Program Design

A ***program*** is a sequence of computer instructions that perform some function.

The program is designed to implement an ***algorithm***.

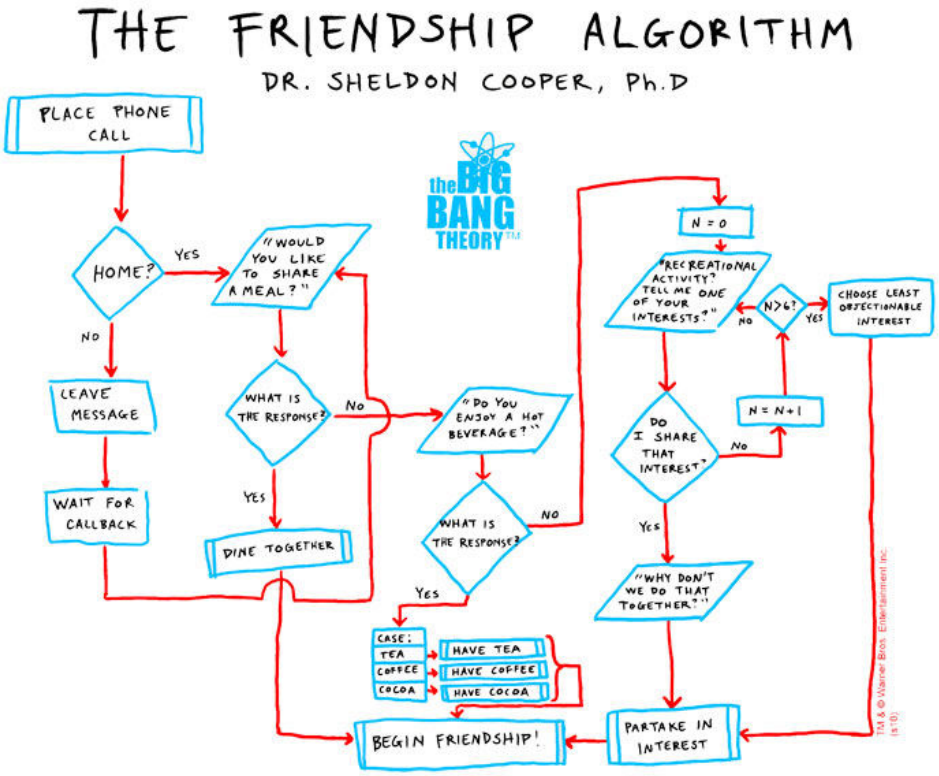
An ***algorithm*** is a procedure consisting of a finite-set of well-defined steps, each step usually consists of one ***instruction***.

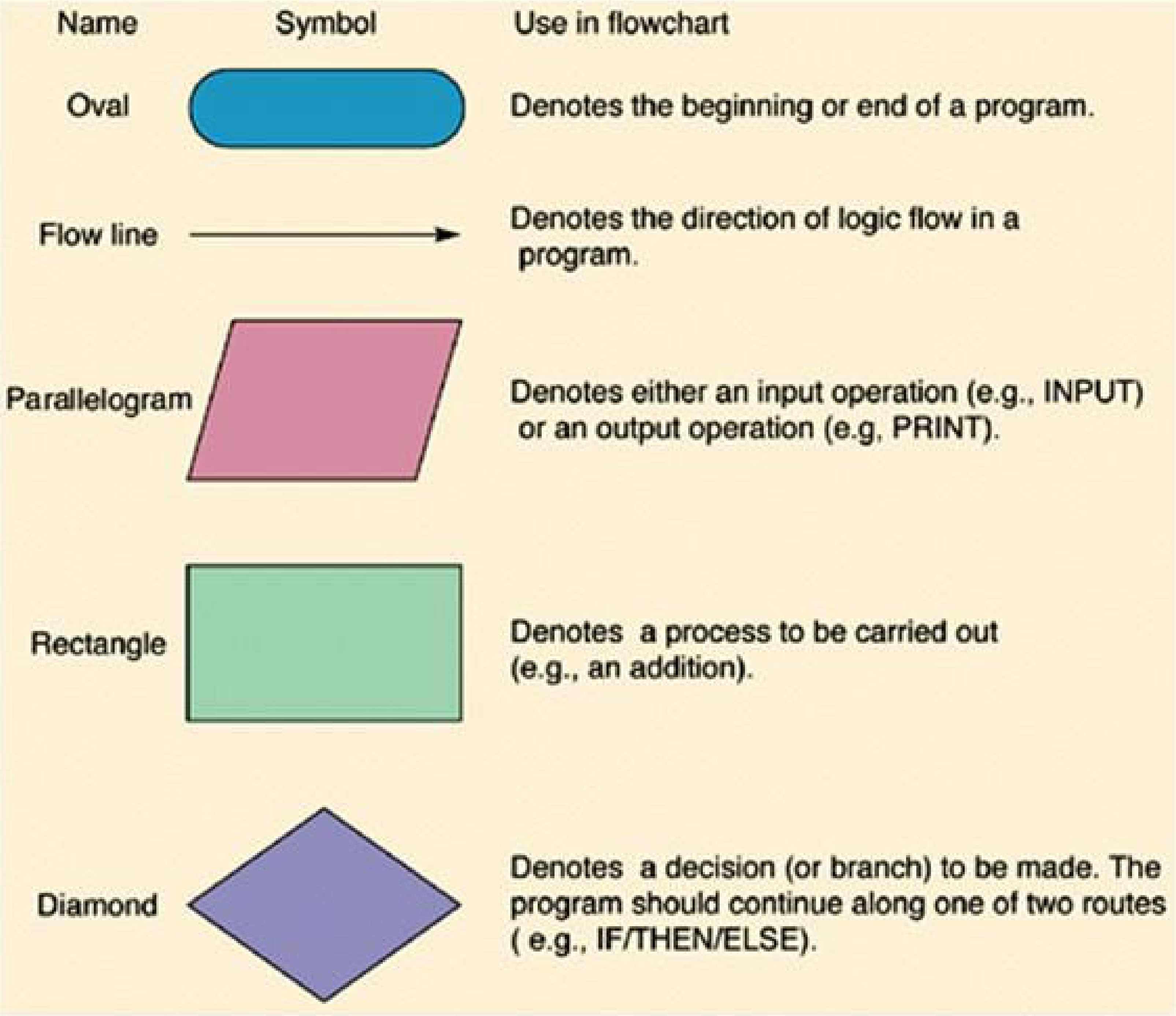
The original “human readable” instructions are known as ***source code statements***.

