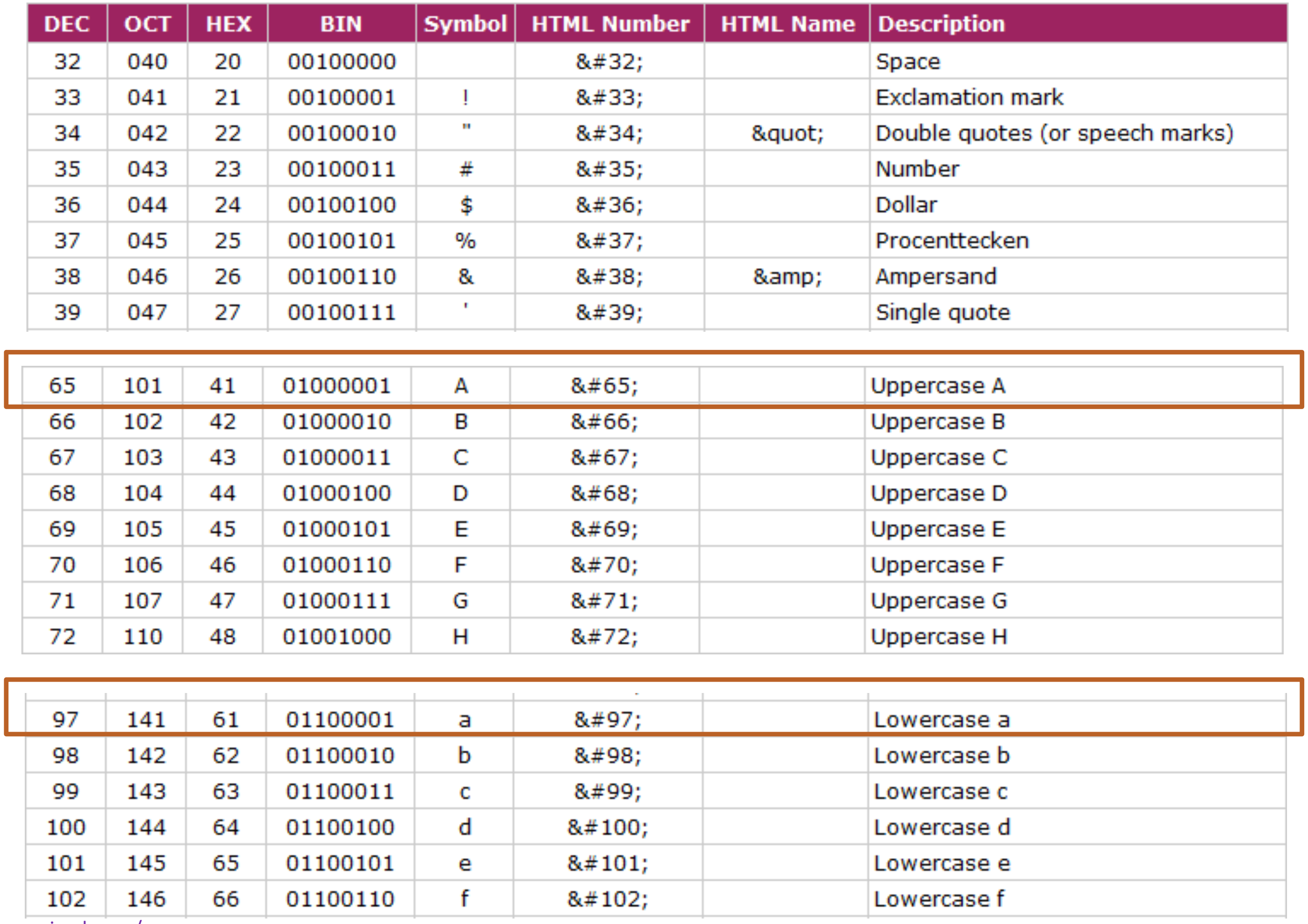
Program Design

Introduction to Arrays

Data Types in Programs (types of variables)

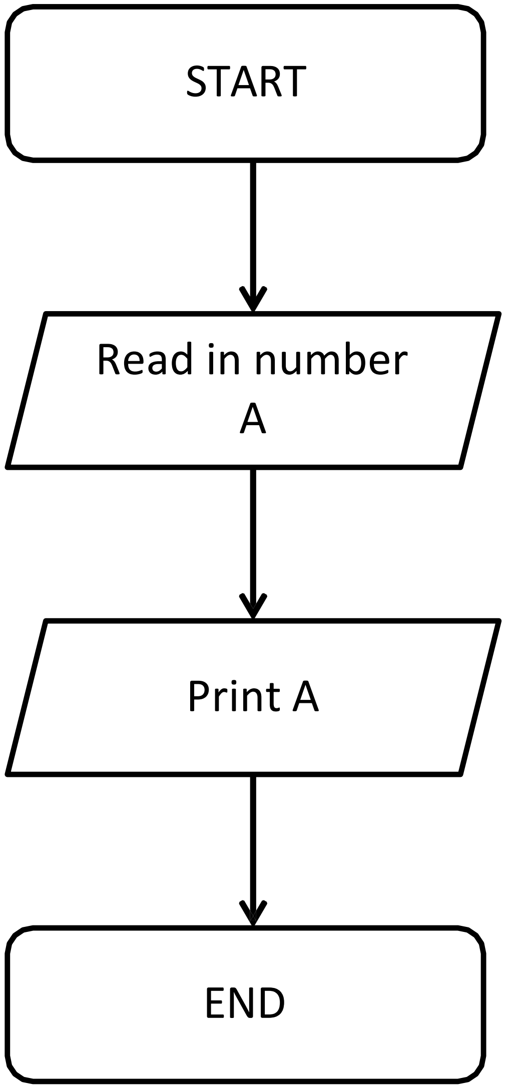
NUMBERS	**Integers** **100, 56, 9000000, 256,**	**Decimal** **3.14, 6.626068 , 100.01**	*int, float, decimal, numeric, tinyint, smallint, bigint, real*
TEXT	**Character** **'A', 'C', 'h', 'a','r'**	**String** **"a text string"**	*char(n), varchar(n), text, nchar(n), nvarchar(n), ntext*
DATES	**DateTime** **2010-12-01**	**TimeStamp** **2010-12-01 23:55:10**	*datetime, timestamp, date, time*
BINARY	**Bit** **1 , 0**	**Binary** **100011100011101010**	*bit, binary(n),varbinary(n), image*
BOOLEAN	**Boolean** **True, false**	*boolean*	**NULL** and **void** are value type in some languages = undefined, value unknown

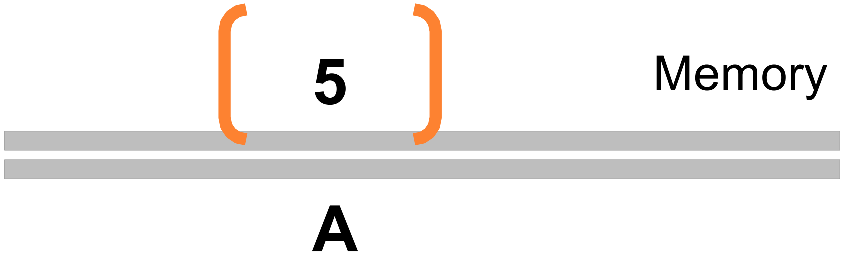
Everything is stored as binary numbers inside the computer – Unicode



DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32	040	20	00100000		&#32;		Space
33	041	21	00100001	!	&#33;		Exclamation mark
34	042	22	00100010	"	&#34;	&quot;	Double quotes (or speech marks)
35	043	23	00100011	#	&#35;		Number
36	044	24	00100100	$	&#36;		Dollar
37	045	25	00100101	%	&#37;		Procenttecken
38	046	26	00100110	&	&#38;	&amp;	Ampersand
39	047	27	00100111	'	&#39;		Single quote
65	101	41	01000001	A	&#65;		Uppercase A
66	102	42	01000010	B	&#66;		Uppercase B
67	103	43	01000011	C	&#67;		Uppercase C
68	104	44	01000100	D	&#68;		Uppercase D
69	105	45	01000101	E	&#69;		Uppercase E
70	106	46	01000110	F	&#70;		Uppercase F
71	107	47	01000111	G	&#71;		Uppercase G
72	110	48	01001000	H	&#72;		Uppercase H
97	141	61	01100001	a	&#97;		Lowercase a
98	142	62	01100010	b	&#98;		Lowercase b
99	143	63	01100011	c	&#99;		Lowercase c
100	144	64	01100100	d	&#100;		Lowercase d
101	145	65	01100101	e	&#101;		Lowercase e
102	146	66	01100110	f	&#102;		Lowercase f

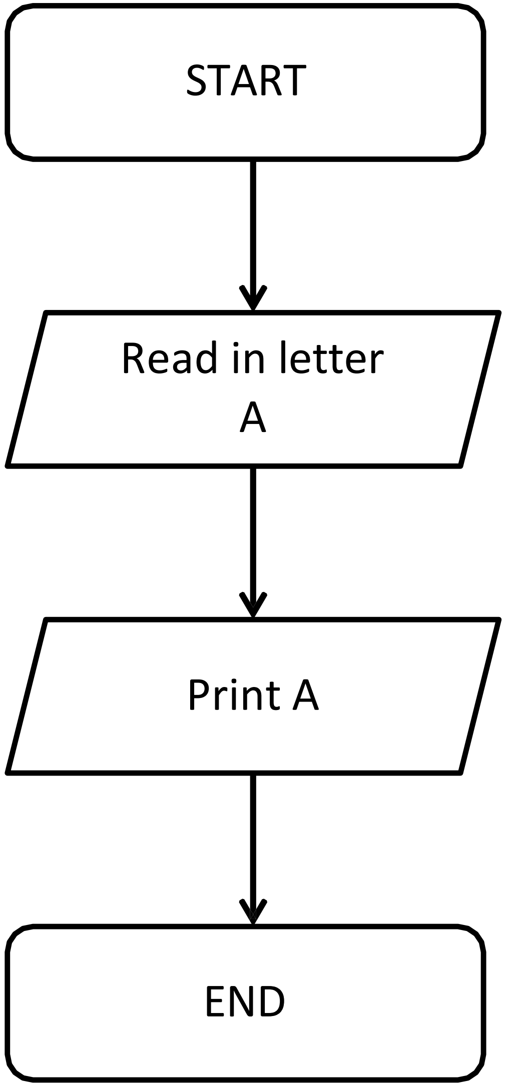
https://home.unicode.org/

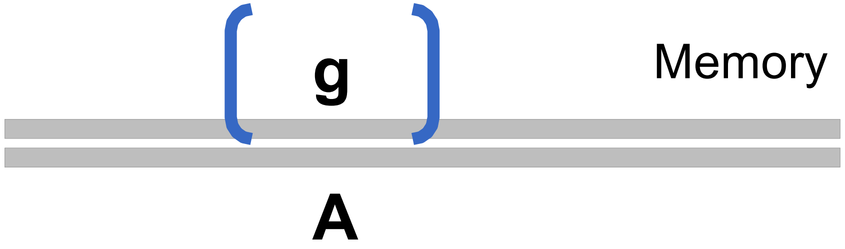




Type : ***integer***

A basic variable holds only <u>one value</u> of a particular type





Type : ***character***

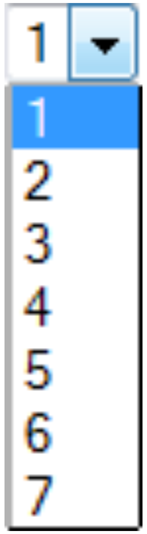
A basic variable holds only <u>one value</u> of a particular type

If we wanted to read in a word
how should we store it ?

