Attendance record

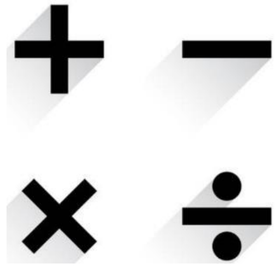
- To check what devices students are using to join class
- To check attendance

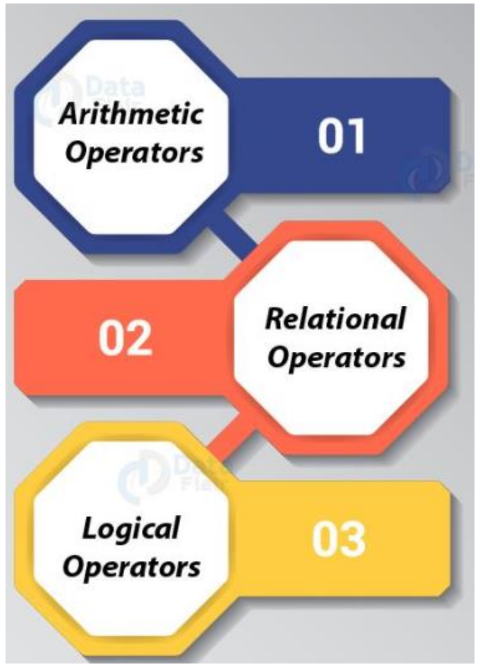
https://forms.gle/8boYn5pFoWChojy77

Start class: **3:05pm**

DATA STRUCTURE & PROGRAMMING I

Chapter 2- Basic Data Type, Operators and Statements





Prepared by: Dr. VALY Dona and Mr. NOU Sotheany

Modified and Lectured by: Mr. BOU Channa

Lecture overview

Overall lectures

Overview

Outline

- Integer
- Real
- Boolean
- Character
  - ASCI code
  - Functions for character
- String
  - Functions for string
- Operators
  - +, -, *, /, DIV, MOD
- Relational operators

Basic data type: Integer

Integer

- Integer is a positive or negative number without fraction number
  - `…, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5,…`
- Declaration
  - `Var <identifier> : Integer`
- Example:
  - `Var age: Integer`
  - `Var x, y, z: Integer`

Basic data type: Integer

Operators for Integer

- Operations: / DIV MOD
  - / : find a result of a division operation
  - DIV : find a quotient of a division operation (not include fractional part)
  - MOD : find a remainder of a division operation
- Operations: + - *
  - + : summation
  - - : subtraction
  - * : multiplication

5 / 2 2.5
5 DIV 2 2
5 MOD 1

%

- Examples:
  - x ← 10+2
  - y ← x-7
  - z ← y/2
  - z ← y MOD 2
  - z ← y DIV 2

```
=> x is 12
=> y is 12-7=5 ✓
=> z is 5/2=2.5
=> z is 5 MOD 2 = 1
=> z is 5 DIV 2 = 2
```

Practice 1

- Let try to predict values of a, b, and c below

```
No.   Suppose:
  1   a ← 5
  2   b ← 8
  3   c ← 10
  4   a ← b MOD 3
  5   b ← b DIV 3
  6   b ← a DIV 3
  7   c ← c DIV 3
  8   b ← c MOD 2
  9   c ← c DIV 4
```

```
Inspecting values:
a is 5, but b and c are unknown
a is 5, b is 8, but c is unknown
a is 5, b is 8 and c is 10
a is 2, b is 8 and c is 10
a is 2, b is 2 and c is 10
a is 2, b is 0 and c is 10
a is 2, b is 0 and c is 3
a is 2, b is 1 and c is 3
a is 2, b is 1 and c is 0
```

Basic data type: Real number

Real number

- Real is a general number including fraction number

  …, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5,…

- Declaration

```
Var <identifier> : Float
Var <identifier> : Double
```

- Examples:

```
Var height: Float
Var a, b, c: Double
```

- Operation: +, -, *, /

```
a ← 10.5 – 2.5          => a is 8.0
b ← a*3.9               => b is 31.2
c ← b/a                 => c is 3.9
```

Boolean

Definition

- Boolean is a logic variable that contains either *true* or *false*
- Declaration

```
Var <identifier>: Boolean
```

- **Examples**:

```
Var is_primary: Boolean
Var p, q, r: Boolean
```

- Operations: AND, OR, NOT

```
Let a, b be boolean values where a
is true and b is false
➢ a AND b : yields true when both
  a and b are true
➢ a OR b : yields true when either
  a or b is true
```