Source code is translated into machine-readable object code, an ***executable program*** is produced.

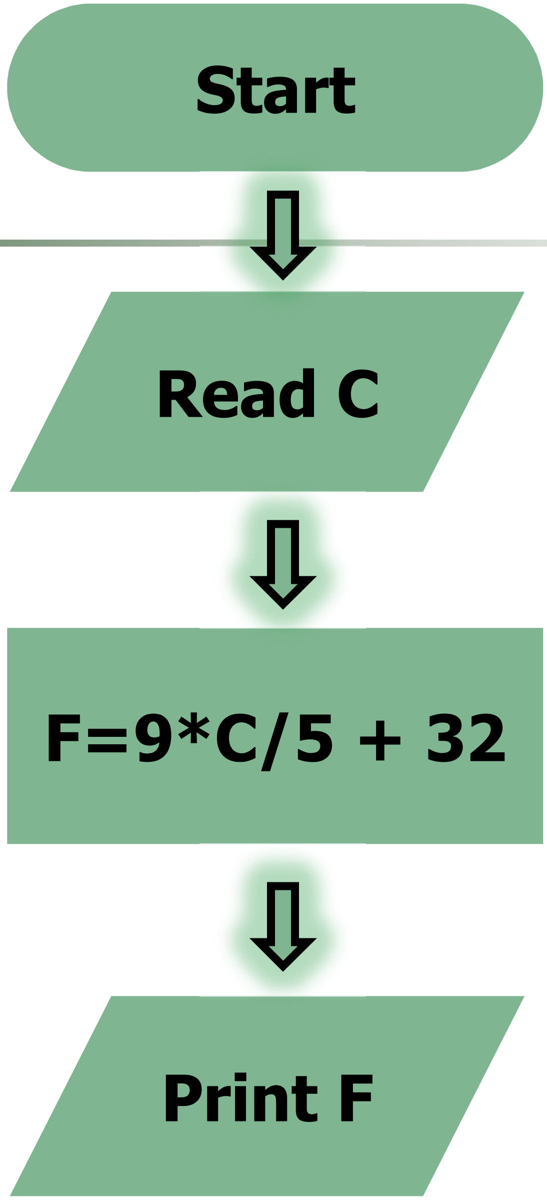
Flowcharts

A flowchart is a ***graphic representation of an algorithm***, often used in the design phase of programming to work out the logical flow of a program.





Name	Symbol	Use in flowchart
Oval	⬭	Denotes the beginning or end of a program.
Flow line	→	Denotes the direction of logic flow in a program.
Parallelogram	▱	Denotes either an input operation (e.g., INPUT) or an output operation (e.g, PRINT).
Rectangle	▭	Denotes a process to be carried out (e.g., an addition).
Diamond	◇	Denotes a decision (or branch) to be made. The program should continue along one of two routes (e.g., IF/THEN/ELSE).



