generations



Sample Code for Different Language Generations

GENERATION	EXAMPLE	SAMPLE CODE
1GL	Machine	**Bits** describe the commands to the CPU. 1110 0101 1001 1111 0000 1011 1110 0110
2GL	Assembly	**Words** describe the commands to the CPU. ADD Register 3, Register 4, Register 5
3GL	FORTRAN, BASIC, C, Java	**Symbols** describe the commands to the CPU. TotalPay = Pay + OvertimePay
4GL	SQL	**More powerful commands** allow complex work to be done in a single sentence. SELECT isbn, title, price, price*0.06 AS sales_tax FROM books WHERE price>100.00 ORDER BY title;
5GL	PROLOG	Programmers can build applications **without specifying an algorithm**. Find all the people who are Mike's cousins as: ?-cousin (Mike, family)

One Size Doesn't Fit All



Popular Programming Languages

PROGRAMMING LANGUAGE	FEATURES	TYPICAL SETTING
C/C++ and C#	• Can create compact code that executes quickly • Provides high- and low-level access	• Used in industrial applications such as banking and engineering
Java	• Is architecture neutral • Is object oriented	• Used to create applets that can be delivered over the web
Objective C	• Has a framework for writing iOS applications	• Used to create applications for OS X and Apple mobile devices
Visual Basic	• Is easy to learn and use • Is object oriented • Has a drag-and-drop interface	• Used in prototype development • Used to design graphical user interfaces
WEB TECHNOLOGIES	**FEATURES**	**TYPICAL SETTING**
AJAX	• Uses a combination of existing technologies like JavaScript, CSS, and XML	• Creates websites that can update without the user refreshing the page
HTML5	• Latest version of HTML	• Introduces tags like <video> and supports drag and drop
VBScript	• Is similar in syntax to Visual Basic • Has classes that represent buttons, drop-down lists, and other web page components	• Creates code that lives on the client machine and adds interaction to web pages
XML	• Enables users to define their own tags	• Facilitates exchange of information from web services
JSON	• Format defined with name/value pairs	• Very common format for exchange of information from web services

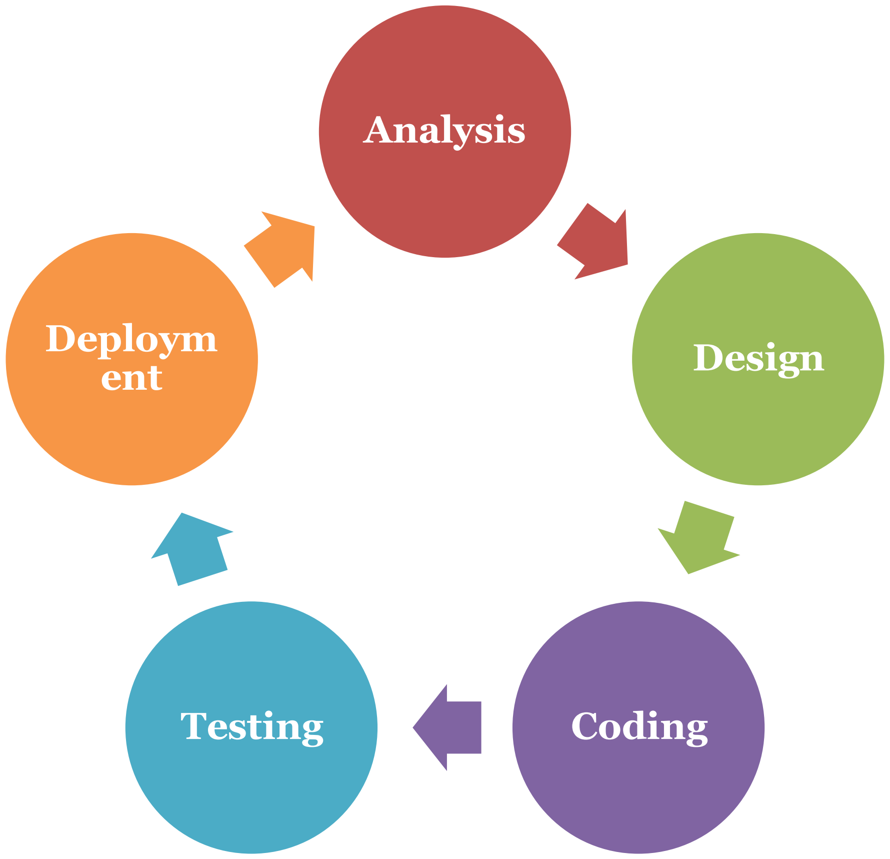
Top Programming Languages of 2016

GreyCampus
JAVA
PHP
RUBY/RAILS
JAVA SCRIPT
C++
PYTHON
IOS
SQL
C#
TOP 9 PROGRAMMING LANGUAGES OF 2016

Program Development Life Cycle (PDLC)