Variable	Value (T:True, F:False)
p	T
q	F
p AND q	F
NOT(p)	F
NOT(NOT(q))	F

Practice 2

Boolean operator

- Example

```
Var p, q, r: Boolean
Begin
        p ← true
        q ← NOT(p)
        r ← (p AND q) OR (NOT(p) AND NOT(NOT(q)))
End
```

```
Value:

p=True
q=False
r=False
```

Period: 3mn

Let a true and b true

c <- a AND b	TRUE
d <- a AND NOT(c)	FALSE
e <- c AND D	FALSE

Boolean

Relational operators

- The relational operators allow you to write relational expressions
- Some relational operators:

  ==, !=, >, <, >=, <=

- The results of relational operators are Boolean
- Syntax:

  expression1 operator *expression2*

- **Examples**:

```
a ← 10
b ← 20
```

Do relational operator	Boolean
a<3	? F
a==b	? F
b<100	? T
a==10	? T
a>=10	? T
a>10 AND b<100	? F
a != 100	? T

Break 5mn

Back: 4:25pm

Character

Definition

- A character is a **single** lowercase/uppercase letter, number, punctuation mark, space, tab, newline and other special operation, which is written in single quote ' '
  - E.g: 'a', 'B', '+', '.', '8', ...

- Declaration

```
Var <identifier>: Character
```

- Examples:

```
Var sex, grade: Character
sex ← 'M'
```

Function	Description	Examples	
		What?	Output
next(*character*)	Give the next character after *character*	next('b')	'c'
prev(*character*)	Give the previous character before *character*	prev('E')	'D'
ord(*character*)	Give the ASCII code of *character*	ord('A')	65
chr(*integer*)	Give the character of the ASCII code *integer*	chr(65)	'A'

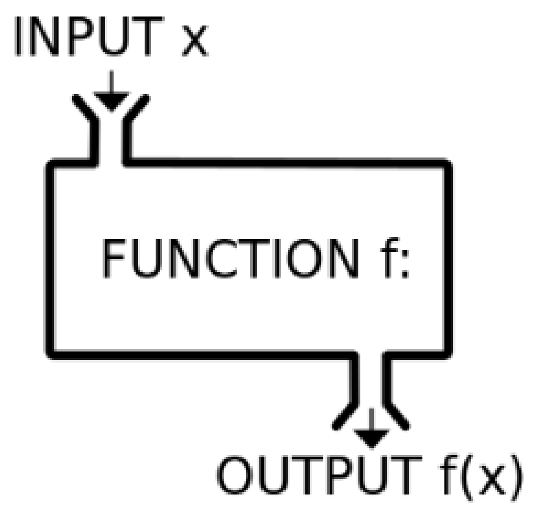
Function

What?

- A function is used to achieve/do something
- A function may take no input/parameter/argument
- A function may take one or more parameters

- A function may be used to calculate and return a value
- A function may return a value or not return any value.
- Once defined, a function can be called anytime

- A function can be:
  - Built-in function : existing function
  - Custom function : user newly defines it (user-defined function)



