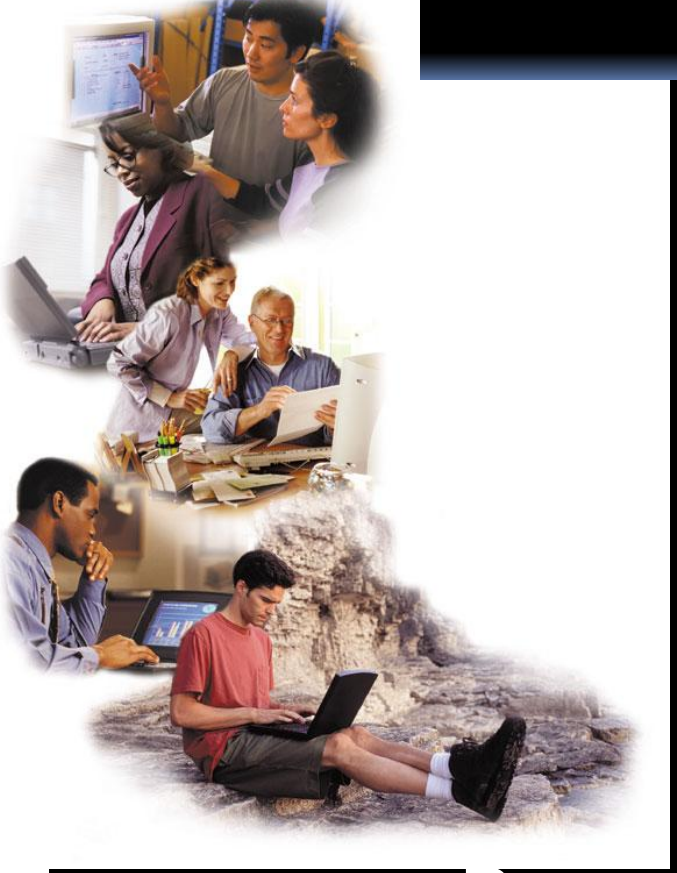


Programming Languages and Program Development

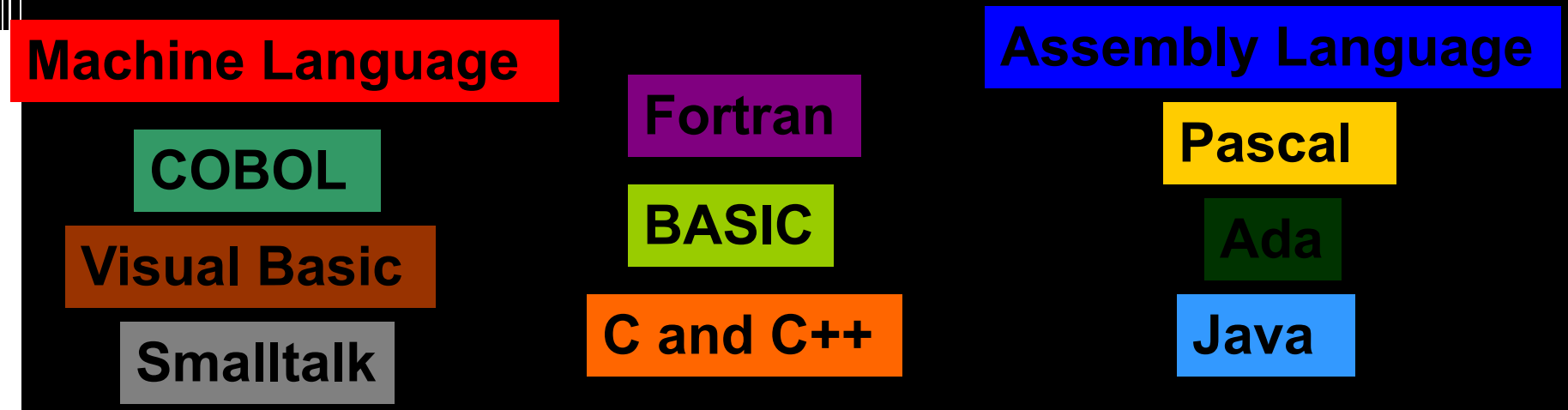
What You Will Learn About

- What a programming language is
- Machine language and assembly language
- High-level programming languages
- The shortcomings of early languages
- Popular programming languages

