Mahidol University

ITCS113
Fundamentals of Programming

Lecture 1

Instructor: Asst. Prof. Dr. Akara Supratak

Contact: akara.sup@mahidol.edu

Course Overview

- Lecture 2 hr. + Lab 2 hr.
- Objectives: Teach you the basics of computer programming starting from preparing environments to a complete program.
- Learning Platform
  - Lecture: MyCourse
  - Quizzes: MyCourse
  - Labs: Github Classroom
  - Announcements: MyCourse Email
- Textbook
  - Paul Deitel and Harvey Deitel, C How to Program, Global Edition, 2016
  - Bronson, Gary J., and Andy Hurd. A first book of ANSI C. Brooks/Cole, 2001.

Course Overview

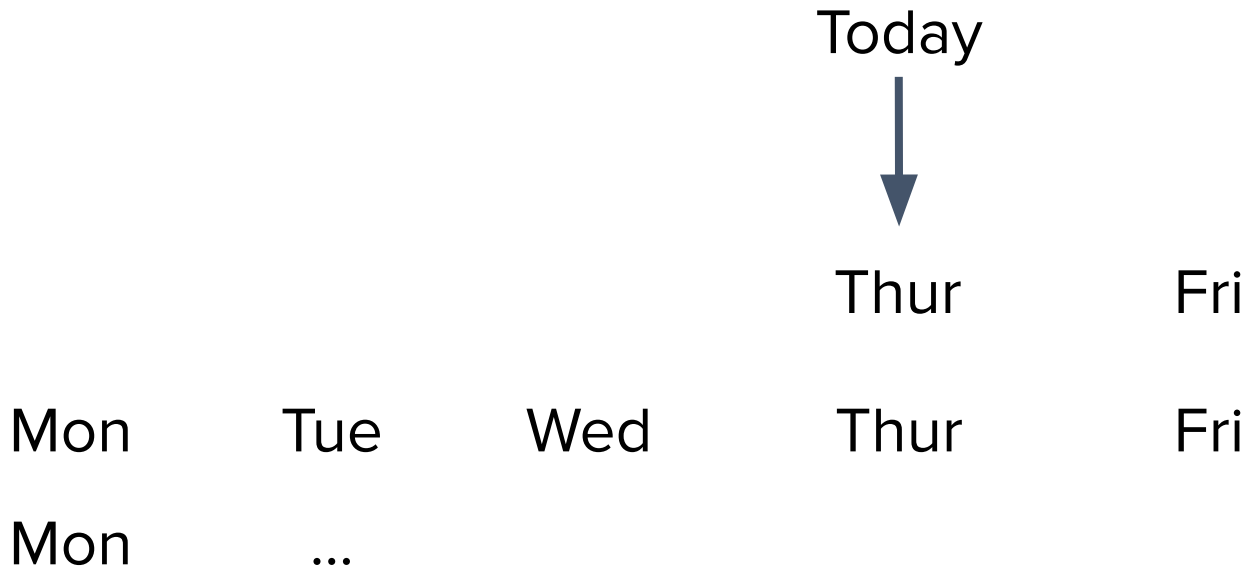
- To pass this subject, you need a C grade
- Closed book quizzes and exams
  - A4 paper + Provided cheatsheet
- Course Syllabus
  - Check on MyCourse Website

Evaluation

- Lab (12): 15%
- Quiz (6): 25%
  - 5% for each quiz
  - We will consider your top 5 quizzes
- Midterm Exam: 25%
- Final Exam: 35%

