The Software Development Process

Step 1: Algorithm Design

Developing the underlying logic of the program

Step 2: Program Composition

Writing the program in computer language

Step 3: Debugging and Testing

Ensuring that the program is error-free and reliable

Step 4: Documentation

Making the program easy to use and understand

Step 5: Storage and Maintenance

Saving the program and improving in light of experience

Modular Design

- Break large problems into smaller, logical subproblems (modules, subroutines or M-files) that can be developed and tested independently
- Modules should be as independent and self contained as possible
- A calling or main program invokes these modules as needed. The main program orchestrates each of the parts in a logical fashion
- Modular design makes it easier to debug and test a program since errors can be isolated
- Program maintenance and modification are facilitated

Top Down Design

- Systematic development process that starts with the most general objectives and successively divides into more detail
- Identifies well defined modules

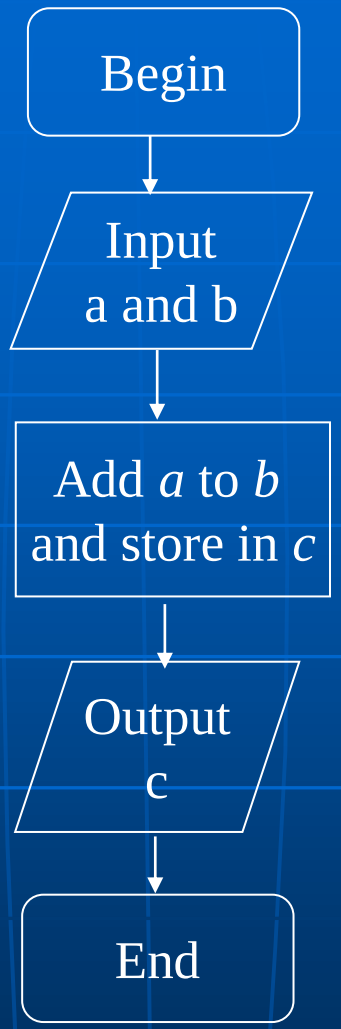
Structured Programming

- Deals with how the code is developed so that it is easy to understand, correct and modify
- Set of rules that prescribe good style habits

Algorithm Design

An algorithm is a sequence of logical steps required to solve a problem. *Flowcharts* and *psuedocode* are used for algorithm development.

1. Flowchart
   - Visual or graphical representation of an algorithm
2. Pseudocode
   - A way to express an algorithm that bridges the gap between flowcharts and computer code
   - Uses codelike statements in place of graphical symbols
   - Looks more like a computer program than a flowchart, thus it is easier to develop a computer program with it and share with others

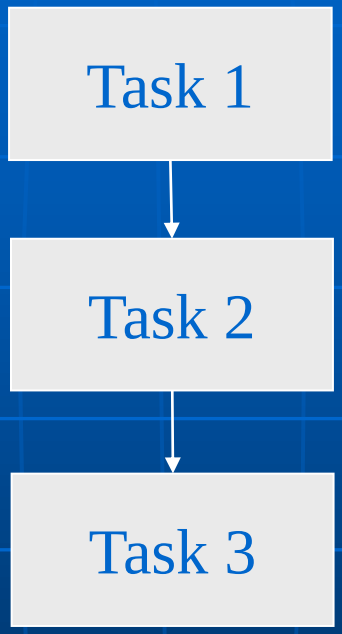
Algorithm	Flowchart	Pseudocode
Step 1: Begin Step 2: Input variables *a* and *b* Step 3: Add the variables Step 4: Output the result Step 5: End		BEGIN *Adder* INPUT *a* and *b* *c* = *a* + *b* PRINT *c* END *Adder*