What You Will Learn About

- The six phases of the program development life cycle (PDLC)
- Why top-down programming makes programs easier to debug and maintain
- The three basic types of control structures
- Syntax errors and logic errors in programs

Programming Languages



- **Programming languages** are artificial languages created to tell the computer what to do
- They consist of vocabulary and a set of rules to write programs
- The **program development life cycle** (**PDLC**) is an organized method of software development

Development of Programming Languages

- Programming languages are classified by levels or generations
- Lower-level languages are the oldest
- The five generations of programming languages are:
  - Machine languages
  - Assembly languages
  - Procedural languages
  - Problem-oriented languages
  - Natural languages

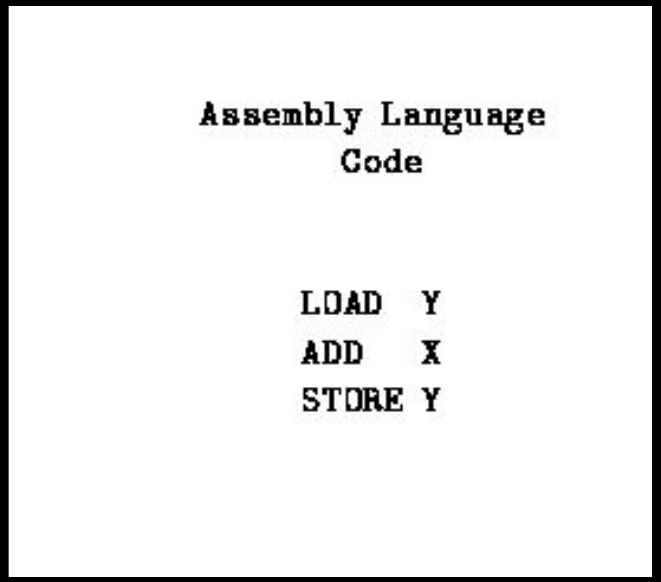
First-Generation Languages

- **Machine language**:
  - Consists of binary numbers (0s and 1s)
  - Is the earliest programming language
  - Is the only language the computer understands without translation
  - Is 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
```

Second-Generation Languages

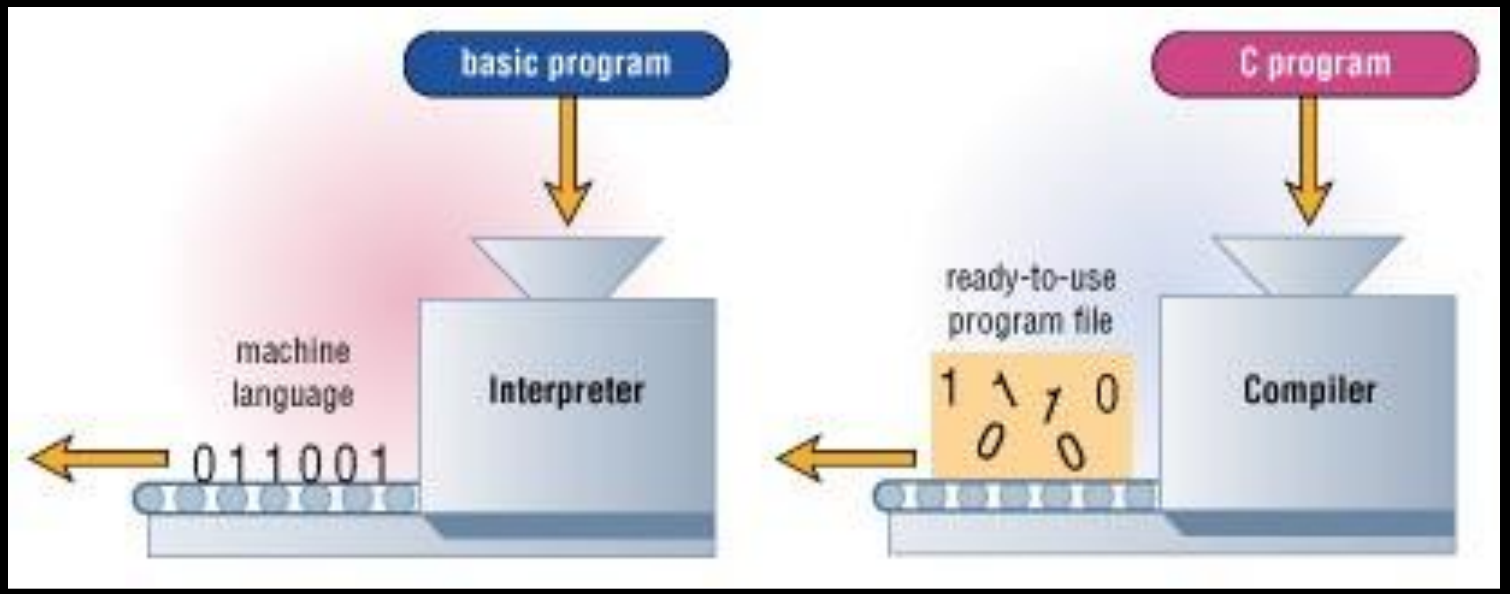
- **Assembly language**:
  - Resembles machine language
  - Is a low-level language
  - Uses brief abbreviations for program instructions.
    - Abbreviations are called **mnemonics**
  - A program is written in **source code** (text file) and translated into machine language by an **assembler**



Third-Generation Languages

- **Procedural languages**:
  - Are high-level languages that tell the computer what to do and how to do it
  - Create programs at a high level of abstraction
  - Are easier to read, write, and maintain than machine and assembly languages
  - Use a compiler or interpreter to translate code
- Fortran and COBOL are third-generation languages

Compilers and Interpreters



- A compiler is a program that changes source code to object code
- An interpreter translates source code one line at a time and executes the instruction

Third-Generation Languages (continued)

- Spaghetti Code and the Great Software Crisis:
  - GOTO statements resulted in programs that were difficult to follow
  - This problem led to the software crisis of the 1960s
