Software Development
Programming & Languages

Programming: A Five-Step Procedure
- Define the problem
- Design a solution
- Code the program
- Test the program
- Document the program

Five Generations of Programming Languages
- Machine Languages, 1945--
- Assembly Languages, 1950s--
- High-Level Languages, 1960s--
- Very-High-Level Languages (4GL), 1970s--
- Natural Languages, 1980s--

Applications of some important programming languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Origin</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
<td>FORmula TRANslator(1954)</td>
<td>Scientific</td>
</tr>
<tr>
<td>COBOL</td>
<td>Common Business-Oriented Language(1959)</td>
<td>Business</td>
</tr>
<tr>
<td>BASIC</td>
<td>Beginner's All-purpose Symbolic Instruction Code (1965)</td>
<td>Education, Business</td>
</tr>
<tr>
<td>Pascal</td>
<td>Named after French inventor Blaise Pascal(1971)</td>
<td>Education, systems programming</td>
</tr>
<tr>
<td>C</td>
<td>Invested at Bell Labs(1972)</td>
<td>Systems programming, general use</td>
</tr>
</tbody>
</table>

A natural language

Hello!
How may I help you?
Who are my customers in Chicago?
Just a sec. I'll see.
The customers in that city are:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ballard</td>
<td>Ballard and Sons, Inc.</td>
</tr>
<tr>
<td>Fremont</td>
<td>Henry Fremont Associates</td>
</tr>
<tr>
<td>Greenlake</td>
<td>Greenlake Consortium</td>
</tr>
<tr>
<td>Wallingford</td>
<td>Wallingford, Inc.</td>
</tr>
</tbody>
</table>

What can I do for you now?

What is Fremont's balance?
Hang on. I'll see.
Accounts Receivable  563.47
Unapplied Credit    79.16
Balance              484.31

What else can I do for you?
Give me Fremont's phone number!
Please wait while I check the files?
(312)789-5562

What can I do for you now?

Language Translators
- An **assembler** is a program that translates the assembly-language program into machine language.
- A **compiler** is a language translator that converts the entire program of a high-level language into machine language before computer executes the program.
- An **interpreter** is a language translator that converts each high-level language statement into machine language and executes it immediately, statement by statement.
Preparing a program for execution

Object-Oriented Programming
Object-oriented programming (OOP) is a programming method that combines data and instructions for processing that data (called methods) into a self-sufficient "object" that can be used in other programs. It involves three important concepts:

- **Encapsulation**: Encapsulation means an object contains (encapsulates) both data and the instructions for processing it.
- **Inheritance**: Inheritance allows traits of a class of objects to be inherited by its subclass.
- **Polymorphism**: Polymorphism allows a message to produce different results based on the object that it sent to. (A message is a call of some method).

Visual Programming

- Visual programming is a method of creating programs in which the programmer makes connections between objects by drawing, pointing, and clicking on diagrams and icons.

Internet Programming

- **HTML (HyperText Markup Language)** is a type of programming language that embeds simple commands within standard ASCII text documents to provide an integrated, two-dimensional display of text, graphics, and sounds.
- **VRML (Virtual Reality Markup Language)** is a type of programming language used to create three-dimensional Web-pages.
- **Java** (developed by Sun Microsystems) is a programming language that allows users to create applications (called Applets) that can be downloaded and executed on the remote host via the internet connection.

Flowchart and pseudocode for averaging numbers

```
1. You can design a program in many ways. But usually, the first step is to figure out the requirement of the program--determine exactly what you want the program to do. In this example, the requirement is to add a sequence of numbers from 1 to a number specified by the user.

2. One of the traditional ways to design a program is to use a flow chart. The top of the flow chart indicates input from the user. Rectangles contain statements, and diamonds indicate decisions.

3. Pseudocode refers to expressing program instructions in English as a way of describing the operation of the program without worrying about following the syntax of a particular computer language.

4. There are many techniques for designing complex programs. In fact, there are programs that can help you design software and keep track of the data flow through the components of your application. These software-engineering tools are most frequently used for large projects that need many programmers.
```
Machine Language

Assembly Language

Segment PARA PUBLIC "CSEG"
ADDSEG CSSEG, DSSEG
ENDPT

PUBLIC SEGMENT PARA "CSEG"
FORMAT (1X, A30, 2X, A1, 4X, F4.2)

character*30 course(5)
end

format ( 'Average' , 30x , f4.2)
write(*,*)
write(*,*)  'Elementary French' , 'Pre-Calculus' , 'Volleyball 1' /
write(*,*)  'Student Grade Report: Jennifer Jonson'
write(*,*)  'Class Grade Number'
write(*,*)
c do the rest of this 5 times
total = 0
if ( grade(i) .EQ. 'A' ) gradenum=4
do 10,  i =1, 5

data grade / 'A' , 'B', 'B', 'C', 'A' /
write (* , 99)  average
average = total / 5
10  continue
write (* , 99 )   course(i) , grade(i) , gradenum
if ( grade(i) .EQ. 'D' ) gradenum=1
if ( grade(i) .EQ. 'C' ) gradenum=2
if ( grade(i) .EQ. 'B' ) gradenum=3
if ( grade(i) .EQ. 'A' ) gradenum=4
if ( grade(i) .EQ. 'E' ) gradenum=0

