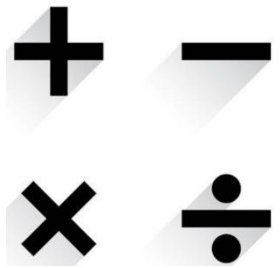
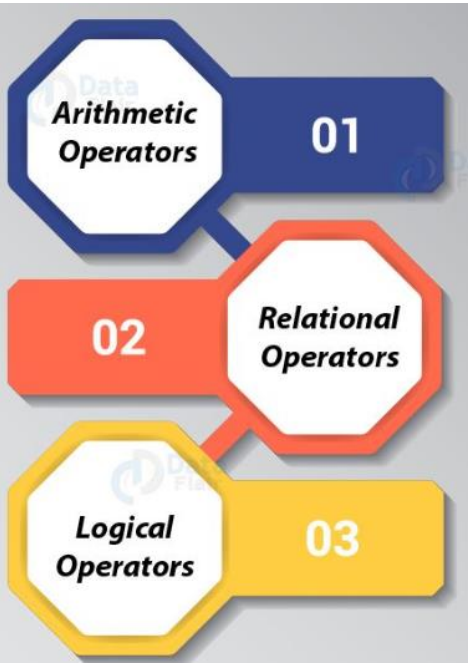
DATA STRUCTURE & PROGRAMMING I

Topic 2: Variable, Basic Data Type, Operators and Statements





Variable/Identifier

Variable/Identifier

What and Why?

- A **variable** (also called **identifier**) is a holder of value which can be used and changed throughout the program
- It is used to store value and its value can be changed during the program run

- **Why need variable?**
  - Understandability and Readability
  - Maintainability
  - Ex: Exchange rate computation

```
t1 = a1 * 4100
t2 = a2 * 4100
t3 =  a3 * 4100
```



```
rate = 4100
t1 = a1 * rate
t2 = a2 * rate
t3 =  a3 * rate
```

What if the multiplier is changed to 4050?
=>You will need to change 3 times.
What if you have 100 lines using the multiplier (4100)?
=>Change 100 times ???

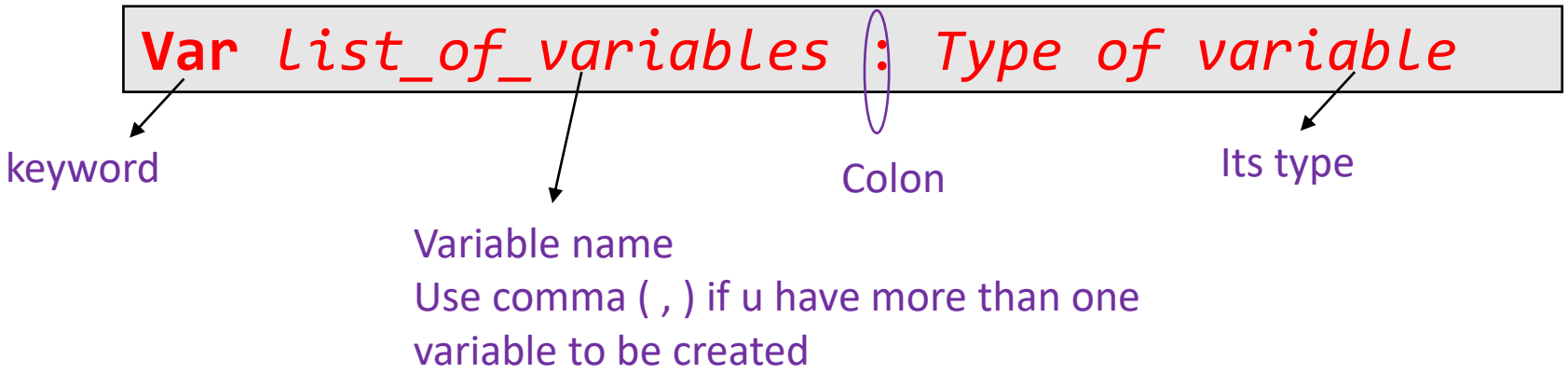
What if the multiplier is changed to 4050?
=>You just need to update value of **rate** to 4050

Variable/Identifier

How to create a variable?