    - Programs were not ready on time
    - Programs exceeded their budgets
    - Programs contained too many errors
    - Customers were not satisfied

Third-Generation Languages (continued)

- **Structured programming languages**:
  - Were developed to improve software development
  - Include Algol and Pascal
  - Forbid the use of GOTO statements
  - Use control structures
    - IF-THEN-ELSE

Third-Generation Languages (continued)

- **Modular programming languages**:
  - Were developed because of problems in structured programming languages
  - Are used to create programs that are divided into separate modules
    - Each module carries out a special function
  - Require specified input to produce specified output

Fourth-Generation Languages

- Types of fourth-generation languages include:
  - Report generators
    - Languages for printing database reports
  - Query languages
    - Languages for getting information out of databases
- Fourth-generation languages are **nonprocedural**
  - They do not force programmers to follow procedures to produce results

Object-Oriented Programming

- **Object-oriented programming (OOP)**:
  - Relies on **component reusability**
    - The ability to produce program modules that perform a specific task
  - Eliminates the distinction between programs and data
  - Uses **objects** that contain data and procedures

Objects

- **Objects** are units of information that contain data as well as methods that process and manipulate the data
- **Classes of objects**:
  - Hierarchy or category of objects
  - Objects at the top of the category are broader in scope than the subclass objects
- **Inheritance** refers to an object's capacity to "pass on" its characteristics to its subclasses

Common Business-Oriented Language (COBOL)

COBOL:

- The earliest (1959) high-level language
- The most widely used business language
- A proven way to do accounting, inventory, billing, and payroll
- Requires programmers to explain what the program is doing at each step

Sample Cobol program

```
      $ SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID.  Multiplier.
AUTHOR.  Michael Coughlan.
* Example program using ACCEPT, DISPLAY and MULTIPLY to
* get two single digit numbers from the user and multiply them together

DATA DIVISION.