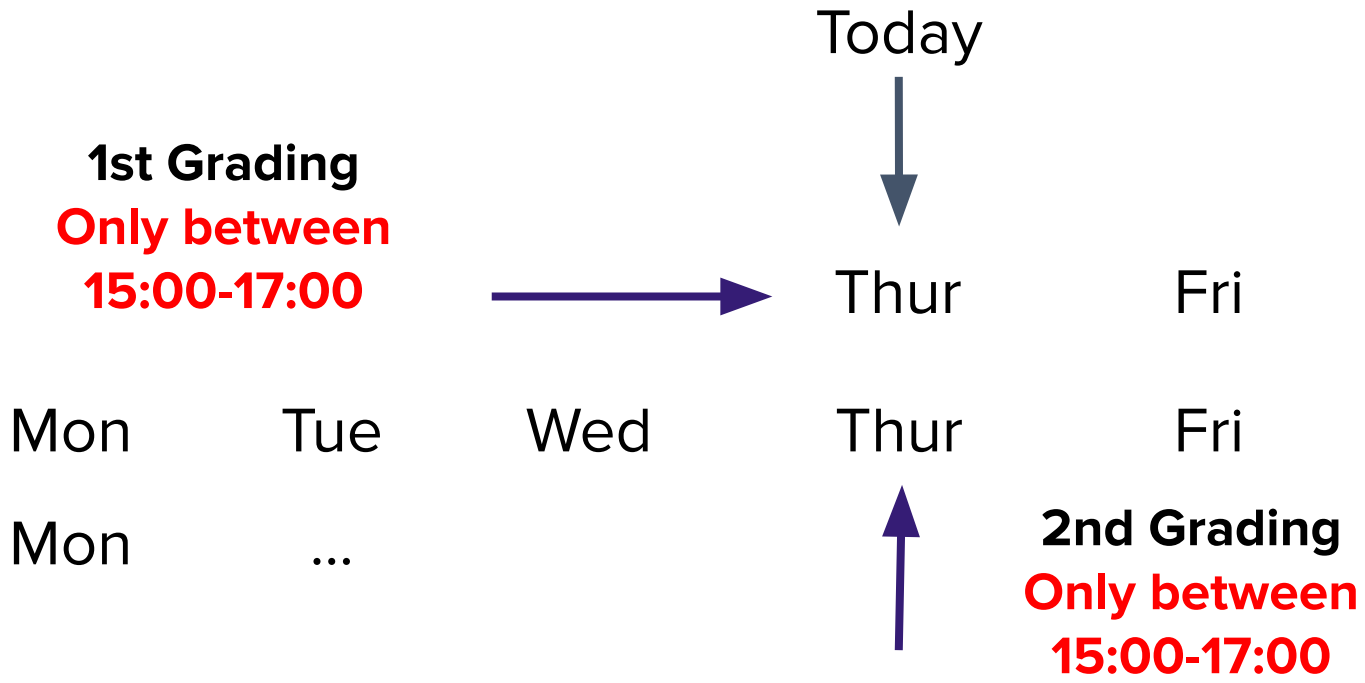
Lab Assignments

- Every week you have approx. 2 hour to do the lab assignment in class and can submit no later than the end of the next lab exercises.



Lab Assignments

- Every week you have approx. 2 hour to do the lab assignment in class and can submit no later than the end of the next lab exercises.



If you miss the 2nd grading, you will get ZERO scores for that lab.

Lab Assignments

To get a full score for each question:

- Your code MUST pass all *test cases* by running the provided script.
- Your code CAN be compiled using GCC and CAN be run via command line (i.e., no errors).
- You CAN answer questions from LAs.
- Follow instructions correctly.

We will check for code similarity.

- If we found that your code is similar to your friend (i.e., you copy the solution from your friends), you and your friends will automatically receive a **ZERO** score as a minimum penalty (more severe if repeated).