#include <stdio.h>
main() {
    char * course[5]="Freshman English ", "Roman Empire ", "Volleyball 1 ", "Pre-Calculus ", "Elective ";
    float num , total=0;
    for   (a=0 ; a<5 ; a++)     {
        printf(" 
" ) ;
        printf(" Class	        Grade Number
 " ) ;
        printf(" 			---------
 " ) ;
        printf (" %s 	%c   %g 
 "  , course[a] , grade[a] , num) ;
        total=total+num;
    }
    printf("Average		          %g
 "    ,   total/5) ;
    printf(" 			--------
 " ) ;
}
case 'E' : num=0 ; break :
case 'D' : num=1 ; break ;
case 'C' : num=2 ; break ;
case 'B' : num=3 ; break ;
case 'A' : num=4 : break :
case 'F' : num=-1 ; break ;
for (i = 0; i < 5; i++) {
    if (course[i] == "freshman English")
        course[i] = "Freshman English ";
    if (course[i] == "Roman Empire")
        course[i] = "Roman Empire ";
    if (course[i] == "store number")
        course[i] = "store number ";
}.
printf("Grade Report: Jennifer Jonson
") ;
printf("Total number of courses:
") ;
printf("Average:
") ;
printf("Grade:
") ;
printf("Course:
") ;
}.
#include "curses.h"
#include <stdio.h>
main() {
    char * course[5]="Freshman English ", "Roman Empire ", "Volleyball 1 ", "Pre-Calculus ", "Elective ";
    float num , total=0;
    for   (a=0 ; a<5 ; a++)     {
        printf(" 
" ) ;
        printf(" Class	        Grade Number
 " ) ;
        printf(" 			---------
 " ) ;
        printf (" %s 	%c   %g 
 "  , course[a] , grade[a] , num) ;
        total=total+num;
    }
    printf("Average		          %g
 "    ,   total/5) ;
    printf(" 			--------
 " ) ;
}
case 'E' : num=0 ; break :
case 'D' : num=1 ; break ;
case 'C' : num=2 ; break ;
case 'B' : num=3 ; break ;
case 'A' : num=4 : break :
case 'F' : num=-1 ; break ;
for (i = 0; i < 5; i++) {
    if (course[i] == "freshman English")
        course[i] = "Freshman English ";
    if (course[i] == "Roman Empire")
        course[i] = "Roman Empire ";
    if (course[i] == "store number")
        course[i] = "store number ";
}.
printf("Grade Report: Jennifer Jonson
") ;
printf("Total number of courses:
") ;
printf("Average:
") ;
printf("Grade:
") ;
printf("Course:
") ;
}.
Machine Language

Assembly Language

Segment PARA PUBLIC "CSEG"
ADDSEG CSSEG, DSSEG
ENDPT

PUBLIC SEGMENT PARA "CSEG"
FORMAT (1X, A30, 2X, A1, 4X, F4.2)

character*30 course(5)
end

format ( 'Average' , 30x , f4.2)
write(*,*)
write(*,*)  'Elementary French' , 'Pre-Calculus' , 'Volleyball 1' /
write(*,*)  'Student Grade Report: Jennifer Jonson'
write(*,*)  'Class Grade Number'
write(*,*)
c do the rest of this 5 times
total = 0
if ( grade(i) .EQ. 'A' ) gradenum=4
do 10,  i =1, 5

data grade / 'A' , 'B', 'B', 'C', 'A' /
write (* , 99)  average
average = total / 5
10  continue
write (* , 99 )   course(i) , grade(i) , gradenum
if ( grade(i) .EQ. 'D' ) gradenum=1
if ( grade(i) .EQ. 'C' ) gradenum=2
if ( grade(i) .EQ. 'B' ) gradenum=3
if ( grade(i) .EQ. 'A' ) gradenum=4
if ( grade(i) .EQ. 'E' ) gradenum=0

