Introduction to Programming

Subodh Sharma
svs@cse.iitd.ac.in
https://subodhvsharma.github.io

IIT Delhi, Computer Science Department

1 Course Logistics

2 Introduction to Computing

- What is computing?
- Computation in STEM and Humanities
- Example of a Computation
- Understanding the Computational Process

3 Computers, Languages, Algorithms

Course Logistics

- All **content-specific discussions on Piazza**. You will be added to COL1000's Piazza. No individual emails will be entertained.
- **Remember to check after every class/lab**: https://moodlenew.iitd.ac.in
- For personal (non-content related) queries, you can send an email to col1000admin@cse.iitd.ac.in

What is computing?

- Computing is a **process** of counting or performing calculation.

What is computing?

- Computing is a **process** of counting or performing calculation.
- Computing **technology** may include various tools such as: sticks & stones, paper & pencil, abacus, straight edge & compass, calculator, computer

What is computing?

- Computing is a **process** of counting or performing calculation.
- Computing **technology** may include various tools such as: sticks & stones, paper & pencil, abacus, straight edge & compass, calculator, computer
  - History of computing is *older* than the history of computing technology

What is computing?

- Computing is a **process** of counting or performing calculation.
- Computing **technology** may include various tools such as: sticks & stones, paper & pencil, abacus, straight edge & compass, calculator, computer
  - History of computing is *older* than the history of computing technology
    - One of the oldest algorithms – Euclid's method to compute gcd

Computation in STEM and Humanities

- **Mechanical/Applied Mechanics**: Autonomous vehicles, 3D printing

Computation in STEM and Humanities

- **Mechanical/Applied Mechanics**: Autonomous vehicles, 3D printing
- **Civil/Material**: Structural modeling & analysis, construction methods, Computational material discovery

Computation in STEM and Humanities

- **Mechanical/Applied Mechanics**: Autonomous vehicles, 3D printing
- **Civil/Material**: Structural modeling & analysis, construction methods, Computational material discovery
- **Chemical/Chemistry/Biochecmical/Biotech**: Computational modeling of *Transport phenomena*: Momentum, energy and mass transfer as unit operations.

Computation in STEM and Humanities

- **Mechanical/Applied Mechanics**: Autonomous vehicles, 3D printing
- **Civil/Material**: Structural modeling & analysis, construction methods, Computational material discovery
- **Chemical/Chemistry/Biochecmical/Biotech**: Computational modeling of *Transport phenomena*: Momentum, energy and mass transfer as unit operations.
- **CSE/Electrical**: AI-based coding (Eg:CoPilot), IoT, 5G, etc.

Computation in STEM and Humanities

- **Mechanical/Applied Mechanics**: Autonomous vehicles, 3D printing
- **Civil/Material**: Structural modeling & analysis, construction methods, Computational material discovery
- **Chemical/Chemistry/Biochecmical/Biotech**: Computational modeling of *Transport phenomena*: Momentum, energy and mass transfer as unit operations.
- **CSE/Electrical**: AI-based coding (Eg:CoPilot), IoT, 5G, etc.
- **Energy Science**: HPC for efficient tapping of renewable energy, sustainable Fusion(?), etc.

Computation in STEM and Humanities

- **Mechanical/Applied Mechanics**: Autonomous vehicles, 3D printing
- **Civil/Material**: Structural modeling & analysis, construction methods, Computational material discovery
- **Chemical/Chemistry/Biochecmical/Biotech**: Computational modeling of *Transport phenomena*: Momentum, energy and mass transfer as unit operations.
- **CSE/Electrical**: AI-based coding (Eg:CoPilot), IoT, 5G, etc.
- **Energy Science**: HPC for efficient tapping of renewable energy, sustainable Fusion(?), etc.
- **Textile**: wearable electronic textiles – information gathering garments, Smart textiles responding to the enviroment, etc.

Computation in STEM and Humanities

- **Mechanical/Applied Mechanics**: Autonomous vehicles, 3D printing
- **Civil/Material**: Structural modeling & analysis, construction methods, Computational material discovery
- **Chemical/Chemistry/Biochecmical/Biotech**: Computational modeling of *Transport phenomena*: Momentum, energy and mass transfer as unit operations.
- **CSE/Electrical**: AI-based coding (Eg:CoPilot), IoT, 5G, etc.
- **Energy Science**: HPC for efficient tapping of renewable energy, sustainable Fusion(?), etc.
- **Textile**: wearable electronic textiles – information gathering garments, Smart textiles responding to the enviroment, etc.
- **Humanities/Management**: Linguistics, Cognitive science, Politics, etc.