Grading:

- 2 = Complete and can answer LA's questions
- 1 = Submitted
- 0 = No submission

Environment

- **Text editors** (recommend VS Code): write your code offline
- **GCC Compiler**: compile your code
- **Github Classroom**: write and test your code with the provided test cases

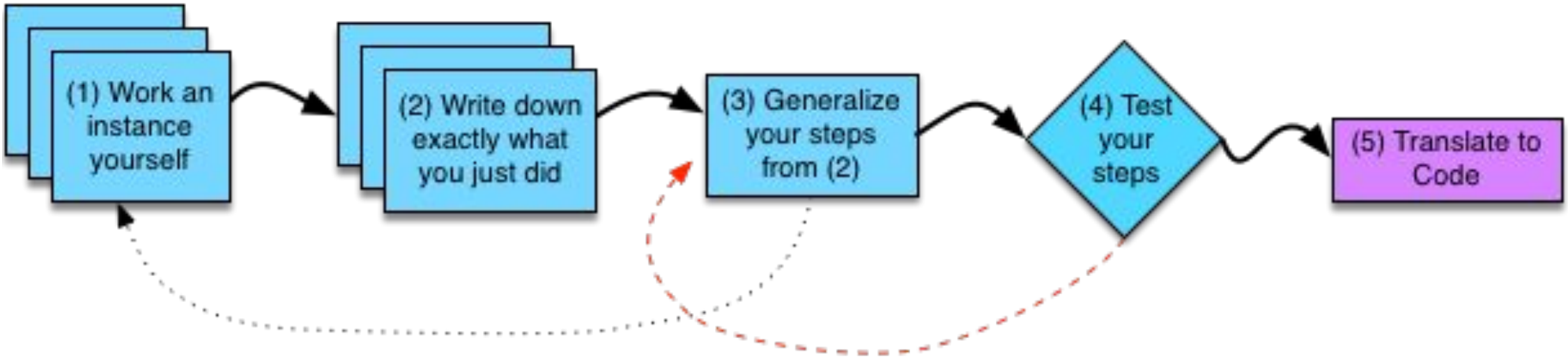
We BAN Assistive Generative AIs

- This course BAN ALL assistive generative AIs
  - e.g., ChatGPT
- If we found out that you use any of them in any lab exercises, quizzes and exams, you will face VERY severe punishments.

Basic C Program

Step to Compile and Run Program

Tips for Designing Algorithms



ref: https://www.coursera.org/learn/programming-fundamentals

Structure of a Basic C Program

Data Types

Data Types

Integer Data Types: `int`

- `int` data type
- **Declaration**: `int var_name;`
  - Whole numbers and + or – signs
  - Values between -2,147,483,648 to 2,147,483,647

- *Example*
  - Valid integer constant:

```
5     -10     +25     1000
253   -26351  +36
```

  - Invalid integer constant:

