

한눈에 보는 컴퓨터 공학

PART 1

23학번 정수호

컴퓨터 공학

컴퓨터 구조

6

이진법

17

자료형

14

자료구조

9

알고리즘/함수

14

동적 프로그래밍

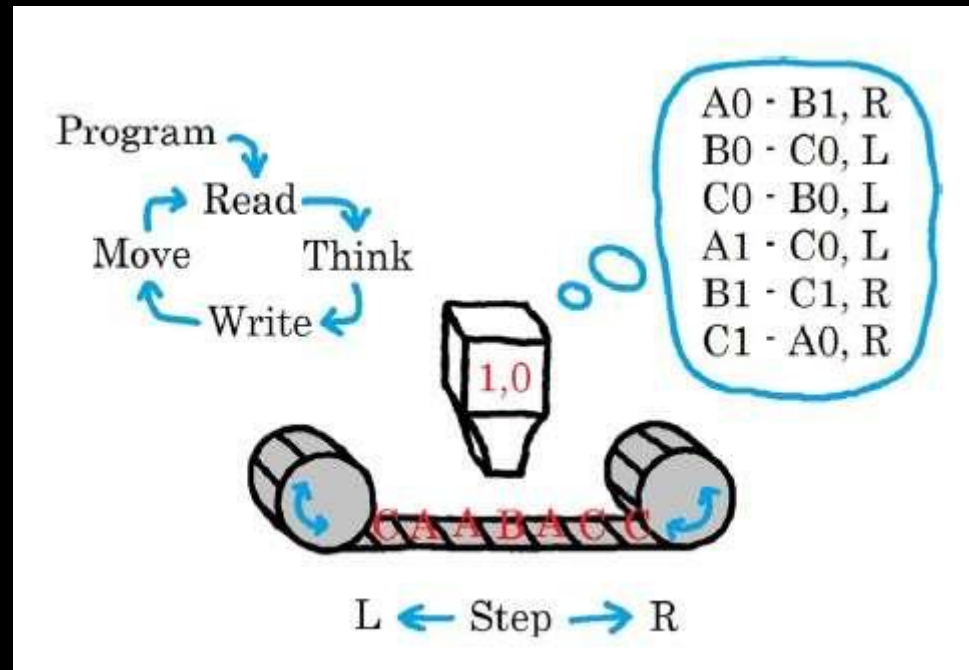
13

구현

27

1. 튜링 기계

- 긴 테이프의 적힌 0과 1을 읽고 수정할 수 있는 기계 (aka. 컴퓨터)