An organized plan for breaking down the task of program development into manageable parts

PDLC Phases



1. Analysis

- Defining the problem
- Interviews, questionnaires and observation
- System analysts collect the user requirements and specify the program specifications **specs**

- Specs define **IPO**
  - Input data
  - Processing
  - Output data

- Specs also include the appearance of user interface

2. Design

- **Program design** identifies components of the program
  - **Top-down program design** breaks program into small, manageable, highly focused subroutines
  - **Structured design** uses **control structures**
  - **Algorithm**
- They are not mutually exclusive

Desk Checking

- An important **design tool**
- The main purpose of desk checking the algorithm is to **identify** major logic errors early, so that they may be easily corrected
- **Logic errors** are bugs that cause program to run incorrectly and produce undesired outputs
- **Test data** needs to be walked through each step in the algorithm, to check that the instructions described in the algorithm will actually do what they are supposed to

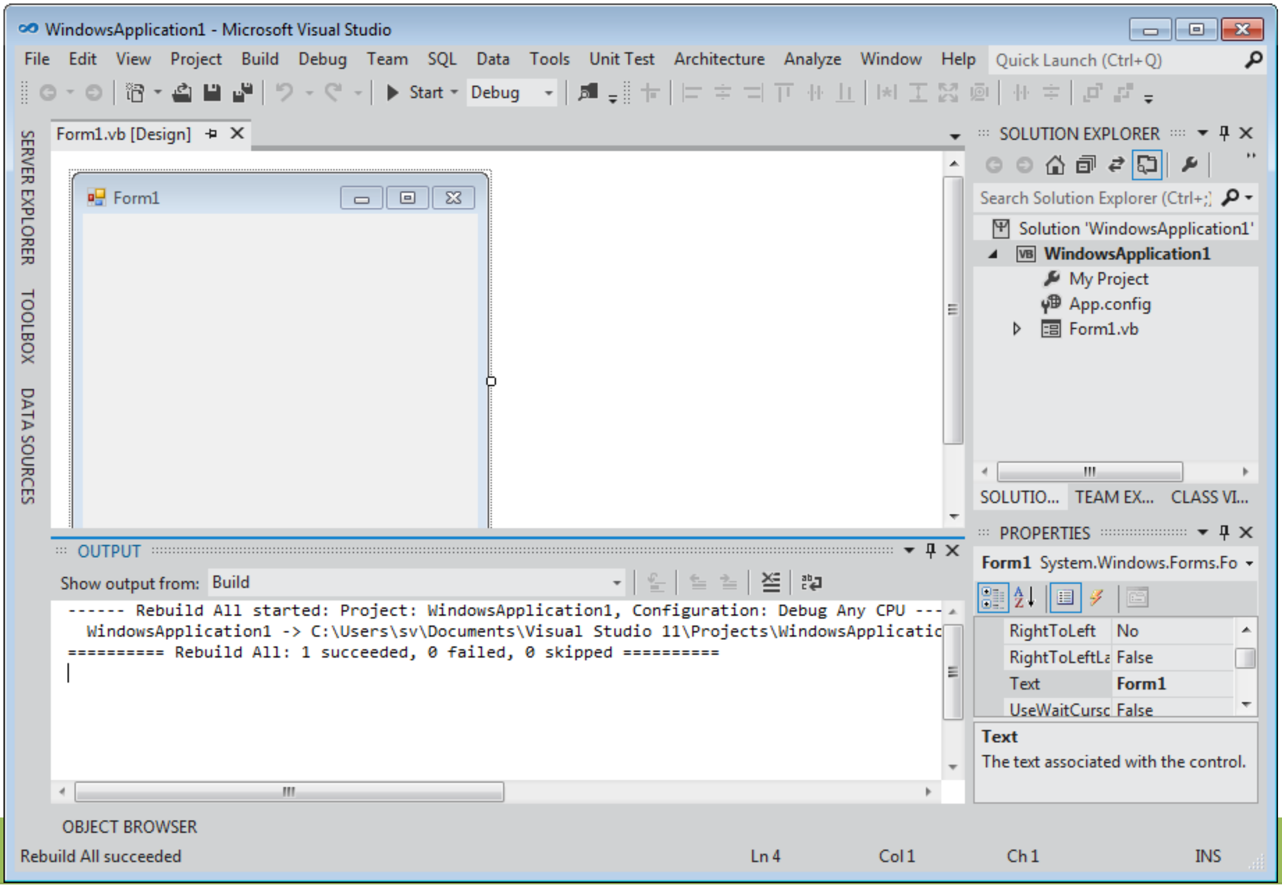
3. Coding

- Process of creating SW applications
- Programmers use programing languages to convert algorithms into source code

- **Syntax errors**
  - Mistakes in the construction of the programming commands
  - Must be corrected for the program to run

3. Coding *cont.*

- Integrated Development Environment (IDE) helps programmers write and test their programs.



RightToLeft	No
RightToLeftLa	False
Text	Form1
UseWaitCursc	False

Text
The text associated with the control.

```
------ Rebuild All started: Project: WindowsApplication1, Configuration: Debug Any CPU ---
  WindowsApplication1 -> C:\Users\sv\Documents\Visual Studio 11\Projects\WindowsApplicatic
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

3. Coding *cont.*

Version Control SW

- SW tools that help team members manage changes to the source code over time
- Keeps track of every modification by each contributor
- Developers can turn back to earlier versions to help fix any mistakes
- Example tools:

Git, CVS, SVN, Mercurial and Bazaar….

4. Testing

- Assessing the functionality of SW program
- Two main categories: Dynamic and static

- Static Testing

Manual or automated review for

  - Code (static code reviews)
  - Requirements and design documents (technical reviews)

- Dynamic Testing

Check the functional behavior of a SW unit by entering test data and comparing results to the expected results

Dynamic Testing Opacity

Opacity (view of code)

- Black-box testing