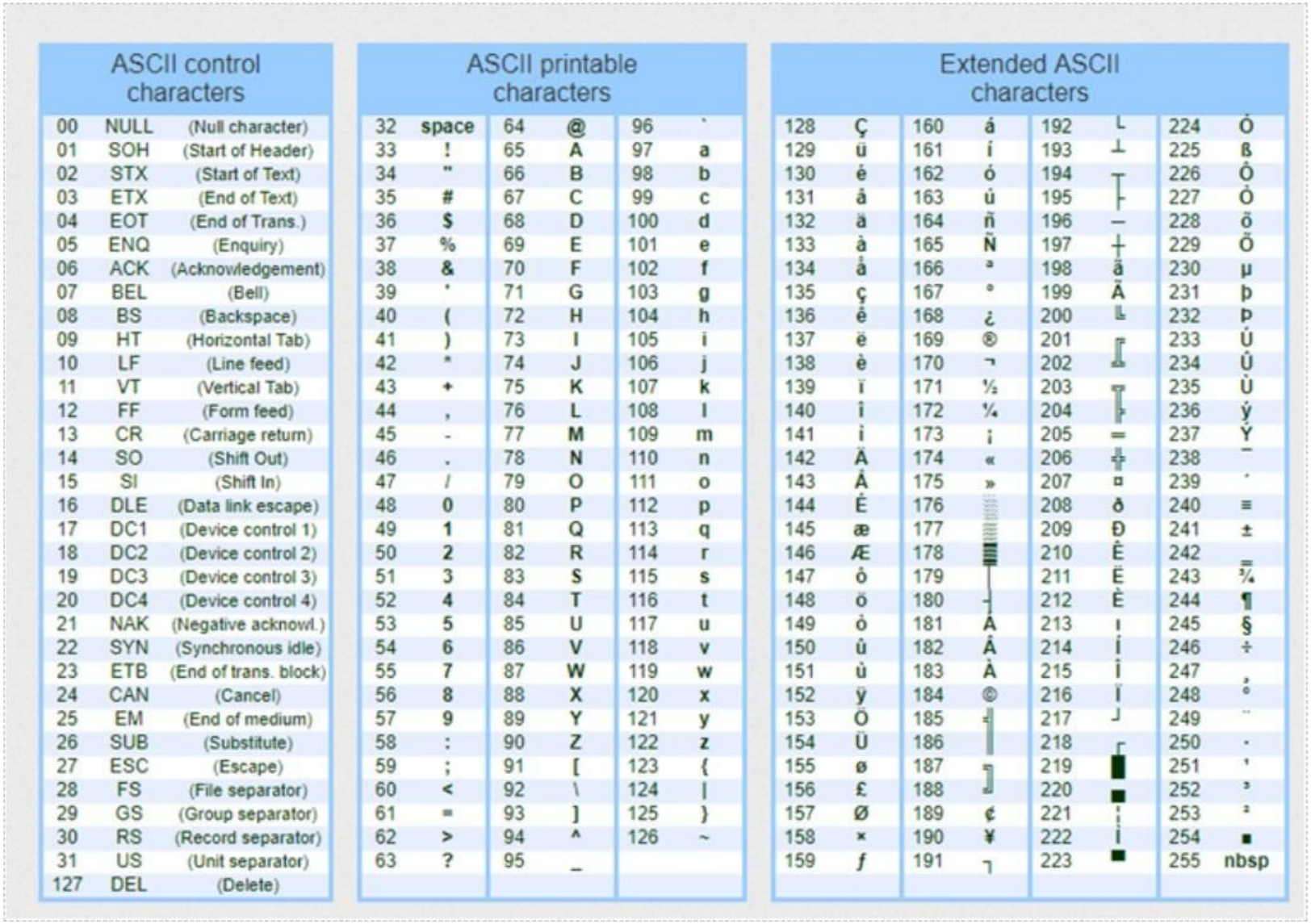
```
$255.62   2,523   3.  6,243,892    1,492.89 +6.0
```

Integer Data Types: `char`

- `char` data type
- **Declaration**: `char var_name;`
  - Store individual character
  - Printable character: letters, digits, and special symbols

- *Example*
  - Letters: `'L'  'o'  'l'`
  - Digits: `'1'  '0'  '5'`
  - Special symbols: `'$'  '#'  ','`

ASCII code: Character encoding standards



ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

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

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	▒	208	ð	240	≡
145	æ	177	▓	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	▉	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Floating-point Data Types

- Also called “real number”
- Can be number zero or any positive or negative number that contains a decimal point

- *Example*
  - Valid floating-point constant:

```
+10.6255   5.      -6.2    3251.92
0.0          0.33   -6.67   +2.
```

  - Invalid floating-point constant:

```
5,326.25    24    123    6,459    $10.29
```

The Escape Character

Table 2.5 Escape Sequences

Escape Sequence	Character Represented	Meaning	ASCII Code
`\n`	Newline	Move to a new line	00001010
`\t`	Horizontal tab	Move to next horizontal tab setting	00001001
`\v`	Vertical tab	Move to next vertical tab setting	00001011
`\b`	Backspace	Move back one space	00001000
`\r`	Carriage return	Carriage return (moves the cursor to the start of the current line—used for overprinting)	00001101
`\f`	Form feed	Issue a form feed	00001100
`\a`	Alert	Issue an alert (usually a bell sound)	00000111
`\\`	Backslash	Insert a backslash character (places an actual backslash character within a string)	01011100
`\?`	Question mark	Insert a question mark character	00111111
`\'`	Single quotation	Insert a single quote character (places an inner single quote within a set of outer single quotes)	00100111
`\"`	Double quotation mark	Insert a double quote character (places an inner double quote within a set of outer double quotes)	00100010
`\`*nnn*	Octal number	The number *nnn* (*n* is a digit) is to be considered an octal number	—
`\x`*hhhh*	Hexadecimal number	The number *hhhh* (*h* is a digit) is to be considered a hexadecimal number	—
`\0`	Null character	Insert the null character, which is defined as having the value 0	00000000