WORKING-STORAGE SECTION.
01  Num1                                PIC 9  VALUE ZEROS.
01  Num2                                PIC 9  VALUE ZEROS.
01  Result                              PIC 99 VALUE ZEROS.

PROCEDURE DIVISION.
    DISPLAY "Enter first number  (1 digit) : " WITH NO ADVANCING.
    ACCEPT Num1.
    DISPLAY "Enter second number (1 digit) : " WITH NO ADVANCING.
    ACCEPT Num2.
    MULTIPLY Num1 BY Num2 GIVING Result.
    DISPLAY "Result is = ", Result.
    STOP RUN.
```

Formula Translator (Fortran)

- **Fortran:**
  - Began in the 1950s
  - Is suited to scientific, mathematical, and engineering applications
  - Is used to solve complex equations
  - Features simplicity, economy, and ease of use

Sample 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 '
```

Ada

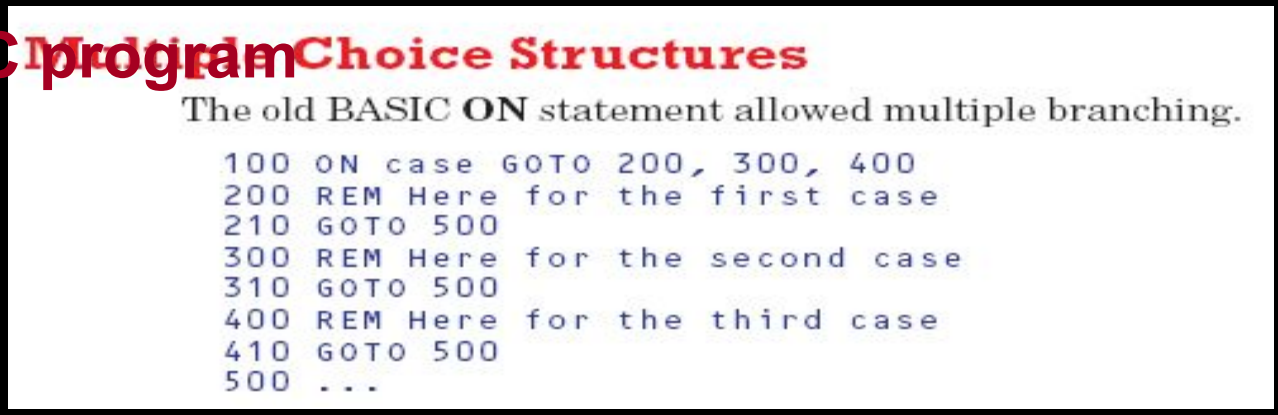
- **Ada:**
  - Named after Augusta Ada Byron
  - Incorporates modular programming
  - The required language for the U.S. Defense Department
  - Suitable for control of real-time systems (missiles)

Sample Ada program

```
      Examples involving scalar subtype conversions:

(22)
      I, J : Integer range 1 .. 10 := 5;
      K    : Integer range 1 .. 20 := 15;
       ...
(23)
      I := J;  --  identical ranges
      K := J;  --  compatible ranges
      J := K;  --  will raise Constraint_Error if K > 10

(24)
      Examples involving array subtype conversions:

(25)
      A : String(1 .. 31);
      B : String(3 .. 33);
       ...
(26)
      A := B;  --  same number of components
(27)
      A(1 .. 9)  := "tar sauce";
      A(4 .. 12) := A(1 .. 9);  --  A(1 .. 12) = "tartar sauce"
```

Beginner's All-Purpose Symbolic Instruction Code (BASIC)