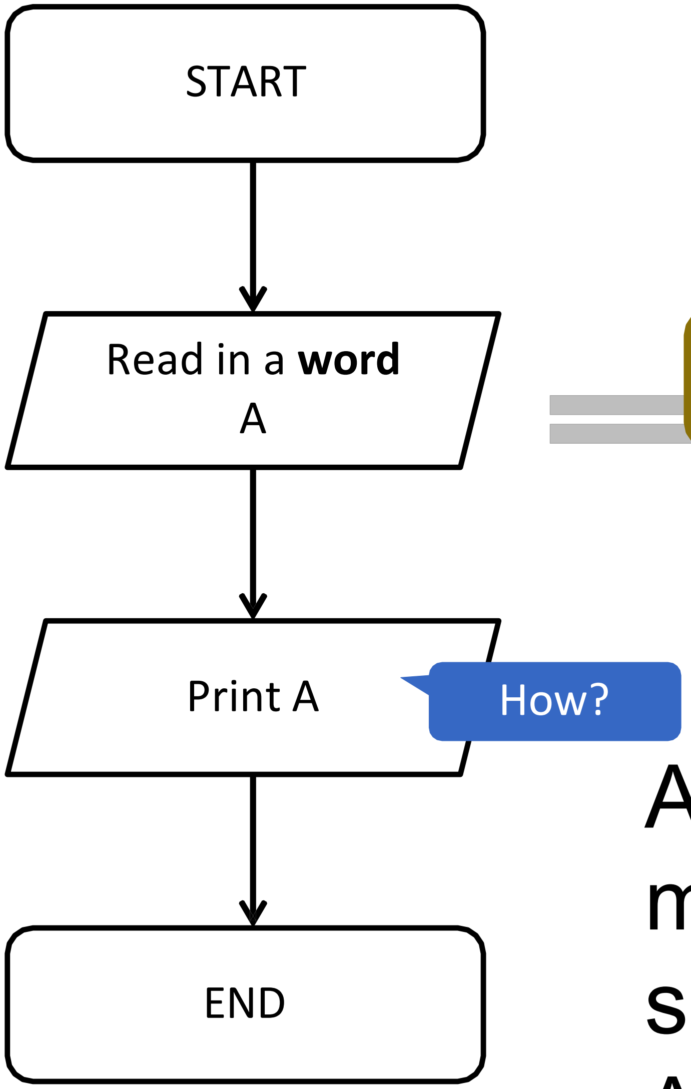
Arrays

- An array is **a variable** that **can contain more than one value**
- But the values must all be of the same **same type**
- An egg box in memory, with a name
- Or a **list** in memory

myVariable →



myVariable →



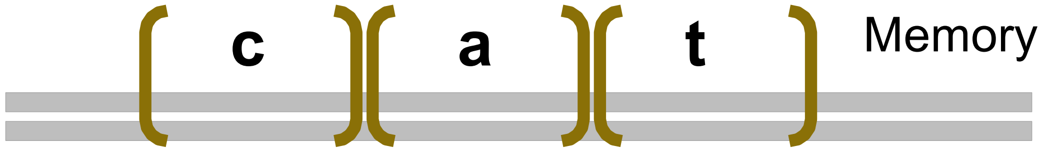
START

Read in a **word**
A

Print A

How?

END



c a t

Memory

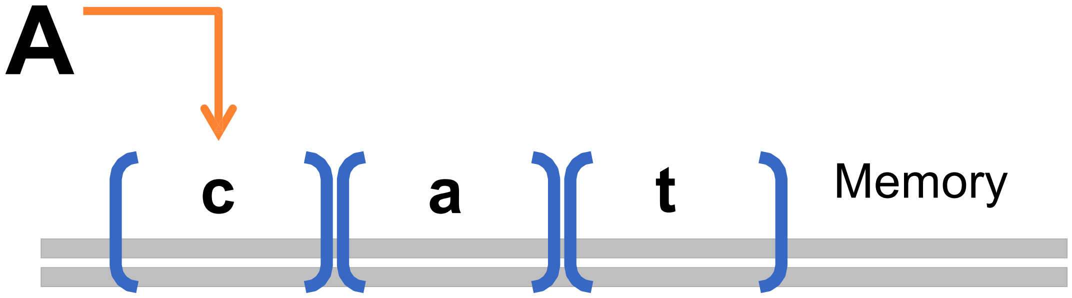
A [3]

*Type : **character array = string***

An array variable holds multiple <u>value</u> of the same type.
A name <u>chain of values</u>

Elements in an array

The string variable



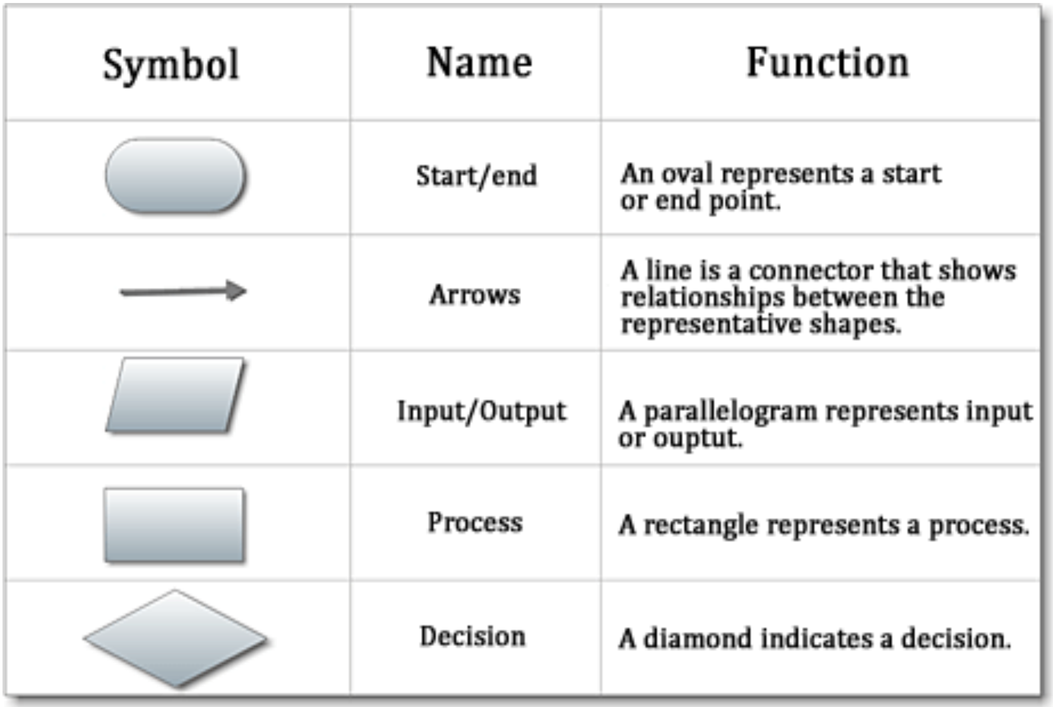
0	1	2	Memory slot reference number
A[0]	**A[1]**	**A[2]**	Getting the value of a memory slot

Type : ***character array = string***

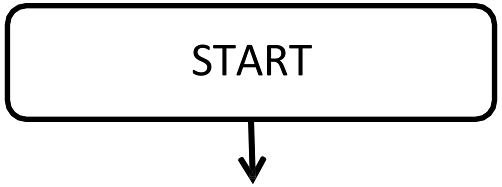
Flowcharts

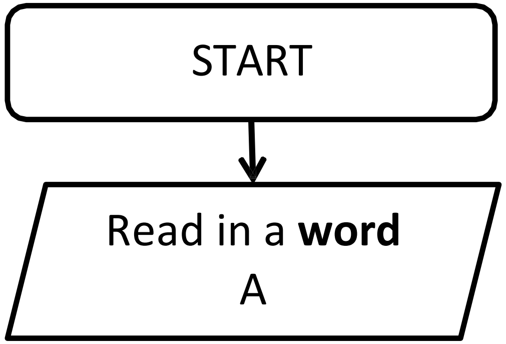
- So let’s say we want to express the following algorithm:

  Print out each character in the word “cat”.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.

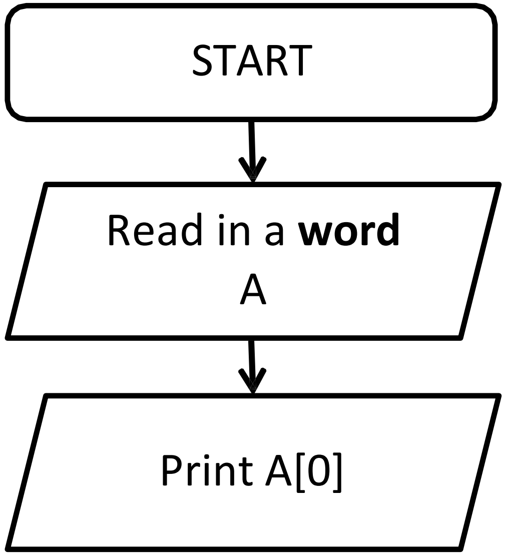

START



START

Read in a **word**
A

0	1	2
c	a	t



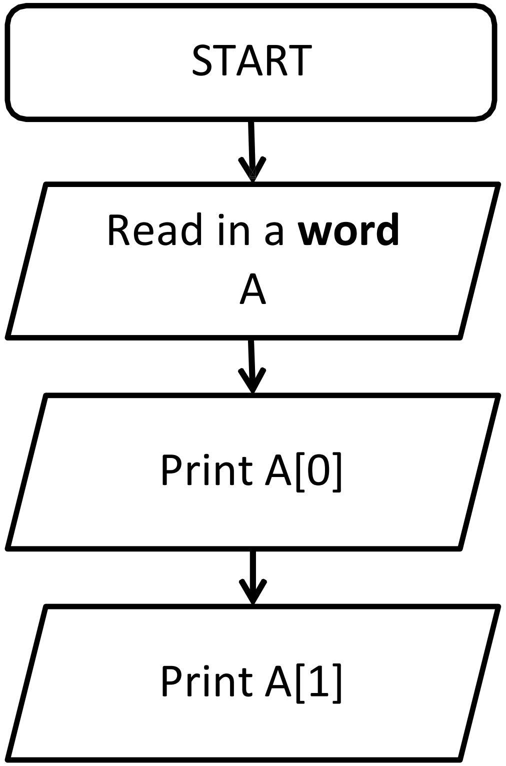
START

Read in a **word**
A

Print A[0]

0	1	2
c	a	t

c



START

Read in a **word** A

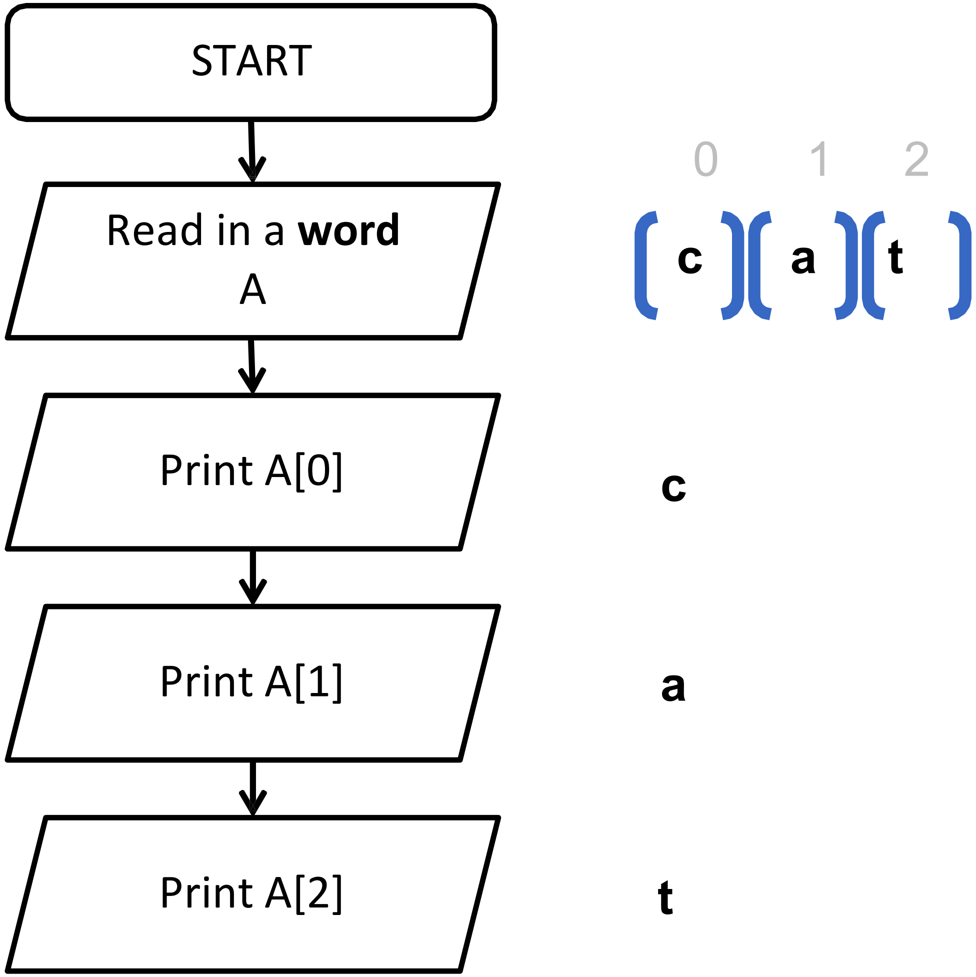
Print A[0]