⋮

Example of a Computation: The Computation Tool

- Pick a tool for computation: Straight-edge and Compass

Example of a Computation: The Computation Tool

- Pick a tool for computation: Straight-edge and Compass
  - **Straight-edge**: It is unmarked! Therefore, **cannot** specify lengths, but can specify lines rays and line segments.

Example of a Computation: The Computation Tool

- Pick a tool for computation: Straight-edge and Compass
  - **Straight-edge**: It is unmarked! Therefore, **cannot** specify lengths, but can specify lines rays and line segments.
  - **Compass**: Can define arcs and circles; Can specify **arbitrary non-zero** lengths.

Example of a Computation: The Problem

- Doubling a Square: Given a square ABCD of side $a > 0$

Example of a Computation: The Problem

- Doubling a Square: Given a square ABCD of side $a > 0$
- Computation steps:

Example of a Computation: The Problem

- Doubling a Square: Given a square ABCD of side $a > 0$
- Computation steps:
  1. Draw a diagonal $\overline{AC}$.

Example of a Computation: The Problem

- Doubling a Square: Given a square ABCD of side $a > 0$
- Computation steps:
  1. Draw a diagonal $\overline{AC}$.
  2. Complete the square ACEF.

Example of a Computation: The Problem

- Doubling a Square: Given a square ABCD of side $a > 0$
- Computation steps:
  1. Draw a diagonal $\overline{AC}$.
  2. Complete the square ACEF.
- Step (1) above is a primitive operation.

Example of a Computation: The Problem

- Doubling a Square: Given a square ABCD of side $a > 0$
- Computation steps:
  1. Draw a diagonal $\overline{AC}$.
  2. Complete the square ACEF.
- Step (1) above is a primitive operation.
- However step (2) is a complex operation that requires further computation (called the *refinement* of the computational process).

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.
  1. Construct two lines l_1 and l_2 perpendicular to $\overline{PQ}$ passing through P and Q, respectively.

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.
  1. Construct two lines l_1 and l_2 perpendicular to $\overline{PQ}$ passing through P and Q, respectively.
  2. On the same side of $\overline{PQ}$ mark points R on l_1 and S on l_2 such that $PR = PQ = QS$.

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.
  1. Construct two lines l_1 and l_2 perpendicular to $\overline{PQ}$ passing through P and Q, respectively.
  2. On the same side of $\overline{PQ}$ mark points R on l_1 and S on l_2 such that $PR = PQ = QS$.
  3. Draw $\overline{RS}$. Claim: $PQSR$ is of side b.

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.
  1. Construct two lines l_1 and l_2 perpendicular to $\overline{PQ}$ passing through P and Q, respectively.
  2. On the same side of $\overline{PQ}$ mark points R on l_1 and S on l_2 such that $PR = PQ = QS$.
  3. Draw $\overline{RS}$. Claim: $PQSR$ is of side b.
- This time step (1) is complex and step (2) is primitive.

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.
  1. Construct two lines l_1 and l_2 perpendicular to $\overline{PQ}$ passing through P and Q, respectively.
  2. On the same side of $\overline{PQ}$ mark points R on l_1 and S on l_2 such that $PR = PQ = QS$.
  3. Draw $\overline{RS}$. Claim: $PQSR$ is of side b.
- This time step (1) is complex and step (2) is primitive.
- Perpendiculars: Subsequent refinement of step (1)

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.
  1. Construct two lines l_1 and l_2 perpendicular to $\overline{PQ}$ passing through P and Q, respectively.
  2. On the same side of $\overline{PQ}$ mark points R on l_1 and S on l_2 such that $PR = PQ = QS$.
  3. Draw $\overline{RS}$. Claim: $PQSR$ is of side b.
- This time step (1) is complex and step (2) is primitive.
- Perpendiculars: Subsequent refinement of step (1)
  1. Choose a length $c > 0$.