- A variable is created by variable declaration.
- We **declare a variable** in order to reserve memory space for it

- Syntax to declare a variable (for writing algorithm)



Variable/Identifier

Basic types of variables

- **Number**
  - **Integer** : A non-fractional number, e.g: 5, 8, 9, 199
  - Float : A non-fractional and fractional number (real number), e.g: 4, 4.0 or 5.8
- **Character** : A single letter. It is placed inside a single quote, e.g: 'M'
- **Sequence of character** : A string (a set of letters). It is placed inside a double quote, e.g: "Sok"
- NOTE: Different types of variables consume different memory space

- Examples of variable declaration:

```
Var gender: Character
Var age, number: Integer
Var name, title: Sequence of character
Var person_weight: Float
```

Variable or Identifier

Naming a variable

Rules of giving a variable name

- ✓ Start with a letter (small or capital is okay) or underscore
- ✓ Not start with number or any symbol (+, -, @, #, !, &, *, /, ., …. etc)
- ✓ Don't use name that already exist with reserved keywords
- ✓ No space is allowed.
  - If the name is long, use underscore (_ , person_weight) or camel case (myVar, personWeight)

- Naming convention (good practice)
  - Name of a variable should reflect the value it will store
  - The name should be readable and not too s

- Examples:

```
Var age: Integer
Var price: Float
Var name: Sequence of character
Var _tel: Integer
Var studentName: Sequence of character
Var work_position: Sequence of character
```

```
Var 7age: Integer
Var @price: Float
Var +name: Sequence of character
Var *tel: Integer
Var &student Name: Sequence of character
Var work Position: Sequence of character
```

Variable or Identifier

NOTE: Naming a variable

- Name of a variable is **case sensitive** (Ex: age, Age, AGE are different names)

```
Var age: Integer
Var Age: Integer
Var aGe: Integer
Var AGE: Integer
```



- Each variable can't have the same name
  - Once a name is given to a variable, that name can't be used for naming another variable

```
Var sex: Character
Var sex: Sequence of character
Var price: Integer
Var price: Float
```



Variable or Identifier

Assign value to a variable

- Assign value to a variable **means that** you give a value to that variable
- Syntax for assigning a value to a variable:
  - *<variable>* ← *<value* **or** *other variable (same type)* **or** *expression>*
- The type of value assigning to the variable must be the same as variable's type
- A variable can be used once a value was assigned to that variable
- Examples:

```
Var name, name2 : Sequence of character
Var val : Integer
name  ← "Sok"
name2 ← "Sao"
name  ← name2
val   ← 5
val   ← val*2
```



```
Var name, name2 : Sequence of character
Var val : Integer
name  ← "Sok"
name  ← "name2"
name  ← name2
val   ← name
val   ← "Dara"
```



Algorithm

Constant

- It is a non-changeable value (its value is fixed for all algorithm)
- Syntax for declaring a constant:

```
const(identifier: Type) ← value or expression
```

keyword — name — Its type — Fixed value or result of an operation

- Examples:

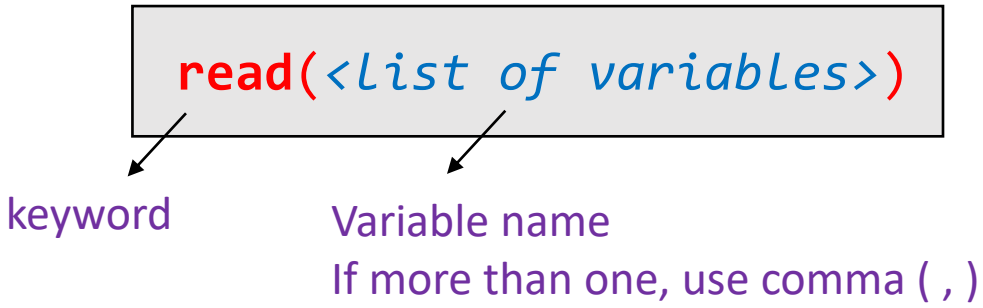
```
const(YEAR: Integer) ← 2018
const(NEXT3YEAR: Integer) ← YEAR+3
const(EXCHANGE_RATE: Integer) ← 4100
```

- Rule for naming a constant and naming convention)
  - Same as variables' rules, **but**
  - All letters should be capitalized

Algorithm

Entering data / Get user's inputs

- Syntax for entering a data:



- It is an instruction to get input from user and assign to a variable
- **Examples:**
  - `read(number)` : ask a user for an input and store it in a variable named `number`
  - `read(name, surname)` : ask a user for two inputs and store it in the variables namely `name, surname`
  - `read(age)` : ask a user for an input and store it in the variable named `age`

Algorithm

Displaying the information or data

- It is an instruction to display information
- Syntax for displaying a data:

```
write("static text", <variables, constants, or expression>)
```

keyword

Display what you write
(placed inside double quotes)

Placeholder/value
(just put a variable name or constant name or an expression/operation)

- Examples:
  - `write("Enter a number: ")`
  - `read(number)` `//Suppose user inputs number 7 here`
  - `write("This is the input from user: ", number)`
    - This is the input from user: 7
  - `write("The sum of ", number, " and 3 ", "is ", number+3)`
    - The sum of 7 and 3 is 10

Examples of Algorithms

How to add two numbers

- Writing an algorithm for adding input two numbers. Then display the result.

```
Var num1, num2: integer
Var result: integer
Begin
    write(“Enter the first number: ”)
    read(num1)
    write(“Enter the second number: ”)
    read(num2)

    result ← num1 + num2
    write(“The sum is: ”, result)
    write(“The sum of ”, num1, “ and ”, num2, “is: ”, result)
End
```

Output: …?