Print A[1]

0	1	2
c	a	t

c

a



START

Read in a **word** A

Print A[0]

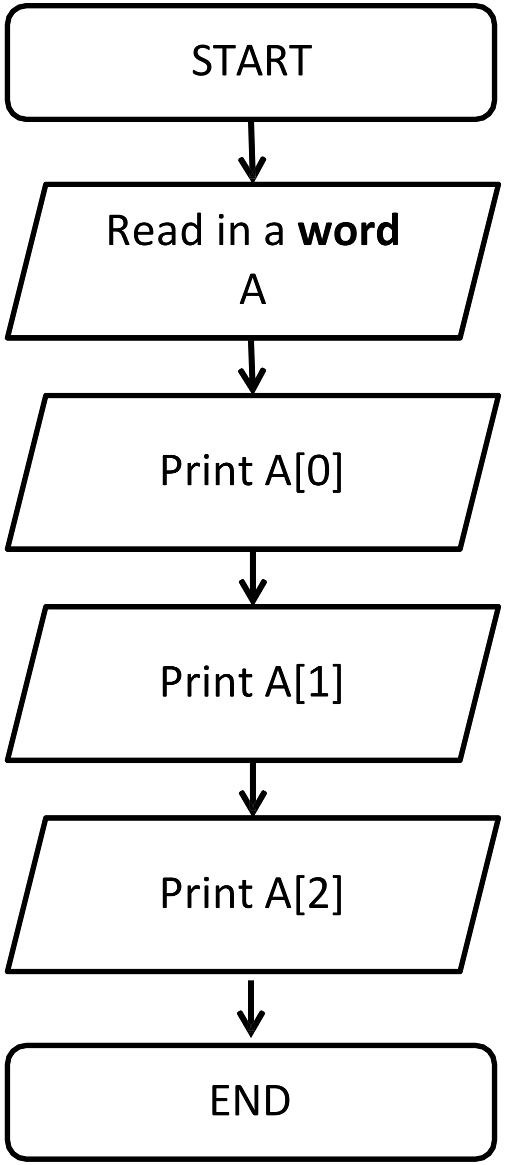
Print A[1]

Print A[2]

c

a

t



START

Read in a **word**
A

Print A[0]

Print A[1]

Print A[2]

END

0	1	2
c	a	t

c

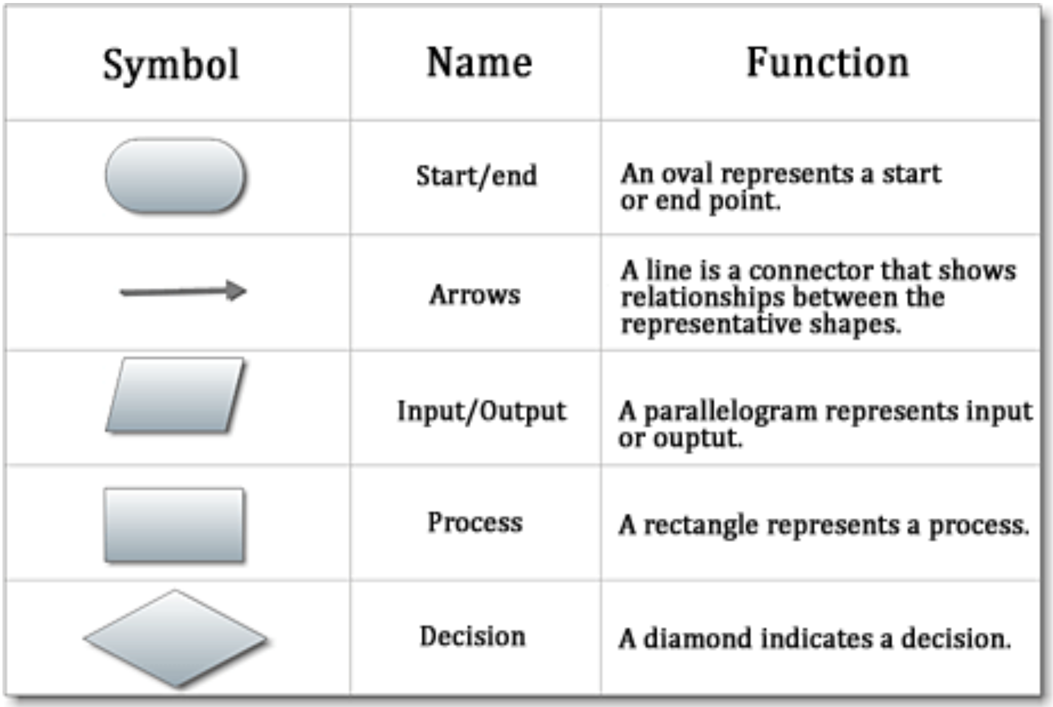
a

t

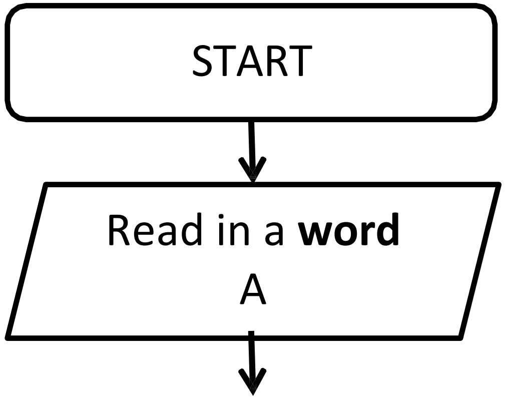
Flowcharts

- So let’s say we want to express the following algorithm:

Print out each character in any word that is entered.



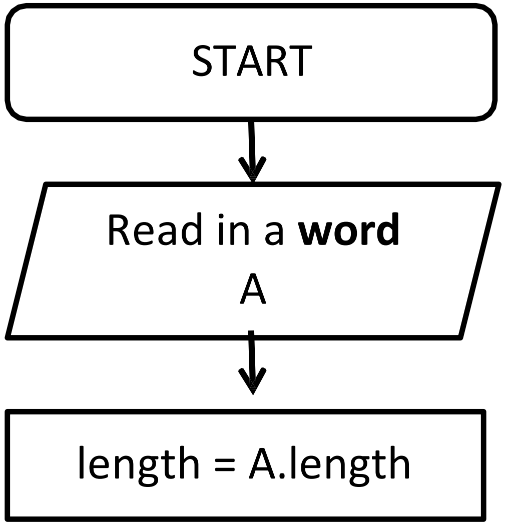
Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.



