ITC213: STRUCTURED PROGRAMMING

Bhaskar Shrestha
National College of Computer Studies
Tribhuvan University

Lecture 03: Program Development Life Cycle

Readings: Not Covered in Textbook

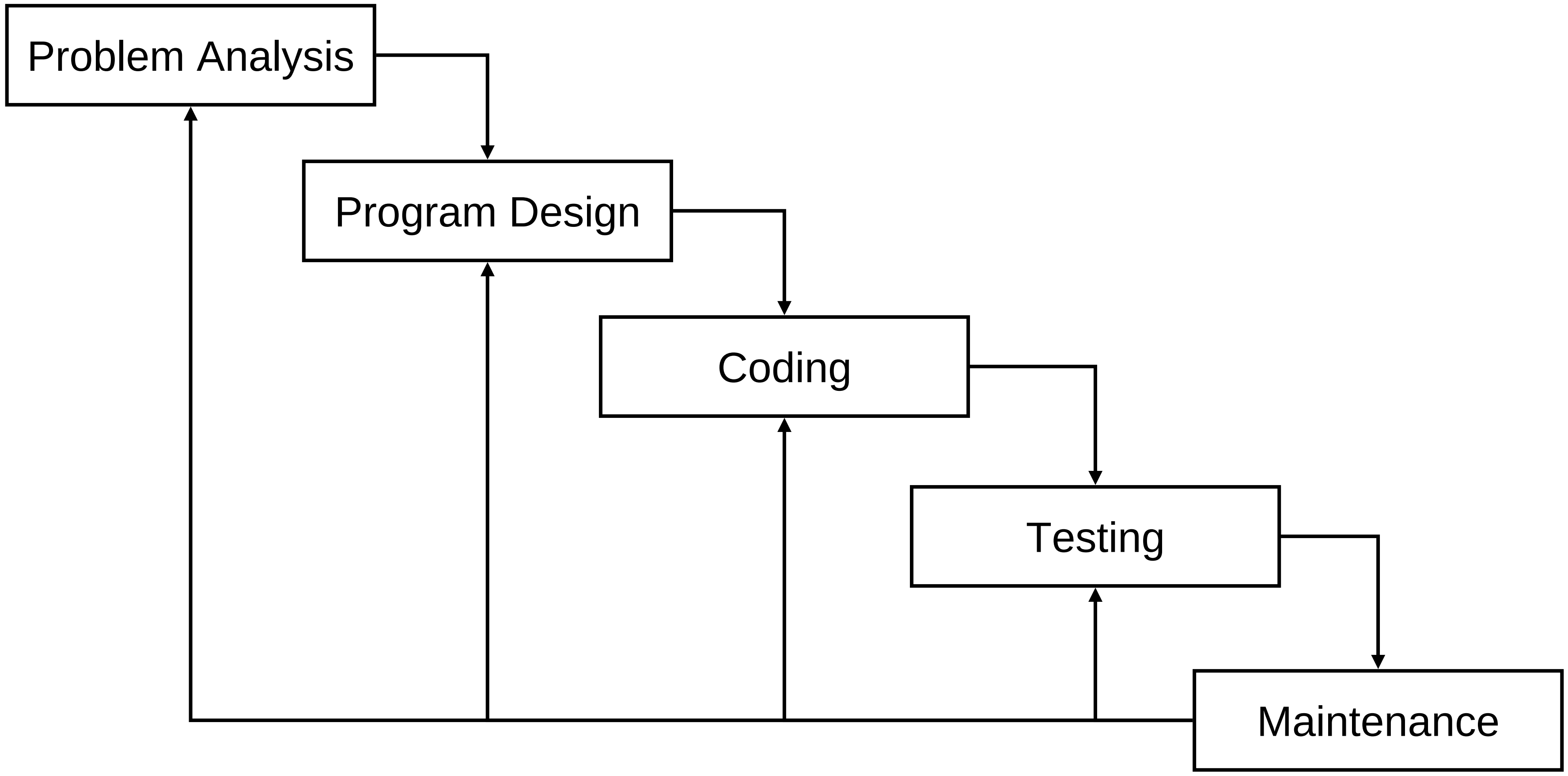
Program Development Life Cycle

- Creating new programs is called ***program development***
- The process associated with creating successful applications programs is called the ***program development life cycle*** (PDLC)
- It includes a set of activities which produce application programs
- There are various approaches in program development each with different perspectives