Fundamental Control Structures

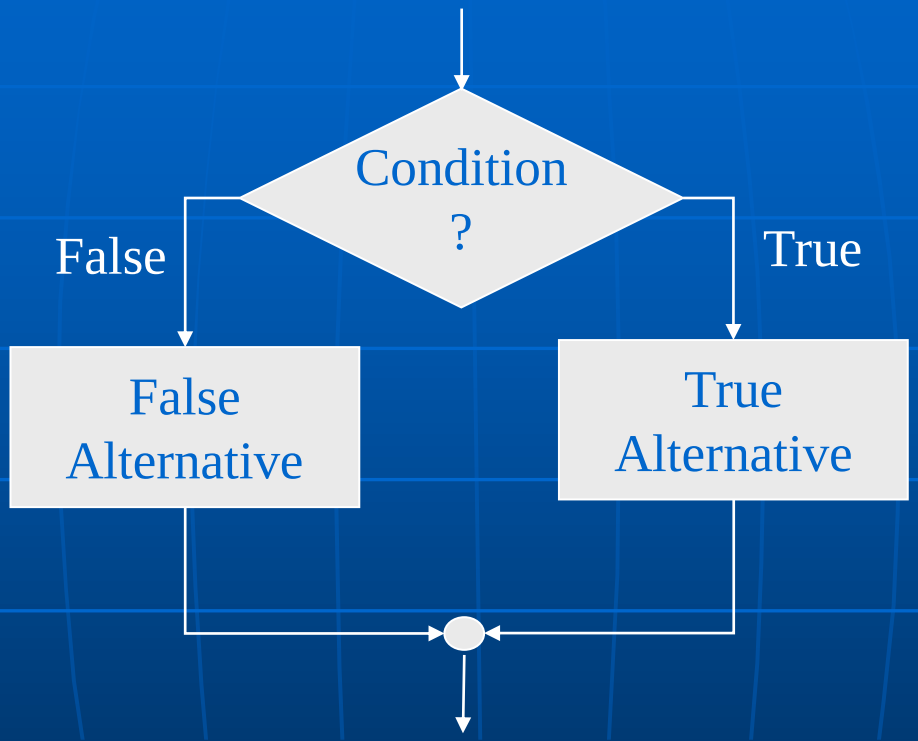
- Sequence
- Selection (IF statement)
- Repetition (FOR or WHILE statements)

Any numerical algorithm can be developed from these 3 fundamental control structures