Sample BASIC program

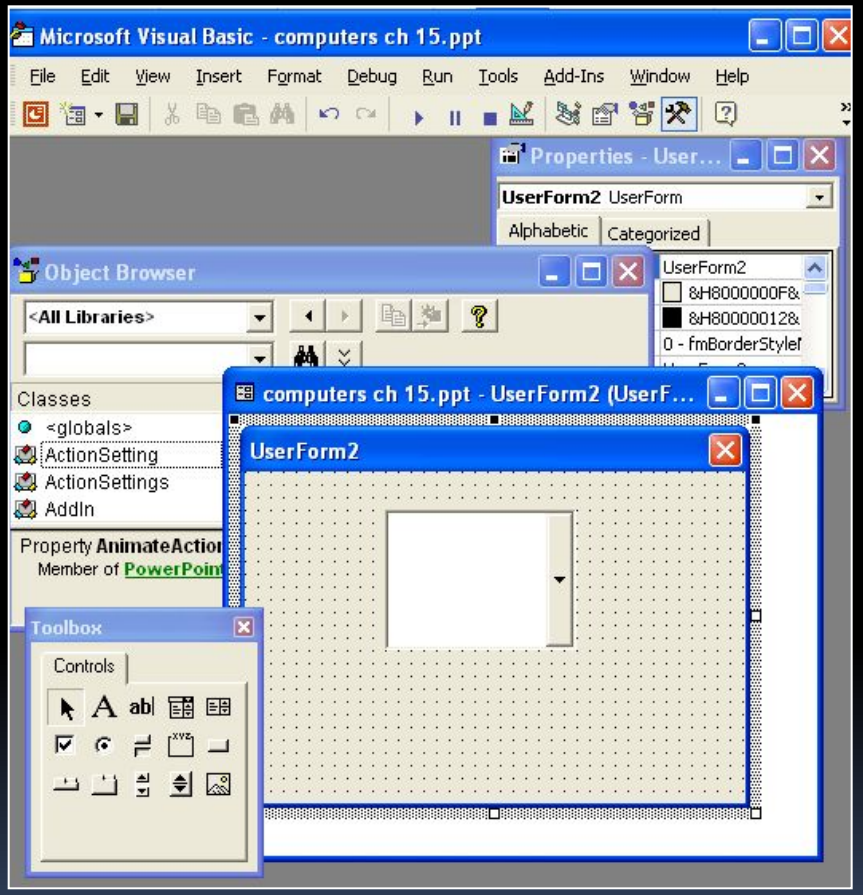


- **BASIC:**
  - An easy-to-use language available on personal computers
  - Widely taught in schools as a beginner's programming language
  - Designed as an interpreted language

Visual Basic (VB)

- **Visual Basic:**
  - Is widely used in program development packages
  - Uses event-driven programming
  - Enables the programmer to develop an application by using on-screen graphical user interfaces

Sample Visual Basic



Pascal

Sample Pascal program

```
var                                          {outer-block variables}
          {small variables...}
   ktr           : integer;

          {big variables...}
   big_array     : array [0..9999] of integer;
...
procedure foo;
   var
             {big variables...}
      big_array_2 : array [0..9999] of integer; {40K bytes}

             {small variables...}
      ktr2       : integer;
```

- **Pascal:**
  - Is named after Blaise Pascal
  - Encourages programmers to write well-structured programs
  - Widely accepted as a teaching language
  - Has been updated to reflect new approaches to programming

C

- **C:**
  - Was developed by AT&T's Bell Labs in the 1970s
  - Combines high-level programming language with assembly language
  - Programmers manipulate bits of data within a processing unit

Sample C program

```
#include <stdio.h>

int main()
{
    printf("This is output from my first program!\n");
    return 0;
}
```

- Difficult to learn and programming is time consuming

Smalltalk

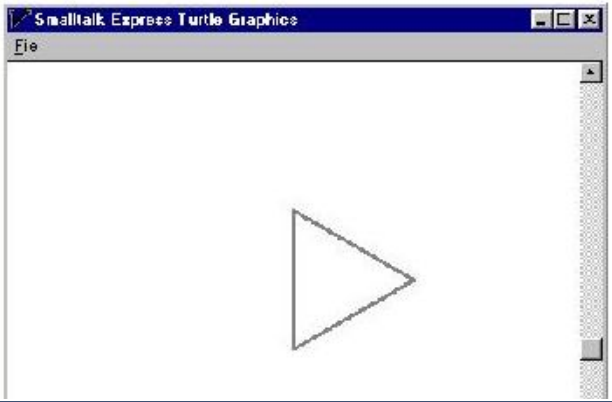
- **Smalltalk:**
  - Developed in the 1970s by Xerox Corp
  - "100% pure" object-oriented programming language
  - Not often chosen for software development