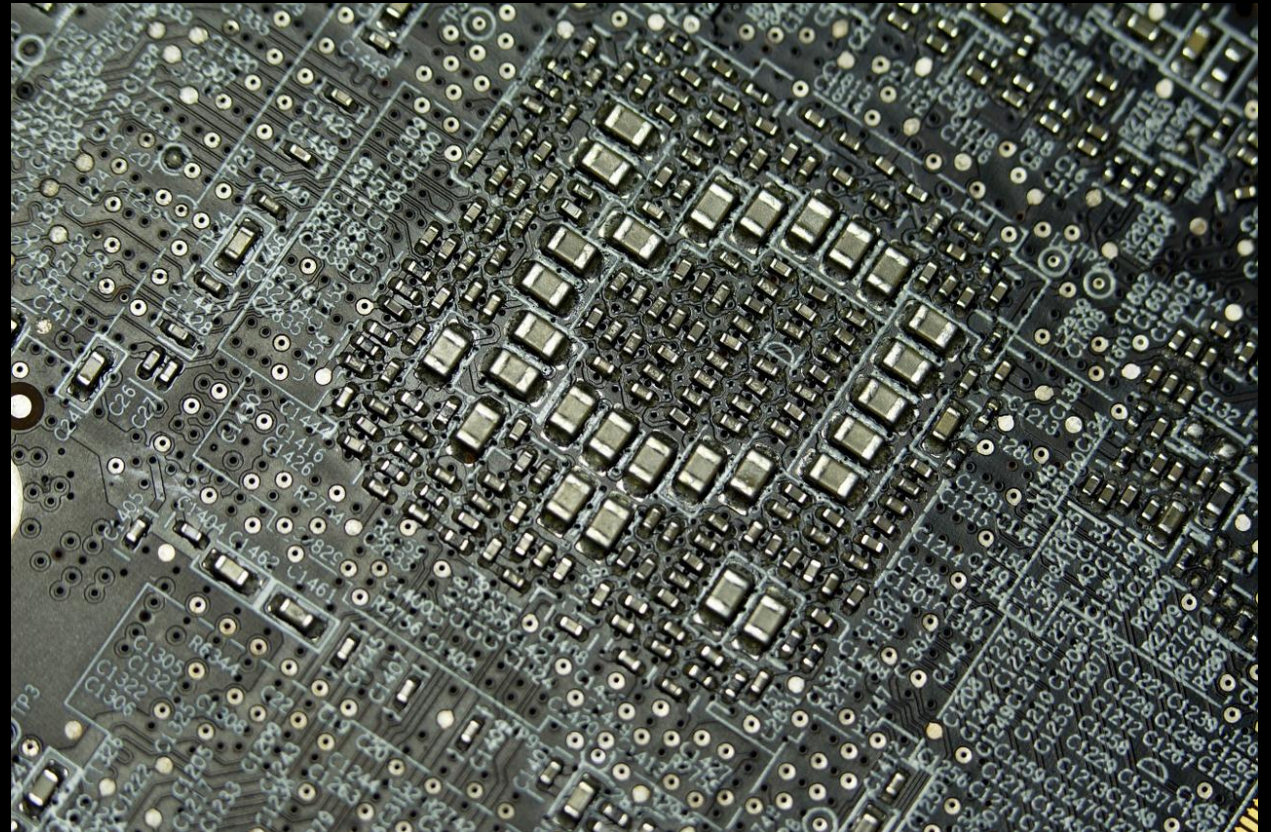
2. CPU

- CPU(Central Processing Unit)은 컴퓨터의 뇌 비슷한 존재

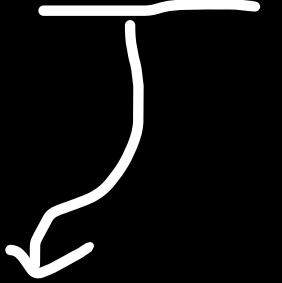


3. 트랜지스터

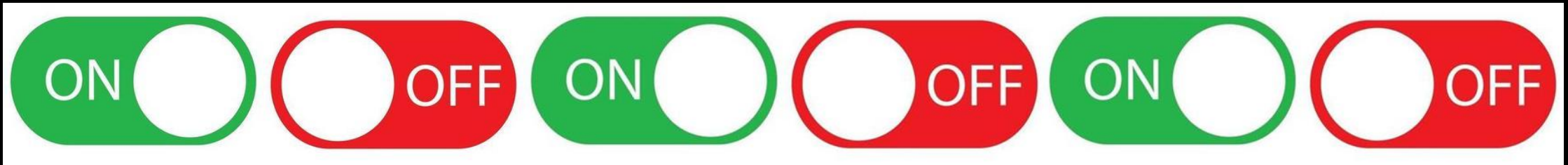
- 매우 작은 온/오프 스위치



4. 비트

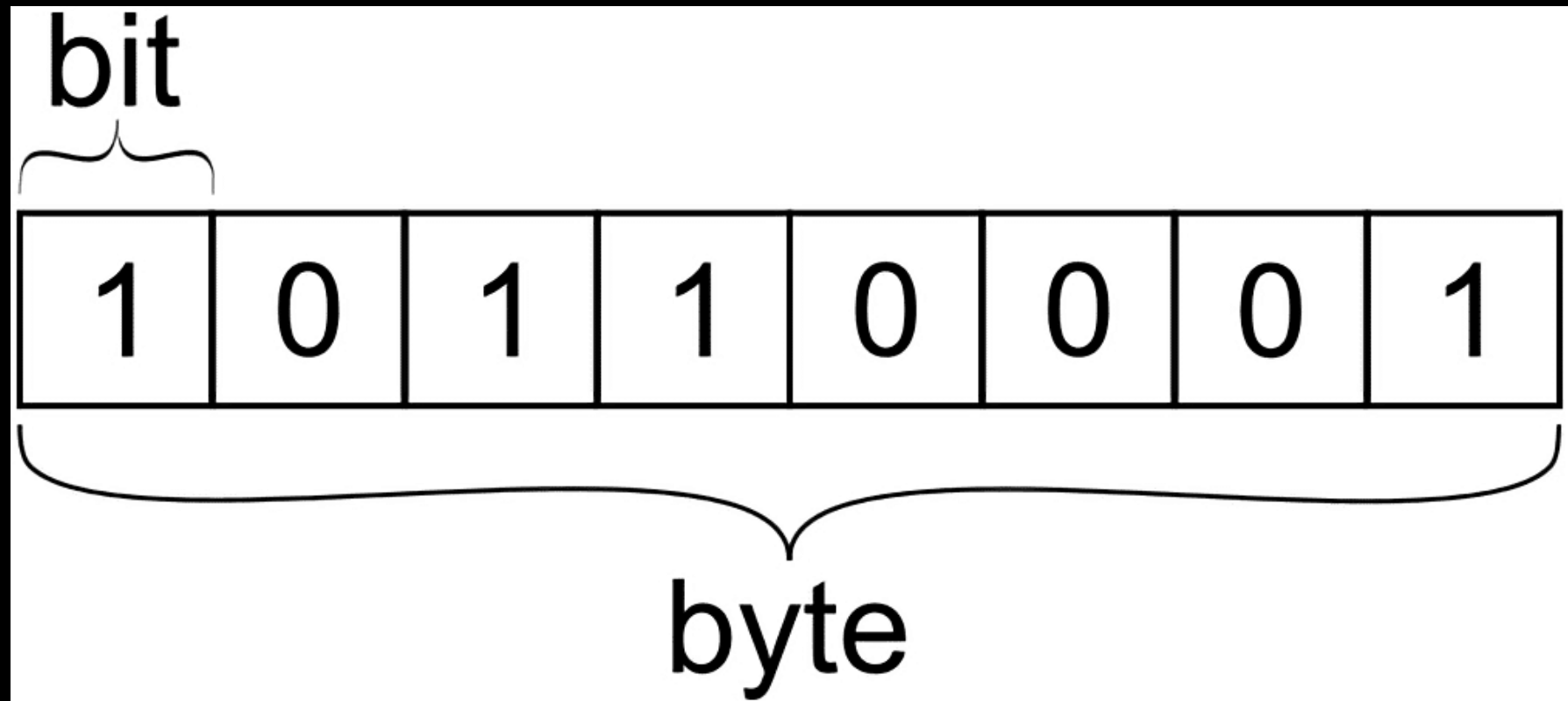


- 트랜지스터 스위치 하나



5. 바이트

- 1비트는 쓸모 없음 -> 8개의 비트를 묶어 바이트로 256개의 숫자 나타냄



6. ASCII 코드

- 키보드로 문자를 칠 때 글자와 숫자가 서로 대응되는 기본적인 표

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

7. 이진법

- 컴퓨터가 계산하는데 사용하는 숫자체계

base-2

x128 x64 x32 x16 x8 x4 x2 x1

0	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

0 + 64 + 32 + 16 + 8 + 0 + 2 + 1

8. 십육진법

- 사람들은 이진법을 계산하기 어려워해서 2진법을 대신해서 사용

1101

1110

1100

1010

1111

D

E

C

A

F

9. 니블

- 1 바이트의 절반, 4비트, 4개로 16진법의 한 숫자를 구별

1101

D

1110

E

1100

C

1010

A

1111

F

10. 기계어

- 만약 코드를 짜면 생성되는 이진법 형식의 코드로 컴퓨터가 실행함
- CPU가 해석+실행함

```
.org $8000  loop:
              ror
              sta $6000
reset:
  lda #$ff
  sta $6002
              jmp loop

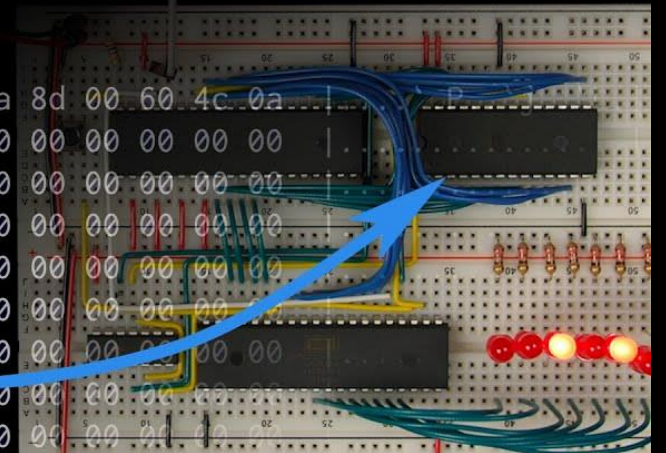
  lda #$50
  sta $6000

.org $fffc
.word reset
.word $0000
```