More Examples of Writing Algorithms

Ex1: Calculate a given price with tax

- Suppose that the tax is 3% and a user is required to input a price. As a result we calculate and display the final price including the tax.

```
Var .........: …Type…
Begin

        ....



End
```

More Examples of Writing Algorithms

Ex1: Calculate a given price with tax

- Suppose that the tax is 3% and a user is required to input a price. As a result we calculate and display the final price including the tax.

```
const(TAX: int) ← 3
const(TITLE: string) ← "Result"
Var price, priceWithTax: float
Begin
    write("Give me the price exclude tax:")
    read(price)
    priceWithTax ← price + (price*TAX)/100
    write(TITLE)
    write(price, "dollars exclude tax.", priceWithTax, "dollars include tax")
End
```

Output: …?

Examples of Algorithms

Ex2: Student information

- Ask a student's information (name, department, birth's year). Then greet him/her with the following information:

```
Enter your name:
Which department are you from?:
Which year were you born?:

  Hello <name>! Welcome to <department>.
  You are <age> years old.
```

```
Var name, department: Sequence of characters
Var year_birth: Integer
Var age: Integer
Begin
    write("Enter your name: ")
    read(name)
    write("Which department are you from?: ")
    read(department)
    write("Which year were you born?": )
    read(year_birth)

    age ← 2021 – year_birth
    write("Hello", name, "! Welcome to ", department, ".")
    write("You are ", age, "years old.")
End
```

Examples of Algorithms

Ex3: Currency exchange program

- Write an algorithm for computing an exchange rate USD-Riel. Suppose that 1USD= 4100 R. A user is required to input an amount of money in USD then a program will convert it into riel currency and display the following information

```
How much in US dollars do you want
to exchange?:
   Your exchange amount of
   <amountUSD> dollars are equal to:
   <amountRiel> when the exchange
   rate is 1 USD = 4100 riels
```

```
CONST(RATE: Float) ← 4100
Var amountUSD: Float
Var amountRiel: Float
Begin
    write("How much in US dollars do you want to exchange?: ")
    read(amountUSD)
    amountRiel ← amountUSD * RATE
    write("Your exchange amount of ", amountUSD, " dollars are
         equal to: ", amountRiel, "when the exchang rate is
         1 USD = ", RATE, "riels")
End
```

Practise

Write an algorithm for each of the question below:

1. Read a last name and first name from a user. Then display a phrase as follows:

```
What is your last name?
What is your first name?
      Welcome <lastname> <firstname>!
```

2. Read a number from a user and calculate square of that number then display its result.

```
Enter a number:
     The square of <number> is ….
```

3. Read two numbers (say n1 and n2) from a user then display their summation, subtract, and multiplication.

```
Enter the first number:
Enter the second number:
     The summation of <n1> and <n2> is: …
     The subtraction of <n1> and <n2> is: …
     The multiplication of <n1> and <n2> is: …
     The division of <n1> and <n2> is: …
```

Practise

Write an algorithm for each of the question below:

4. A program to ask user for firstname, lastname and department. Then display this message:

   Welcome to *department*, *lastname* *firstname*!

5. Ask a user to input height and base of a triangle. Calculate the surface of this triangle and display.

6. Ask a user for a, b and c length of a triangle. Calculate its surface using heron formula.

Types of Variables

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

- **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

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:

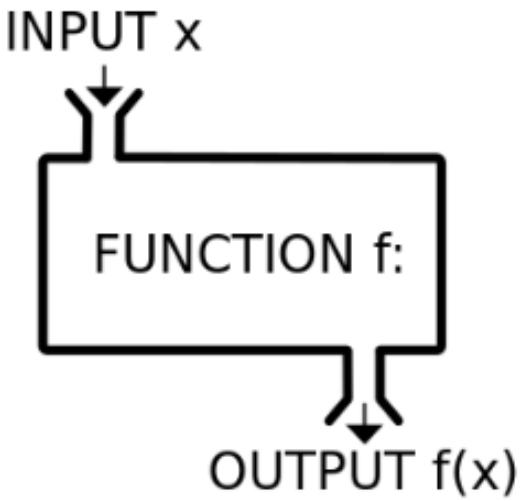
```
Var sex, grade: Character
sex ← ‘M’
```

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

www.lookuptables.com

http://www.asciitable.com/

ASCII TABLE

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

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`

Q&A

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

Practice

- 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

1. Write an algorithm to ask for firstname and lastname from a user. Then make another variable username. This variable is the result of contatenation between firstname, underscore (_), and lastname.