Waterfall model

- The first published model of program development process was derived from other engineering process (Royce, 1970)
- It is a multistage model with each stage concerned with separate activities of program development
- Each stage in the model follows a linear sequence one after another, often with feedback loops
- Also known as classic life cycle, linear sequential model

Waterfall Model



Problem Analysis: Define the Problem (1/2)

- The stage begins with reviewing the program specifications indicating what the new system should do
- During this stage, the systems analyst and programmer review the specifications and possibly talk with users in order to fully understand what the software should do.
- Documentation resulting from this phase consists of the program specifications, timetable, which language will be used, how the program will be tested, and what documentation is required

Problem Analysis: Define the Problem (2/2)

- The program services, constraints and objectives are established by consultation with system users
- Consists of the following tasks:
  - Define Objectives of the Program
  - Determine desired outputs
  - Determine input requirements
  - Determine processing requirements
  - Evaluate feasibility of the program
  - Document the analysis

Program Design: Outline the Solution

- During program design, the specifications are used to express the algorithm needed to solve the problem, usually in the form of any number of program design tools
- Program Design Tools
  - Structure charts
  - Flowcharts
  - Pseudocode
  - Data modeling
- Documentation from the program design step includes all the design specifications (flowcharts, pseudocode, etc.)

Structure Charts and Data Modeling

- **Structure charts** depict the overall organization of a program, and how the modules of a program—logically related operations that perform a well-defined task—are defined and how they connect to each other hierarchically
- Program modules should be arranged hierarchically, in a *top-down* fashion, so that their relationship to each other is apparent
- **Data modeling** is a technique used to illustrate the data in an application and is frequently used with object-oriented programming
- In a data model, the objects in the program are identified, along with their variables and class

Structured Programming

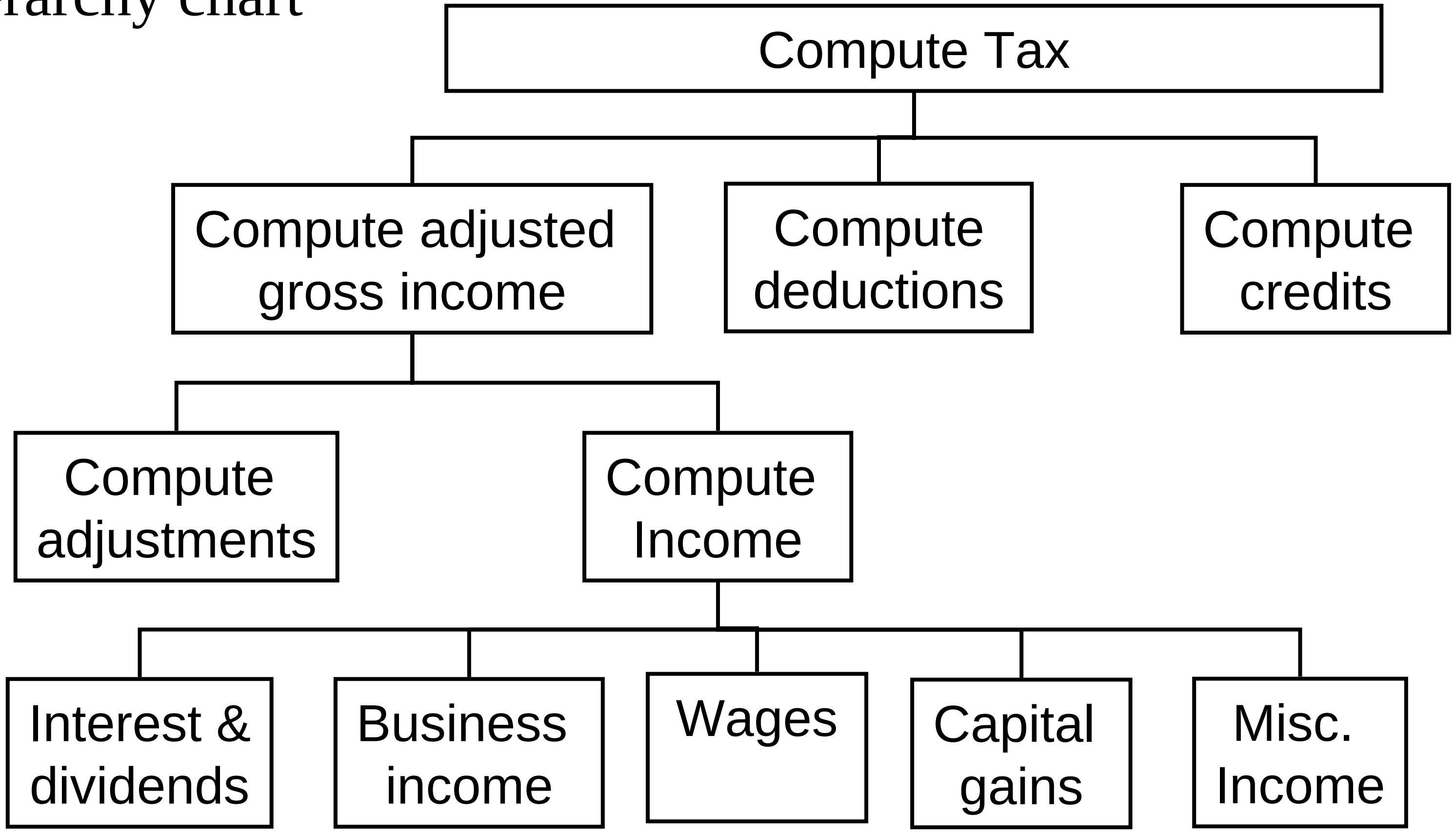
- Today most programmers use a design approach called ***structured programming***
- ***Structured programming*** takes a top-down approach that breaks programs into modular forms
- It uses standard logic tools called control structures
- Developed by Bohm and Jacopini
- Key design considerations
  - Top down design
  - Modular design
  - Structure Theorem