Assembly language

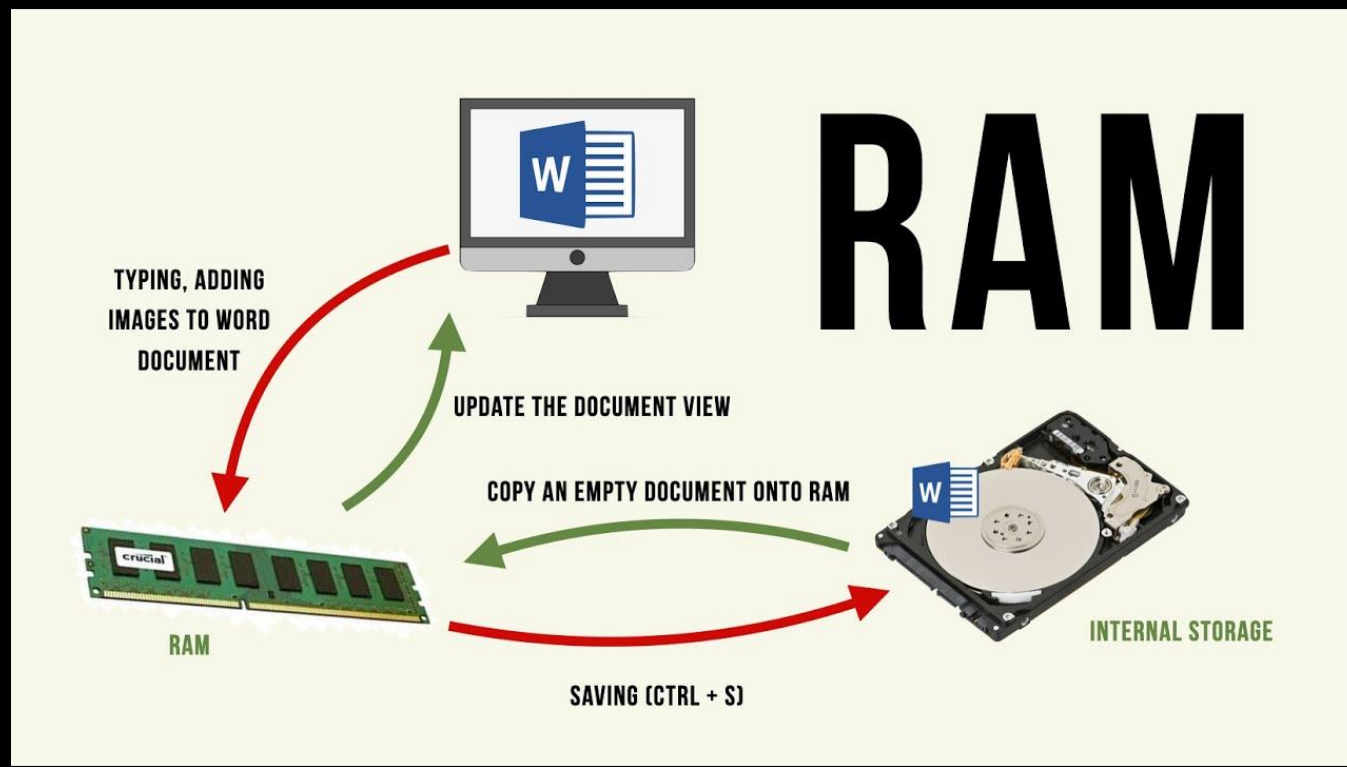
```
00000000  a9 8d 02 60 a9 50 8d 00 60 6a 8d 00 60 4c 0a
00000010  80 00 00 00 00 00 00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040  00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Machine code



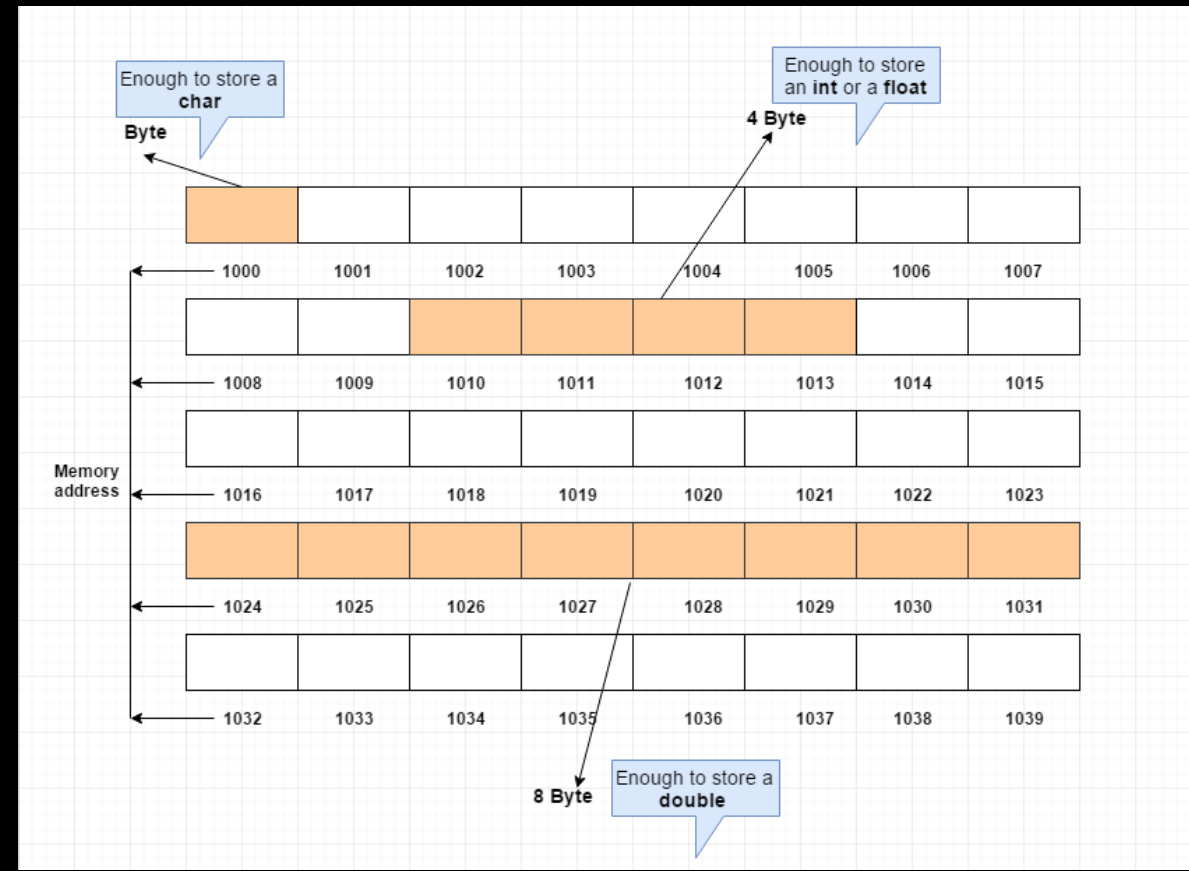
11. RAM

- RAM(Random Access Memory) 컴퓨터의 데이터 저장소
- 거대한 데이터를 위한 아파트 단지다.
- CPU가 읽고 사용할 수 있다



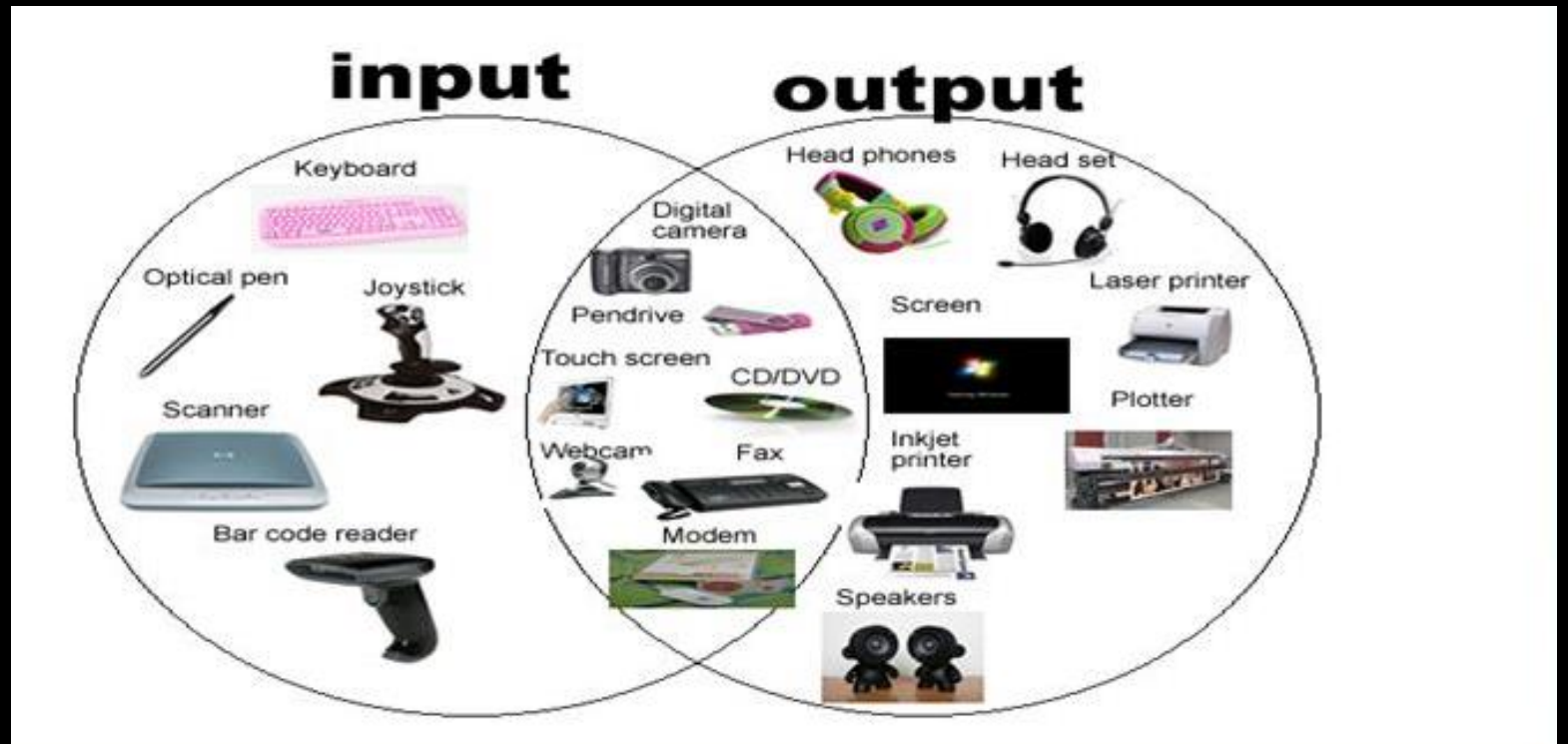
12. 기억장치 주소

- 각 데이터가 살고 있는 아파트 단지 내 집의 주소이다



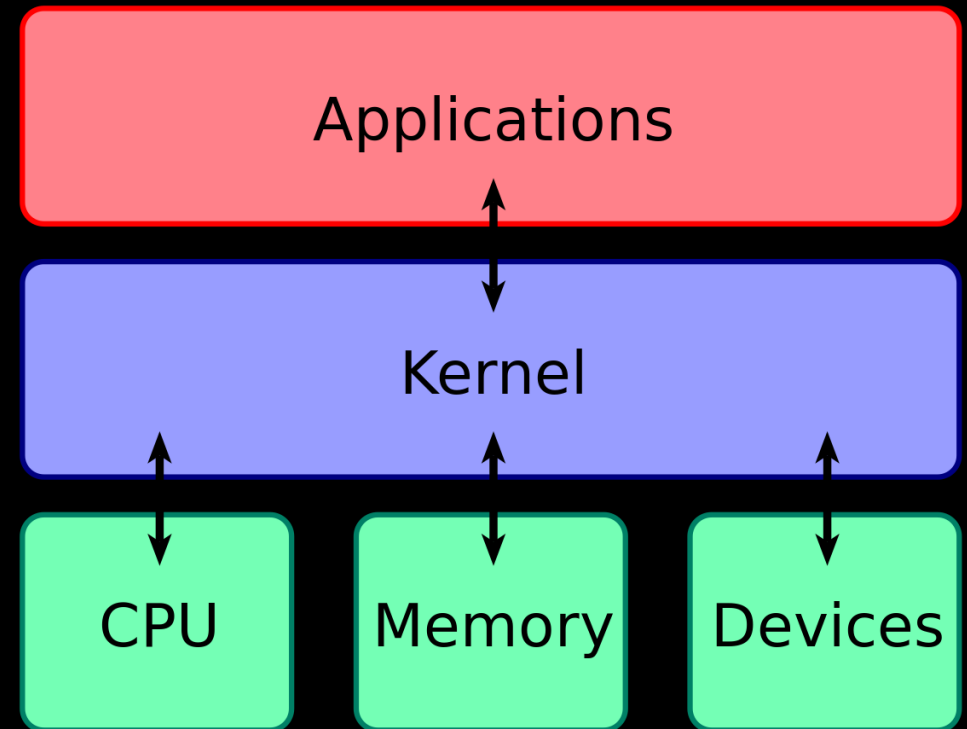
13. 데이터 입력/출력

- 입력 장치: 컴퓨터 안으로 데이터를 집어 넣는 장치(예. 키보드)
- 출력 장치: 컴퓨터 밖으로 데이터를 출력 하는 장치(예. 모니터)



14. 운영체제

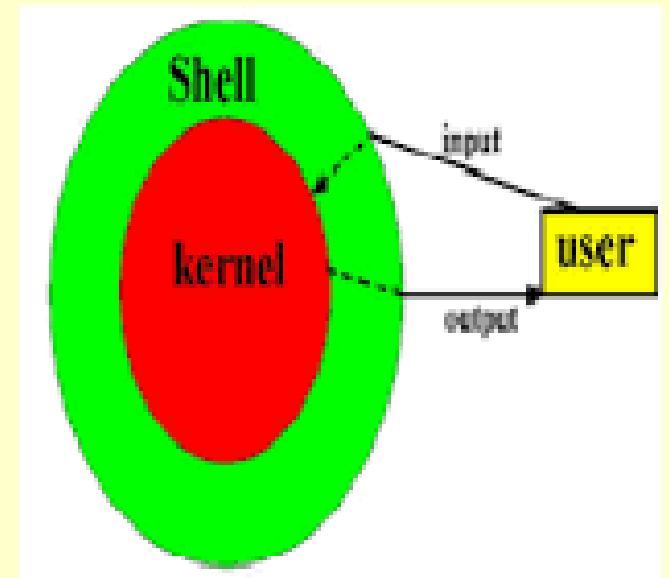
- 컴퓨터의 하드웨어적 요소와 소프트웨어적 요소를 이어줌
- 예시: windows, mac, linux...



15. 셸

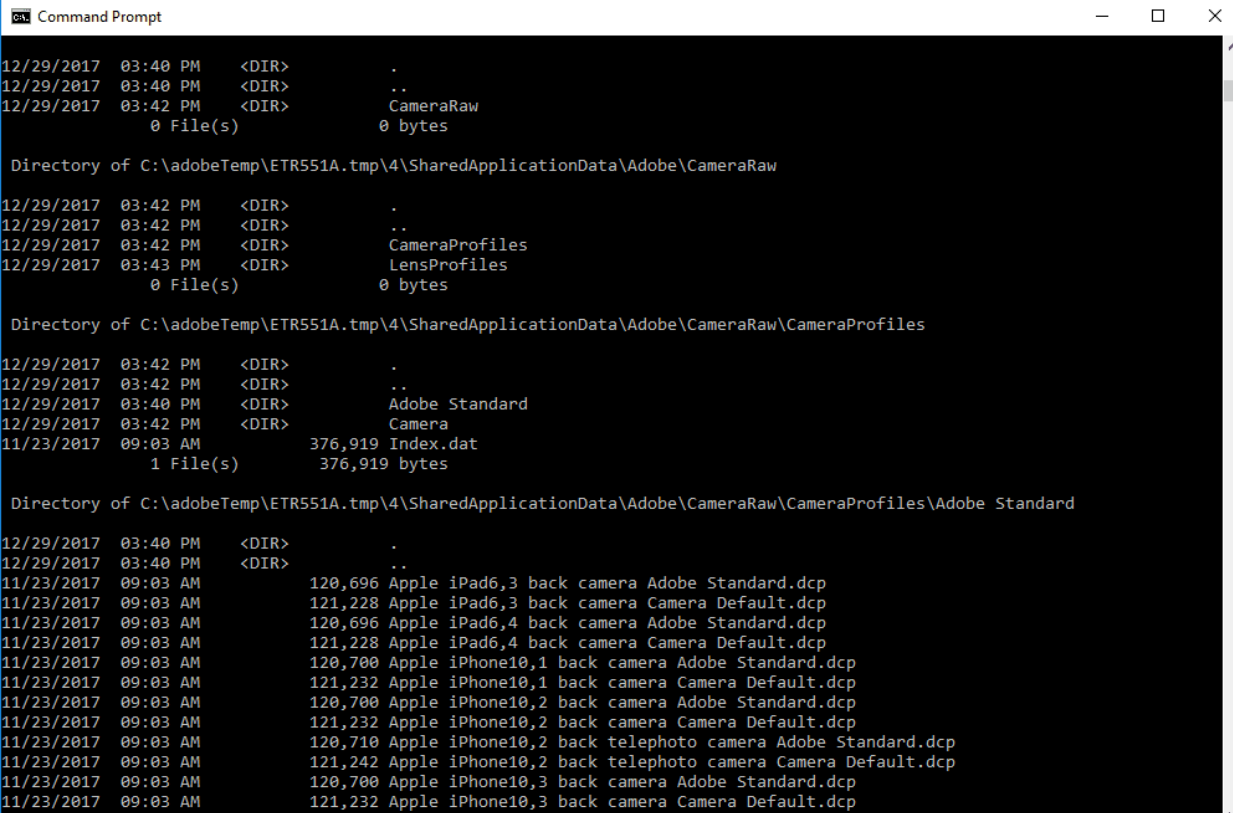
- 운영 체제를 감싸고 있는 프로그램
- 다양한 운영체제 기능과 서비스를 제공

A shell is a UNIX program that interprets the commands that users type on their terminal keyboards.



16. 명령 줄 인터페이스

- 사용자가 특정 단어들로 이루어진 코드를 입력하면 그에 대응되는 답변을 출력해준다



```
Command Prompt

12/29/2017 03:40 PM <DIR> .
12/29/2017 03:40 PM <DIR> ..
12/29/2017 03:42 PM <DIR> CameraRaw
0 File(s) 0 bytes

Directory of C:\adobeTemp\ETR551A.tmp\4\SharedApplicationData\Adobe\CameraRaw

12/29/2017 03:42 PM <DIR> .
12/29/2017 03:42 PM <DIR> ..
12/29/2017 03:42 PM <DIR> CameraProfiles
12/29/2017 03:43 PM <DIR> LensProfiles
0 File(s) 0 bytes

Directory of C:\adobeTemp\ETR551A.tmp\4\SharedApplicationData\Adobe\CameraRaw\CameraProfiles

