Chapter 13
Programming and Languages

Lecture Guide

- **Learning Objectives**
  - Define programming and describe the six steps of programming.
  - Compare design tools including top-down design, pseudocode, flowcharts, and logic structures.
  - Describe program testing and the tools for finding and removing errors.
  - Describe CASE tools and object-oriented software development.
  - Explain the five generations of programming languages.

Chapter Outline

- **Programs and Programming**
  - A **program** is a list of instructions for the computer to follow to accomplish the task of processing data into information.
  - The instructions are made up of statements used in a programming language such as C++, Java, or Visual Basic.
  - Can off-the-shelf software do the job, or should it be custom written?
    - One of the first things that should to be decided in programming.
    - Programming is a problem solving procedure
  - Programming, also known, as software development, is a six-step procedure used by software engineers or programmers for creating the list of instructions.
  - **SDLC**-the software development life cycle consists of six steps.
    - *Program specification*: The program's objectives, outputs, inputs, and processing requirements are determined.
    - *Program design*: A solution is created using programming techniques such as top-down program design, pseudocode, flowcharts, and logic structures.
    - *Program code*: The program is written or coded using a programming language.
    - *Program test*: The program is tested or debugged by looking for syntax and logic errors.
    - *Program documentation*: Documentation is an ongoing process throughout the programming process. This phase focuses on formalizing the written description and processes used in the program.
    - *Program maintenance*: Completed programs are periodically reviewed to evaluate their accuracy, efficiency, standardization, and ease of use.
    - Computer professional, known as **software engineers** or **programmers**, use the six-step procedure.
- **Step 1: Program Specification**
  - Also called **program definition** or **program analysis,** it requires that the programmer specify five items:
    - Program objectives - Make a clear statement of the problem you are trying to solve. Determine your objectives.
    - Desired Outputs - list what you want to get out of the computer system, specify outputs before inputs.

        - Input data - Once you know the output you want, you can determine the input data and the source of this data.
        - Processing Requirements - define the processing tasks that must happen for input data to be processed into output.
        - Documentation – Create a program specifications document recording program objectives, desired outputs, needed inputs, and required processing. Ongoing documentation is essential.
- **Step 2: Program Design**
    - Plan a solution preferably using structured programming techniques.
    - Determine the outputs and inputs for the program, and then use top-down program design to identify the program's processing steps.
        - Program modules
        - Each module made up of logically related program statements
    - **Pseudocode** - an outline of the logic of the program to be written. See Figure 13-7
    - **Flowcharts** - graphically present the detailed sequence of steps needed to solve a programming problem. See Figures 13-8 through 13-9.
    - **Logic structures**; sequential, selection, and repetition
        - Sequential structure - one program statement follows another. See Figure 13-10.
        - Selection structure - occurs when a decision must be made. The outcome of the decision determines which of two paths to follow. (IF-THEN-ELSE structure).
        - Repetition or Loop structure - describes a process that may be repeated as long as a certain condition remains true. The structure is called a "loop" or "iteration" because the program loops around or repeats again and again.
    - The last thing to do before leaving the program design step is to document the logic of the design. This report typically includes pseudocode, flowcharts, and logic structures.
- **Step 3: Program Code**
    - Writing the program is called **coding**.
    - Write the "program code" that instructs the computer what to do.
    - A Good Program
        - Should work under most conditions.
        - Should catch obvious and common input errors.
        - Should be well documented and understandable by programmers other than the programmer who wrote it.
        - One of the best ways to code effective programs is to write **structured programs** using logic structures.
    - Coding
        - **Code** or write the program using the appropriate computer language.
            - Focus on assigning meaning to different pieces of content.
        - A **programming language** uses a collection of symbols, words, and phrases that instruct a computer to perform specific operations.
            - Focus on processing data and information for a wide variety of different types of applications (such as C++)