START

Read in a **word** A

0	1	2	3	4
e	**x**	**a**	**m**	**s**



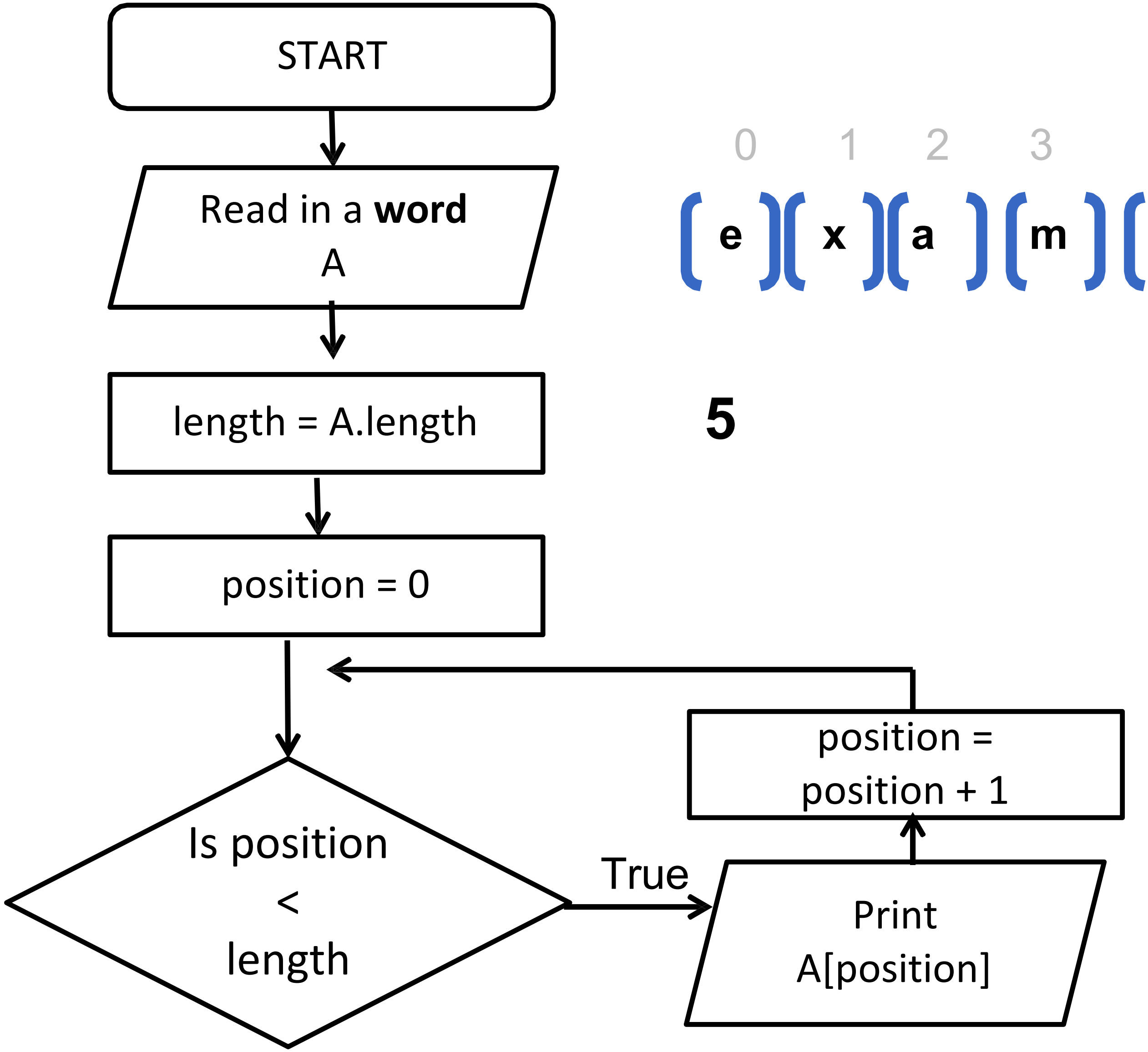
START

Read in a **word**
A

length = A.length

0	1	2	3	4
e	**x**	**a**	**m**	**s**

5



START

Read in a **word**
A

length = A.length

position = 0

Is position
<
length

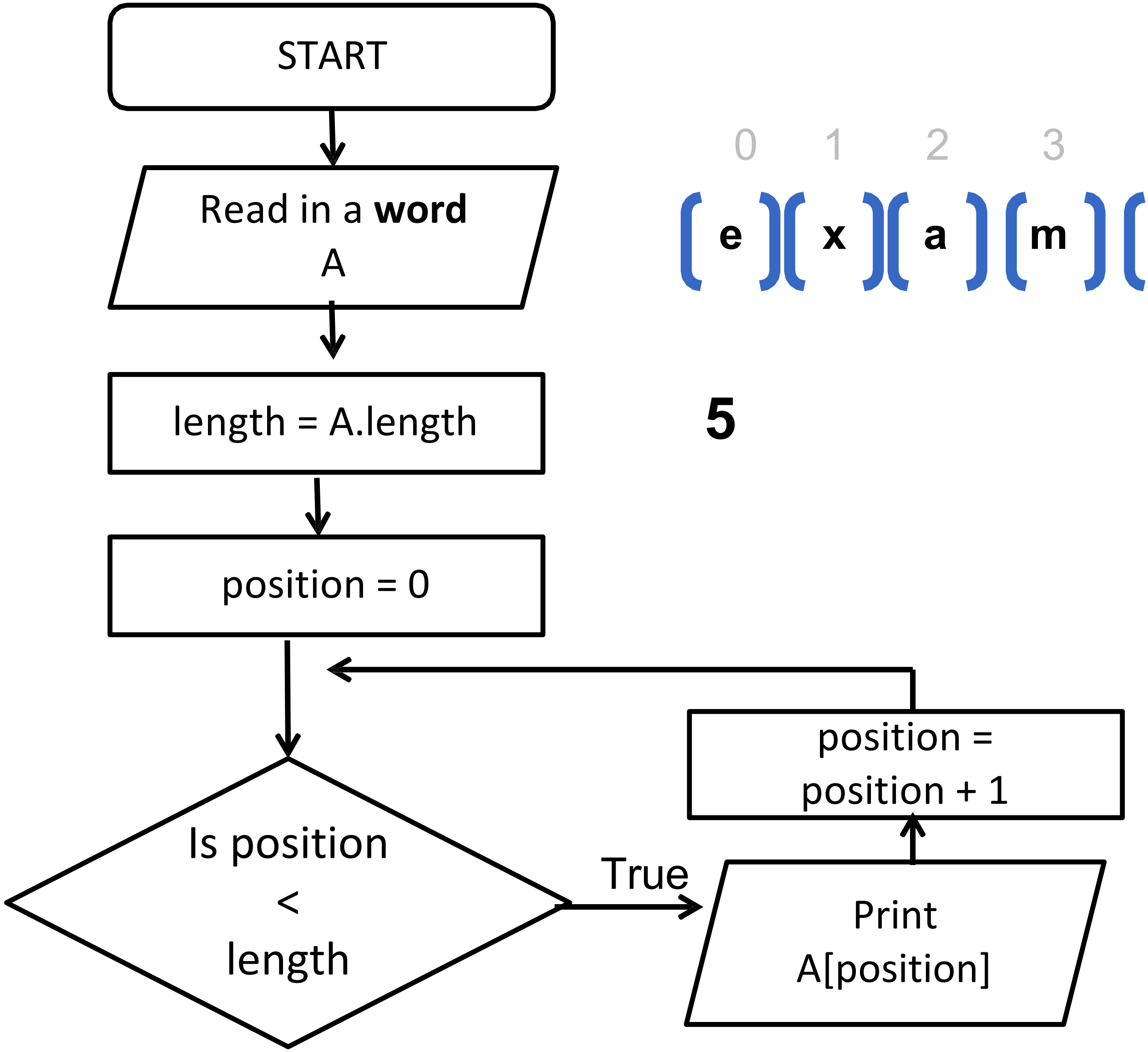
True

Print
A[position]

position =
position + 1

0	1	2	3	4
e	x	a	m	s

5



START

Read in a **word**
A

length = A.length

position = 0

Is position
<
length

True

Print
A[position]

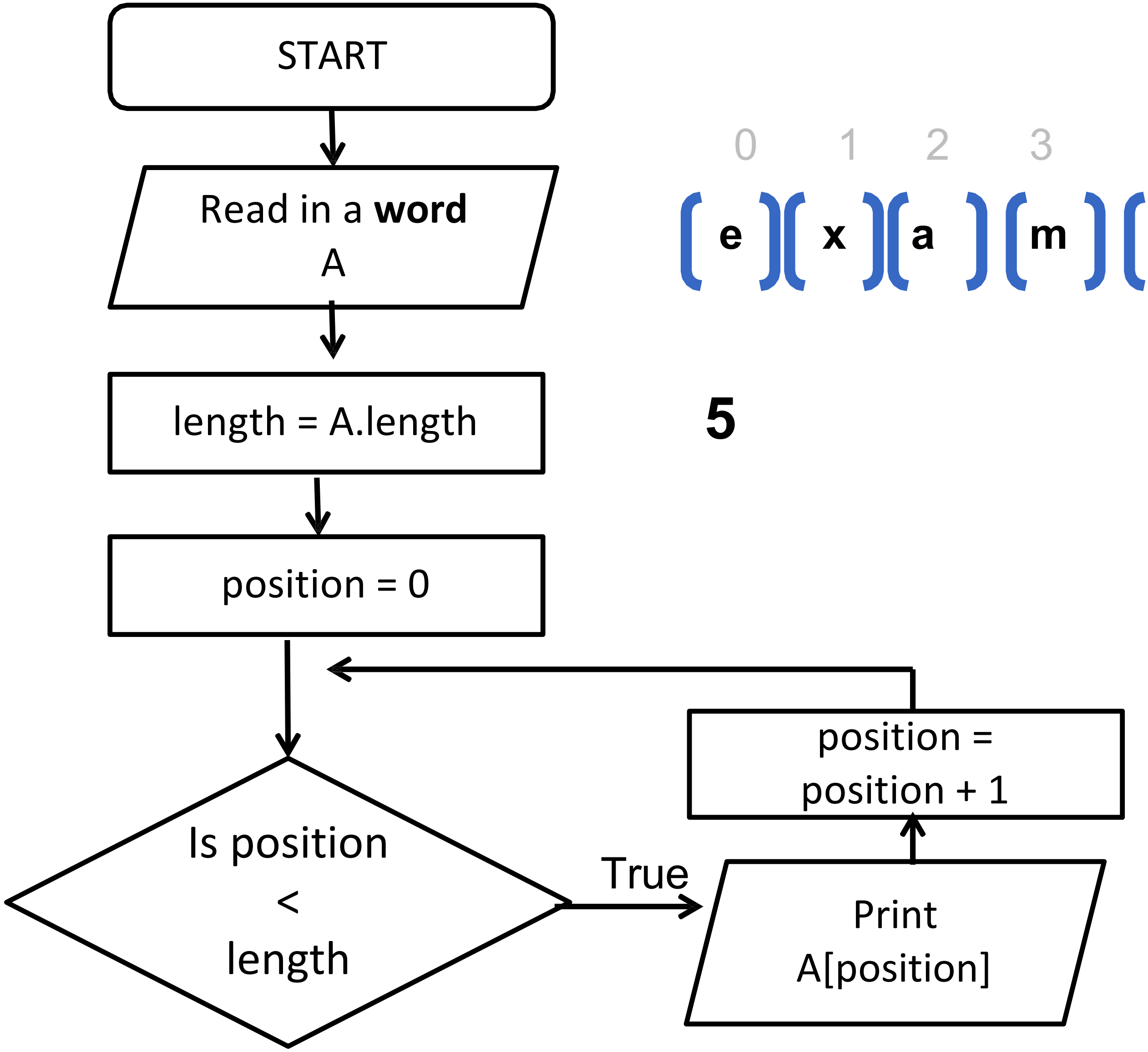
position =
position + 1

0	1	2	3	4
e	x	a	m	s

5

position	Array ref	value
0	A[0]	e

e



START

Read in a **word** A

length = A.length

position = 0

Is position < length

True

Print A[position]

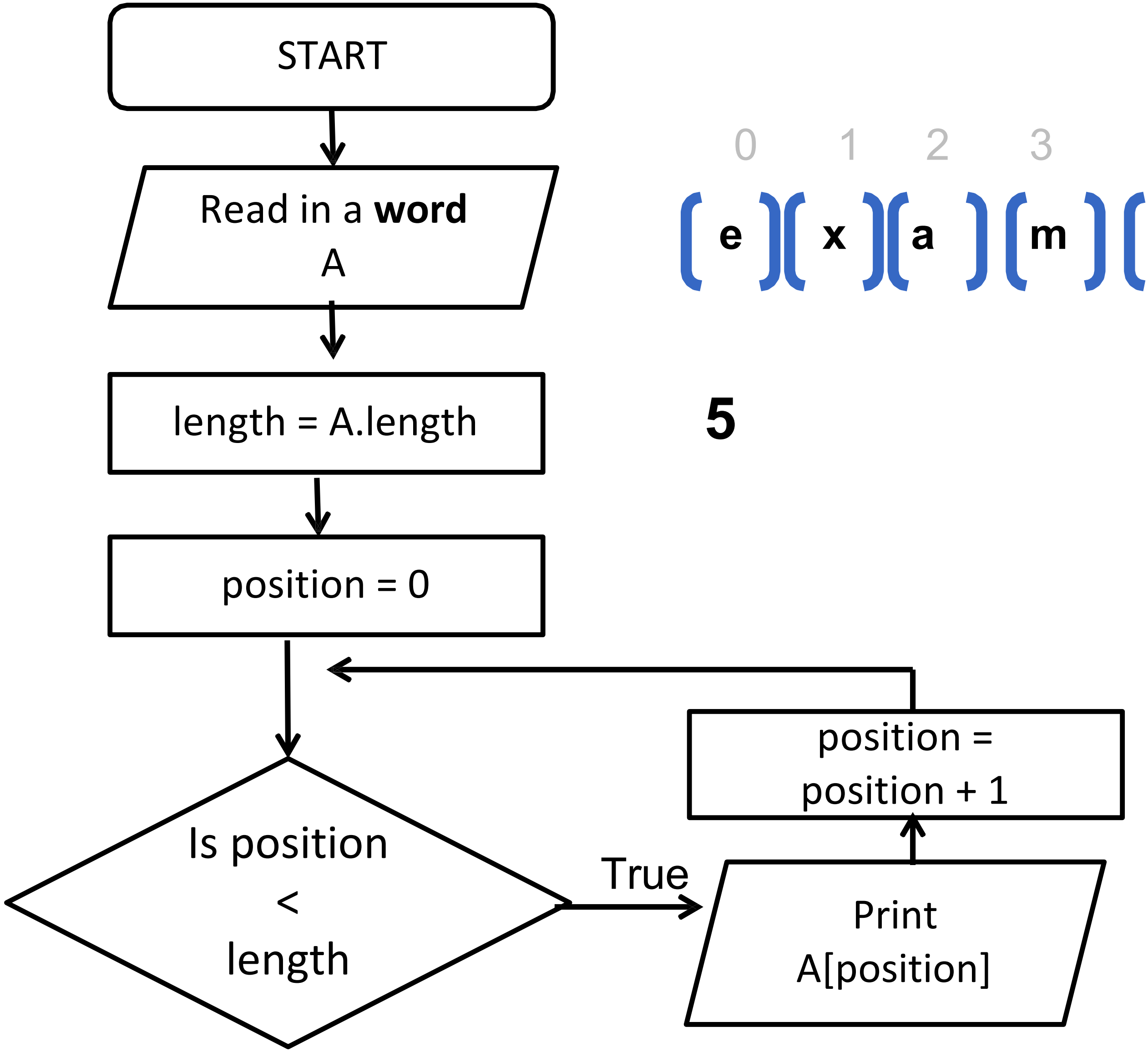
position = position + 1

0	1	2	3	4
e	x	a	m	s

5

position	Array ref	value
0	A[0]	e
1	A[1]	x

x



START

Read in a **word**
A

length = A.length

position = 0

Is position
<
length

True

Print
A[position]