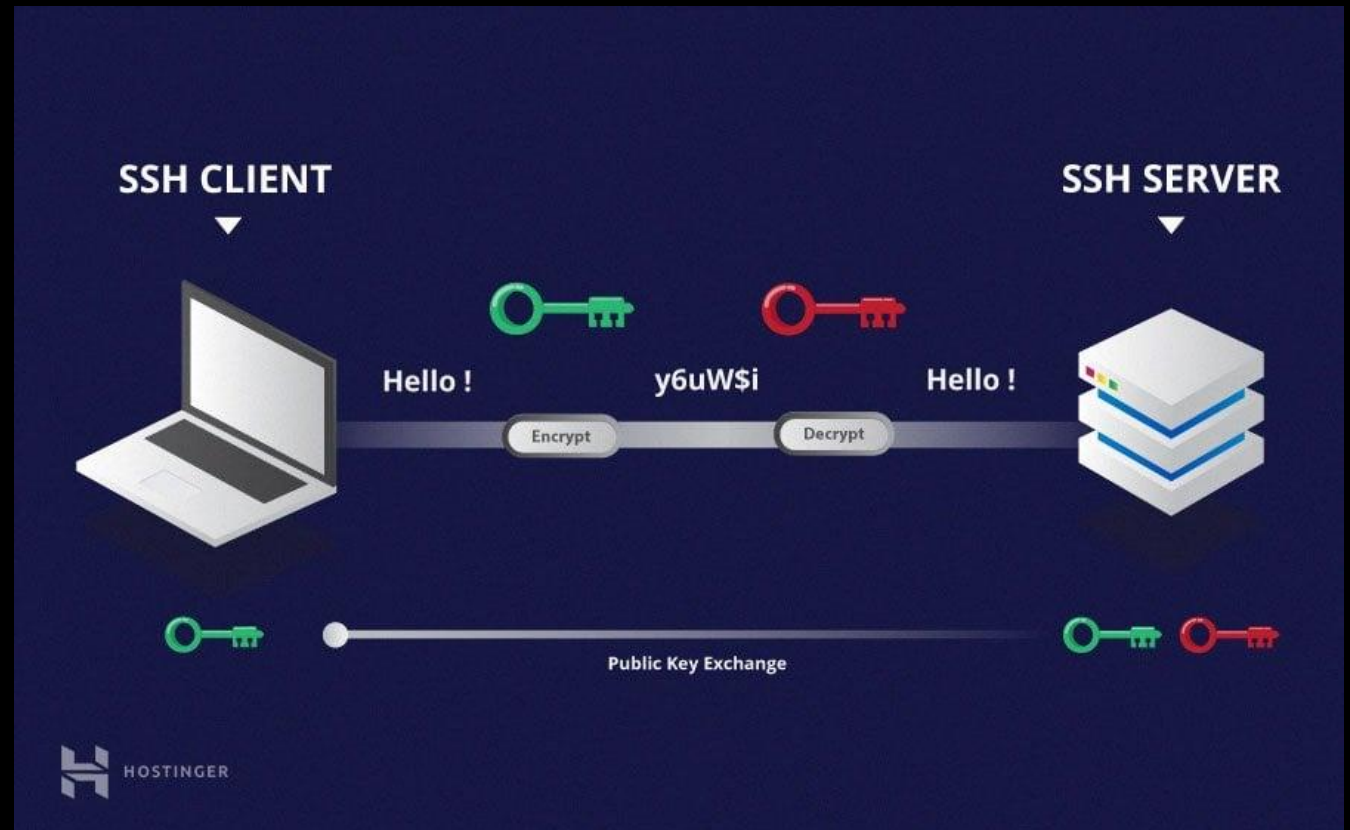
12/29/2017 03:42 PM <DIR> .
12/29/2017 03:42 PM <DIR> ..
12/29/2017 03:40 PM <DIR> Adobe Standard
12/29/2017 03:42 PM <DIR> Camera
11/23/2017 09:03 AM 376,919 Index.dat
1 File(s) 376,919 bytes

Directory of C:\adobeTemp\ETR551A.tmp\4\SharedApplicationData\Adobe\CameraRaw\CameraProfiles\Adobe Standard

12/29/2017 03:40 PM <DIR> .
12/29/2017 03:40 PM <DIR> ..
11/23/2017 09:03 AM 120,696 Apple iPad6,3 back camera Adobe Standard.dcp
11/23/2017 09:03 AM 121,228 Apple iPad6,3 back camera Camera Default.dcp
11/23/2017 09:03 AM 120,696 Apple iPad6,4 back camera Adobe Standard.dcp
11/23/2017 09:03 AM 121,228 Apple iPad6,4 back camera Camera Default.dcp
11/23/2017 09:03 AM 120,700 Apple iPhone10,1 back camera Adobe Standard.dcp
11/23/2017 09:03 AM 121,232 Apple iPhone10,1 back camera Camera Default.dcp
11/23/2017 09:03 AM 120,700 Apple iPhone10,2 back camera Adobe Standard.dcp
11/23/2017 09:03 AM 121,232 Apple iPhone10,2 back camera Camera Default.dcp
11/23/2017 09:03 AM 120,710 Apple iPhone10,2 back telephoto camera Adobe Standard.dcp
11/23/2017 09:03 AM 121,242 Apple iPhone10,2 back telephoto camera Camera Default.dcp
11/23/2017 09:03 AM 120,700 Apple iPhone10,3 back camera Adobe Standard.dcp
11/23/2017 09:03 AM 121,232 Apple iPhone10,3 back camera Camera Default.dcp
```

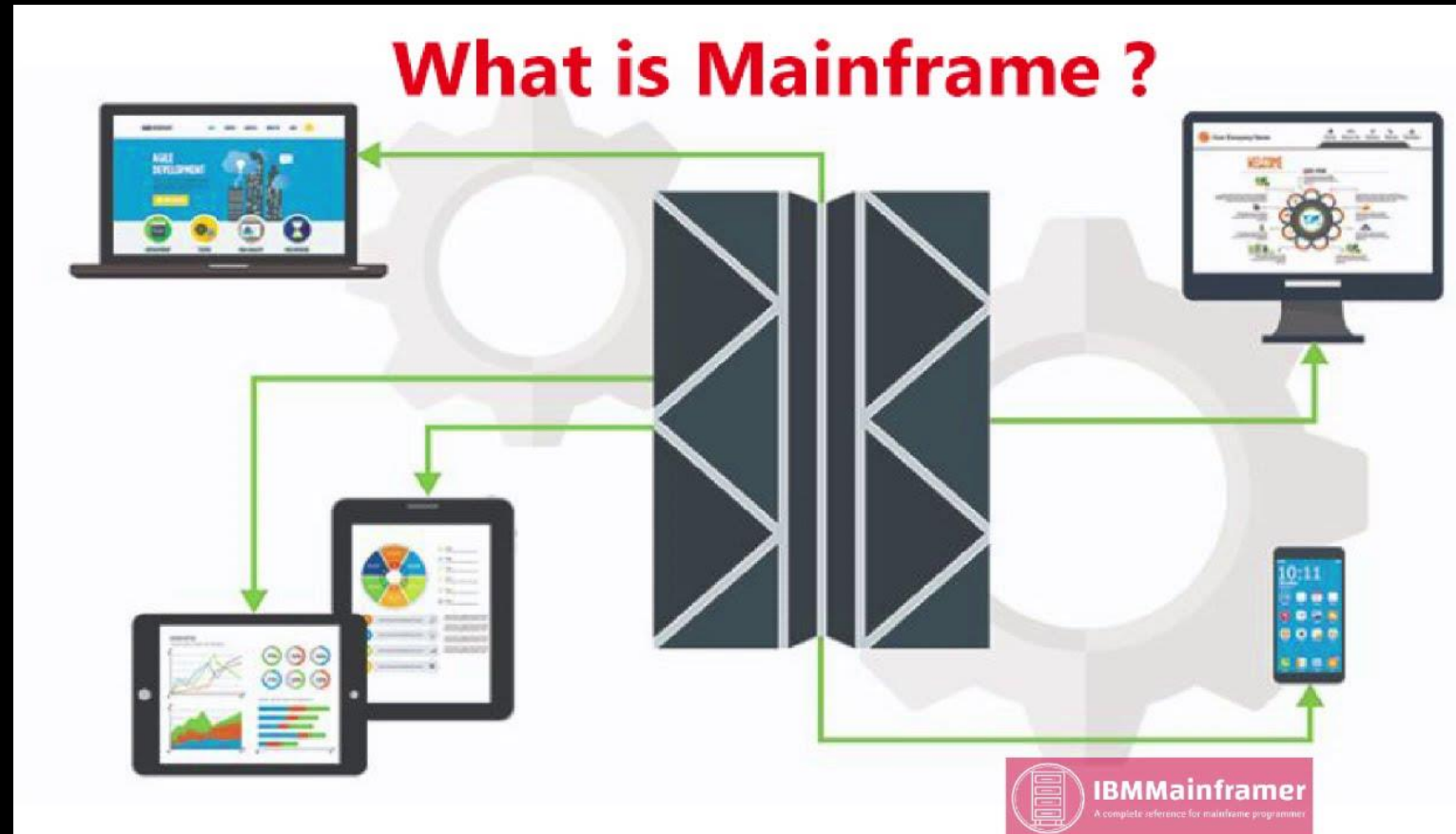
17. SSH

- SSE(Secure Shell Protocol) 네트워크에 있는 다른 컴퓨터와 연결하는 보안 프로토콜



18. 메인프레임

- 큰 데이터를 처리하는 거대한 컴퓨터



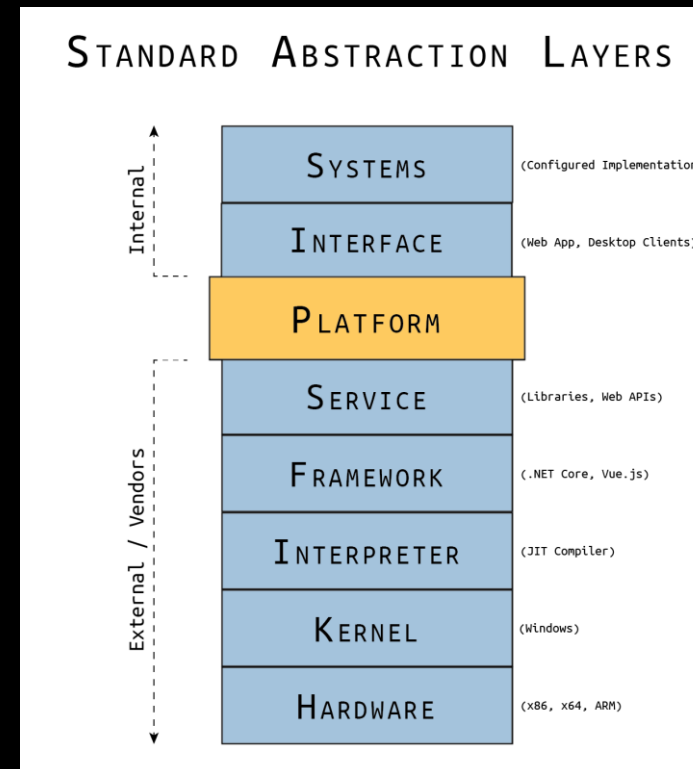
19. 프로그래밍 언어

- 컴퓨터 시스템을 구동 시키는 소프트웨어를 작성하기 위해 만들어진 언어



20. 컴퓨터 추상화

- 프로그래밍 언어가 사용하는 개념으로 사용자 혹은 엔지니어가 사용하는 컴퓨터의 내부 구조를 몰라도 사용할 수 있도록 해준다



21. INTERPRETER LANGUAGE

- interpreter가 코드를 한 줄 씩 해석하여 실행하는 형식의 컴퓨터 코드
- 조금 느리지만 바로 수정가능



