

Hanyang Univ.

자료구조론 2주차

강진영

CONTENTS

1

실습 환경 구축

2

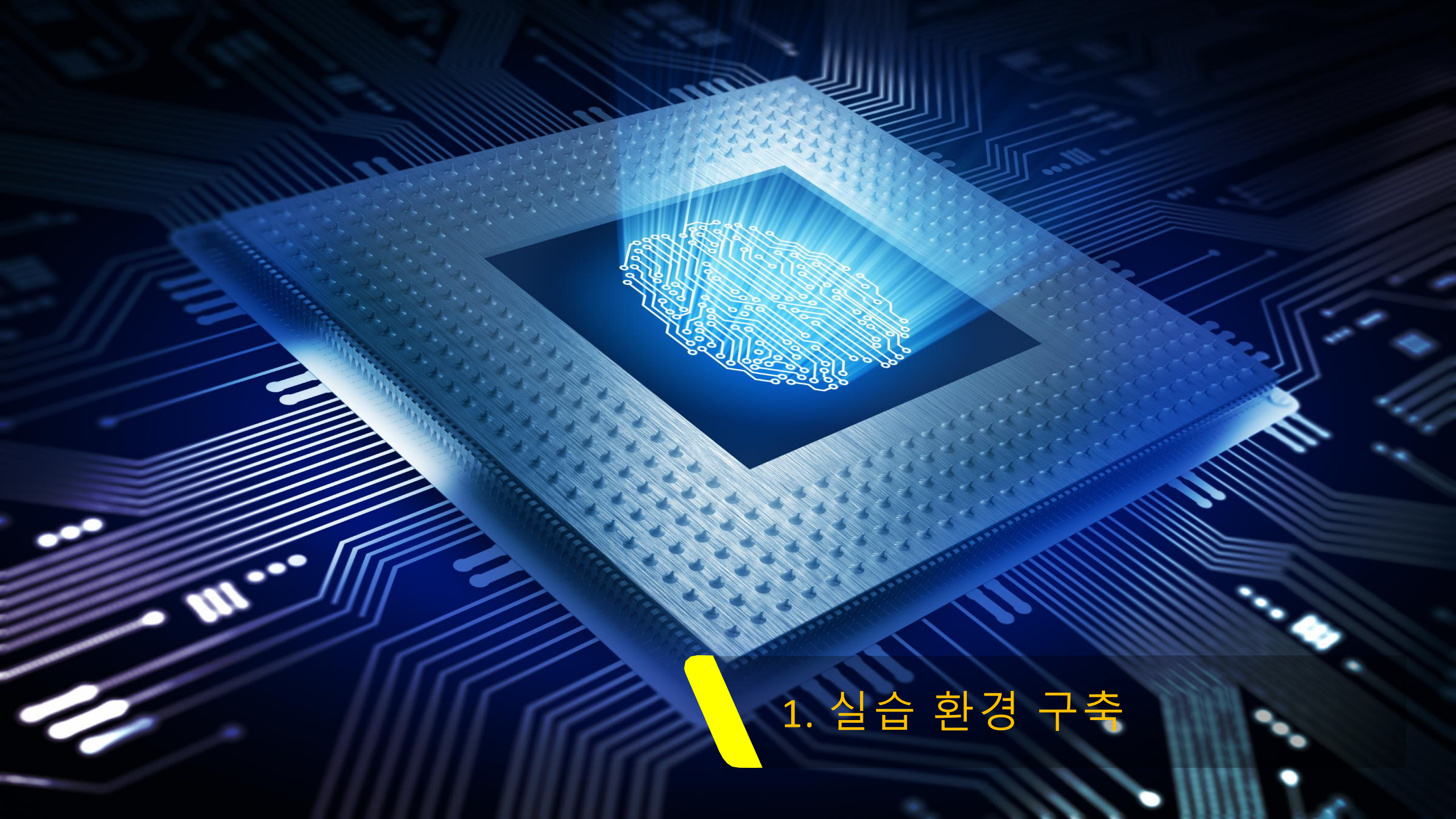
자료형과 형변환

3

배열의 이해

4

배열과 포인터 함께 이해하기



1. 실습 환경 구축

1. 실습 환경 구축

Contents

1. 실습 환경 구축

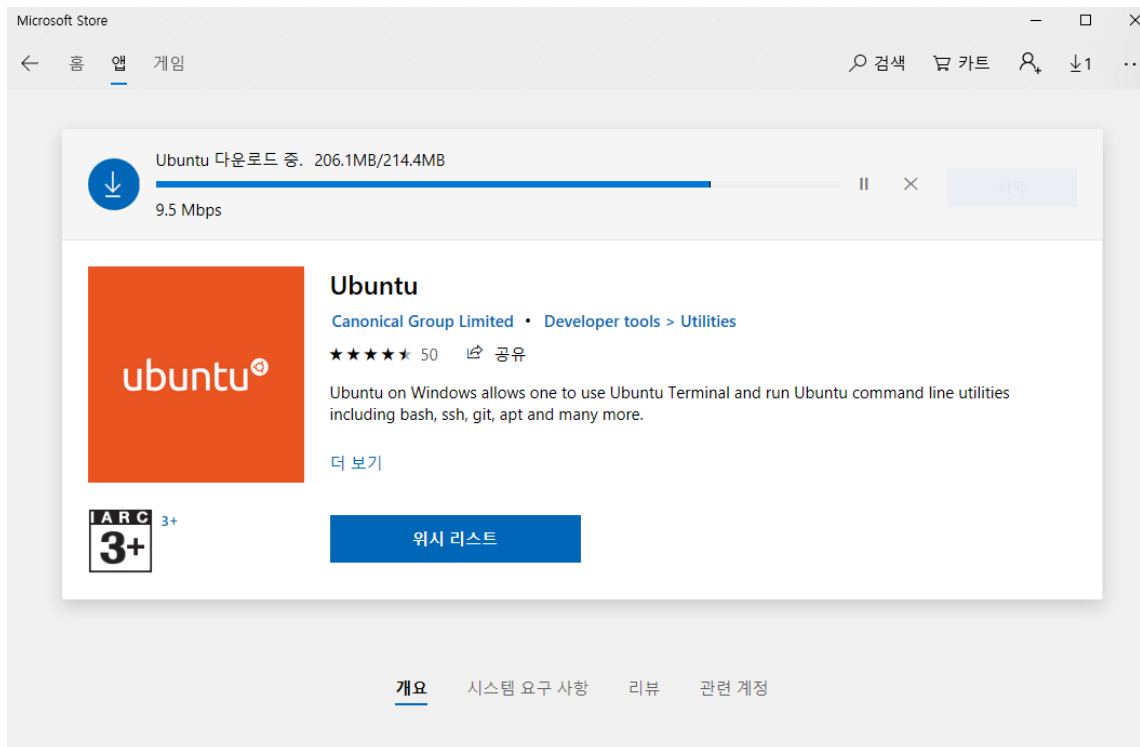
2. 자료형과 형변환

3. 배열

4. 배열과 포인터

Linux용 Windows 하위 시스템

```
C:\Windows\System32\bash.exe
Linux용 Windows 하위 시스템에 배포가 설치되어 있지 않습니다.
아래의 Microsoft Store에서 배포를 설치할 수 있습니다.
https://aka.ms/wslstore
계속하려면 아무 키나 누르세요...
```