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.
  1. Construct two lines l_1 and l_2 perpendicular to $\overline{PQ}$ passing through P and Q, respectively.
  2. On the same side of $\overline{PQ}$ mark points R on l_1 and S on l_2 such that $PR = PQ = QS$.
  3. Draw $\overline{RS}$. Claim: $PQSR$ is of side b.
- This time step (1) is complex and step (2) is primitive.
- Perpendiculars: Subsequent refinement of step (1)
  1. Choose a length $c > 0$.
  2. With P as a centre mark off arcs Y and Z on either side of P.

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.
  1. Construct two lines l_1 and l_2 perpendicular to $\overline{PQ}$ passing through P and Q, respectively.
  2. On the same side of $\overline{PQ}$ mark points R on l_1 and S on l_2 such that $PR = PQ = QS$.
  3. Draw $\overline{RS}$. Claim: $PQSR$ is of side b.
- This time step (1) is complex and step (2) is primitive.
- Perpendiculars: Subsequent refinement of step (1)
  1. Choose a length $c > 0$.
  2. With P as a centre mark off arcs Y and Z on either side of P.
  3. Draw circles of radius $2c$ from centre points Y and Z.

Example of a Computation: Refinement

- Square: Given a line segment $\overline{PQ}$, s.t. $|\overline{PQ}| = b$, construct a square of length b.
  1. Construct two lines l_1 and l_2 perpendicular to $\overline{PQ}$ passing through P and Q, respectively.
  2. On the same side of $\overline{PQ}$ mark points R on l_1 and S on l_2 such that $PR = PQ = QS$.
  3. Draw $\overline{RS}$. Claim: $PQSR$ is of side b.
- This time step (1) is complex and step (2) is primitive.
- Perpendiculars: Subsequent refinement of step (1)
  1. Choose a length $c > 0$.
  2. With P as a centre mark off arcs Y and Z on either side of P.
  3. Draw circles of radius $2c$ from centre points Y and Z.
  4. Join the points of intersection of the two circles.

Example of a Computation: Correctness?

- Diagonal $\overline{AC}$ length = $\sqrt{2}a$

Example of a Computation: Correctness?

- Diagonal $\overline{AC}$ length = $\sqrt{2}a$
- Area of ACEF = $2a^2$

Example of a Computation: Correctness?

- Diagonal $\overline{AC}$ length = $\sqrt{2}a$
- Area of ACEF = $2a^2$
- Where the two circles drawn from Y and Z of radius $2c$ is perpendicular to YZ.

Understanding the Computational Process: Essential Ingredients

- **Primitive operations & expressions**: These represent the simplest objects of the computational process. Eg: Drawing a line, drawing an arc etc.

Understanding the Computational Process: Essential Ingredients

- **Primitive operations & expressions**: These represent the simplest objects of the computational process. Eg: Drawing a line, drawing an arc etc.
- **Methods of combination**: This specifies how primitive expressions and objects can be combined to form compound expressions and objects. Eg: Drawing a perpendicular.

Understanding the Computational Process: Essential Ingredients

- **Primitive operations & expressions**: These represent the simplest objects of the computational process. Eg: Drawing a line, drawing an arc etc.
- **Methods of combination**: This specifies how primitive expressions and objects can be combined to form compound expressions and objects. Eg: Drawing a perpendicular.
- **Methods of abstraction**: Naming compound objects be named and used/manipulated as a unit. Eg: square-construction process from the diagonal of a given square. It is useful in:

Understanding the Computational Process: Essential Ingredients

- **Primitive operations & expressions**: These represent the simplest objects of the computational process. Eg: Drawing a line, drawing an arc etc.
- **Methods of combination**: This specifies how primitive expressions and objects can be combined to form compound expressions and objects. Eg: Drawing a perpendicular.
- **Methods of abstraction**: Naming compound objects be named and used/manipulated as a unit. Eg: square-construction process from the diagonal of a given square. It is useful in:
  1. separating logical subproblems. Eg: drawing a perpendicular from a point is logically separate from drawing a square on a line segment.

Understanding the Computational Process: Essential Ingredients

- **Primitive operations & expressions**: These represent the simplest objects of the computational process. Eg: Drawing a line, drawing an arc etc.
- **Methods of combination**: This specifies how primitive expressions and objects can be combined to form compound expressions and objects. Eg: Drawing a perpendicular.
- **Methods of abstraction**: Naming compound objects be named and used/manipulated as a unit. Eg: square-construction process from the diagonal of a given square. It is useful in:
  1. separating logical subproblems. Eg: drawing a perpendicular from a point is logically separate from drawing a square on a line segment.
  2. Avoiding repetitions in specifying solutions. Eg: drawing perpendiculars from two separate points are instances of the same computational process.