- **Step 4: Program Test**
  - Debugging refers to the process of testing and then eliminating errors.
  - Programming errors are of two types: syntax errors and logic errors.
    - Syntax errors
      - A violation of the rules of the programming language.
    - Logic errors
      - Occur when the programmer uses an incorrect calculation or leaves out a programming procedure.
  - Testing Process
    - **Desk checking (code review)** - The programmer proofreads a printout of the program listing line by line checking for syntax and logic errors.
    - **Manually testing with sample data** – Checking for programming logic errors, the programmer compares the manually calculated values to those calculated by the program(s).
    - **Attempt at translation** - The program is run through a computer, using a translator program. The translator attempts to translate the written program from the programming language (such as C++) into the machine language. It will identify any syntax errors.
    - **Testing sample data on the computer** - the program is tested for logic errors using sample data after all syntax errors have been corrected.
    - **Testing by a select group of potential users -** called beta testing, potential users try out the program and provide feedback.
- **Step 5: Program Documentation**
  - Consists of written descriptions and procedures about a program and how to use it.
  - Continues throughout all the programming steps.
  - All the prior documentation is reviewed, finalized, and distributed.
  - Documentation is important for people who may be involved with the program in the future (users, operators and programmers)
    - Two examples of documentation are:
      - printed manuals
      - Help option
- **Step 6: Program Maintenance**
  - As much as 75 percent of the total lifetime cost for an application program is for maintenance.
  - The purpose of program maintenance is to ensure that current programs are operating error free, efficiently, and effectively.
  - Activities in program maintenance
    - **Operations -** concerned with locating and correcting operational errors, making programs easier to use, and standardizing software using structured programming techniques.
    - Programming modifications or corrections are often referred to as **patches**. For software that is acquired, it is common for the software manufacture to periodically send patches or updates for their software. If these patches are significant, they are known as **software updates**.

  - **Changing needs -** Programs need to be adjusted for a variety of reasons, including new tax laws, new information needs, and new company policies. Some projects start before all requirements are known
    - SDLC (Systems Design Life Cycle) – then becomes more cyclical
    - **Agile development** is a popular development methodology
      - Starts by getting core functionality
      - Expands on project until customer is satisfied with the results

- **CASE and OOP**
  - **CASE -** Computer Aided Software Engineering tools provide some automation and assistance in program design, coding, and testing.
  - **Object - Oriented Programming (OOP) –** focuses less on the procedures and more on defining the relationships between previously defined procedures or "objects".
    - Process of organizing a program into objects.
    - Each **object** contains both the data and processing operations needed to perform a task.
    - Objects are reusable, self-contained components
- **Generations of Programming Languages**
  - **Levels** or **generations** of programming languages, range from "low" to "high."
  - Programming languages are called **lower level** when they are closer to the language the computer itself uses. The computer understands the 0s and 1s that make up bits and bytes.
  - Programming languages are called **higher level** when they are closer to the language humans' use—that is, for English speakers, more like English.
  - **First generation – Machine language**
    - Data represented in 1's and 0's
    - Machine languages also vary according to make of computer
  - **Second generation – Assembly languages**
    - Use abbreviations or mnemonics such as ADD that are automatically converted to the appropriate sequence of 1's and 0's.
    - Much easier for humans to understand and to use.
    - Vary from computer to computer.
  - **Third Generation – High-level procedural languages**
    - Portable – they can be moved from one computer to another
    - They are more "English-like" programming languages
    - Still require training to use higher-level languages
    - Known as 3GLs
    - Designed to express the logic—the procedures—that can solve general problems
    - Most widely used languages to create software applications
    - C++ is a widely used procedural language
    - **Compiler -** converts a programmer's procedural language program, called the source code, into a machine language code, called the object code. This object code can then be saved and run later.
    - **Interpreter** - converts the procedural language one statement at a time into machine code just before it is to be executed. No object code is saved.