Sample Smalltalk program

Source Code

```
Window turtleWindow: 'Turtle Graphics'.
   Turtle
        defaultNib: 2;
        foreColor: ClrDarkgray;
        home;
        go: 100;
        turn: 120;
        go: 100;
        turn: 120;
        go: 100;
        turn: 120
```

Sample Run



C++

- **C++:**
  - Incorporates object-oriented features
  - Is widely used for professional program development

Sample C++ program

```
#include <iostream.h>

int main()

{

  cout<<"HEY, you, I'm alive!  Oh, and Hello World!";

  return 0;

}
```

Java

- **Java:**
  - Developed by Sun Microsystems
  - An object-oriented, high-level programming language with a twist
  - First true cross-platform programming language
  - Gained acceptance faster than any other programming language
  - A simplified version of C++

Java

- **Java,** continued **:**
  - Java is designed to run on any computer platform
  - **Java Virtual Machine** enables cross-platform use
  - **Java applets** or small programs are downloaded to computers through networks
  - Weaknesses include:
    - The security risk in downloading applets
    - The speed in running the programs

Sample 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();
}
```

Web-Based Languages

- **Markup languages:**
  - **Hypertext markup language** (**HTML**) sets the attributes of text and objects within a Web page
  - **Extensible markup language** (**XML**) is used for sharing data and objects in a Web environment
- **Scripting languages:**
  - **VBScript** is used to write short programs (scripts) that are embedded in Web pages
  - **JavaScript** is used to write scripts on Web pages
- **Visual Studio .NET:**
  - Used for the development of scripts and programs that are accessible from the Web

The Program Development Life Cycle (PDLC)

- The PDLC was introduced in the 1970s to address problems in creating programs
- It provides an organized plan for breaking down the task of program development into manageable parts
- Six phases of the PDLC:
  1. Defining the problem
  2. Designing the program
  3. Coding the program
  4. Testing and debugging the program
  5. Formalizing the solution
  6. Implementing and maintaining the program

Phase 1: Defining the Problem

- The first step in program development
- Systems analysts provide program specifications (specs) to programmers
- Specs define:
  - Input data
  - Processing
  - Output
  - Appearance of user interface

Phase 2: Designing the Program

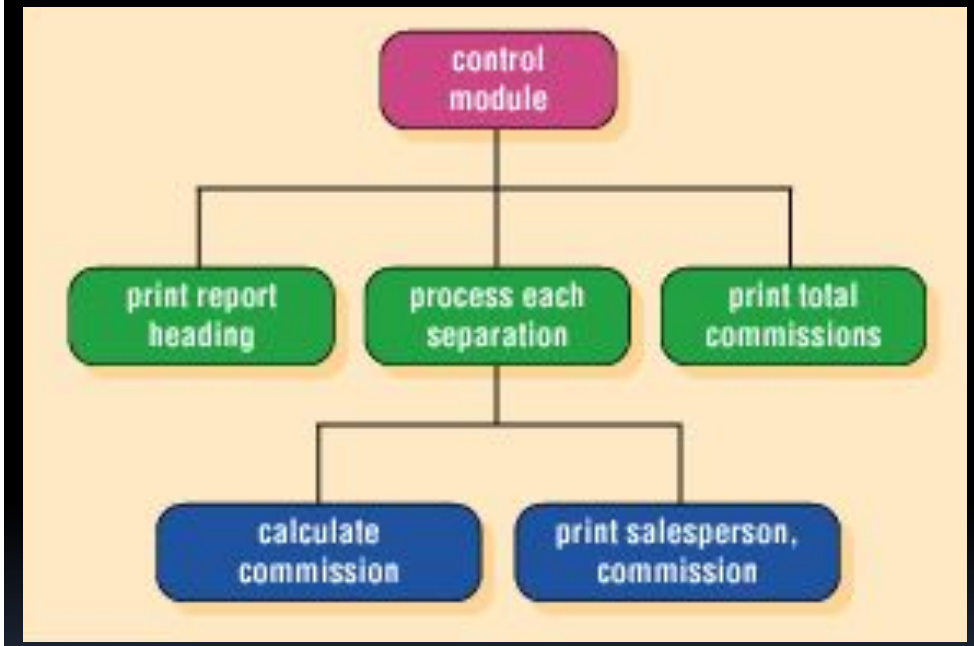
- Programmers create the program's design