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.
- **Algorithm**: It is a finite sequence of well-defined instructions (Eg: combination of primitives of a computation tool) to solve a problem.

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.
- **Algorithm**: It is a finite sequence of well-defined instructions (Eg: combination of primitives of a computation tool) to solve a problem.
  1. It works with a definite input and output

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.
- **Algorithm**: It is a finite sequence of well-defined instructions (Eg: combination of primitives of a computation tool) to solve a problem.
  1. It works with a definite input and output
  2. It is unambiguous (Eg: How would one evaluate $E_1 + E_2 * E_3$)

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.
- **Algorithm**: It is a finite sequence of well-defined instructions (Eg: combination of primitives of a computation tool) to solve a problem.
  1. It works with a definite input and output
  2. It is unambiguous (Eg: How would one evaluate $E_1 + E_2 * E_3$)
  3. The number of steps executed to arrive at a solution is finite

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.
- **Algorithm**: It is a finite sequence of well-defined instructions (Eg: combination of primitives of a computation tool) to solve a problem.
  1. It works with a definite input and output
  2. It is unambiguous (Eg: How would one evaluate $E_1 + E_2 * E_3$)
  3. The number of steps executed to arrive at a solution is finite
- **Programming Language**: It is a vocabulary (with a syntax – also called the grammar of the language), which is used to

NOTE: The "form" is usually a Program. The program is developed by conforming to the grammatical rules of the language.

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.
- **Algorithm**: It is a finite sequence of well-defined instructions (Eg: combination of primitives of a computation tool) to solve a problem.
  1. It works with a definite input and output
  2. It is unambiguous (Eg: How would one evaluate $E_1 + E_2 * E_3$)
  3. The number of steps executed to arrive at a solution is finite
- **Programming Language**: It is a vocabulary (with a syntax – also called the grammar of the language), which is used to
  1. Translate the algorithm into a 'form"

NOTE: The "form" is usually a Program. The program is developed by conforming to the grammatical rules of the language.

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.
- **Algorithm**: It is a finite sequence of well-defined instructions (Eg: combination of primitives of a computation tool) to solve a problem.
  1. It works with a definite input and output
  2. It is unambiguous (Eg: How would one evaluate $E_1 + E_2 * E_3$)
  3. The number of steps executed to arrive at a solution is finite
- **Programming Language**: It is a vocabulary (with a syntax – also called the grammar of the language), which is used to
  1. Translate the algorithm into a 'form"
  2. Communicate with the computation machine

  NOTE: The "form" is usually a Program. The program is developed by conforming to the grammatical rules of the language.

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.
- **Algorithm**: It is a finite sequence of well-defined instructions (Eg: combination of primitives of a computation tool) to solve a problem.
  1. It works with a definite input and output
  2. It is unambiguous (Eg: How would one evaluate $E_1 + E_2 * E_3$)
  3. The number of steps executed to arrive at a solution is finite
- **Programming Language**: It is a vocabulary (with a syntax – also called the grammar of the language), which is used to
  1. Translate the algorithm into a 'form"
  2. Communicate with the computation machine

  NOTE: The "form" is usually a Program. The program is developed by conforming to the grammatical rules of the language.
- **Computing Tool**: A programming langauage and the computer together form a computing tool.

Computers, Languages, Algorithms

- **Computer**: Yet another tool for performing computation.
- **Algorithm**: It is a finite sequence of well-defined instructions (Eg: combination of primitives of a computation tool) to solve a problem.
  1. It works with a definite input and output
  2. It is unambiguous (Eg: How would one evaluate $E_1 + E_2 * E_3$)
  3. The number of steps executed to arrive at a solution is finite
- **Programming Language**: It is a vocabulary (with a syntax – also called the grammar of the language), which is used to
  1. Translate the algorithm into a 'form"
  2. Communicate with the computation machine

  NOTE: The "form" is usually a Program. The program is developed by conforming to the grammatical rules of the language.
- **Computing Tool**: A programming langauage and the computer together form a computing tool.
- Thus, each program uses *only* the primitives of the computing tool.