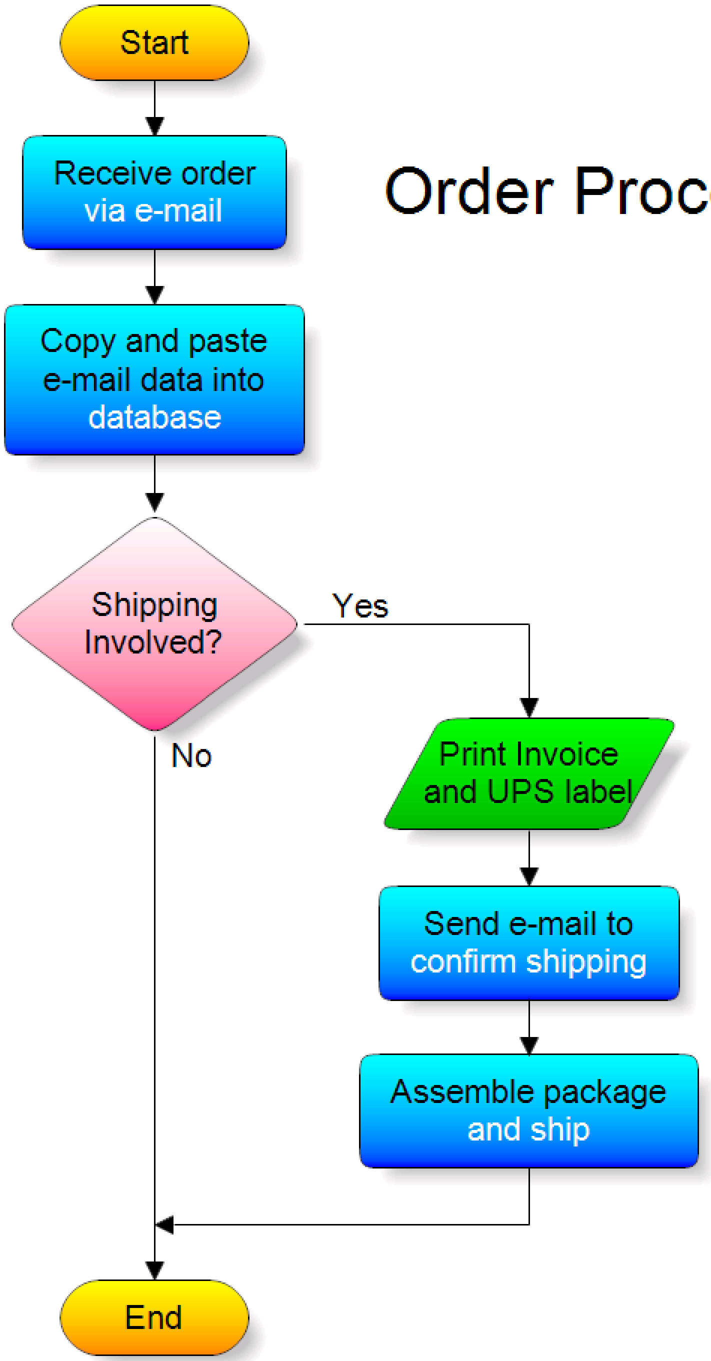
Start

Read C

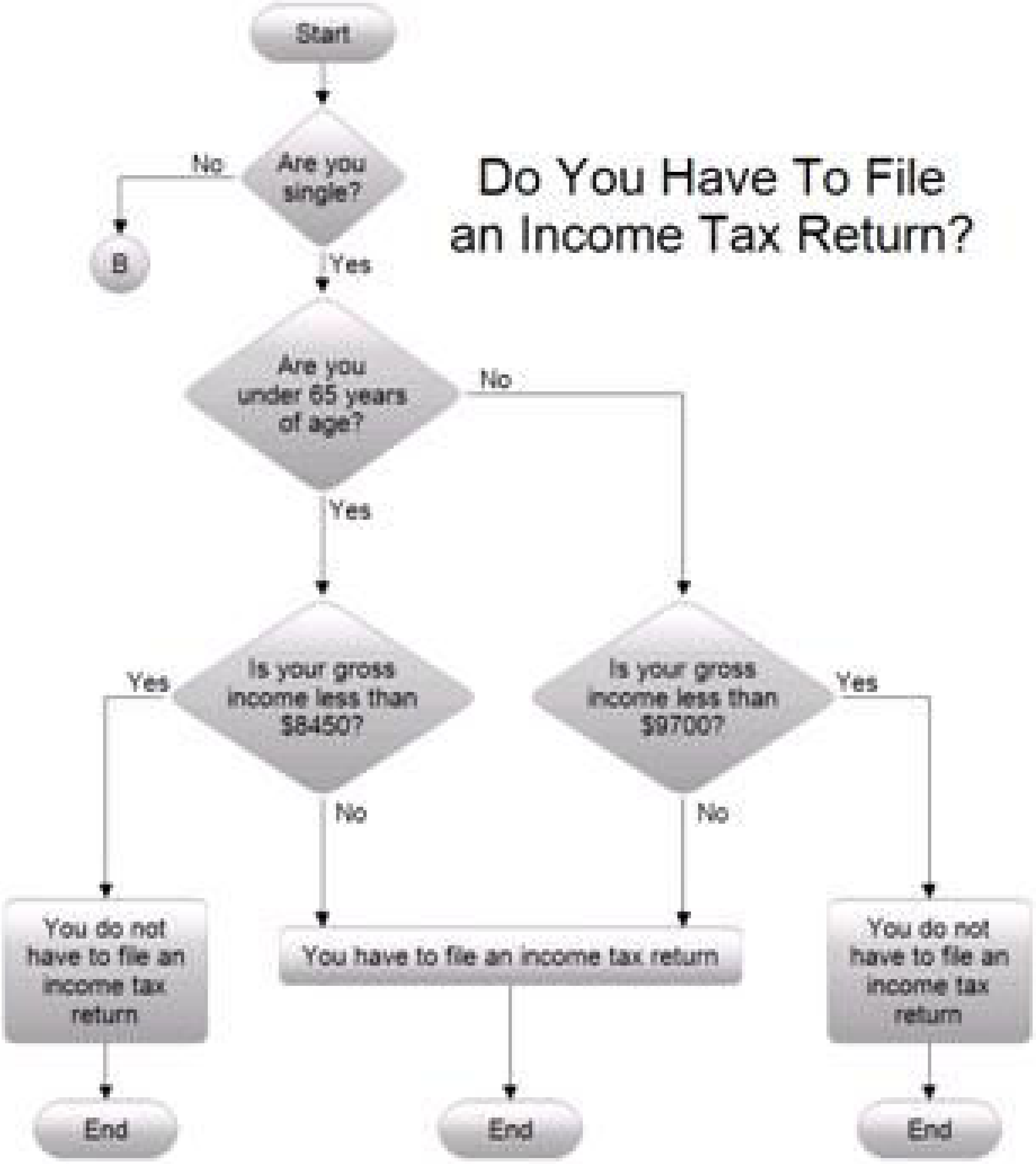
F=9*C/5 + 32

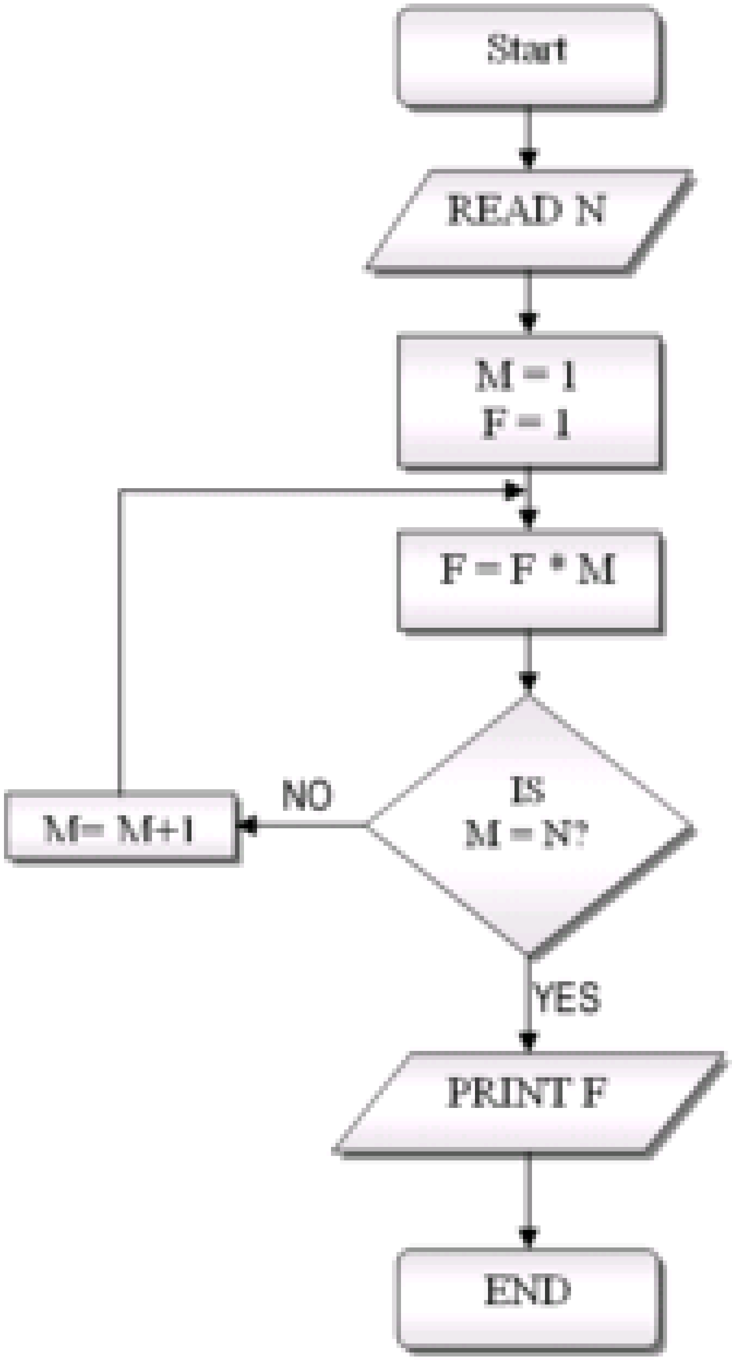
Print F

Order Processing


Start
Receive order via e-mail
Copy and paste e-mail data into database
Shipping Involved?
Yes
No
Print Invoice and UPS label
Send e-mail to confirm shipping
Assemble package and ship
End

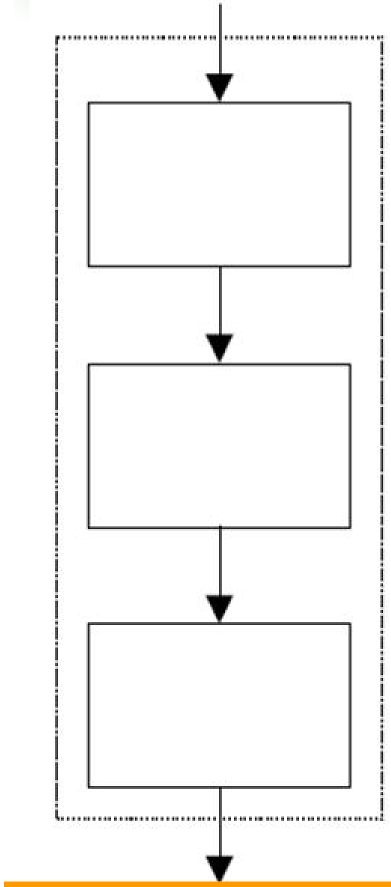