  - Tester has no knowledge of code
  - Often done by someone other than the coder

- White-box testing
  - Testing all possible logic paths in the software unit
  - Deep knowledge of the logic
  - Makes each program statement execute at least once

5. Deployment and Maintenance

- Deployment

All processes involved in getting new SW up and running properly in its environment, including installation, configuration running, testing and making necessary changes

- Maintenance

Evaluate the program on a regular basis

Documenting the Program

- Throughout the PDLC
- Documentation includes
  - Program design work
  - Overview of program functionality
  - Thorough explanation of main features
  - Tutorials
  - Reference documentation of program commands
  - Description of error messages

Complete

- The ----- phase of PDLC includes identifying syntax errors.
- Testing all possible logic paths in the software unit, with thorough knowledge of the logic is called -----.
- ----- allows complex operations to be processed in one statement, e.g. report generators and query languages.
- ----- testing checks functional behavior of SW by entering test data and comparing results to expected results.
- ----- are brief abbreviations for program instructions that make assembly language easier to use.
- Interviews, questionnaires and observation are employed in the ----- phase of PDLC.

Complete

- ----- is the vocabulary and rules of a programming language.
- ----- is the only programming language that a computer (CPU) understands directly.
- ----- converts source code into object program.
- ----- are mistakes in construction of programming commands which must be corrected for program runs.
- ----- are the processes involved in getting new SW up and running properly in its environment.
- ----- and ----- are machine dependent languages.
- ----- code is full of GOTO statements, which make code difficult to follow and prone to errors.

Complete

- The compilation process takes ----- time than the interpretation process.
- ----- testing is conducted when tester has no knowledge of code.
- Assessing the functionality of SW program is called -----.
- The main purpose of ----- the algorithm is to identify major logic errors early to be easily corrected.
- Evaluating the program on a regular basis is called -----.
- ----- translates source code one line at a time and execute instructions without creating machine code.
- Phases of PDLC are -----, -----, -----, ----- and -----.

Differentiate Between

- Source and machine code
- White and black box testing
- Logic and syntax errors
- Compiler and interpreter
- Dynamic and static testing

Thank You