Character

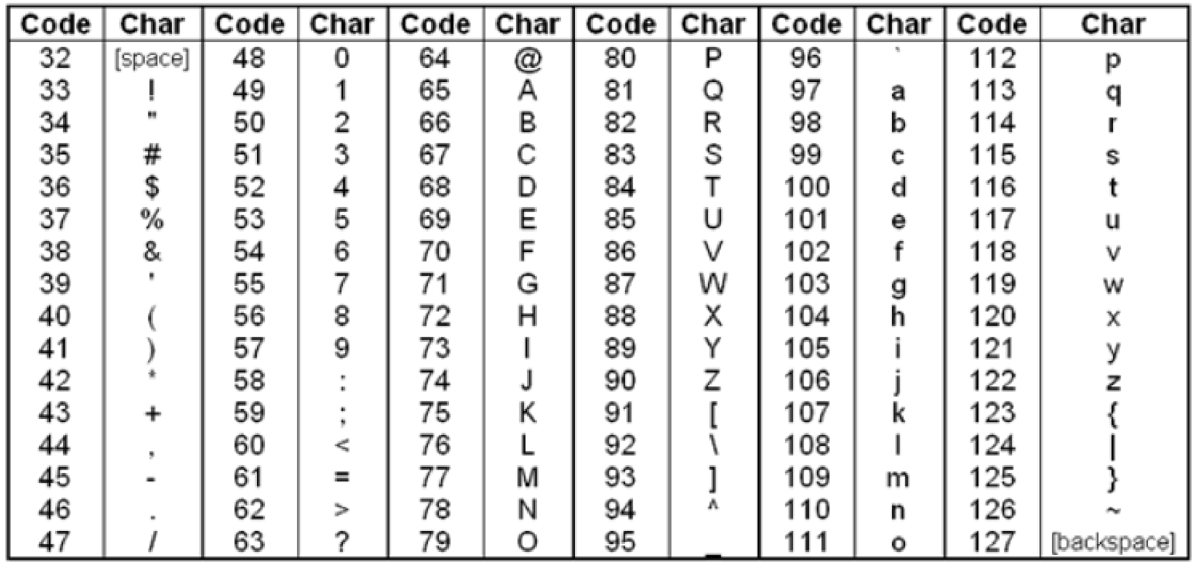
ASCII CODE

- ASCII: American Standard Code for Information Interchange
- Each character has its equivalent numeric code (integer)
- The basic ASCII characters represent in 7 bits
- There are 128 possible characters (from 0 to 127)
- Example

```
Var c: Character
Var n: Integer
Begin
    c ← 'A'
    write(c)
    write('c')
    write(ord(c))
    n ← ord(c)+1
    write(n)
End
```