Do You Have To File an Income Tax Return?


Start
Are you single?
No
B
Yes
Are you under 65 years of age?
No
Yes
Is your gross income less than $8450?
Is your gross income less than $9700?
Yes
No
No
Yes
You do not have to file an income tax return
You have to file an income tax return
You do not have to file an income tax return
End
End
End

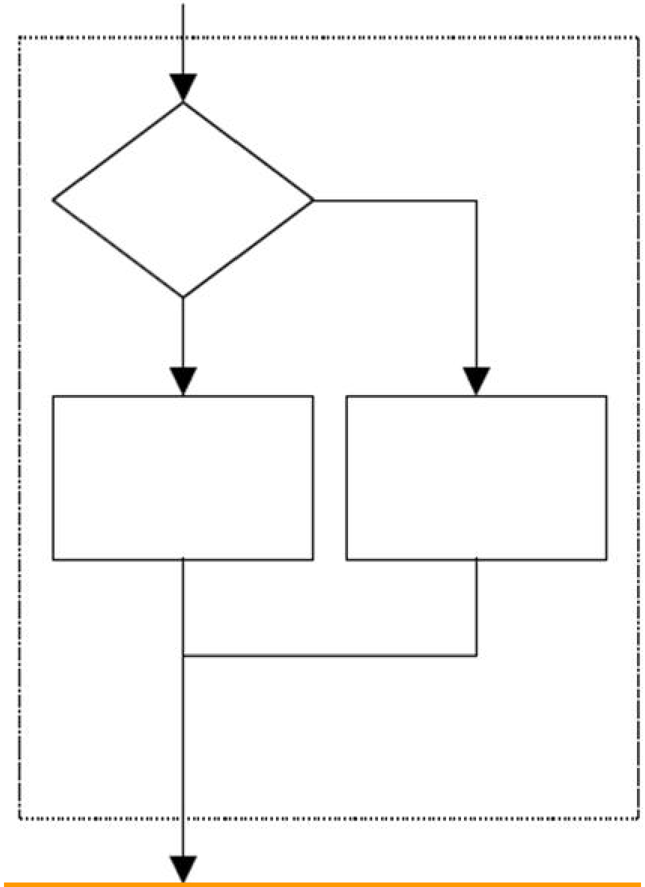


What is wrong with this algorithm?