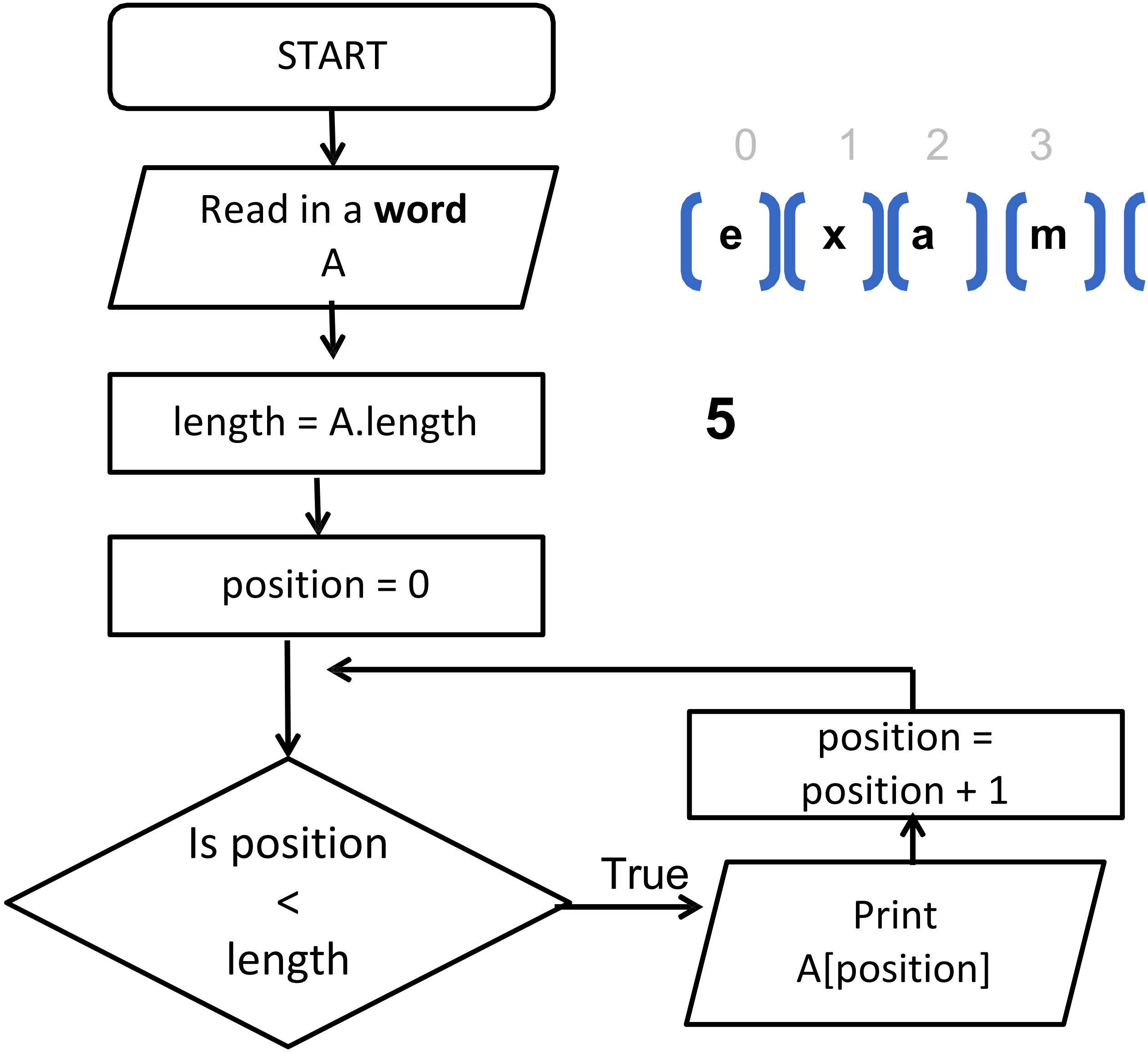
position =
position + 1

0	1	2	3	4
e	x	a	m	s

5

position	Array ref	value
0	A[0]	e
1	A[1]	x
2	A[2]	a

a

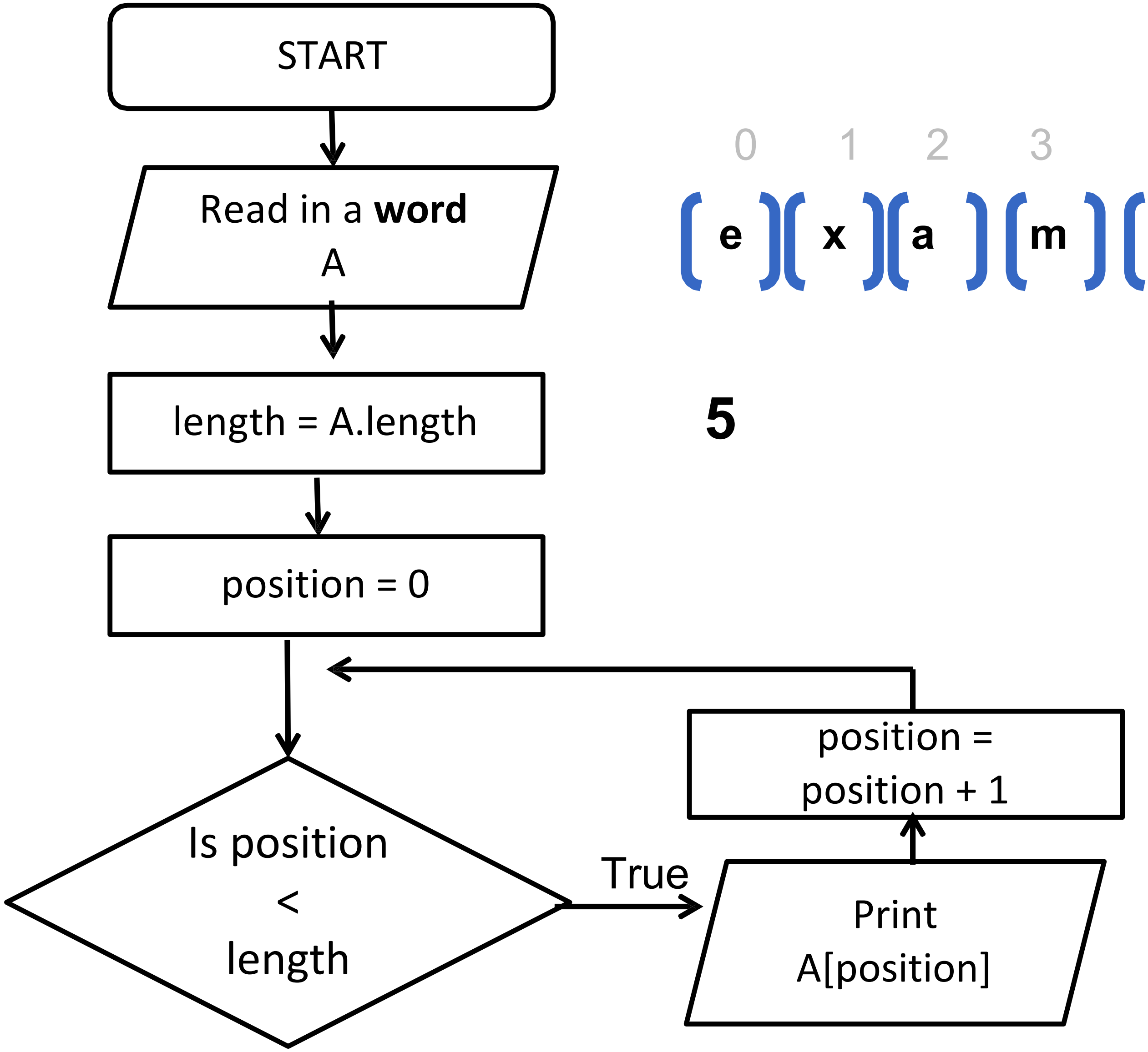


0	1	2	3	4
e	x	a	m	s

5

position	Array ref	value
0	A[0]	e
1	A[1]	x
2	A[2]	a
3	A[3]	m

m



START

Read in a **word** A

length = A.length

position = 0

Is position < length

True

Print A[position]

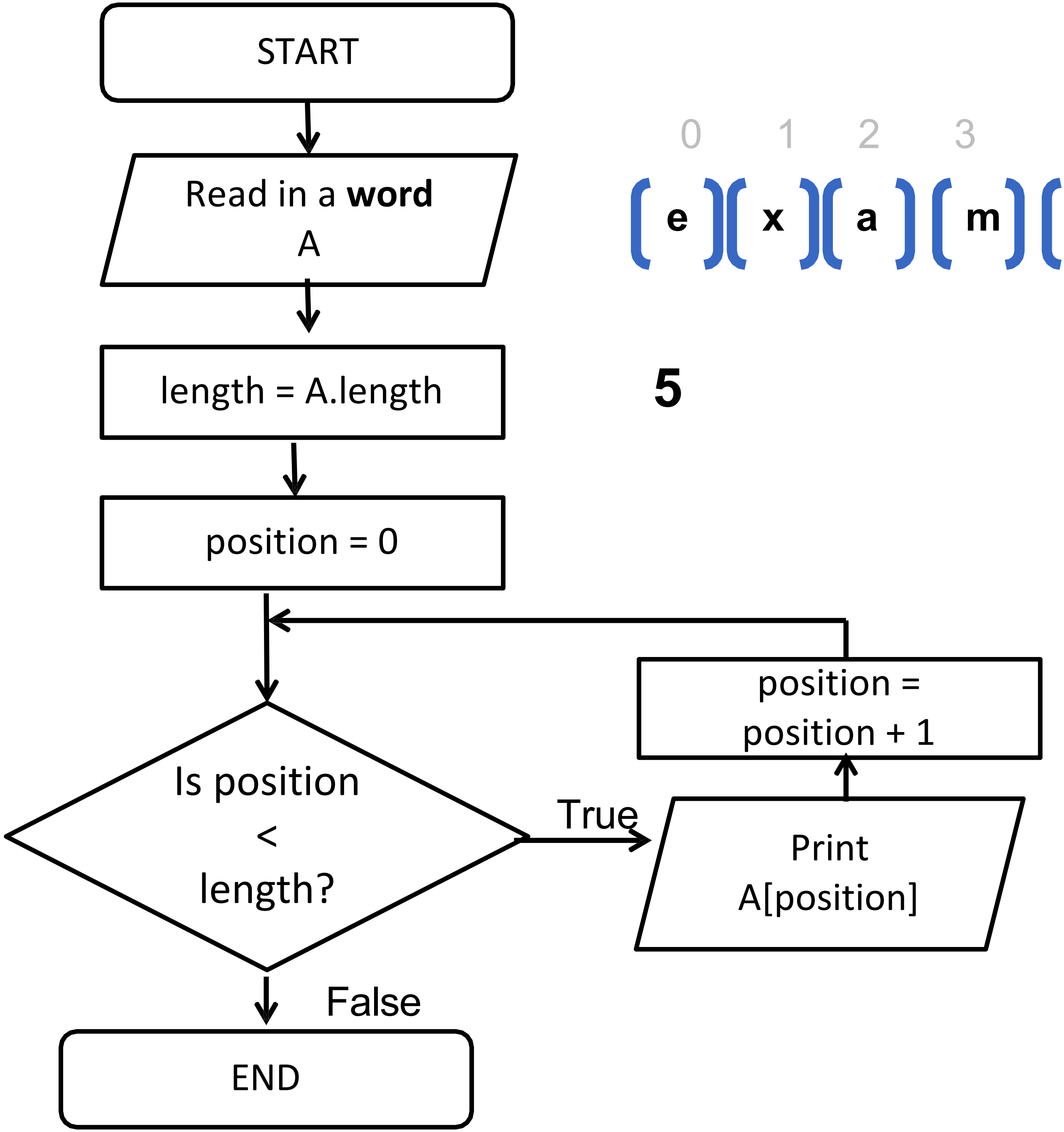
position = position + 1

0	1	2	3	4
e	x	a	m	s

5

position	Array ref	value
0	A[0]	e
1	A[1]	x
2	A[2]	a
3	A[3]	m
4	A[4]	s

s



START

Read in a **word** A

length = A.length

position = 0

Is position < length?