Top-down Design

- Top-down program design proceeds by identifying the function of the program and then breaking it down, in hierarchical fashion, to specify the processing steps, or modules, required to perform that function, down to the lowest level of detail
  - Start with what you want
  - Break it into parts
  - If the parts are non trivial then
    - apply top-down again
  - else
    - Design is done

Example

- Top-down program design can be represented graphically in a hierarchy chart



Modular Design

- ***Modularization*** simplifies program design by allowing the individual program modules to be developed and tested separately
- Groups tasks which perform the same function
- A ***module*** (sometimes called a subprogram or a subroutine) is a self-contained processing step, consisting of logically related program statements
- It is best if each module has only a single function, just as an English paragraph should have a single, complete thought

Structured Theorem

- Advocates of structured programming have shown that any program can be constructed out of three fundamental control structures: sequence, selection, and iteration
- A control structure, or logic structure, controls the logical sequence in which computer program instructions are executed
  - A ***sequence*** control structure is simply a series of procedures that follow one another.
  - The ***selection*** control structure involves a choice: It offers two or more paths to follow at points in the program where a decision must be made
  - An ***iteration*** is an operation that repeats until a certain condition is met
- Eliminates the GOTO (used in unstructured programming)

Structured Programming

- The point of structured programming is to make programs more efficient and better organized (more readable), and to have better notations so that they have clear and correct descriptions
- In structured programming, the program is designed in three mini-steps:
  - Determine the Program Logic, Using a Top-Down Approach
  - Design Details, Using Pseudocode, Flowcharts, and Control Structures
  - Do a Structured Walkthrough

Structured Walkthrough

- In the structured walkthrough, a programmer leads other people in the development team through a design segment
- The ***structured walkthrough***, the final part of the design phase, consists of a formal review process in which others—fellow programmers, systems analysts, and perhaps users
- The team reviews the segment of the program for errors, omissions, and duplications in the processing tasks
- Because the whole program is still on paper at this point, these matters are easier to correct than they will be later

Coding: Code the Program

- Once the design has been developed and reviewed in a walkthrough, the next step is the actual writing of the program, called coding
- ***Coding*** consists of translating the logic requirements from pseudocode or flowcharts into a programming language—the letters, numbers, and symbols arranged according to syntax rules (language rules) that make up the program.
- The program coding stage results in finished source code, which includes enough internal documentation to make the source code understandable and easy to update

Coding: Code the Program

- One of the first steps in the coding process is deciding which programming language to use
- Languages for specific applications are often chosen with respect to such criteria as suitability, compatibility with other applications, organizational standards, programmer availability, portability, and speed
- For a program to work, you have to follow the syntax, the rules of the programming language that specify how words and symbols are put together