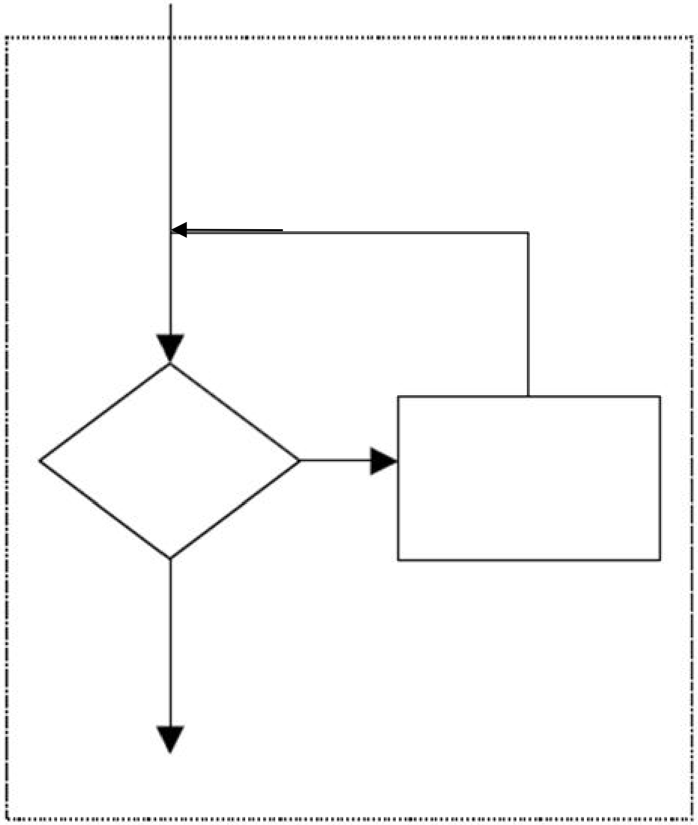
Basic flowchart structures



SEQUENCE

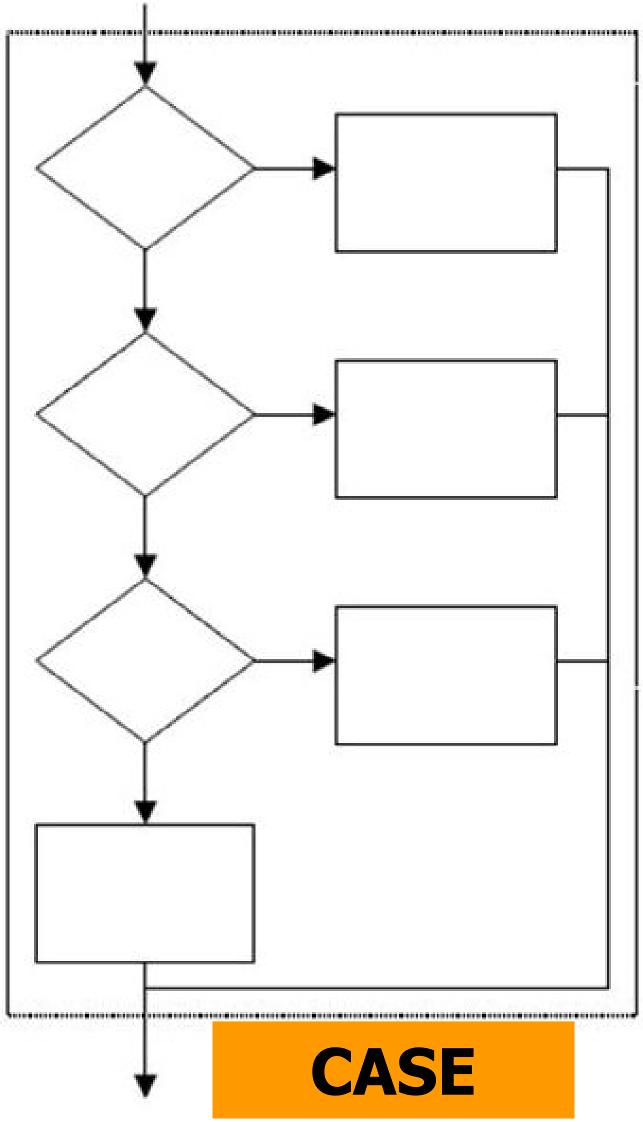


IF-THEN-ELSE

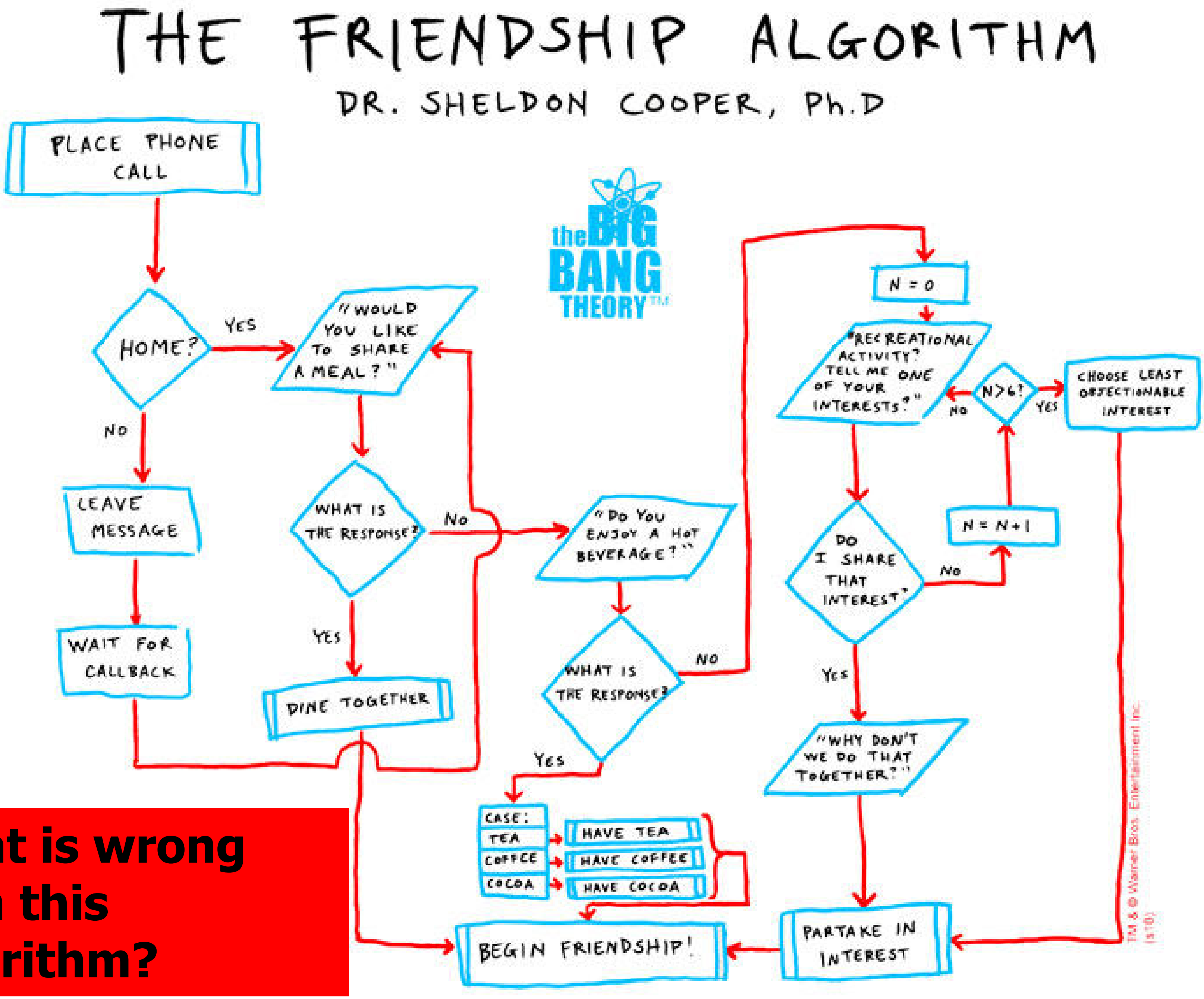


WHILE

Derived flowchart structures



CASE



What is wrong with this algorithm?

Languages

Programs are written in specific languages:

- Low-level languages:
  - Machine languages
  - Assembly languages

- High-level languages (FORTRAN, C, MATLAB…)

Low-Level Languages

Machine language:

Instructions are intrinsically compatible with and understood by the computer’s CPU.

Each instruction ultimately must be expressed as a series of bits – ***intrinsic machine code***. (Octal and hexadecimal more convenient)

An instruction normally consists of two parts: the ***operation*** to be performed and the ***operand*** expresses as a storage location.

Coding a machine language is ***tedious*** and seldom done by hand.

Low-Level Languages

Assembly language:

Is ***more symbolic*** than machine language.

The operand are referred to by ***variable names*** rather than by the addresses.

Blocks of code that are to be repeated verbatim at multiple locations in the program are know as macros (***macro instructions***). Macros are written only once and are referred to by a symbolic name.

Translated into machine language by an ***assembler***.

High-Level Language

The instructions ***resemble English.***

Translated into machine language by either a ***compiler*** (a true stand-alone executable program is created) or an ***interpreter*** (no stand-alone program is produced).

language	instruction
Machine language	**0001110010000110**
Assembly language	**ADD R6, R2, R6**