```
hello.py > ...  
1  msg = "Hello World"  ← interpreter  
2  print(msg)  
3  |
```

22. COMPILER LANGUAGE

- compiler가 코드 전부를 기계어로 바꾸고 나서 프로그램 실행
- 빠르지만 바로 수정은 힘들다

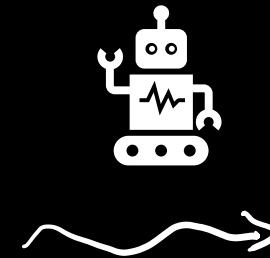


```
// Hello World in C++ (pre-ISO)

#include <iostream.h>

main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

compiler



23. EXE FILE

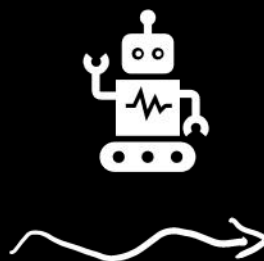
- 다른 추가적인 도움 없이 운영체제에서 바로 실행할 수 있는 파일
- Compiler Language를 실행시키면 만들어짐

```
// Hello World in C++ (pre-ISO)

#include <iostream.h>

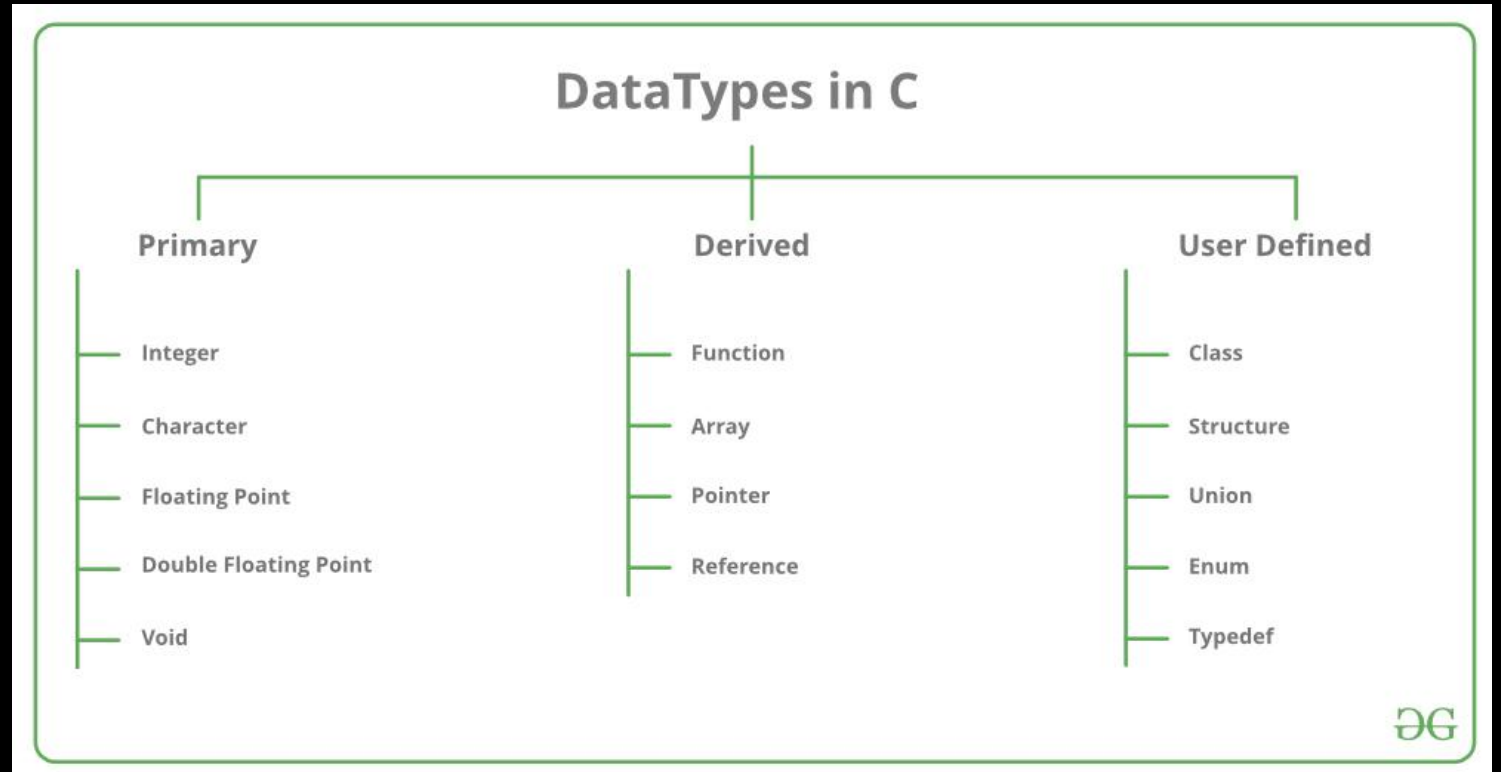
main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

compiler



24. 자료형

- 컴퓨터 안에서 사용하는 데이터가 무엇인지 나타내기 위해서 사용
- 예) int, char, string...



25. 변수

- 데이터에 이름을 붙여서 코드 내에서 다시 사용할 수 있게 해준다
- 자료형이 여기서 사용된다

```
hello.py > ...  
1  msg = "Hello World"  
2  print(msg)  
3  |
```

26. 동적 타입 언어

- 인터프리터가 그 시점의 변수 값을 기반으로 런타임에 변수에 타입을 할당하는 것
- 따로 자료형을 지정해줄 필요가 없다

```
hello.py > ...  
1  msg = "Hello World"  
2  print(msg)  
3  |
```

27. 정적 타입 언어

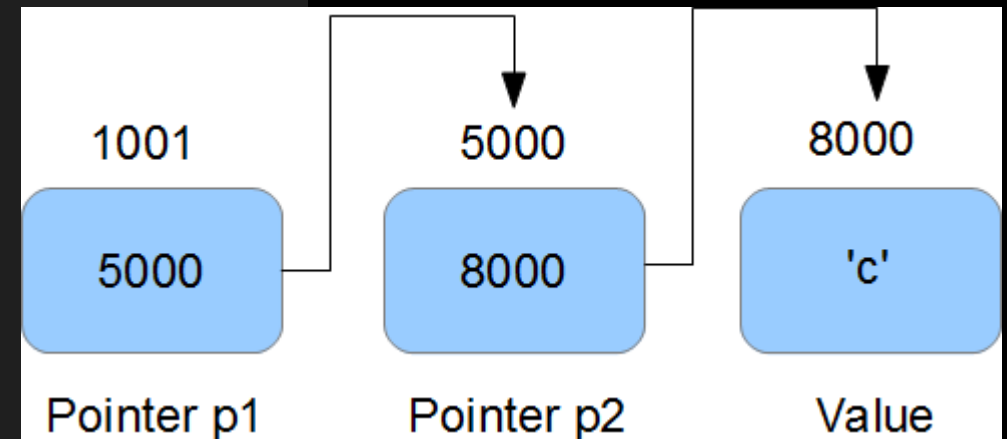
- 코드 작성자가 직접 변수 타입을 지정해주어야 한다

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdbool.h>
4  #include <time.h>
5
6  int main(){
7      struct timespec start, end;
8      double cpuTime;
9
10     clock_gettime(CLOCK_MONOTONIC, &start);
11
12     //some functions
13
14     clock_gettime(CLOCK_MONOTONIC, &end);
15
16
17     cpuTime = (end.tv_sec - start.tv_sec) + (double)(end.tv_nsec - start.tv_nsec) / 1e9;
18     printf("time taken: %f\n",cpuTime);
19 }
```


28. 포인터

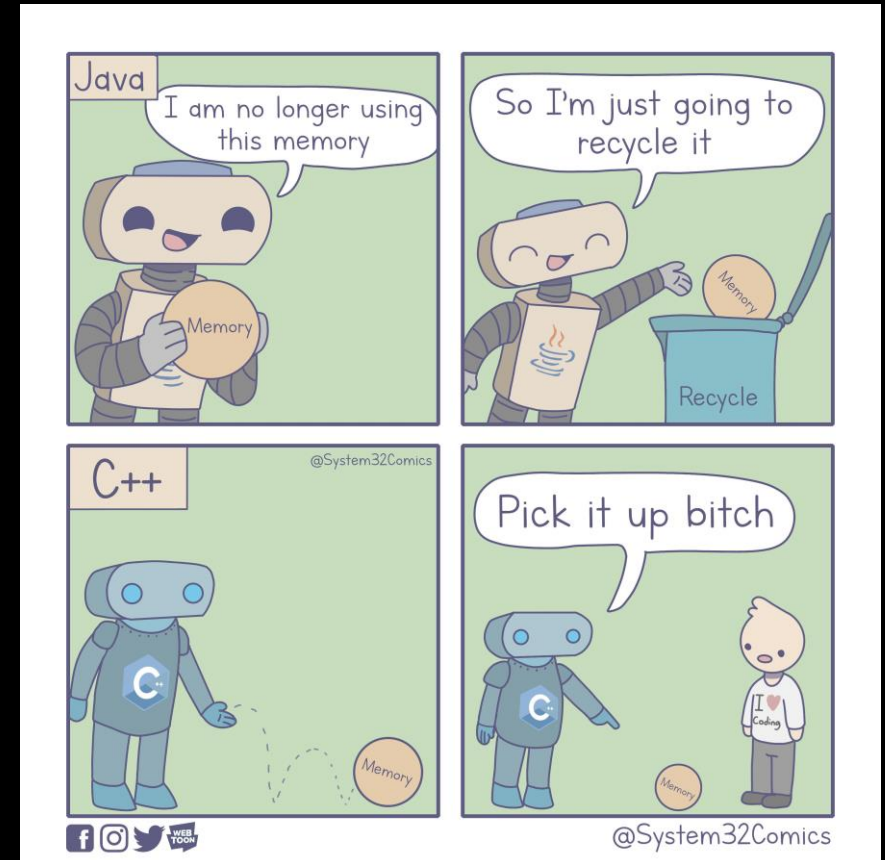
- 프로그램안에 데이터의 주소를 값으로 가지는 변수

```
1  #include <stdio.h>
2
3  int main(){
4      double a[5], lowest, *p;
5      int index;
6      p = &a[0];
7      for (int i=0;i<5;i++){
8          printf("enter %dth double: ", i+1);
9          scanf("%lf", p+i);
10     }
11     lowest = *p;
12     index=0;
13     for (int i=0;i<5;i++){
14         if (lowest > *(p+i)){
15             lowest = *(p+i);
16             index = i;
17         }
18     }
19     printf("lowest number: %lf\n", lowest);
20     printf("index: %d", index);
21
22     return 0;
23 }
```



