Algorithm

A description of an operation as a sequence of simpler operations.
Examples

• Recipes in a cookbook

• Directions for getting from one place to another

• Long division
Flowchart

1. Read a number a
2. Read a number b
3. Calculate \(c = a + b\)
4. Output c
Flowchart

Memory:

\[ a = 3 \]

1. Read a number \( a \)
2. Read a number \( b \)
3. Calculate \( c = a + b \)
4. Output \( c \)
Flowchart

Read a number a

Read a number b

Calculate c = a + b

Output c

Memory:

a = 3
b = 5
Flowchart

Read a number a

Read a number b

Calculate c = a + b

Output c

Memory:

a = 3
b = 5
c = 8
Flowchart

Memory:

a = 3
b = 5
c = 8

Output

8

Read a number a

Read a number b

Calculate c = a + b

Output c
Decision

Read a number a

Is a > 10?

No
Output "Small..."

Yes
Output "Large!"
Memory:
\[ a = 8 \]

Decision

Read a number \( a \)

Is \( a > 10 \)?

No → Output "Small..."

Yes → Output "Large!"
Memory:

\[ a = 8 \]

**Decision**

1. **Read a number** \( a \)
2. **Is** \( a > 10 \)?
   - **No**: Output "Small..."
   - **Yes**: Output "Large!"
Memory:

\[ a = 8 \]

Read a number \( a \)

Is \( a > 10? \)

No

Output "Small..."

Yes

Output "Large!"

Output

Small...
Loop

Read a number a

Is a > 0?

No

Output "The end"

Yes

Decrease a by 1

Output a
Loop

Memory:  
a = 3

Read a number a  

Is a > 0?  

No  
Output "The end"

Yes  
Output a  
Decrease a by 1
Loop

Memory:
\[ a = 3 \]

1. Read a number a
2. Is a > 0?
   - No: Output "The end"
   - Yes: Decrease a by 1

3. Output a
Loop

Memory:
\[ a = 3 \]

Read a number \( a \)

Is \( a > 0? \)

No

Output "The end"

Yes

Decrease \( a \) by 1

Output \( a \)
Loop

Memory:

\( a = 2 \)

Read a number \( a \)

Is \( a > 0? \)

No

Output "The end"

Yes

Output a

Decrease a by 1

Output

3
Loop

Memory:

\[ a = 2 \]

1. Read a number \( a \)
2. Is \( a > 0 \)?
   - No: Output "The end"
   - Yes: Decrease \( a \) by 1, then output \( a \)

Output:

3
Memory: 
\[ a = 2 \]

Loop

1. Read a number \( a \)
2. Is \( a > 0 \)?
   - No: Output "The end"
   - Yes: Decrease \( a \) by 1 and go back to step 1
3. Output \( a \)

Example:
1. \( a = 2 \)
2. \( a > 0 \): Decrease \( a \) by 1
3. \( a = 1 \)
4. \( a > 0 \): Decrease \( a \) by 1
5. \( a = 0 \)
6. \( a > 0 \): Decrease \( a \) by 1
7. \( a < 0 \): Output "The end"
Loop

Memory:

\[ a = 1 \]

Read a number \( a \)

Is \( a > 0 \)?

Yes

Output \( a \)

Decrease \( a \) by 1

No

Output "The end"
Loop

Read a number a

Is a > 0?

No
Output "The end"

Yes
Output a
Decrease a by 1

Memory:
a = 1

Output
3
2
Loop

Memory:
\( a = 1 \)

Read a number \( a \)

Is \( a > 0 \)?

No

Output "The end"

Yes

Output \( a \)

Decrease \( a \) by 1

Output

3
2
1
Loop

Memory:
a = 0

Read a number a

Is a > 0?

No → Output "The end"

Yes → Output a

Decrease a by 1

Output:

3 2 1
Loop

Memory:
a = 0

Read a number a

Is a > 0?

No
Output "The end"

Yes
Output a

Decrease a by 1
Loop

Memory:
\[ a = 0 \]

Read a number \( a \)

Is \( a > 0 \)?

No

Output "The end"

Yes

Output \( a \)

Decrease \( a \) by 1
Higher or lower

► Player A chooses a number.

► Player B has to guess the number that player A chose.

► The game involves repeated rounds. In each round:

  ► Player B guesses a number.

  ► If the number player B guessed is equal to the number A chose, the game ends.

  ► Otherwise, if the number A chose is lower than the guess, player A says “Lower”. If the number A chose is higher than the guess, player A says “Higher”.

Higher or lower – simple algorithm

Output "Got it!"

Is response = "Higher"?

Yes

guess ← guess + 1

Output guess

No

guess ← 1

Read response string
Higher or lower – better algorithm

low ← 1
hi ← 100
Higher or lower – better algorithm

low ← 1
hi ← 100

guess ← (hi + low) / 2

Output guess

Read response string
Higher or lower – better algorithm

low ← 1
hi ← 100

guess ← (hi + low) : 2

Output guess

Is response = "Higher"?

Read response string
Higher or lower – better algorithm

low ← 1
hi ← 100

guess ← (hi + low) ÷ 2

low ← guess + 1

Is response = "Higher"?
Yes
No

Output guess
Read response string
Higher or lower – better algorithm

- **low** ← 1
- **hi** ← 100
- **guess** ← \((hi + low) : 2\)

Output **guess**

- **low** ← **guess** + 1

**Is response = "Lower"?**

- Yes
  - **low** ← **guess** + 1

  **Is response = "Higher"?**
  - Yes
  - Output **guess**

  **Is response = "Lower"?**
  - Yes
  - **low** ← 1
  - **hi** ← 100

  **guess** ← \((hi + low) : 2\)
  - No
  - **low** ← 1
  - **hi** ← 100

  **guess** ← \((hi + low) : 2\)
Higher or lower – better algorithm

1. **initialization**
   - hi ← 100
   - low ← 1

2. **guess calculation**
   - guess ← (hi + low) : 2

3. **response handling**
   - Read response string
   - Is response = "Higher"?
     - Yes: hi ← guess - 1
     - No: low ← guess + 1
   - Is response = "Lower"?
     - Yes: low ← 1
     - No: hi ← 100

4. **output**
   - Output guess

---

**Flowchart:**

- **Node:** Is response = "Lower"?
  - **Flow:** Yes → hi ← guess - 1
  - **Flow:** No → low ← guess + 1
- **Node:** Is response = "Higher"?
  - **Flow:** Yes
  - **Flow:** No
- **Node:** low ← 1
  - **Flow:** hi ← 100
- **Node:** Output guess
  - **Flow:** Read response string
higher or lower – better algorithm

Output "Got it!"

No

Is response = "Lower"?

Yes

hi ← guess - 1

No

low ← guess + 1

Yes

Is response = "Higher"?

No

low ← 1
hi ← 100

guess ← (hi + low) : 2

Output guess

Read response string
Further study

Introductory programming exercises:
http://code.org