True

Print A[position]

position = position + 1

False

END

0	1	2	3	4
e	x	a	m	s

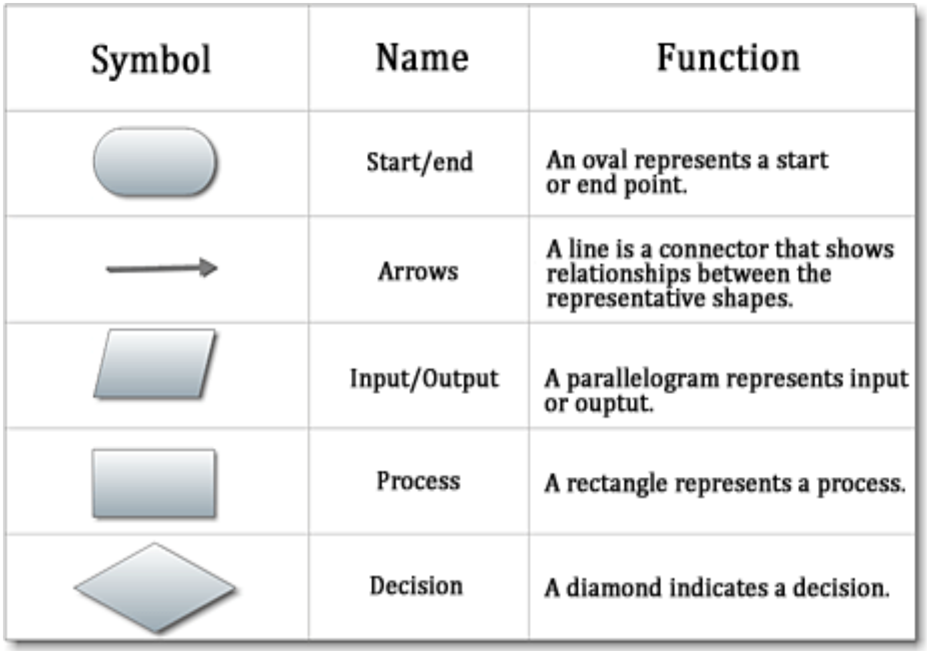
5

position	Array ref	value
0	A[0]	e
1	A[1]	x
2	A[2]	a
3	A[3]	m
4	A[4]	s

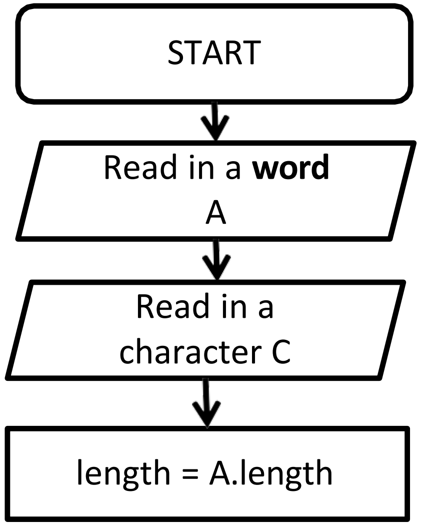
Flowcharts

- So let’s say we want to express the following algorithm:

Given a word and a character from that word print out the number of times the character is in the word.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.



START

Read in a **word** A

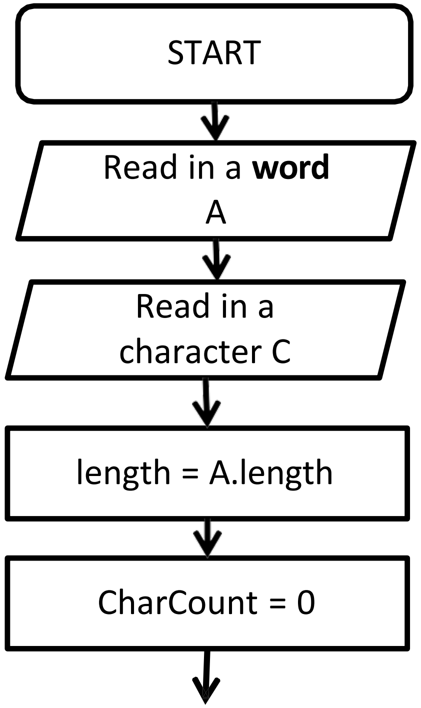
Read in a character C

length = A.length

0	1	2	3	4	5
c	h	e	e	s	e

e

6



START

Read in a **word** A

Read in a character C

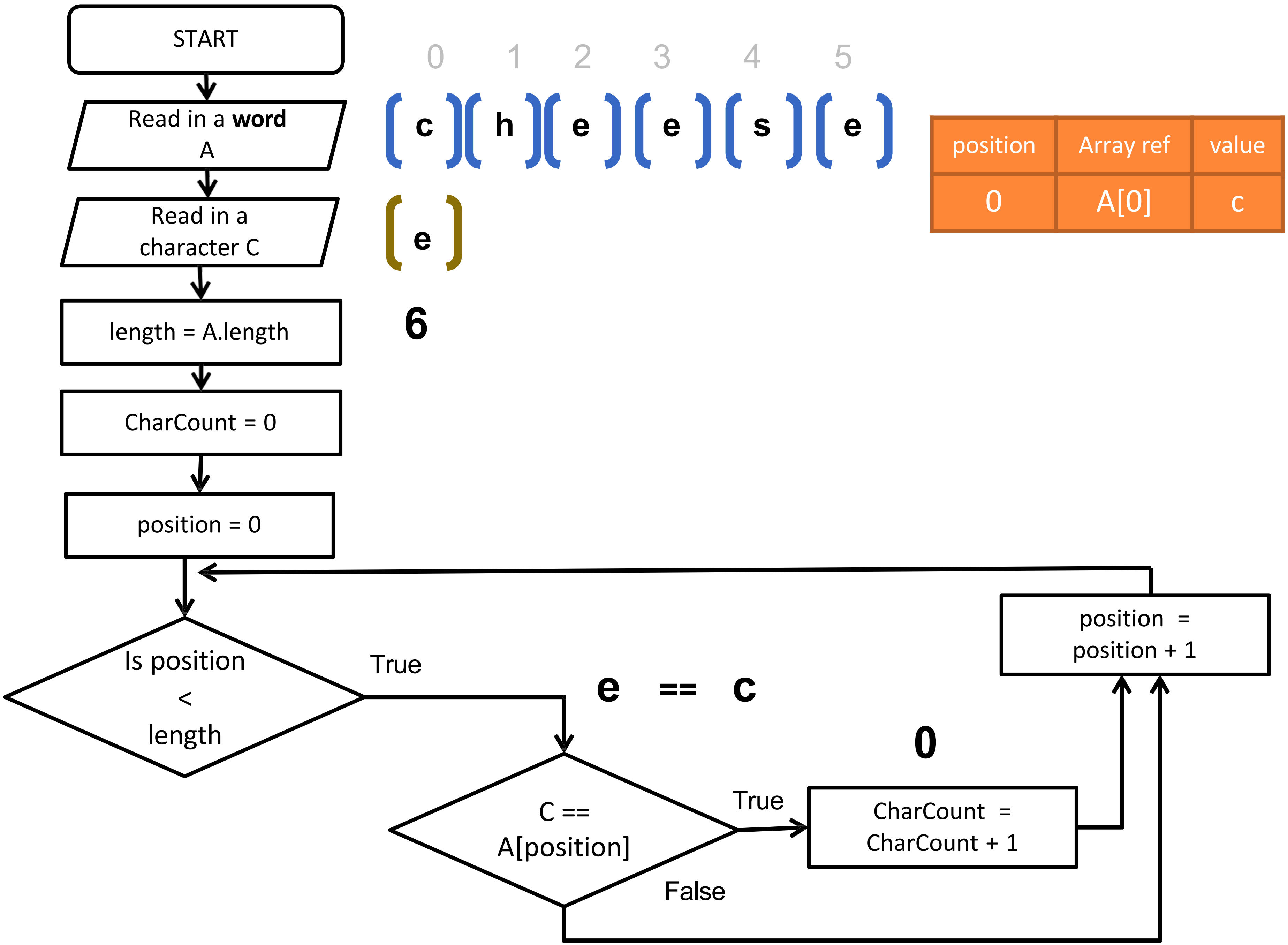
length = A.length

CharCount = 0

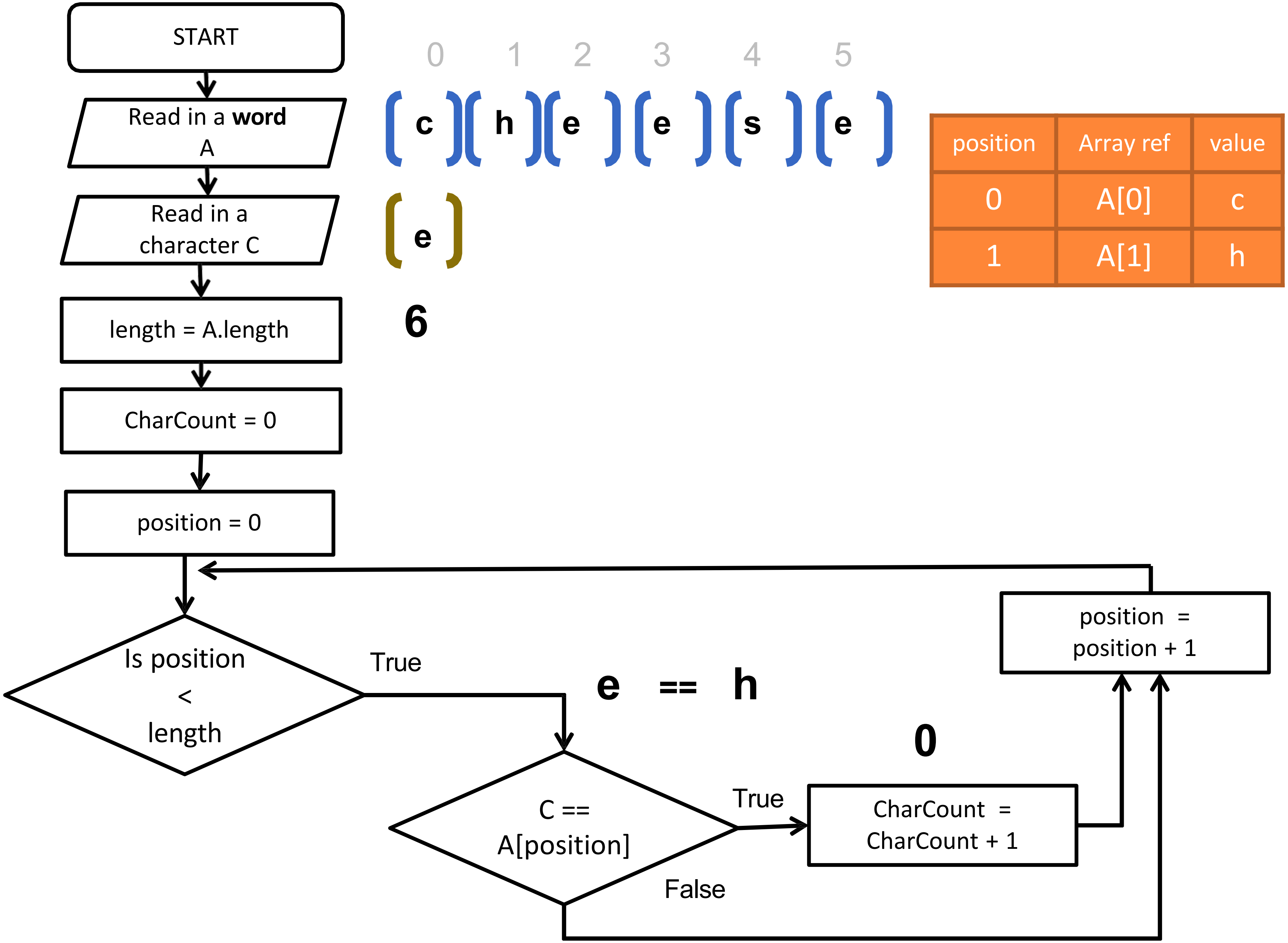
0	1	2	3	4	5
c	h	e	e	s	e

e

6



position	Array ref	value
0	A[0]	c



START

Read in a **word** A

Read in a character C

length = A.length

CharCount = 0

position = 0

Is position < length

True

C == A[position]