29. 쓰레기 수집

- 프로그램이 동적으로 할당했던 메모리 영역 중에서 필요 없게 된 영역을 자동으로 해제하는 기능
- 포인터를 쓰기 힘들어서 나온 기법



30. INT

- 작은 정수를 나타내는 숫자 단위

자료형			크기 (byte)	범위
정수	signed	short	2	-32,768 ~ 32,767
		int	4	-2,147,483,648 ~ 2,147,483,647
		long	4	-2,147,483,648 ~ 2,147,483,647
		long long	8	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
	unsigned	unsigned short	2	0 ~ 65,535
		unsigned int	4	0 ~ 4,294,967,295
		unsigned long	4	0 ~ 4,294,967,295
		unsigned long long	8	0 ~ 18,446,744,073,709,551,615
문자	signed	char	1	-128 ~ 127
	unsigned	unsigned char	1	0 ~ 255

31. SIGNED

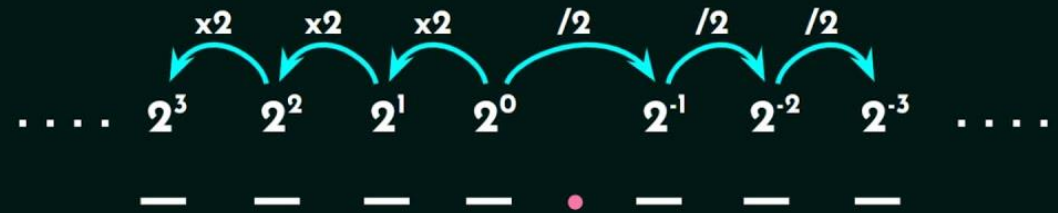
- 음수를 나타내기 위해서 사용됨
- 예) unsigned int ($0 \sim 2^{32}-1$) vs signed int ($-2^{31} \sim 2^{31}-1$)

자료형			크기 (byte)	범위
정수	signed	short	2	-32,768 ~ 32,767
		int	4	-2,147,483,648 ~ 2,147,483,647
		long	4	-2,147,483,648 ~ 2,147,483,647
		long long	8	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
	unsigned	unsigned short	2	0 ~ 65,535
		unsigned int	4	0 ~ 4,294,967,295
		unsigned long	4	0 ~ 4,294,967,295
		unsigned long long	8	0 ~ 18,446,744,073,709,551,615
문자	signed	char	1	-128 ~ 127
	unsigned	unsigned char	1	0 ~ 255

32. FLOATING POINT

- 좁은 범위 내의 소수를 표현 하려고 할 때 사용
- 7개의 자리 수 허용

Floating Point Numbers



33. DOUBLE

- float보다 큰 자리수의 소수를 나타낼 때 사용됨
- 총 15개의 자리 수를 표현할 수 있음

자료형		크기 (byte)	수의 표현 범위
char	char	1	$-2^7 \sim 2^7 - 1$ (-128 ~ 127)
	signed char		
	unsigned char		$0 \sim 2^8 - 1$ (0~255)
int	short int	2	$-2^{15} \sim 2^{15} - 1$ (-32,768 ~ 32,767)
	unsigned short int		$0 \sim 2^{16} - 1$ (0 ~ 65,535)
	int	4	$-2^{31} \sim 2^{31} - 1$ (-2,147,483,648 ~ 2,147,483,647)
	unsigned int		$0 \sim 2^{32} - 1$ (0 ~ 4,294,967,295)
long	long int	4	$-2^{31} \sim 2^{31} - 1$ (-2,147,483,648 ~ 2,147,483,647)
	unsigned long int		$0 \sim 2^{32} - 1$ (0 ~ 4,294,967,295)
float	float	4	$-10^{128} \sim 10^{127}$: 소수 6자리 표현
double	double	8	$-10^{128} \sim 10^{127}$: 소수 15자리 표현
	long double	8 또는 그 이상	차이를 많이 보임 : double의 정밀도와 같거나 크다.

34. CHAR

- 하나의 알파벳을 나타낼 때 사용

자료형	크기 (byte)	수의 표현 범위
char	1	char
		signed char
		unsigned char
int	2	short int
		unsigned short int
	4	int
		unsigned int
long	4	long int
		unsigned long int
float	4	float
double	8	double
	8 또는 그 이상	long double

35. STRING

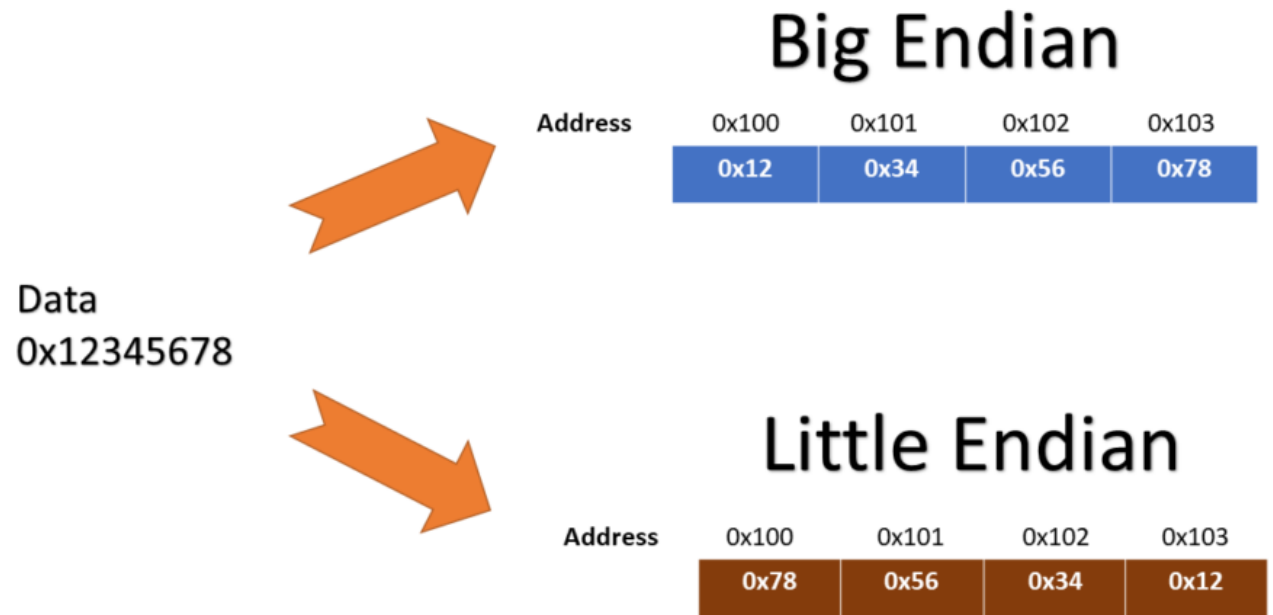
- 여러 개의 알파벳 또는 기호(단어, 문장)등을 나타낼 때 사용

```
my_string1 = "apple"
```

```
my_string2 = 'Hello, world!'
```