Example: Escape Character

- Output: "Hello World
  "
- `printf("\"Hello World\n\"");`

- Output: "5*7=?"
- `printf("\"5*7=\?\"");`

- Output: 'L'o"L"
- `printf("\'L\'o\"L\"");`

Variables

Define a variable

Variables: Initialization

Variables: Assignment

Exercise: What are the value?

- What is the value of `w, x, y,` and `z` ?

```
int w = 0, x, y, z;
x = 10;
x = x + 10;
y = 2 + 2;
```

Exercise: What are the value?

- What is the value of `x` and `y`?

```
int x, y;
int y = 100;
x = z + 210;
```

Implicit Type Conversion

Explicit Type Conversion (Casting)

Example: Cast a variable/expression

Expression	Value before casting	Value after casting
`(int)(9.79)`	`9.790000`	`9`
`(float)(8 * 5 % 2)`	`0`	`0.000000`
`(int)(5 + 1.0/2)`	`5.500000`	`5`
`(int)(2*2.0*2.0)`	`8.000000`	`8`
`(char)(20.5+70)`	`90.5`	`'Z'`

Basic Input / Output

Conversion Control Sequences

- `scanf()` and `printf()` needs a conversion control sequence for each value and variable

Table 2.8 Conversion Control Sequences

Sequence	Meaning
`%d`	Display an integer as a decimal (base 10) number
`%c`	Display a character
`%f`	Display the floating-point number as a decimal number with six digits after the decimal point (pad with zeros, if necessary)