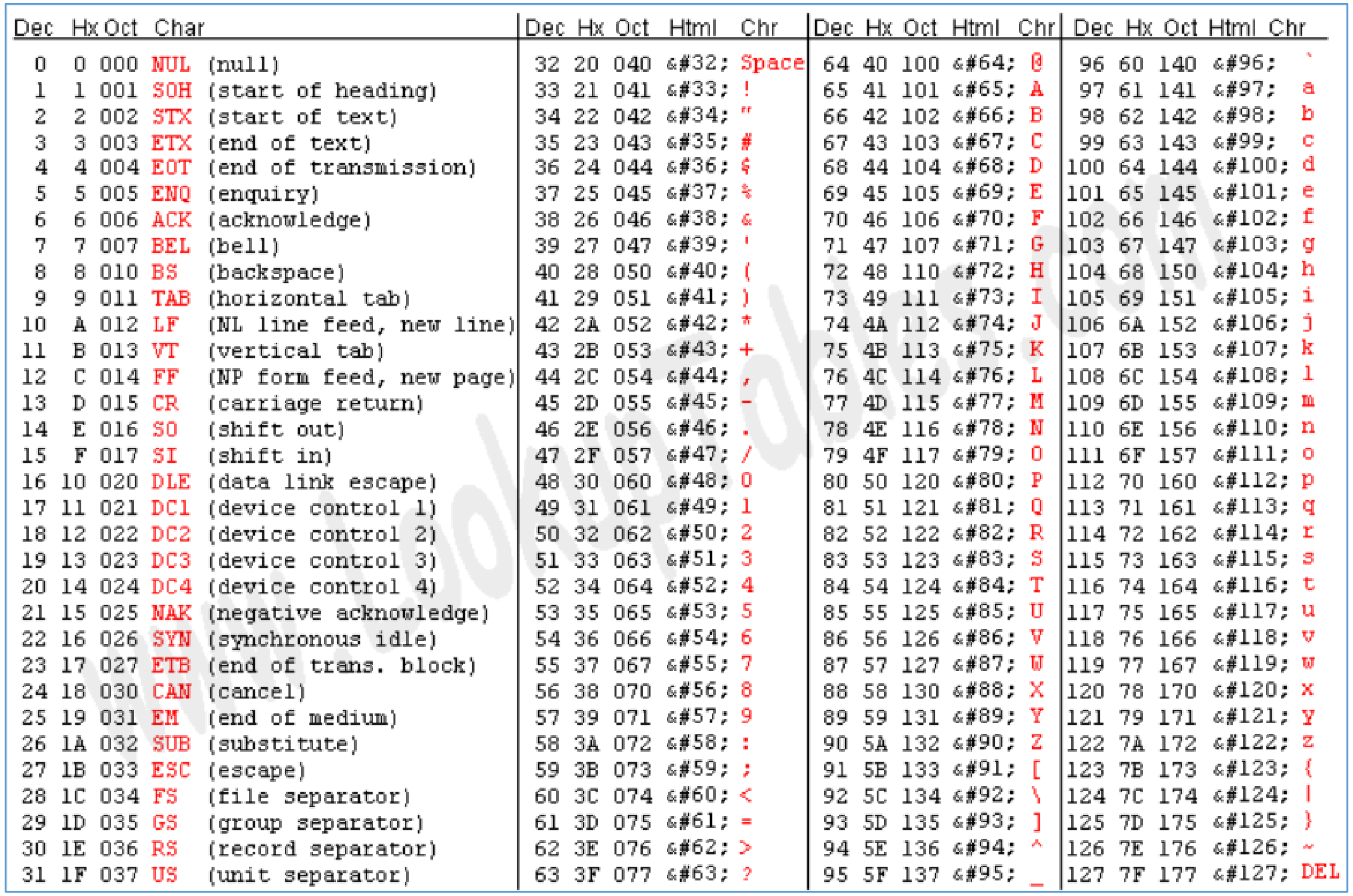
```
Output:



A
c
65

66
```

ASCII Code Table

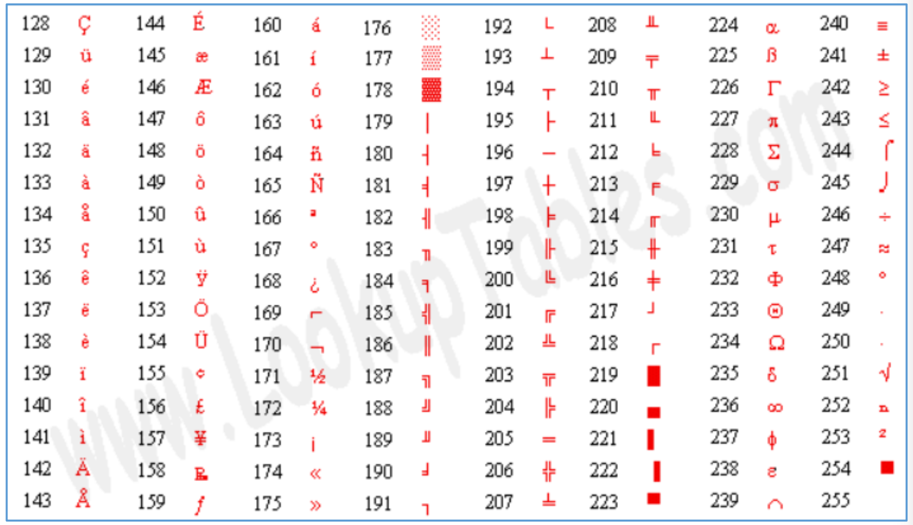


Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

ASCII Code

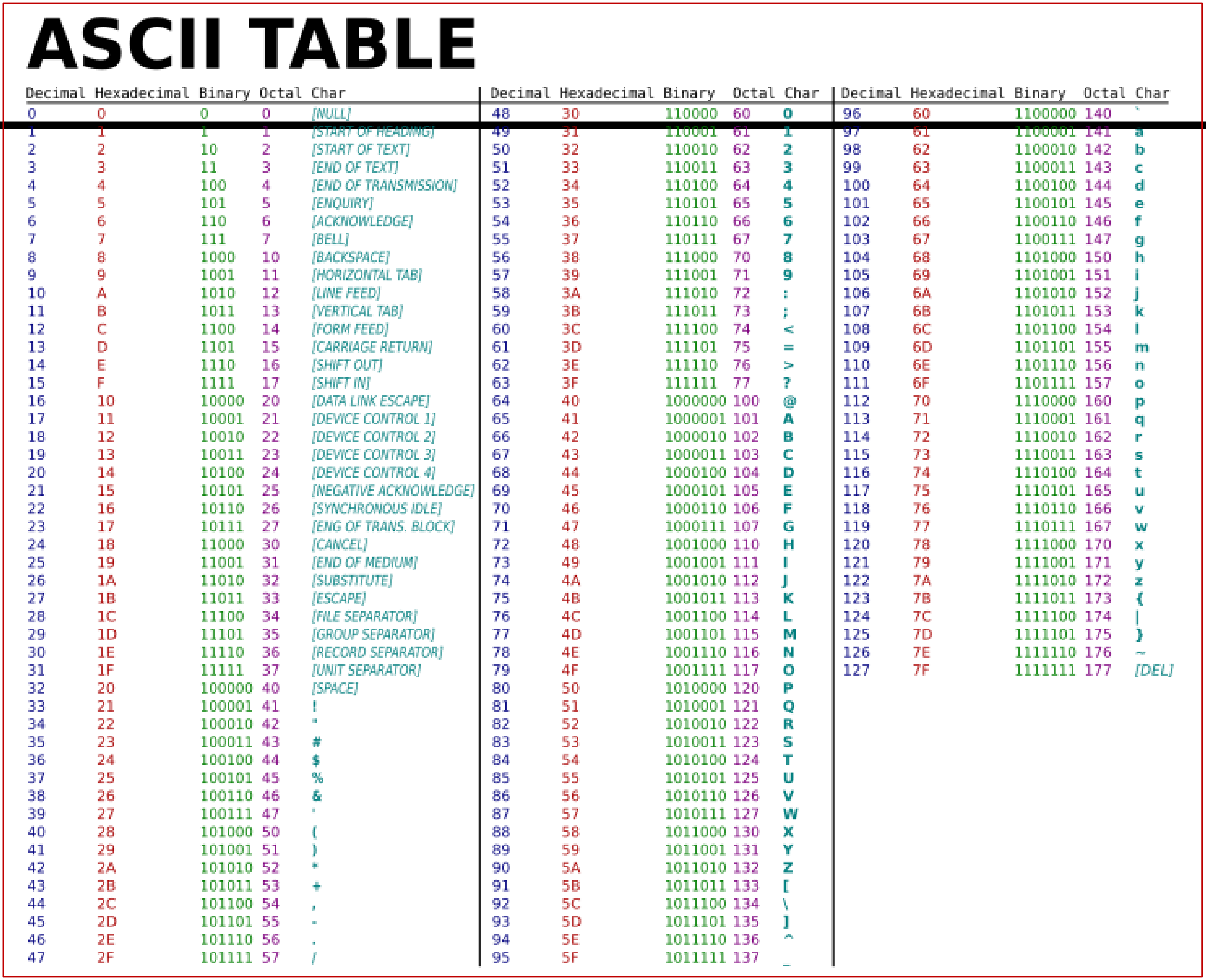


Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL



128	Ç	144	É	160	á	176	░	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

http://www.asciitable.com/



ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

String

Definition

- String is a sequence of character and it is written inside double quotes “ ”
  - E.g: “Hello”, “1234”, “I’m fine”
- Declaration:
  - `Var identifier: Sequence of character`
- Examples
  - `Var name, surname: Sequence of character`
  - `name ← “Marry”`
  - `write(name)                 “Marry”`
  - `write(name[0])              ‘M’`

index	0	1	2	3	4
name	M	a	r	r	y

String

Functions

Function	Description	Examples	
		E.g:	Output
concat(*string1, string2*)	concatenate two strings together	concat(“Hel”, “lo”)	“Hello”

length(*string*)	Find number of character in *string*	length(“hello ”)
		length(“”)
		length(concat(“Thank”, “ you”))
			6
			0
			9

substring(*p*, *n*, *string*)	Find substring positioning at *p* upward for *n* characters from a *string*	substring(2, 4, “Algorithm”)
		substring(0, 5, “I am fine”)
		substring(6, 1, “I am find”)
			“gori”
			“I am ”
			“i”

identifier[*index*]	Find a character positioning at *index*	name ← “Marry” name[0]	 ‘M’

index	0	1	2	3	4
name	M	a	r	r	y

String

Functions

Function	Description	Examples	
		E.g:	Output
stringcomp(*string1, string2*)	Compare two strings • *0: if they are same* • *Otherwise, they are different*	stringcomp(*“Hi”, “Hi”*)	0