Coding: Code the Program

- Many organizations enforce rules called ***coding standards*** to standardize programming styles and make programs more universally readable and easier to maintain
- ***Reusable code*** refers to code segments that can be used over and over again, by several programs
  - The idea behind reusable code is that programs will take less time to write and contain fewer errors if error-free, reusable-code modules can be stitched together to form programs

Testing (1/3)

- Once a program is coded, it must be tested for its correctness
- Program testing involves running various tests
  - **1. Perform Desk-Checking:**
    - Desk-checking is simply reading through, or checking, the program to make sure that it's free of errors and that the logic works.
    - In other words, desk-checking is like proofreading.
    - This step should be taken before the program is actually run on a computer

Testing (2/3)

- **2. Debug the Program:**
  - Debugging means detecting, locating, and removing all errors in a computer program
  - Mistakes may be in syntax errors or logic errors:
    - Syntax errors are caused by typographical errors and incorrect use of the programming language. These are the easiest bugs to fix. Debugging utility programs (sometimes called diagnostics) check program syntax and display syntax-error messages.
    - Logic errors are caused by incorrect use of control structures resulting in incorrect program results.

Debugging Tools

- Many compilers provide the programmer with debugging tools, such as displaying informative error messages indicating the source of many of the errors, colored-coded source code, etc
- Temporary dummy output statements can be used to help identify where program execution goes and display the values of loop counters and other key variables

Testing (3/3)

- **3. Run Real-World Data:**
  - After desk-checking and debugging, the program may run fine—in the laboratory.
  - However, it then needs to be tested with real data, called beta testing
  - It’s mandatory to test with bad data—data that is faulty, incomplete, or in overwhelming quantities—to see if you can make the system crash.
  - The testing process may involve several trials using different test data before the programming team is satisfied the program can be released
- The documentation resulting from this step includes a copy of the finished program code, plus test data and results

Documentation & Maintenance (1/2)

- Preparing documentation is the fifth step in programming.
- The resulting documentation consists of written, graphic, and electronic descriptions of what a program is and how to use it.
- Documentation is needed for everyone who will be involved with the program
  - Preparing User Documentation
  - Prepare Operator Documentation
  - Write Programmer Documentation

Documentation & Maintenance (2/2)

- Virtually every program, if it is to last a long time, requires ongoing maintenance.
- Program maintenance is the process of updating software so that it continues to be useful
- It is a costly process, but can be used to extend the life of a program
- Documentation resulting from this step consists of the amended program package reflecting what problems occurred and what program changes were performed

Tools for Facilitating Program Development

- Program development tools can be used to facilitate the program development process
- An application generator enables both programmers and end users to code new applications quickly
  - Common examples are wizards, macro languages, report generators, form generators, graphics generators, and code generators.

Tools for Facilitating Program Development

- The basic strategy of computer-aided software engineering (CASE) tools is to automate one or more steps of applications software development
  - All CASE tools are different, but many contain such features as action-diagram editors, fourth-generation-language programming, code generators, reusable-code-management routines, and active data dictionaries
- Rapid application development (RAD) refers to a group of tools that enable software development to take place during the entire program development process
  - RAD tools provide CASE-like assistance for developing user interfaces, preparing code for reuse, etc

Programming Paradigms: Procedural Programming

- ***Procedural Programming*** is based upon the concept of the modularity
- A main procedural program is composed of one or more modules. Each module is composed of one or more subprograms.
- Procedural code
  - is easier to read and more maintainable
  - is more flexible
  - facilitates the practice of good program design

Programming Paradigms: Declarative and Object oriented Programming

- ***Declarative programming***
  - describes to the computer a set of conditions and
  - lets the computer figure out how to satisfy them
- ***Object oriented programming***
  - A computer program is composed of a collection of individual units, called objects.
  - Operations are provided for each class of objects.
  - Operations change the state of an object.
  - To make the overall computation happen, the objects interact through their own operations and their own data

Example languages for Different Paradigms

- Procedural
  - C, Pascal, Basic, FORTRAN, COBOL, Ada…
- Declarative
  - LISP, Prolog, …
- Objected oriented
  - Smalltalk, Java, C++, …