- **Fourth generation – Task-Oriented Languages**
  - Require little special training on the part of the user.
  - Known as 4GL's or very high-level languages.
  - Designed to solve specific problems.
  - 4GL's are nonprocedural languages and focus on specifying what the program is to accomplish.
  - 4GLs are more English-like
    - **Query languages** - enable nonprogrammers to use certain easily understood commands to search and generate reports from a database.
      - One of the most widely used query languages is SQL (structured query language).
    - **Application generators** – or a **program coder**, a program that provides modules of prewritten code, greatly reduces the time required to create an application.
- **Fifth generation – Problem and Constraint Languages**
  - The next step in programming languages with be the fifth generation language (5GL).
  - A computer language that incorporates the concepts of artificial intelligence to allow a person to provide a system with a problem and some constraints, and then request a solution.
  - Enables a computer to learn and to apply new information just as people do.
  - Communicate more directly to a computer using **natural languages** or human languages such as English or Spanish.
  - When will fifth-generation languages become a reality? That's difficult to say, however, researchers are actively working on the development of 5GL languages and have demonstrated some success.

- **Careers in IT**
  - Computer programmers create, test, and troubleshoot programs used by computers.
  - Programmers also may update and repair existing programs.
  - Need for programmers to work on the most basic computer functions have decreased. However, demand for computer programmers with specializations in advanced programs continues.
  - Jobs in programming typically require a bachelor's degree in Computer Science or information systems
  - Salaries range from $ 49,000 to $89,000.
- **A Look to the Future**
  - Your Own Programmable Robot
  - Would you like to have your own robot that could help you with all of your chores and understood your words?
  - It is just a matter of time before these robots can understand human instructions instead of complex programming languages.
  - One of earliest robots made available to consumers was the Roomba from iRobot, which is essentially an automated, intelligent vacuum cleaner.

- A company named Aldebaran Robotics has taken a different approach, creating small, humanoid, or human-like, robots that the end-user can program.
- In the future, it will not be necessary for someone to use software or know a programming language to communicate with a robot.
- The hardware components needed to make robots are becoming cheaper. However, the software remains a challenge. Human languages and conversations remain very difficult for a computer to fully understand.

Teaching Tips

- **Programs and Programming**
  - A program is a list of instructions for the computer to follow to accomplish the task of processing data into information.
  - A good way to show how a computer would interpret a process is to play a "Make a Sandwich Game". Bring in different types of meat, cheese, and condiments and have them try to instruct you on how to make the sandwich. Take them as literal as possible.
- **Program Specification**
  - Emphasize the five aspects, especially that outputs must be decided first. Sometimes students have a hard time understanding this.
  - Provide a scenario that requires students to identify the desired outputs; use a white board as a place for students to record outputs to be discussed
- **Program Design**
  - Illustrate the different steps by providing a visual aid that would walk through a flowchart with the various structures.
  - Have students create flowcharts examples using flowchart design
- **Program Code**
  - Use pseudo code to write simple programs
  - Use the different types of structures in the program description to help the students better understand the different concepts.
- **Program Test**
  - Go through the different aspects that need to be tested.
  - You can use the analogy that syntax is to programming as grammar is to English, and how it can affect the understanding of the code.
  - Discuss the consequences if a program is not tested thoroughly.
    - **Desk checking** - The programmer goes through the listing line by line looking for syntax and logic errors.
    - **Manually testing with sample data** - Looking for programming logic errors, the programmer compares the manually calculated values to those calculated by the programs.
    - **Attempt at translation** - The translator attempts to translate the written program from the programming language (such as C++) into the machine language. It will identify syntax errors
    - **Testing sample data on the computer** - the program is tested for logic errors using sample data

        - **Testing by a select group of potential users -** called beta testing, potential users try out the program and provide feedback.
- **Program Documentation**
    - You can demonstrate internal documentation by having the students include remark statements within their lab programs.
- **Program Maintenance**
    - Emphasize this is the bulk of the lifetime cost of the application.
    - You can discuss with the students different types of changing needs that can arise in an organization and relate it to Program Maintenance.
- **CASE and OOP**
    - Emphasize how each tool can aid in program development and save time and money in the process.
- **Generations of Programming Languages**
    - Ask students to use the Internet to find the names of programming languages for each level of generation
    - What distinguishes a lower level program from a higher level program? Explain.