True

CharCount = CharCount + 1

False

position = position + 1

0	1	2	3	4	5
c	h	e	e	s	e

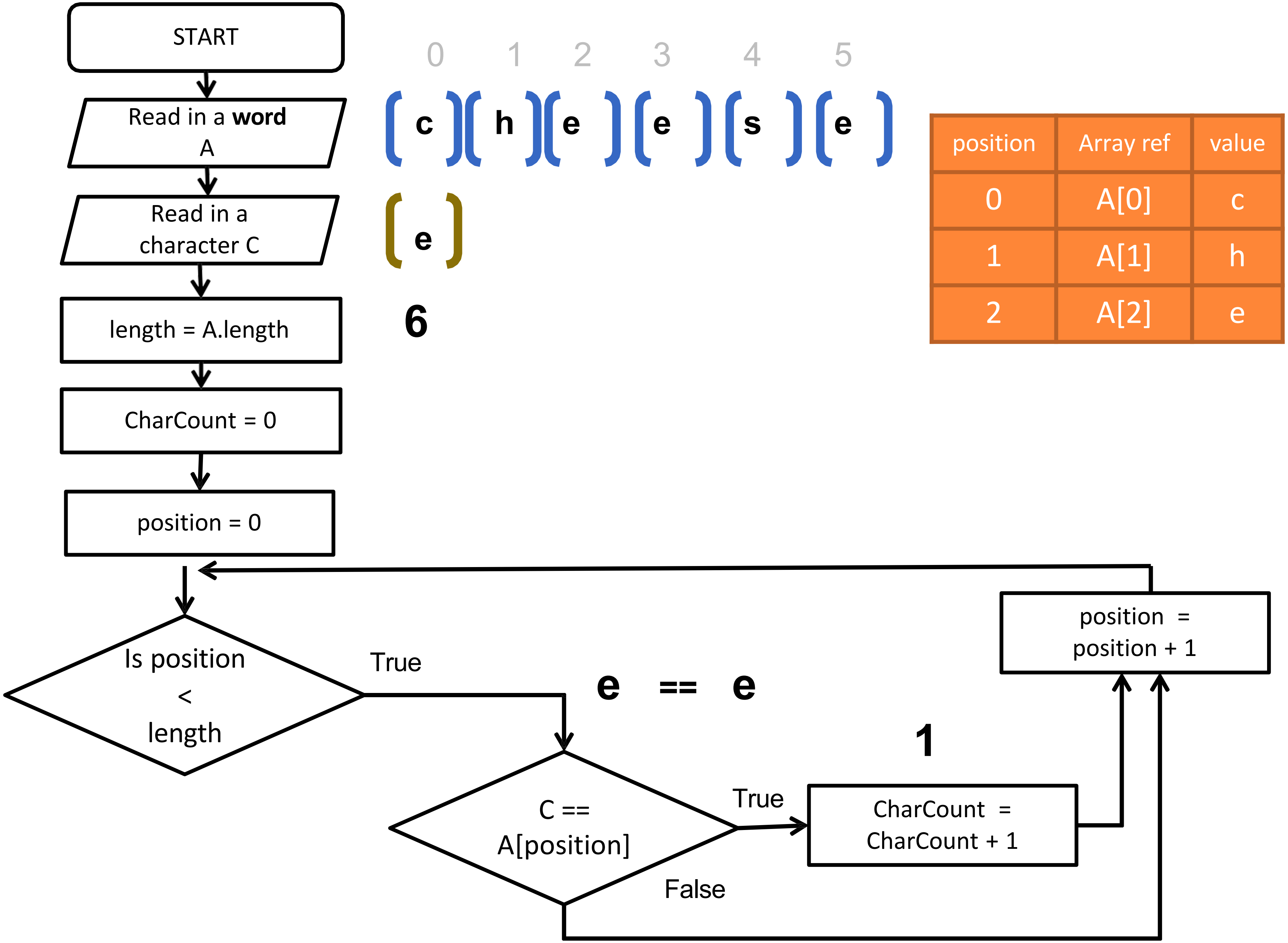
e

6

e == h

0

position	Array ref	value
0	A[0]	c
1	A[1]	h



START

Read in a **word** A

Read in a character C

length = A.length

CharCount = 0

position = 0

Is position < length

True

C == A[position]

True

CharCount = CharCount + 1

False

position = position + 1

0	1	2	3	4	5
c	h	e	e	s	e

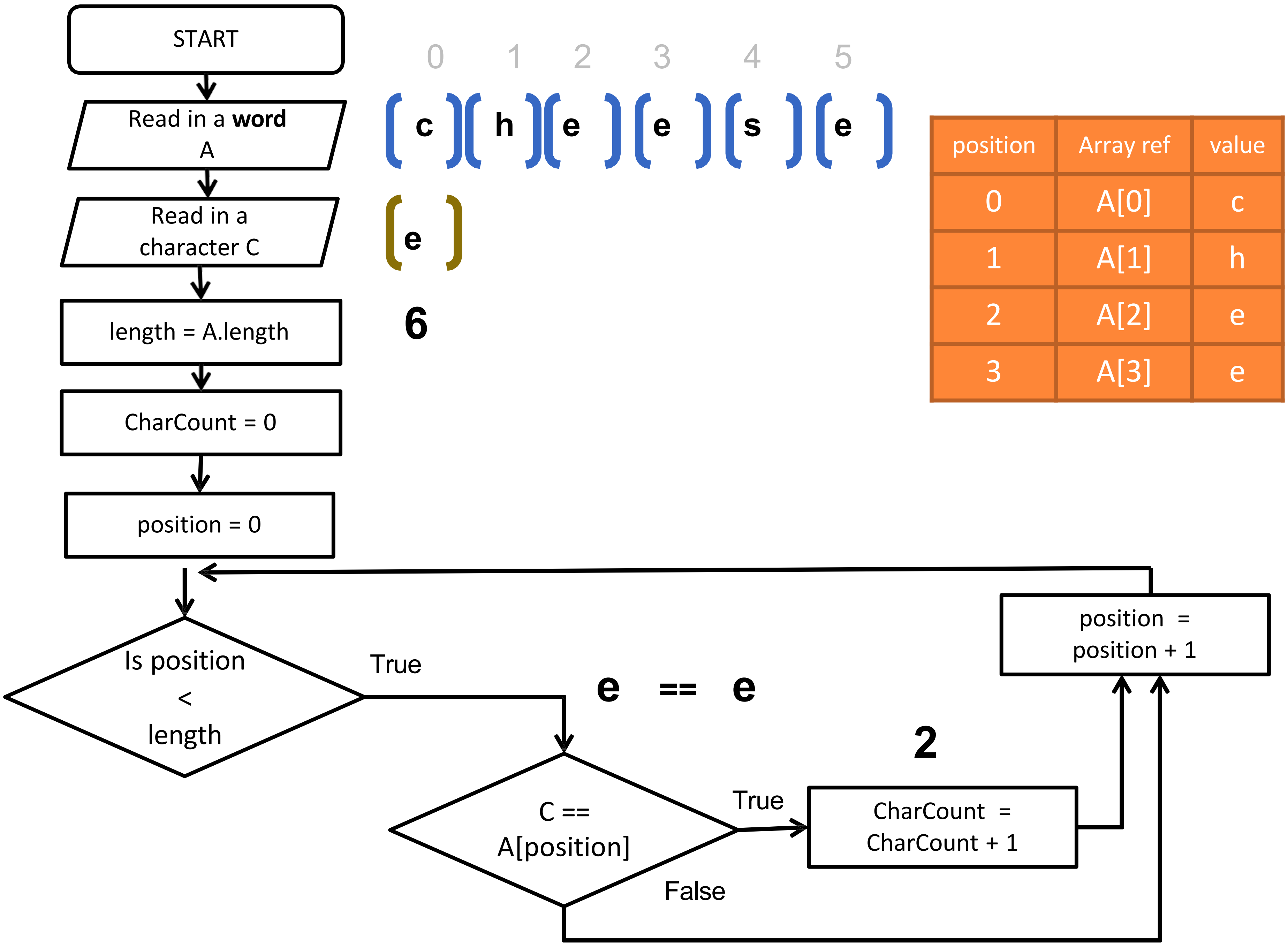
e

6

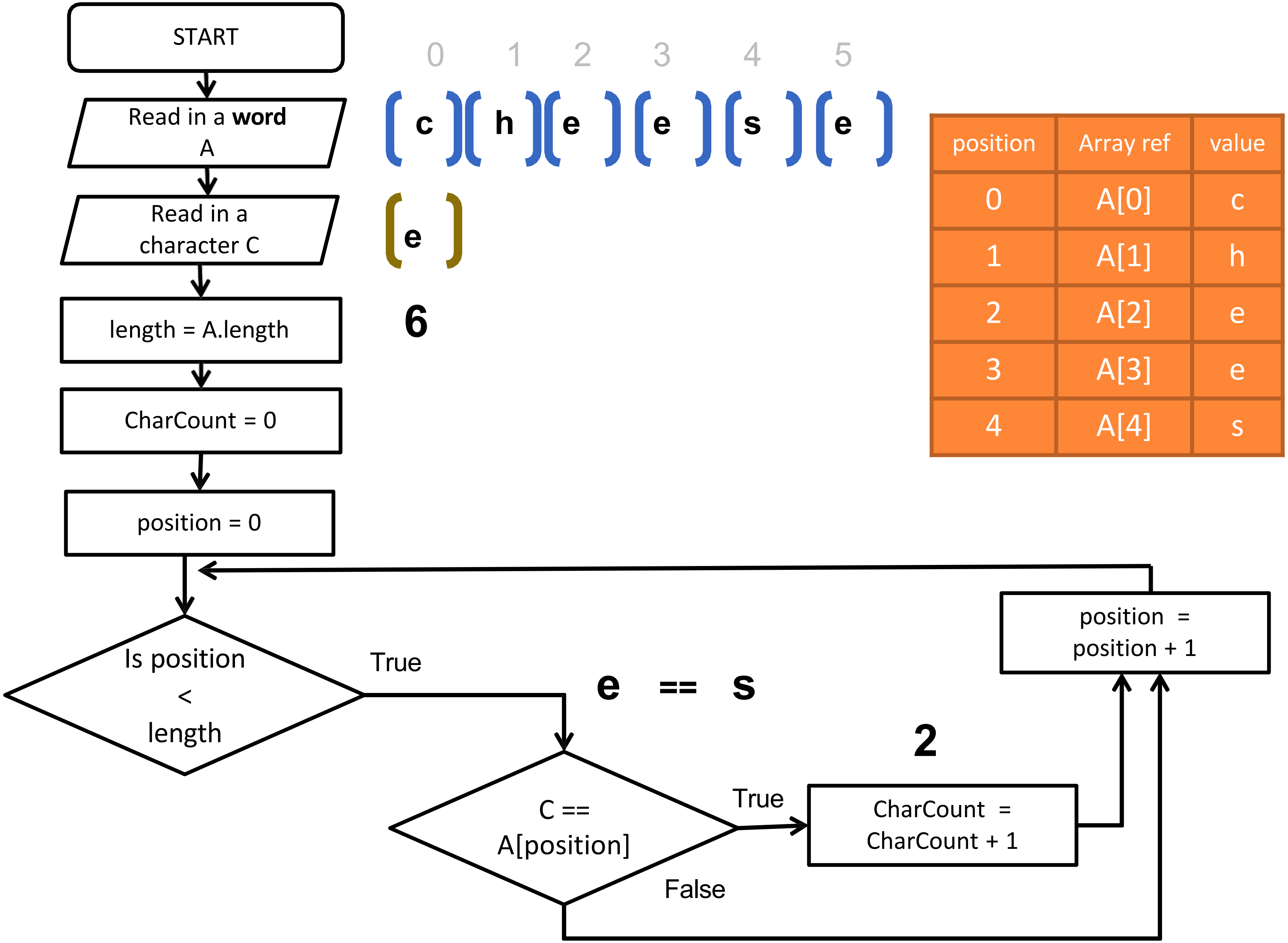
position	Array ref	value
0	A[0]	c
1	A[1]	h
2	A[2]	e