High Level	**R6 = R2 + R6**

Structured programming

•Also known as: ***top-down programming (in your textbook),*** procedure-oriented programming, GOTO-less programming.

•***Divides a procedure*** or algorithm into parts known as ***subprograms***, subroutines, modules, blocks, procedures or ***functions (in MATLAB)***.

•Internal subprograms are written by the programmer; external subprograms are supplied in a library from another source.

•Labels and GOTO commands are avoided.

Program design process (from textbook)

1. **Clearly state the problem that you are trying to solve:**

"A program to solve a system of simultaneous linear equations"
"A program to solve a system of simultaneous linear equations having real coefficients and up to 20 equations"

2. **Define the inputs required and the outputs to be produced:** Make a list of your input and output variables and clearly identify them

3. **Design the algorithm you intend to implement** (use flowcharts or pseudocode or MATLAB):

Decomposition: look for logical divisions within the problem and divides it up into subtasks.
Step-wise refinement: refine each of the divisions

Program design process (from textbook)

4. **Turn the algorithm into MATLAB statements (?)**

5. **Test the resulting program :**

•Start by testing each component.

•Verify that it works correctly for all legal input data sets.

•Testing continues after the program is complete (alpha release, beta release, general use).

Pseudocode

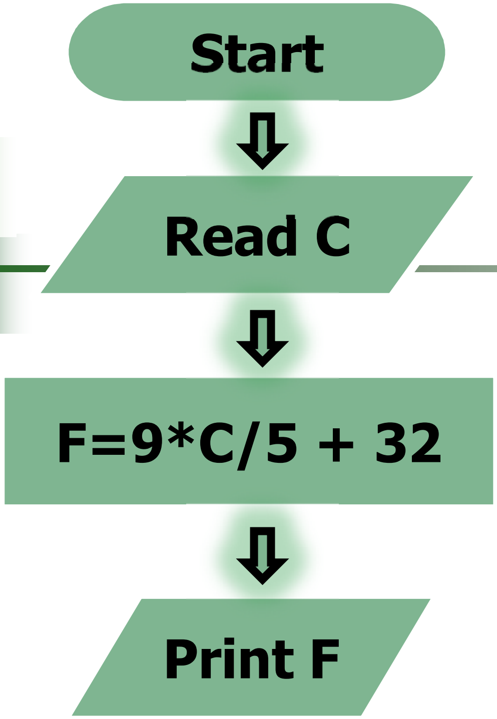
The description of an **algorithm** should be in a standard form that is **easy** for the programmer and other people **to understand**.