stringcopy(*des, source*)	Copy a string	Var var1: Sequence of character stringcopy(*var1, “Hi”*)	“Hi”
string2lowercase(*string)*	Convert string into lowercase	string2lowercase(“Hello”)	“hello”
string2uppercase(*string)*	Convert *string* into uppercase	string2uppercase(“Hello”)	“HELLO”

Summary

Revision

▪ Integer	: non-fractional number `Integer`
▪ Real	: real number `Float`, `Double`
▪ Character	: a single character written in a single quote ‘ ’
▪ Boolean	: true or false value
▪ String	: a sequence of character written in a double quote “ ”
▪ Operators	: + - * / DIV MOD
▪ Relational operators	: == > < >= <= !=
▪ Functions for Character	: `next`, `prev`, `ord`, `chr`
▪ Functions for String	: `concate`, `length`, `substring`, `stringcmp`, `stringcopy`, `string2lowercase`, `string2uppercase`

Practice

Exercises

1. Write an algorithm to determine DIV and MOD of two input numbers from a user
2. Write an algorithm to calculate the summation of two input of real numbers from a user and identify the integer part and fractional part
3. Suppose we have: a true, b false. What is the value for each of:
   - NOT(a), NOT(b), a OR b , NOT(a) OR NOT (b), NOT(a) AND b, a AND b
4. Write an algorithm that allows a user to input two string. Copy the second string to the first one then covert first string to uppercase.

Practice

- Suppose we have: a true, b false. What is the value for each of:
  - NOT(a), NOT(b), a OR b , NOT(a) OR NOT (b), NOT(a) AND b, a AND b

- Write an algorithm that allows a user to input two string. Copy the second string to the first one then covert first string to uppercase.