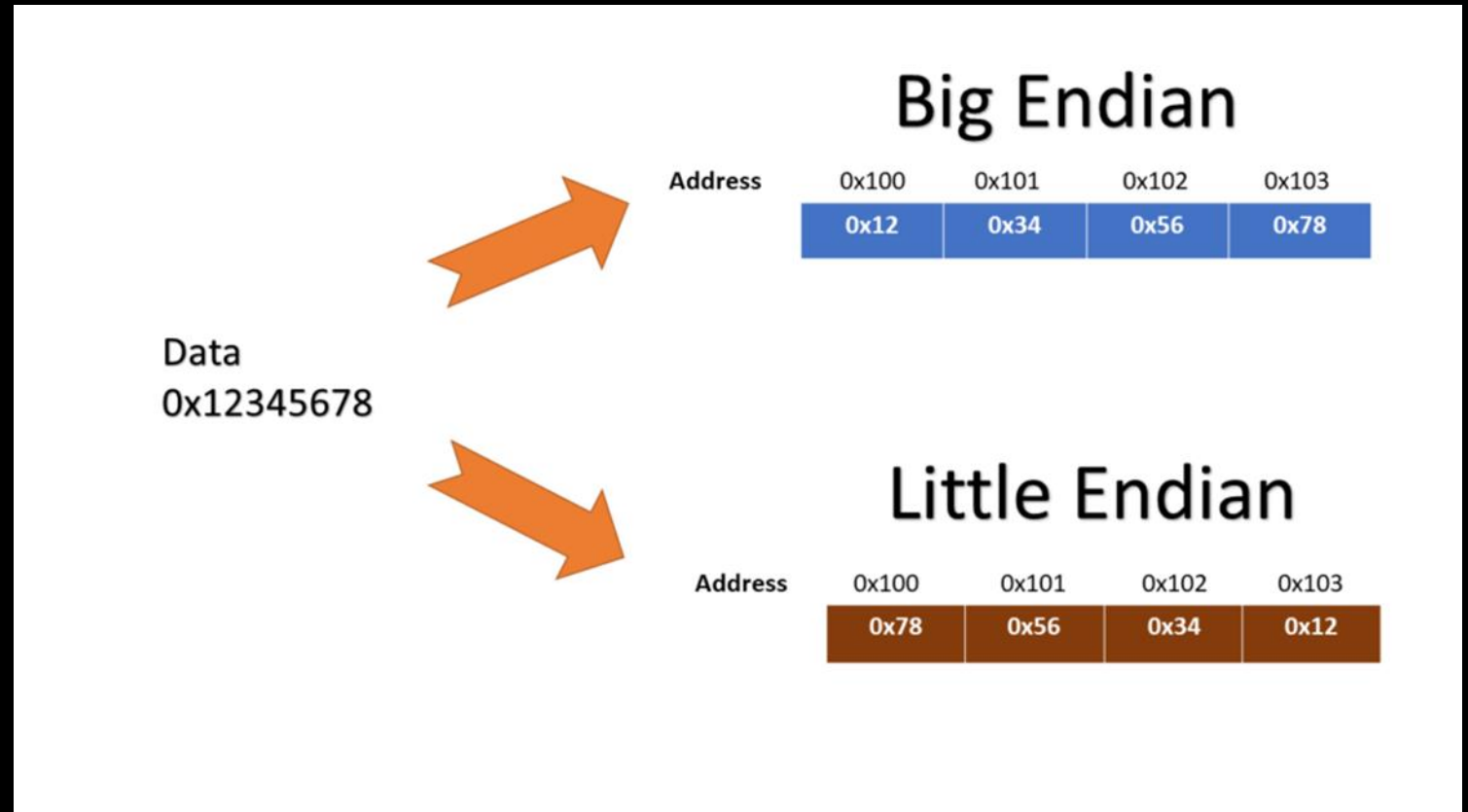
36. 빅 엔디언

- 주소를 오름차순으로 저장하는 방법
- 직관적이어서 많이 차용



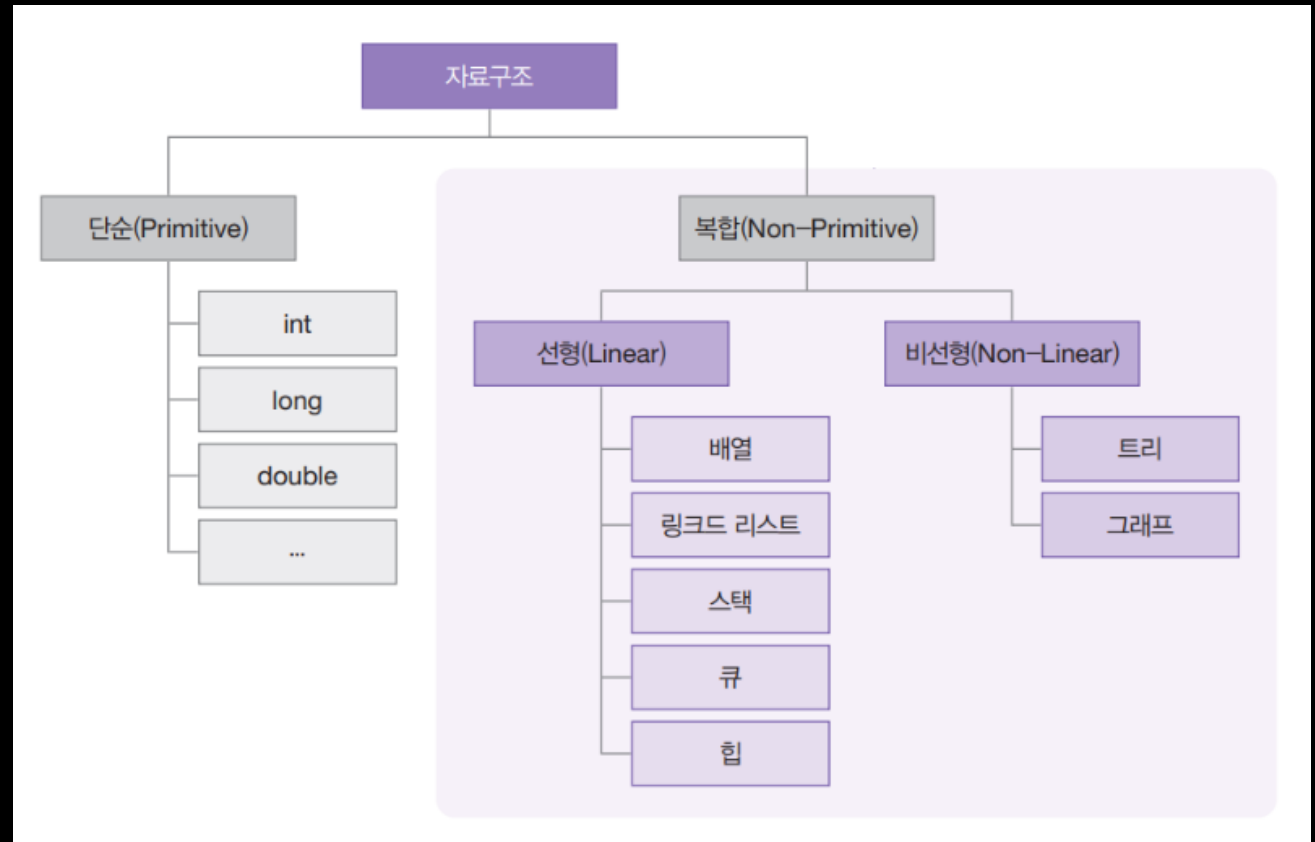
37. 리틀 엔디언

- 주소를 내림차순으로 저장하는 방법
- 속도가 더 빠르다



38. 자료 구조

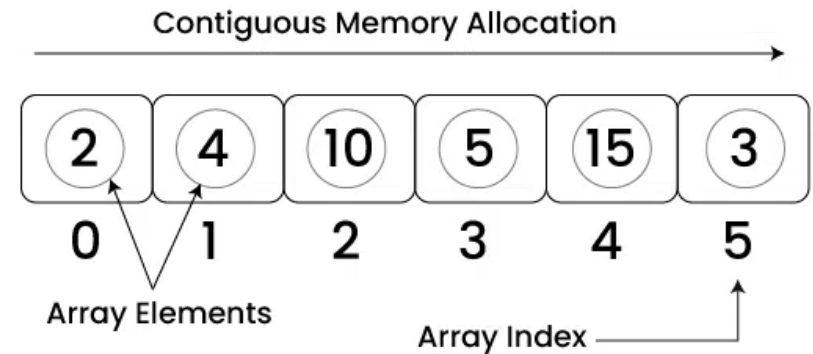
- 효율적인 자료의 저장과 수정을 가능하게 해주는 자료의 조직
- 예시) array, list, tree...



39. ARRAY

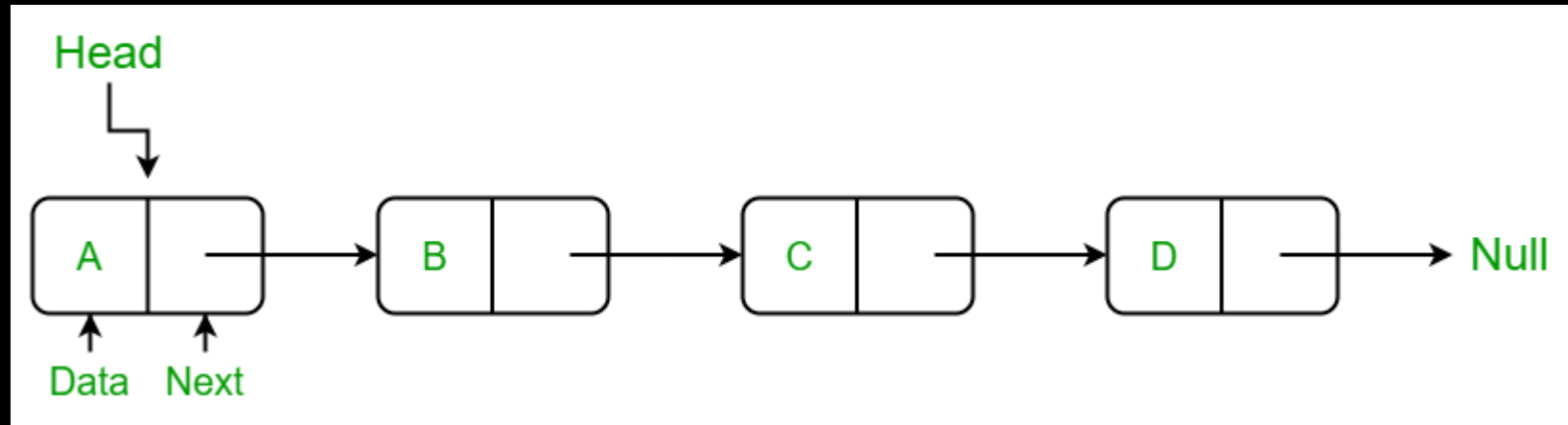
- 가장 기본적인 선형자료구조로 데이터를 쭉 나열해 놓은 자료구조
- 0에서부터 시작함