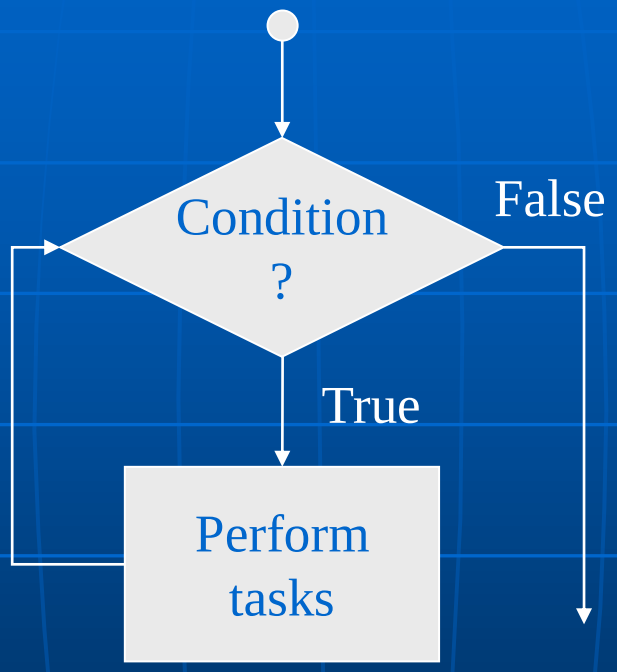
Sequence



Selection



Repetition



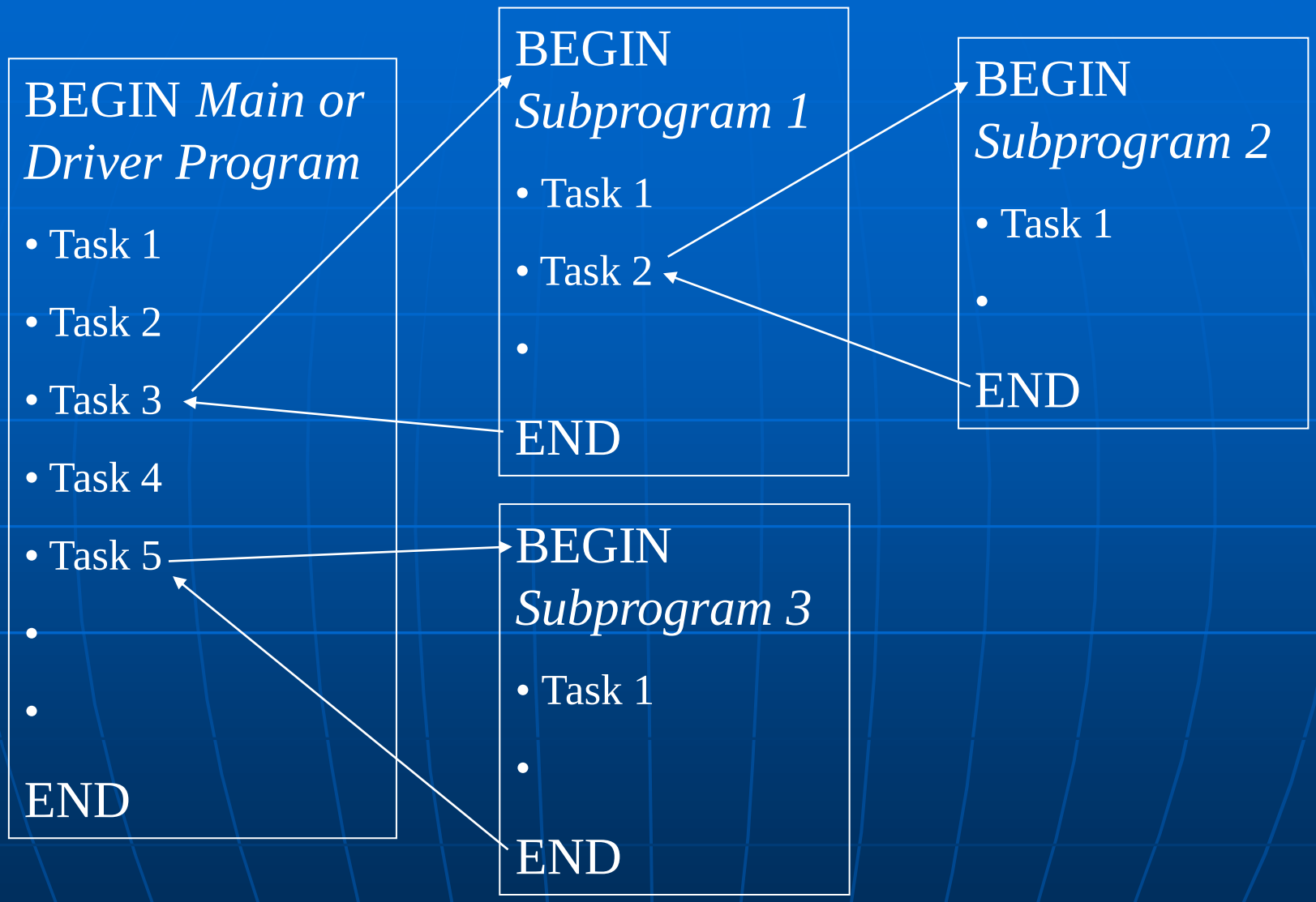
Structured Programming

A exact definition is difficult, however the major idea is embodied in the following *structure principle* :

"The static structure (that is, spread out on the page) of the program should correspond in a simple way to the dynamic structure (that is, spread out in time) of the corresponding computation."

Some General Guidelines for Structured Programming:

- Programs should consist solely of the 3 fundamental control structures
- Each of the structures should have only one entrance and exit
- Unconditional transfers (GO TO's) should be avoided
- The structures should be clearly identified with comments and visual devices such as indentation, blank lines and blank spaces
- Avoid "spaghetti like" code with indiscriminate branching


BEGIN *Main or Driver Program*
• Task 1
• Task 2
• Task 3
• Task 4
• Task 5
•
•
END
BEGIN *Subprogram 1*
• Task 1
• Task 2
•
END
BEGIN *Subprogram 2*
• Task 1
•
END
BEGIN *Subprogram 3*
• Task 1
•
END

Quality Control

Extensive debugging and testing is required.

Errors or “Bugs”

- Syntax errors
- Link or build errors
- Run-time errors
- Logic errors

Testing should proceed in phases:

- Module tests
- Development tests
- Whole system tests
- Operational tests

Documentation

- After debugging and testing , a program must be documented
- Internal documentation:
  titles, headings, sections and descriptive comments inside the program
- External documentation:
  output messages generated when the program is run to make the program more professional and use-friendly

Maintenance

- Upgrading in the light of experience
- Ensuring that the program is safely stored

Computer Languages

- Visual Basic

  Developed at Dartmouth as an instructional language for students in 1960's

- Fortran

  *For*mula *Tran*slation
  Developed by IBM in 1957

- C, C++

  Developed at Bell Labs in 1972

- MatLab

- Mathematica

Interesting Tidbit

Errors or “Bugs”

- Term coined by Admiral Grace Hopper -- a pioneer of computer languages. In 1945, she was working with an early electromagnetic computer when it went dead. She opened it up and found a moth stuck in one of the relays -- the computer had a “bug”. She removed the moth -- “debugged” the machine -- and it worked fine.