#include <stdio.h>
main() {
    char * course[5]="Freshman English ", "Roman Empire ", "Volleyball 1 ", "Pre-Calculus ", "Elective ";
    float num , total=0;
    for   (a=0 ; a<5 ; a++)     {
        printf(" 
" ) ;
        printf(" Class	        Grade Number
 " ) ;
        printf(" 			---------
 " ) ;
        printf (" %s 	%c   %g 
 "  , course[a] , grade[a] , num) ;
        total=total+num;
    }
    printf("Average		          %g
 "    ,   total/5) ;
    printf(" 			--------
 " ) ;
}
case 'E' : num=0 ; break :
case 'D' : num=1 ; break ;
case 'C' : num=2 ; break ;
case 'B' : num=3 ; break ;
case 'A' : num=4 : break :
case 'F' : num=-1 ; break ;
for (i = 0; i < 5; i++) {
    if (course[i] == "freshman English")
        course[i] = "Freshman English ";
    if (course[i] == "Roman Empire")
        course[i] = "Roman Empire ";
    if (course[i] == "store number")
        course[i] = "store number ";
}.
printf("Grade Report: Jennifer Jonson
") ;
printf("Total number of courses:
") ;
printf("Average:
") ;
printf("Grade:
") ;
printf("Course:
") ;
}.
#include "curses.h"
#include <stdio.h>
main() {
    char * course[5]="Freshman English ", "Roman Empire ", "Volleyball 1 ", "Pre-Calculus ", "Elective ";
    float num , total=0;
    for   (a=0 ; a<5 ; a++)     {
        printf(" 
" ) ;
        printf(" Class	        Grade Number
 " ) ;
        printf(" 			---------
 " ) ;
        printf (" %s 	%c   %g 
 "  , course[a] , grade[a] , num) ;
        total=total+num;
    }
    printf("Average		          %g
 "    ,   total/5) ;
    printf(" 			--------
 " ) ;
}
case 'E' : num=0 ; break :
case 'D' : num=1 ; break ;
case 'C' : num=2 ; break ;
case 'B' : num=3 ; break ;
case 'A' : num=4 : break :
case 'F' : num=-1 ; break ;
for (i = 0; i < 5; i++) {
    if (course[i] == "freshman English")
        course[i] = "Freshman English ";
    if (course[i] == "Roman Empire")
        course[i] = "Roman Empire ";
    if (course[i] == "store number")
        course[i] = "store number ";
}.
printf("Grade Report: Jennifer Jonson
") ;
printf("Total number of courses:
") ;
printf("Average:
") ;
printf("Grade:
") ;
printf("Course:
") ;
}.
A COBOL Program

IDENTIFICATION DIVISION
PROGRAM-ID. D, FORRES
AUTHOR. JOHN DOE
DATE-WRITTEN. MARCH 8, 1995.
DATE-COMPILED
* SYSTEM : STUDENT GRADE REPORT
* RUNNING SCHEDULE : ON DEMAND
* OVERVIEW : THIS PROGRAM WILL READ STUDENT DATA,
* CALCULATE AND PRINT STUDENT'S GRADE AVERAGE.
* INPUT : STUDENT DATA
* OUTPUT : REPORT OF STUDENT'S GRADES AND THE GRADE AVERAGE

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-PC.
OBJECT-COMPUTER. IBM-PC
SPECIAL-NAMES. CO1 IS TOP-OF-PAGE.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT STUDENT-DATA
ASSIGN TO "D: \ INPUT . DAT".
SELECT GRADES-REPORT
ASSIGN TO "D: \ OUTPUT . REP".

DATA DIVISION.
FILE SECTION.
FD STUDENT-DATA
RECORD CONTAINS 121 CHARACTERS
DATA RECORD IS STUDENT-REC.

01 STUDENT-REC.
03 STUDENT-NAME OCCURS 5 TIMES.
10 FIRST-NAME PIC X (8).
10 LAST-NAME PIC X (8).
03 CLASS-DATA OCCURS 5 TIMES.
10 CLASS-DESCR PIC X (20).
10 CLASS-GRADE PIC X .

A Pascal Program

Program grades (input , output)

var
A : integer
num , total , average : real;
grade : array[1..5] of char;
class : array[1..5] of string [30];
Begin
for a : = 1 to 5 do
Begin
case grade [a] of
'A' : num: =4;
'B' : num: =3;
'C' : num : =2;
'D' : num : =1;
'F'  : num : =0;
end; (case)
writeln (CLASS[a], ' ', GRADE[a], ' ', NUM: 4: 2);
total : = total+num;
end; {for}
average: = total/5;
writeln('                -------');
writeln (' Average                  ', average: 4: 2);
end

A FORTRAN Program

C COMPUTER THE SUM AND AVERAGE OF 10 NUMBERS
C REAL NUM, SUM, AVG
C INTEGER TOTNUM, COUNTR
C SUM=0.0
C INITIALIZE LOOP CONTROL VARIABLE
C TOTNUM=10
C LOOP TO READ DATA AND ACCUMULATE SUM
20 IF (COUNTR . GE . TOTNUM) GO TO 30
READ , NUM
SUM=SUM + NUM
C UPDATE LOOP CONTROL VARIABLE
COUNTR = COUNTR + 1
GO TO 20
C END OF LOOP - COMPUTE AVERAGE
30 AVE = SUM / TOTNUM
C PRINT RESULTS
PRINT, SUM
PRINT, AVG
STOP

An SQL query for retrieving data

SELECT < columns > FROM < tables > WHERE < condition >

Example:
SELECT CUSTOMER NAME , AMOUNT OWED
FROM CUSTOMER , ORDER
WHERE CUSTOMER.CUSTOMER ID NO = ORDER.CUSTOMER ID NO

This query creates a new record by linking the customer and order files through their common data field, Customer ID No. The new record will show the name of each customer and the amount that customer owes.