e == e

1



position	Array ref	value
0	A[0]	c
1	A[1]	h
2	A[2]	e
3	A[3]	e

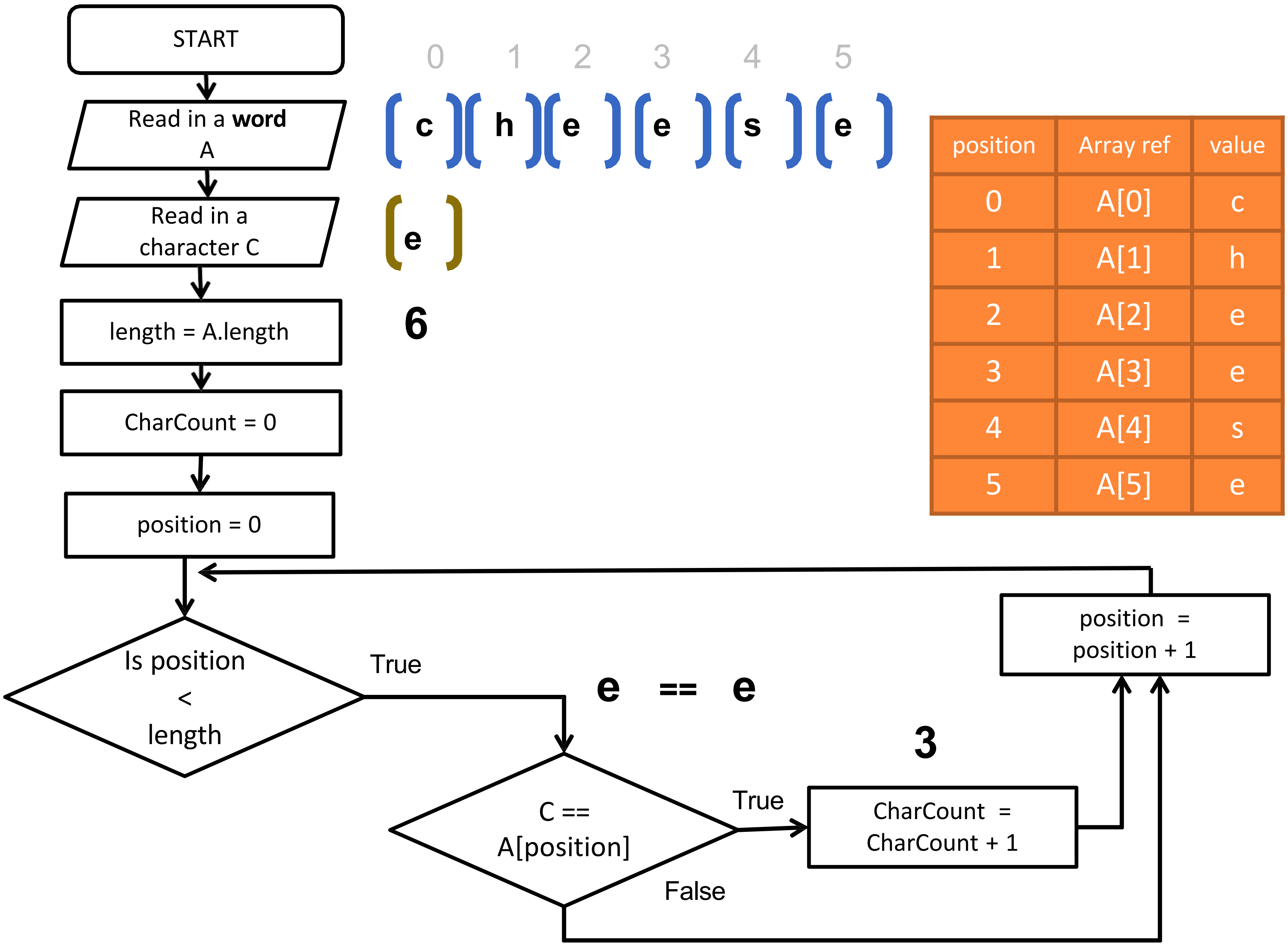


0	1	2	3	4	5
c	h	e	e	s	e

e

6

position	Array ref	value
0	A[0]	c
1	A[1]	h
2	A[2]	e
3	A[3]	e
4	A[4]	s



START

Read in a **word** A

Read in a character C

length = A.length

CharCount = 0

position = 0

Is position < length

True

C == A[position]

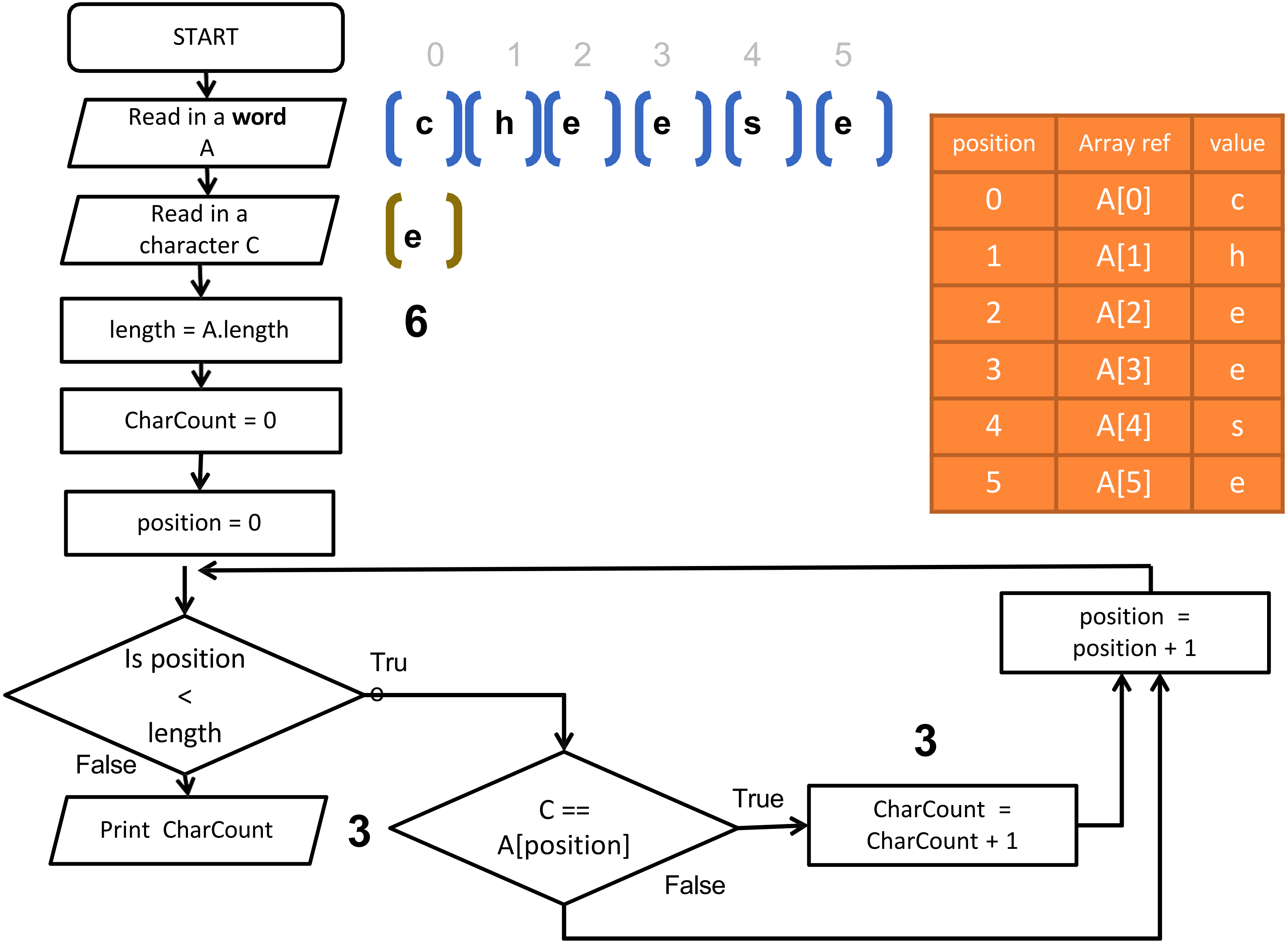
True

CharCount = CharCount + 1

False

position = position + 1

position	Array ref	value
0	A[0]	c
1	A[1]	h
2	A[2]	e
3	A[3]	e
4	A[4]	s
5	A[5]	e



START

Read in a **word** A

Read in a character C

length = A.length

CharCount = 0

position = 0

Is position < length

Tru
e

False

Print CharCount

3

C == A[position]

True

False

CharCount = CharCount + 1

3

position = position + 1

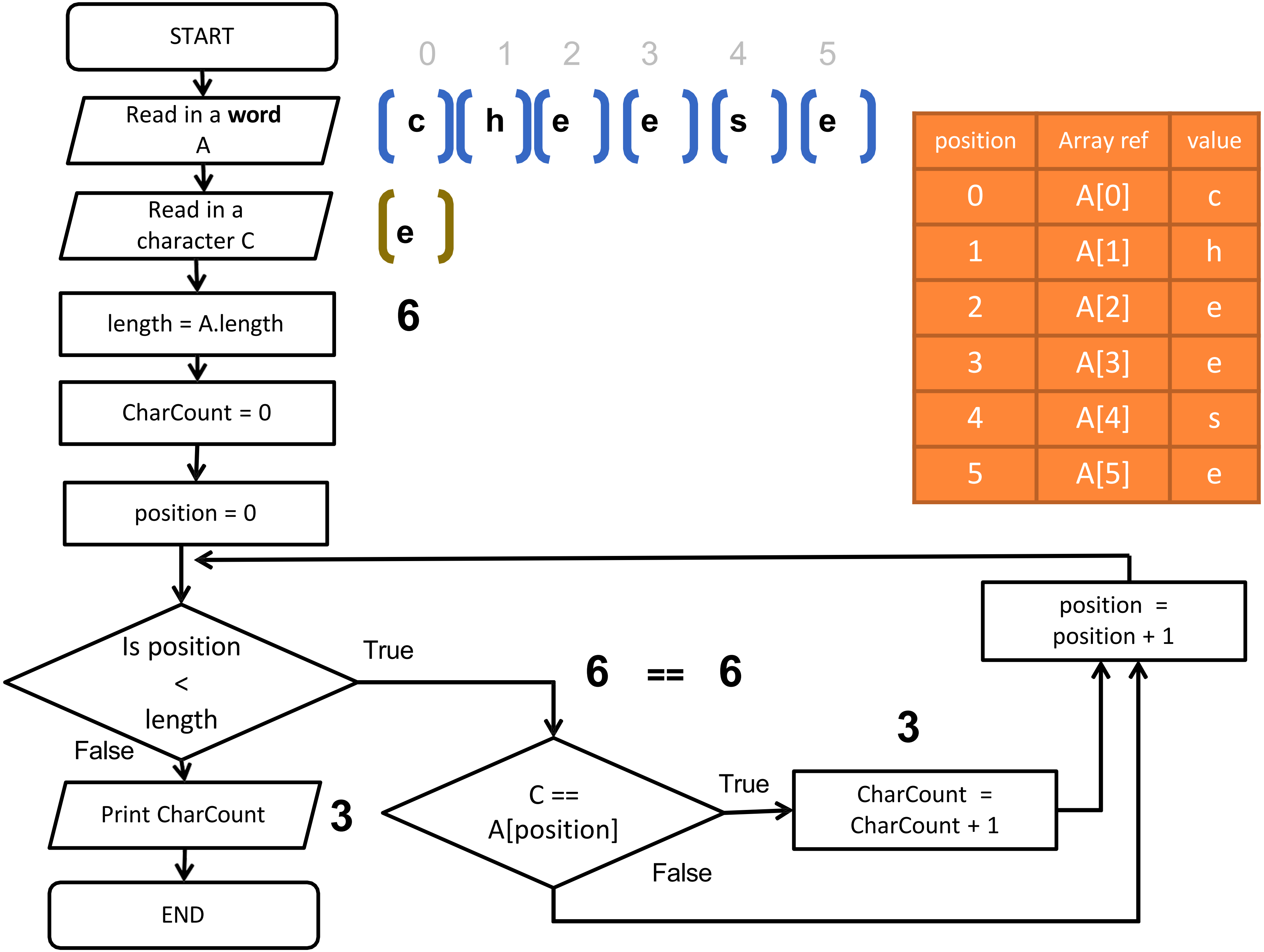
0 1 2 3 4 5

c h e e s e

e

6

position	Array ref	value
0	A[0]	c
1	A[1]	h
2	A[2]	e
3	A[3]	e
4	A[4]	s
5	A[5]	e



START

Read in a **word** A

Read in a character C

length = A.length

CharCount = 0

position = 0

Is position < length

True

False

C == A[position]

True

False

CharCount = CharCount + 1

position = position + 1

Print CharCount

END

position	Array ref	value
0	A[0]	c
1	A[1]	h
2	A[2]	e
3	A[3]	e
4	A[4]	s
5	A[5]	e