The **standard forms** used to described algorithms are called **constructs**.

The constructs can be described in a special way called **pseudocode.**

Pseudocode is constructed with a **separate line for each distinct idea** or segment of code.

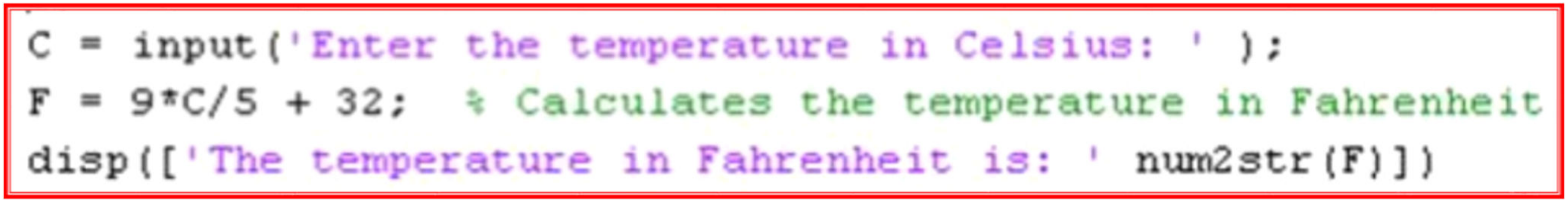
Each line of the pseudocode should describe its idea in **plain, easily understandable English.**



Flowchart

```
Prompt user to enter temperature in °C
Read temperature in °C (C)
F (temperature in °F) = 9*C/5 + 32
Write temperature in °F
```

pseudocode



```
C = input('Enter the temperature in Celsius: ' );
F = 9*C/5 + 32;  % Calculates the temperature in Fahrenheit
disp(['The temperature in Fahrenheit is: ' num2str(F)])
```

MATLAB

Data type (in C)

Int: used to define integer numbers

```
int QQ;
QQ = 12;
```

Float: used to define floating point numbers

```
float DIA;
DIA = 4.8;
```

Double: used to define big floating point numbers, reserves twice the storage for the number

```
double VOL;
VOL = 3250000;
```

Char: defines characters

```
char messg;
messg = ‘hola’;
```

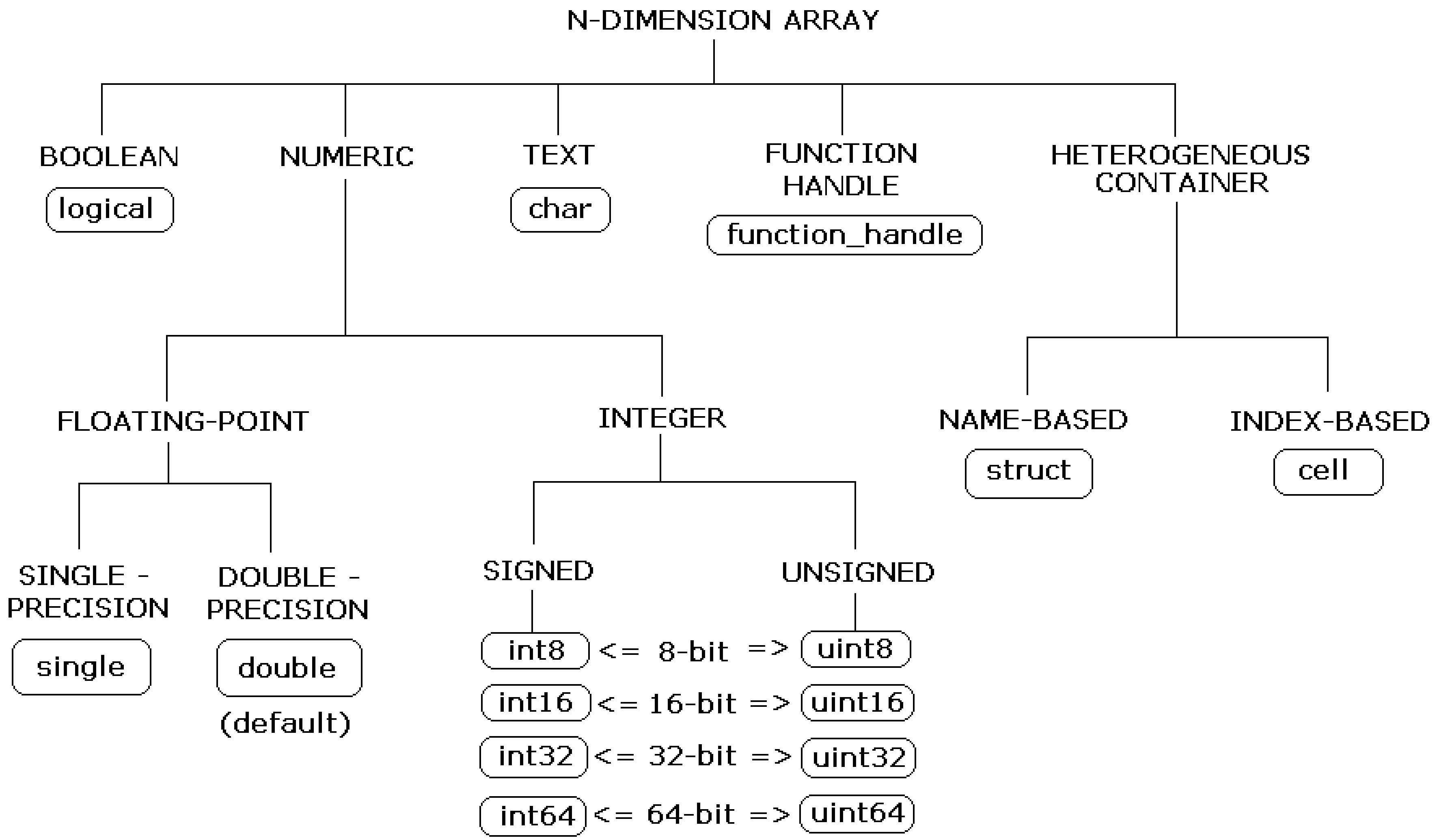
Data type (MATLAB)