Key Terms

Key Term	Definition
agile development	A popular development methodology that gets the core functionality of a program working then expands on it until the customer is satisfied with the results.
application generator	A program that provides modules of prewritten code. (see also program coder)
assembly language	A second generation programming language that uses abbreviations or mnemonics that are automatically converted to the appropriate sequence of 1s and 0s.
beta testing	Potential users try out the program and provide feedback.
code	Writing a program using the appropriate computer language.
code review	Proofreading a printout of a program for syntax and logic errors.
coding	Writing a computer program.
compiler	Converts the programmer's procedural language program into a machine language code.
computer-aided software engineering (CASE) tools	Provide some automation and assistance in program design, coding, and testing.
content-markup language	Programming, www.mhhe.com/ce2015, select Student Edition, choose Chapter 13, and then Flashcards
debugging	Running a program on a computer and then fixing the parts that do not work.
desk checking	A programmer goes through the listing line by line looking for syntax and logic errors.
DO UNTIL structure	A type of loop structure. www.mhhe.com/ce2015, select

	Student Edition, choose Chapter 13, and then Flashcards
DO WHILE structure	A type of loop structure. www.mhhe.com/ce2015, select Student Edition, choose Chapter 13, and then Flashcards
documentation	Written descriptions and procedures about a program and how to use it.
fifth-generation language (5GL)	A computer language that incorporates the concepts of artificial intelligence to allow direct human communication.
fourth-generation language (4GL)	Designed to solve specific problems. (see also very high-level languages)
generation	Classifications of programming languages. (see also level)
higher level	Programming languages that are closer to the language humans use
IF-THEN-ELSE structure	A logic structure that occurs when a decision must be made. (see also selection structure)
IFPS (interactive financial planning system)	A fourth generation programming language used to develop financial models.
interpreter	Converts the procedural language one statement at a time into machine code just before it is to be executed.
level	Classifications of programming languages. (see also generation)
logic error	Occurs when the programmer uses an incorrect calculation or leaves out a programming procedure.
logic structure	Programming techniques that take much of the guesswork out of programming.
loop structure	A programming process that may be repeated as long as a certain condition remains true.
lower level	Programming language closer to the language the computer itself uses. The computer understands the 0s and 1s that make up bits and bytes.
machine language	First generation language where data is represented in 1s and 0s.
maintenance programmer	A person who ensures that current programs are operating error free, efficiently, and effectively, and fixes problems when it isn't.
module	A program's processing steps. (see also programming module)
natural language	Human languages such as English or Spanish.
object	Contains both the data and processing operations necessary to perform a task.
object code	Machine language code.
object-oriented programming (OOP)	A process by which a program is organized into objects, which contain both the data and processing operations necessary to perform a task.
object-oriented software development	A type of programming that focuses on defining the relationships between previously defined procedures or

	“objects.
objective	Problem to be solved.
operator	People responsible for maintaining the system or program.
patches	Programming modifications or corrections.
portable language	Can be run on more than one kind of computer.
procedural language	Programming language designed to express the logic that can solve general problems. (see also third generation language)
program	A list of instructions for the computer to follow to accomplish the task of processing data into information.
program analysis	Requires that five items be specified for the program: the program’s objectives, the desired output, the input data required, the processing requirements, and the documentation. (see also program specification, program definition)
program coder	A program that provides modules of prewritten code. (see also application generator)
program definition	Requires that five items be specified for the program: the program’s objectives, the desired output, the input data required, the processing requirements, and the documentation. (see also program specification, program analysis)
program design	Planning a solution to a programming problem.
program documentation	Documentation that is carried on throughout all the programming steps.
program flowchart	A graphical representation of the detailed sequence of steps needed to solve a programming problem.
program maintenance	To ensure that current programs are operating error free, efficiently, and effectively.
program module	A program’s processing steps.(see also module)
program specification	Requires that five items be specified for the program: the program’s objectives, the desired output, the input data required, the processing requirements, and the documentation. (see also program definition, program analysis)
programmer	A person who creates, tests, and troubleshoots programs used by computers.
programming	A six-step procedure for programs. (see also software development)
programming language	Uses a collection of symbols, words, and phrases that instruct a computer to perform specific operations.
pseudocode	An outline of the logic of the program.
query language	A fourth generation language that uses certain easily understood commands to search and generate reports from a database.
repetition structure	A loop structure that describes a process that may be