  - **Top-down design** focuses on the program's main goal (main routine), then breaks the program into manageable components (subroutines/modules)
  - Control structures are used to see how each subroutine will do its job
- Developing an algorithm is a step-by-step description of how to arrive at a solution
- Program design tools:
  - Structure charts – show the top-down design
  - Flow charts – show the logic of program
  - Pseudo code – alternative to flow charts

Structured Design

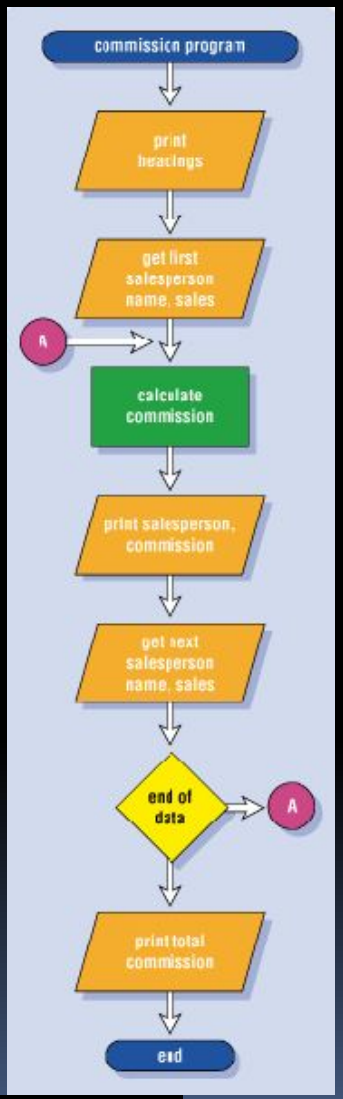
- **Control structures** are logical constructs that specify how the instructions in a program are to be executed
- Three types of control structures:
  - **Sequence control structure** – Instructions are executed in the order in which they appear
  - **Selection control structures** – The program branches to different instructions depending on whether a condition is met; IF…THEN…ELSE
  - **Repetition control structure** – The program repeats the same instructions over and over; DO-WHILE and DO-UNTIL

Structure Chart and Flowchart

Structure Chart



Flowchart



Phase 3: Coding the Program

- Coding requires the translation of the algorithm into specific program instructions
- An appropriate programming language is chosen, and the code is typed according to its syntax rules

Phase 4: Testing and Debugging the Program

- Testing and debugging eliminate all errors
- Syntax and logic errors are corrected
- Debugging is the process of eliminating errors

Phase 5: Formalizing the Solution

- Documentation is created for future use
- The variable names and definitions, a description of the files needed, and the layout of the output are produced
- A user manual is developed to explain how the program works

Phase 6: Implementing and Maintaining the Program

- The program is:
  - Tested by users
  - Thoroughly documented
  - Maintained and evaluated regularly

Summary

- A programming language is an artificial language consisting of a vocabulary and a set of rules
- Machine language is the lowest-level programming language
- Assembly language contains symbols for programming instructions
- Third-generation (high-level) languages require programmers to specify the procedures to be followed
- Object-oriented languages combine procedures and data

Summary, continued

- The PDLC's six phases are:
  - Defining the program
  - Designing the program
  - Coding the program
  - Testing and debugging the program
  - Formalizing the solution
  - Implementing and maintaining the program
- Top-down programming makes programs easier to debug and maintain
- Debugging requires finding and correcting syntax errors and logic errors