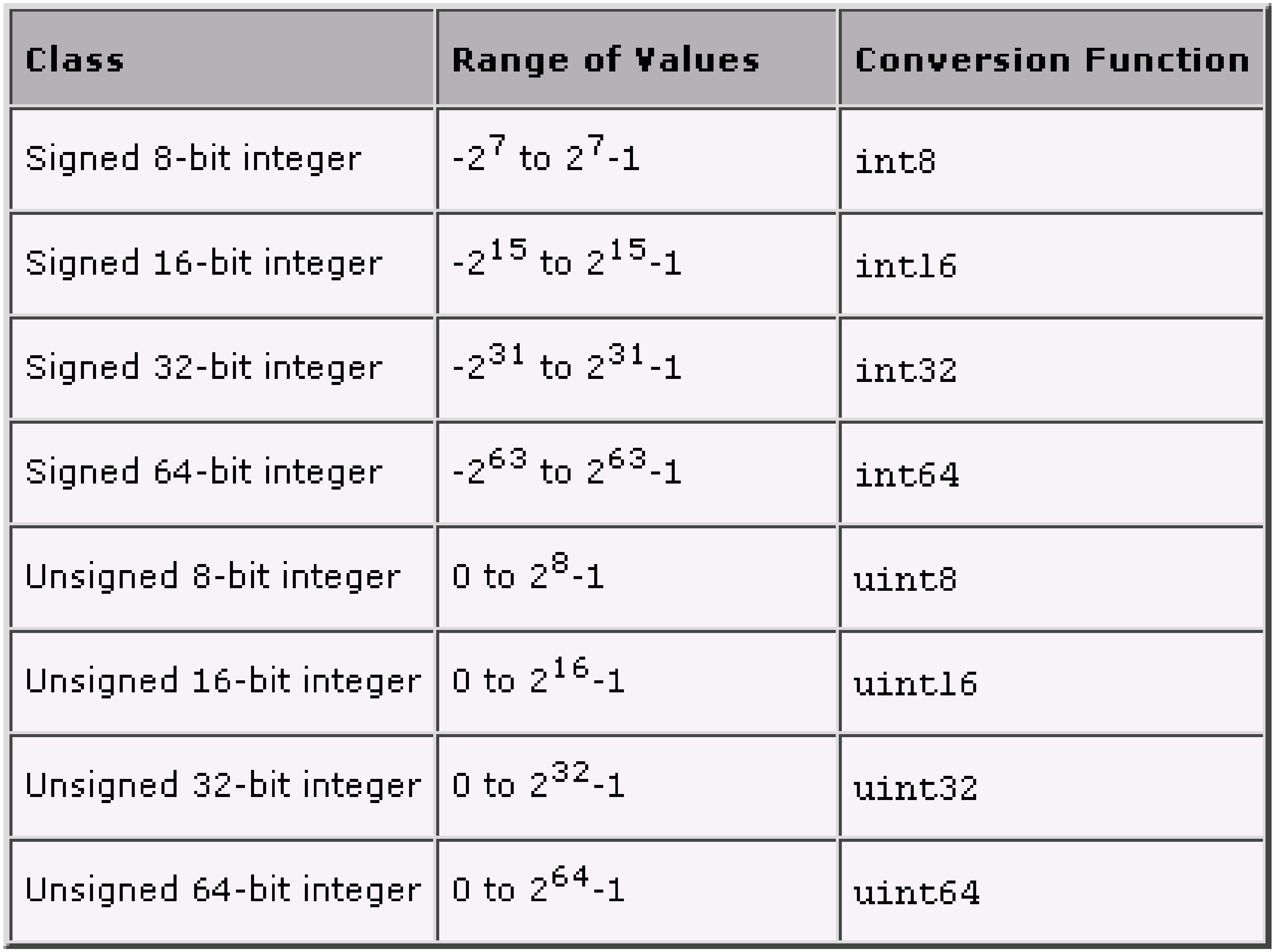
By default, MATLAB stores all numeric values as **double-precision floating point.**

You **cannot change** the default type and precision.

You can choose to store any number, or array of numbers, as integers or as single-precision.

Data type (MATLAB)





Class	Range of Values	Conversion Function
Signed 8-bit integer	-2^{7} to $2^{7}-1$	int8
Signed 16-bit integer	-2^{15} to $2^{15}-1$	int16
Signed 32-bit integer	-2^{31} to $2^{31}-1$	int32
Signed 64-bit integer	-2^{63} to $2^{63}-1$	int64
Unsigned 8-bit integer	0 to $2^{8}-1$	uint8
Unsigned 16-bit integer	0 to $2^{16}-1$	uint16
Unsigned 32-bit integer	0 to $2^{32}-1$	uint32
Unsigned 64-bit integer	0 to $2^{64}-1$	uint64

The range for double is:
-1.79769e+308 to -2.22507e-308
2.22507e-308 to 1.79769e+308

realmax - realmin

Data type (MATLAB)

A variable type double is automatically created whenever a numerical value is assigned to a variable name. The values can be real, imaginary or complex.

```
VR = 10.5
VI  = 5i
VJ  = 6j
VC = 2+3i
```

Type char variables are used to hold character strings. Automatically created when a single character or a character string is assigned to a variable name:

```
VCH = ‘hola’
```