What is
Array
Data Structure



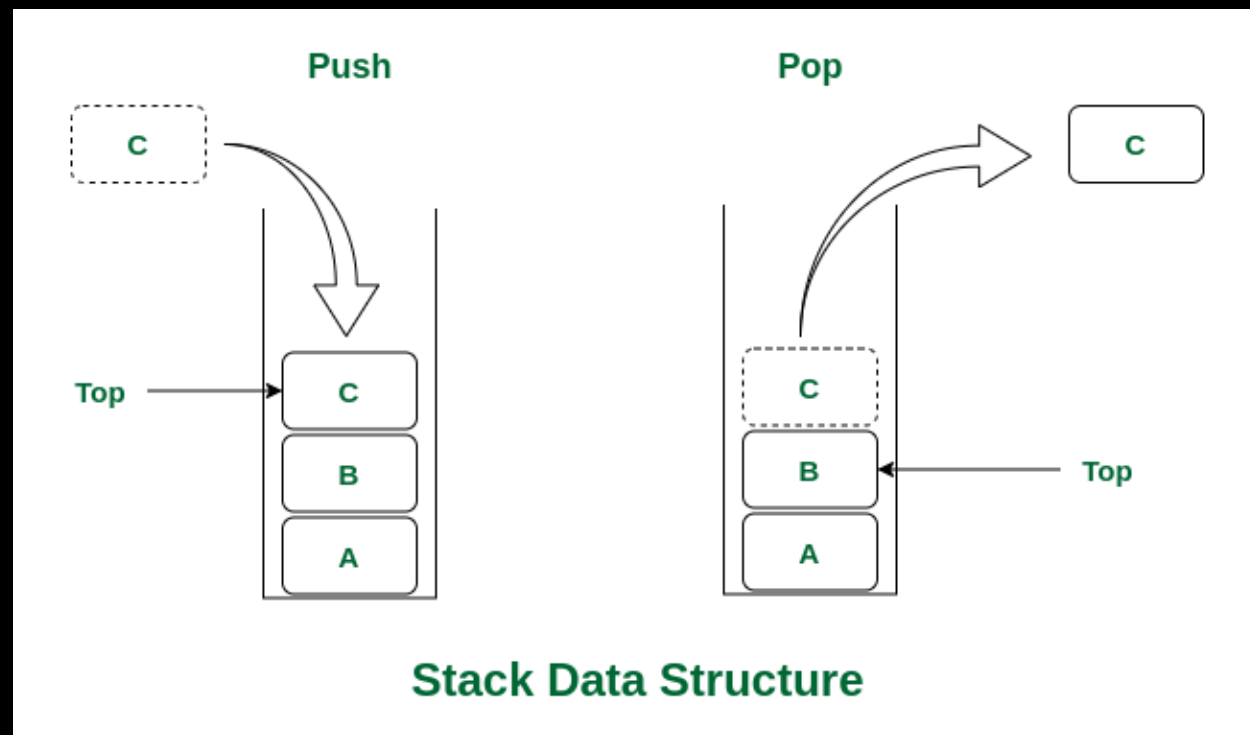
40. LINKED LIST

- 포인터로 데이터를 엮은 형태의 자료구조
- 수정이 편하다는 장점이 있다



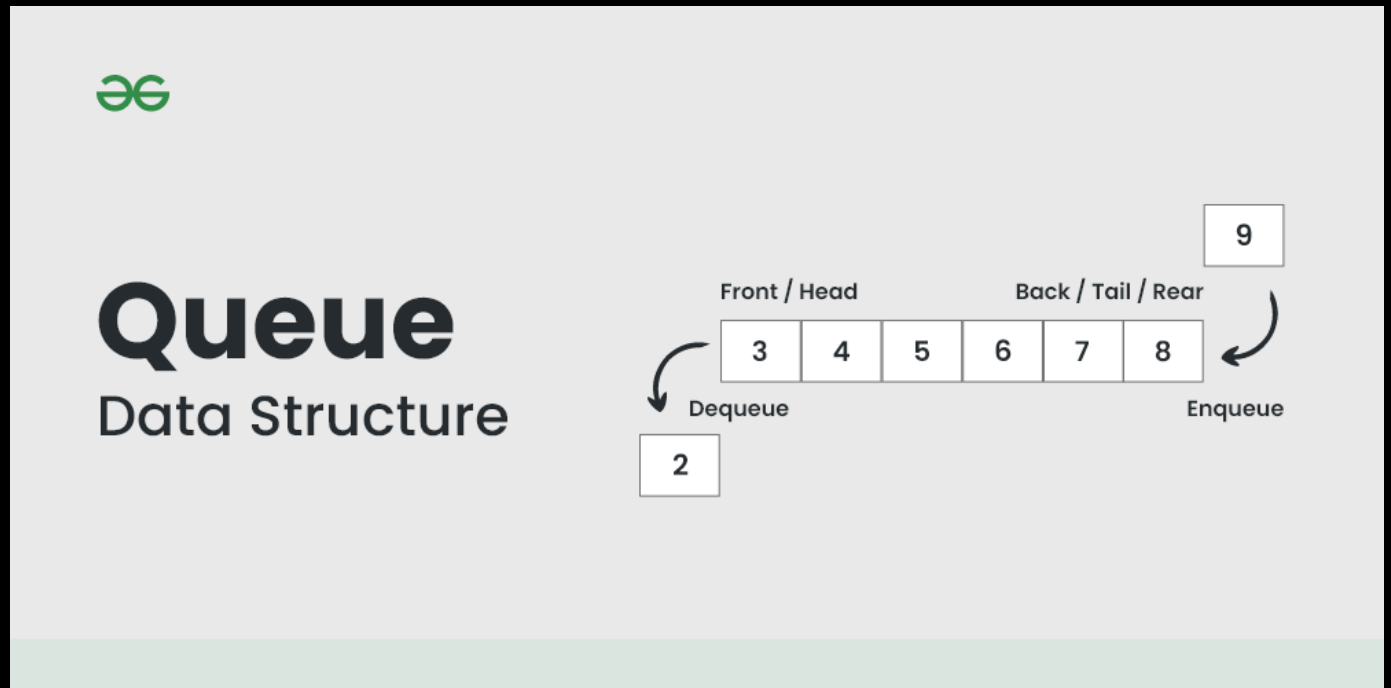
41. STACK

- LIFO (Last In First Out) 규칙을 따르는 선형자료구조
- 자료를 밀어 넣으면 push, 자료를 빼내면 pop이 된다



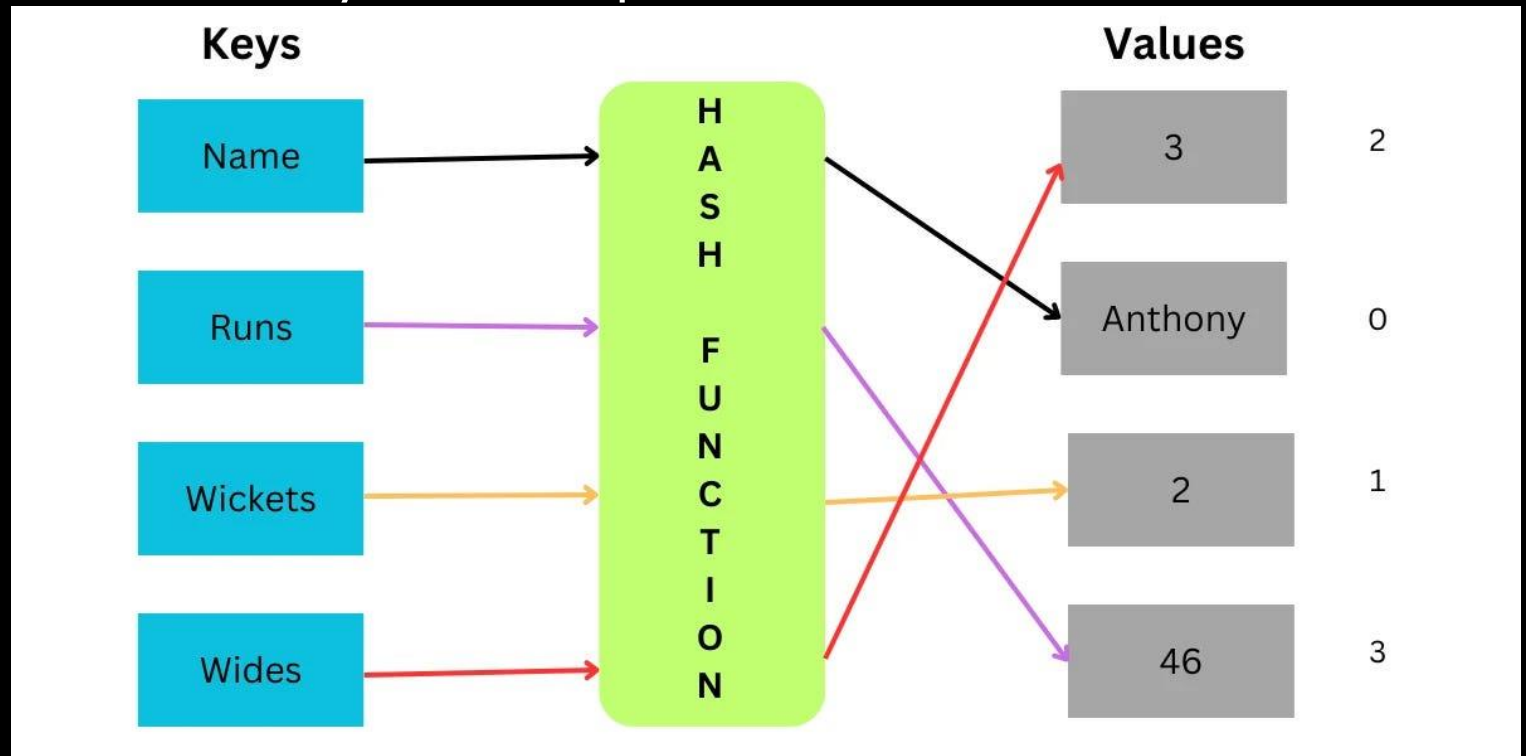
42. QUEUE

- stack와 상반된 선형자료구조, FIFO(First In First Out)을 따른다
- 자료를 넣을 때 put, 자료를 뺄 때 get를 사용한다



43. HASH(MAP/DICT)

- 자료에 키를 할당할 수 있는 자료구조
- 키는 자료를 가리키고 서로 key value pair를 생성한다

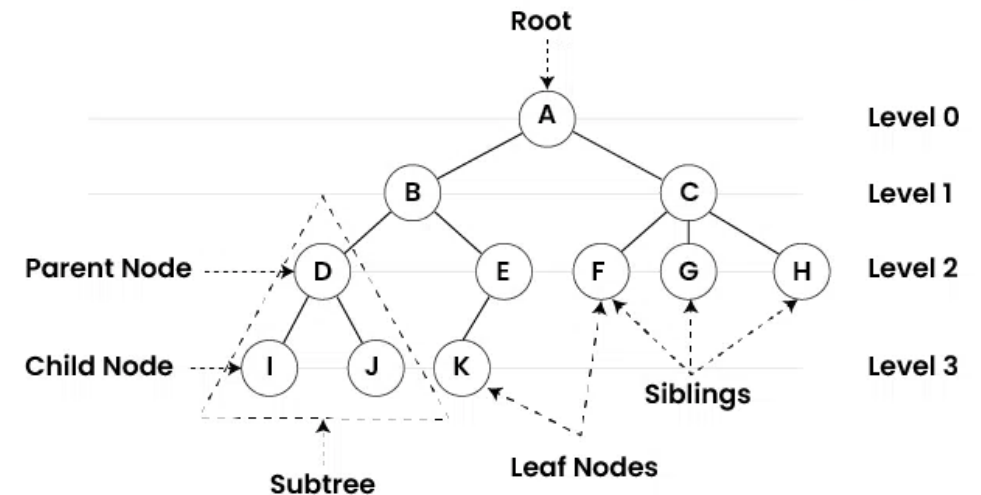


44.TREE

- 비선형적인 자료 구조로써 계급이 나누어진다
- 몇몇 선형자료구조보다 더 빠른 탐색이 가능하다

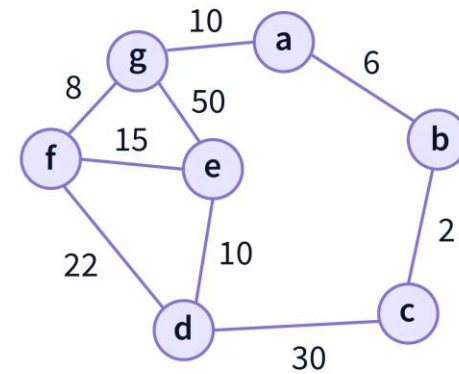
Tree

Data Structure



45. GRAPH

- 다양한 데이터를 나타내는 node와 무수히 많은 경우로 연결할 수 있는 edge로 구성된 자료구조



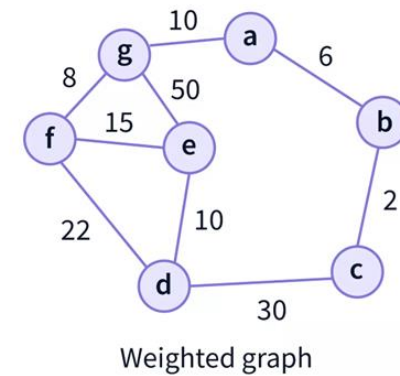
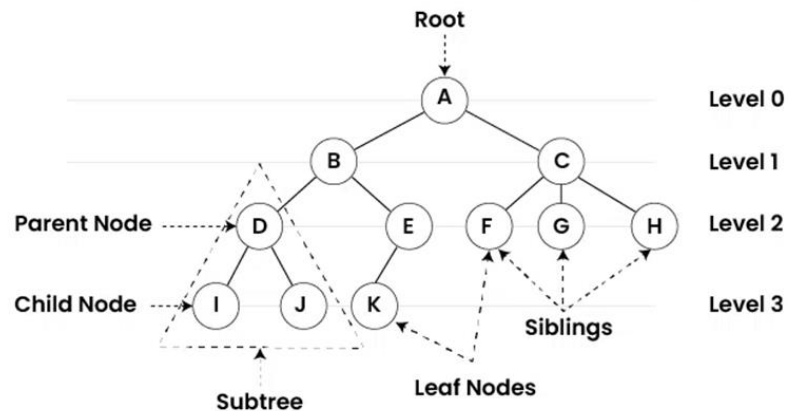
Weighted graph

46. NODE


- Graph, tree 안의 데이터를 나타내는 단위

Tree

Data Structure



SCALER
Topics



감사합니다