Tip: Do not forget to declare a variable before using in both functions

scanf

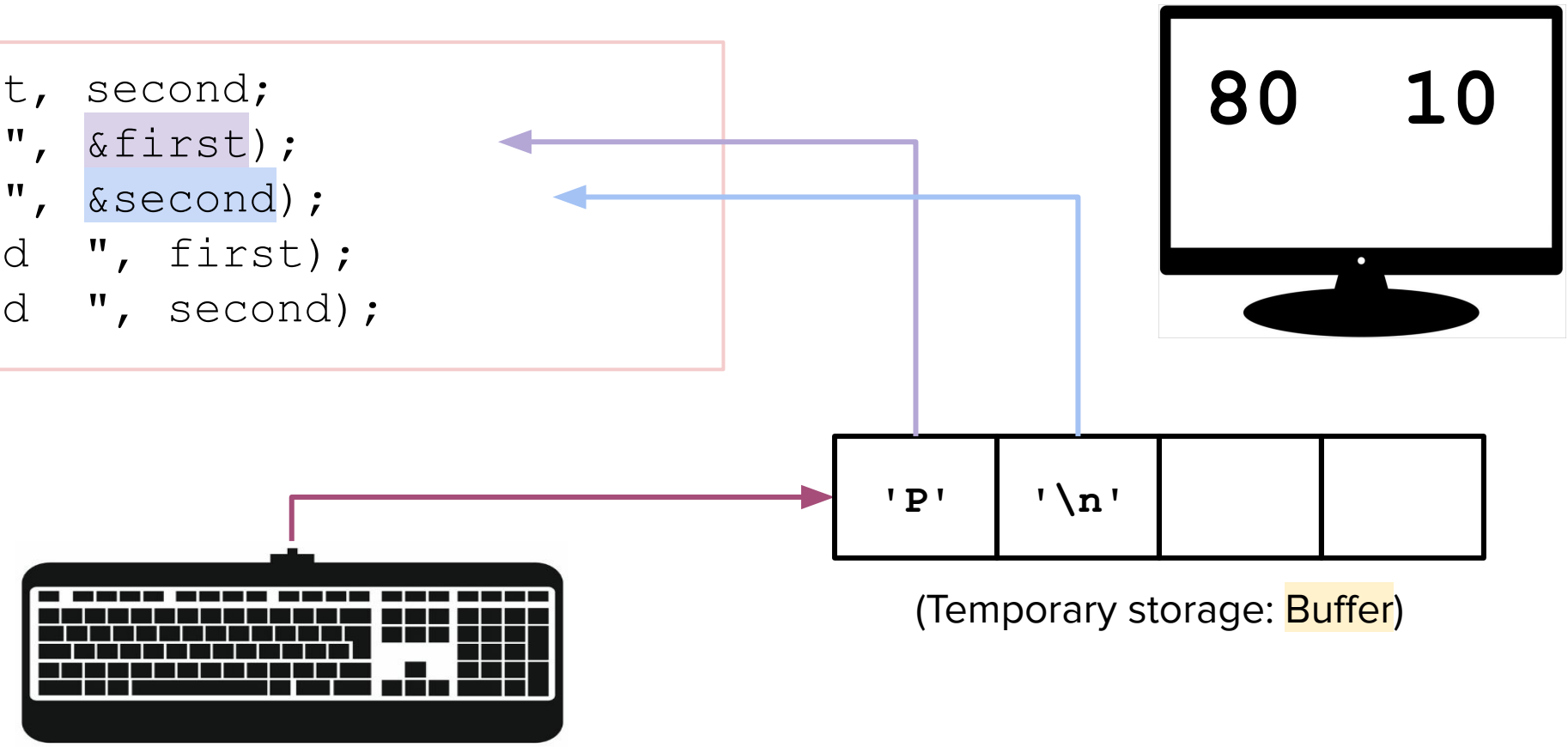
printf

Caution with `scanf("%c", &variable);`

When use multiple `scanf()` with `%c`, **`'\n'`** is accepted as the next variable

Example

```
char first, second;
scanf("%c", &first);
scanf("%c", &second);
printf("%d  ", first);
printf("%d  ", second);
```



Simple Solutions

1) Repeat **`scanf()`**

```
char first, second;
scanf("%c%c", &first, &second);
scanf("%c", &second);
printf("%d  ", first);
printf("%d  ", second);
```

Tip: Do not use for other data type it will generate strange things

2) Use a space in **`scanf()`**

```
char first, second;
scanf("%c ", &first);
scanf("%c", &second);
printf("%d  ", first);
printf("%d  ", second);
```

Arithmetic Operations

Arithmetic Operations

- Data type defines a set of values and a set of operations that can be applied to these values

Data Type	Supplied Operation

Integer
• `int`
• `char`
	`+, -, *, /, %,`
	`=, ==, !=,`
	`<, <=, >, >=`
	`(and more)`

Floating-point
• `float`
• `double`
	`+, -, *, /,`
	`=, ==, !=,`
	`<, <=, >, >=`
	`(and more)`

Exercise: Arithmetic Expression

Expression	Output Type (integer or floating-point)	Value
`5.0 * 2.0`		
`-2 + 5.0`		
`1 / 5`		
`1 % 2`		
`1 / 2.0`		
`5.0 / 2.0`		

Operator Precedence and Associativity

	Operator	Associativity
	`!, (unary) -, ++, --`	right to left
Arithmetic	`*, /, %`	left to right
	`+,- (subtraction)`	left to right
Relational	`<, <=, >, >=`	left to right
	`==, !=`	left to right
Logical	`&&`	left to right
	`||`	left to right
Assignment	`+=, -=, *=, /=`	right to left

Lab Exercises