	repeated as long as certain conditions remain true.
selection structure	A logic structure that occurs when a decision must be made. (see also IF-THEN-ELSE structure)
sequential structure	One program statement follows another.
service updates	Changes over time
software development	Programming is also known as software development
software development life cycle	A six step process for creating new programs
software engineer	Create software required for information systems.
Software updates	Significant patches to a program/software
source code	The programmer's procedural language program,
structured program	The best way to code effective programs by using the logic structures.
structured programming technique	Technique that consist of top-down program design, pseudocode, flowcharts, and logic structures.
syntax error	A violation of the rules of the programming language.
task oriented language	A programming language designed to solve specific problems.
third-generation language (3GL)	Programming languages designed to express the logic that can solve general problems. (see also procedural language)
top-down program design	A design process that identifies the program's processing steps.
user	A person who uses the software to provide the necessary outputs.
very high-level language	Designed to solve specific problems. (see also problem oriented languages, fourth generation language)

Answers to End-of-Chapter Materials
Chapter 13

Num	Multiple Choice Answers (Book)	Matching Answers (Book)	Multiple Choice Answers (www.mhhe.com/ce2015 Only)	Matching Answers (www.mhhe.com/ce2015 Only)
1	A	G	D	G
2	C	H	C	H
3	D	I	B	C
4	D	C	C	F
5	D	A	D	E
6	D	B	C	A
7	D	E	B	D
8	A	D	A	B
9	B	J	C	I
10	A	F	A	J

Open Ended Questions:

1. **Identify and discuss each of the six steps of programming.**
    - **Program specification**: The program's objectives, outputs, inputs, and processing requirements are determined.
    - **Program design**: A solution is created using programming techniques such as top-down program design, pseudocode, flowcharts, and logic structures.
    - **Program code**: The program is written or coded using programming language.
    - **Program test**: The program is tested or debugged by looking for syntax and logic errors.
    - **Program documentation**: Documentation is an ongoing process throughout the programming process. This phase focuses on formalizing the written description and processes used in the program.
    - **Program maintenance**: Completed programs are periodically reviewed to evaluate their accuracy, efficiency, standardization, and ease of use.

2. **Describe CASE tools and OOP. How does CASE assist programmers?**
    - **Computer-aided software engineering tools (CASE) -** provide some automation and assistance in program design, coding, and testing. CASE tools make programmers work easier, faster, and more reliable.
    - **Object-oriented programming (OOP) -** is a process by which a program is organized into objects. Each object contains both the data and processing operations necessary to perform a task.
        - Object-oriented software development focuses less on procedures and more on defining the relationships between previously defined procedures or "objects".

3. **What is meant by "generation" in reference to programming languages? What is the difference between low-level and high-level languages?**
   Computer professionals talk about levels or generations of programming languages, ranging from "low" to "high."
    - Programming languages are called lower level when they are closer to the language the computer itself uses. The computer understands the 0s and 1s that make up bits and bytes.
    - Programming languages are called higher level when they are closer to the language humans use—that is, for English speakers, more like English.
    - There are five generations of programming languages: (1) machine languages, (2) assembly languages, (3) procedural languages, (4) task-oriented languages, and (5) problem and constraint languages.

4. **What is the difference between a compiler and an interpreter?**
    - **Compiler -** converts the programmer's procedural language program, called the source code, into a machine language code, called the object code.
    - **Interpreter** - converts the procedural language one statement at a time into machine code just before it is to be executed.

5. **What are logic structures? Describe the differences between the three types.**
   Logic structures link the various parts of the flowchart. The best way is a combination of three logic structures:
   - Sequential structure - one program statement follows another.
   - Selection structure - occurs when a decision must be made. The outcome of the decision determines which of two paths to follow. (IF-THEN-ELSE structure)
   - Repetition or loop (iteration) structure - describes a process that may be repeated as long as a certain condition remains true.