   E.g: If input for firstname and last name are Jack and Rose, then username should be Jack_Rose

2. Write an algorithm to convert an input string, entered by a user, to uppercase and lowercase. E.g: If a user inputs the word ***Apple***, then the output of the program should be ***APPLE, apple***

3. Write an algorithm to find the length of an input string. E.g: If a user inputs the word ***Book***, then the output of the program should be ***The length of the word "Book" is 4.***

Practice

4. Write an algorithm to display the first character and the last character of an input string entered by a user. E.g: If a user inputs the word *Banana*, then the output of the program should be *In word "Banana", the first character is 'B' and the last character is 'a'*

5. Suppose we have: a=50, b=-50, then what is the Boolean value of each expression below

   - **A)** a>50, **B)** b>=50, **C)** a==50 AND b==50, **D)** a==50 OR b==50,
   - **E)** a>50 AND b<0, **F)** a<0 OR b<0, **G)** a>0 AND b<0,
   - **H)** NOT((NOT(a>50) AND (a>0)) AND NOT(b<0))

6. By taking a look at the ASCII code tables, answer the question below:

   - What are the ASCII codes for the letters a, b, c, …., z ?
   - What are the ASCII codes for the letters A, B, C, …, Z?
   - What are the ASCII codes for the numbers 0, 1, 2, …, 9?
   - What are the ASCII codes for these characters é è ç É È ?

Examples in C programming using string functions

#include <string.h>

String functions

s1&s2 are strings (sequence of character)

- strupr(s1) : convert string to uppercase
- strlwr(s1) : convert string to lowercase
- strlen(s) : find length of a s
- strcpy(s1, s2) : copy string
- strcat(s1, s2) : concatenate string
- strcmp(s1, s2) : compare strings
- strrev(s1) : reverse string

here | string uppercase and lowercase.c

```c
#include<stdio.h>
#include <string.h>
int main(){
    char str[20];

    printf("Enter a string: ");
    scanf("%s", &str);  //read string without space
    //gets(str);//reads string even u have space

    printf("String is: %s",str);
    printf("\nUppercase is: %s",strupr(str));
    printf("\nLowercase is: %s",strlwr(str));
}
```

Examples in C programming using string functions

#include <string.h>

String functions

s1&s2 are strings (sequence of character)

- strupr(s1) : convert string to uppercase
- strlwr(s1) : convert string to lowercase
- strlen(s) : find length of a s
- strcpy(s1, s2) : copy string
- strcat(s1, s2) : concatenate string
- strcmp(s1, s2) : compare strings
- strrev(s1) : reverse string

string function.c

```
2  int main(){
3      char name[20]="Dara";
4      char name2[20];
5
6      //name2="Sok"; error
7      //name2 = name; //error
8
9      strcpy(name2, name);
10     printf("Hi %s %s!",name,name2);
11     printf("\n\n\n");
12
13     //error, can not assign string
14     //Use strcpy instead
15 }
```

Examples in C programming using string functions

#include <string.h>

String functions

s1&s2 are strings (sequence of character)

- strupr(s1) : convert string to uppercase
- strlwr(s1) : convert string to lowercase
- strlen(s) : find length of a s
- strcpy(s1, s2) : copy string
- strcat(s1, s2) : concatenate string
- strcmp(s1, s2) : compare strings
- strrev(s1) : reverse string

string function.c

```c
#include<stdio.h>
int main(){
    char name[20]="Dara";
    char name2[20];

    //name2="Sok"; error
    //name2 = name; //error
    //error, can not assign string
    //Use strcpy instead

    strcpy(name2, name);
    printf("Hi %s %s!",name,name2);
    printf("\n\n");

    strcat(name2, " Hi!");
    printf("Hello %s",name2);
}
```

Assignment

Deadline: 1 week

1. Write a C program to concatenate a string. The user is required to input the name of a student and the name of a course enrolled by that student. For example, the message sample is defined as follows:

   Welcome *studentName* to the course *courseName*.

2. Write a C program to convert an input string, entered by a user, to uppercase and lowercase. E.g: If a user inputs the word *Apple*, then the output of the program should be *APPLE, apple*

3. Write a C program to find the length of an input string. E.g: If a user inputs the word *Book*, then the output of the program should be *The length of the word "Book" is 4.*

4. Write a C program to display the first character and the last character of an input string entered by a user. E.g: If a user inputs the word *Banana*, then the output of the program should be *In word "Banana", the first character is 'B' and the last character is 'a'*