1. 실습 환경 구축

Contents

1. 실습 환경 구축

2. 자료형과 형변환

3. 배열

4. 배열과 포인터

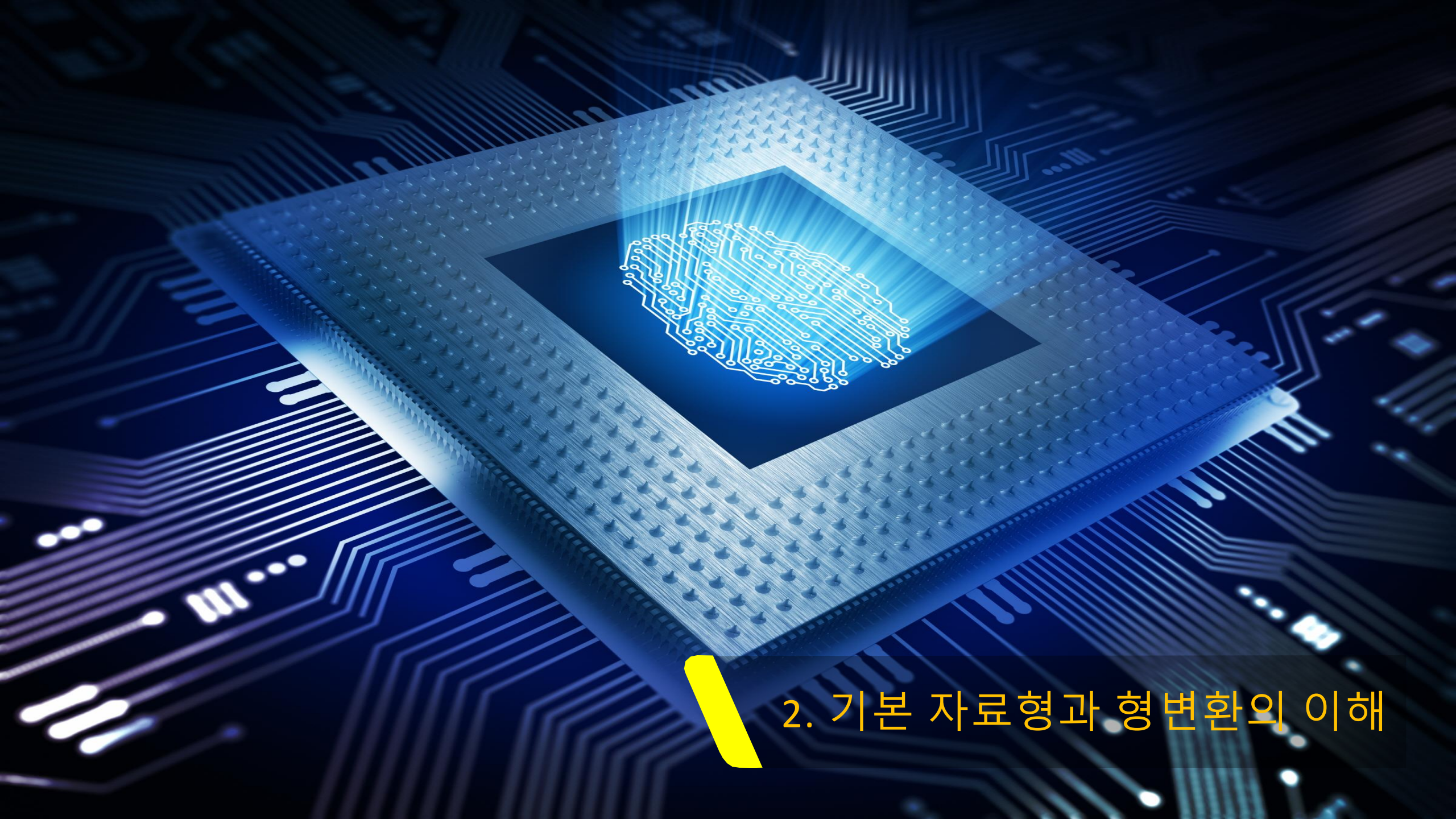
gcc 설치

```
musby@LoveGrace: ~  
musby@LoveGrace:~$ ls  
musby@LoveGrace:~$ pwd  
/home/musby  
musby@LoveGrace:~$ sudo apt install gcc  
[sudo] password for musby:
```

```
musby@LoveGrace: ~  
Get:20 http://archive.ubuntu.com/ubuntu bionic/main amd64 libquadmath0 amd64 8-2  
0180414-1ubuntu2 [134 kB]  
Get:21 http://archive.ubuntu.com/ubuntu bionic/main amd64 libgcc-7-dev amd64 7.3  
.0-16ubuntu3 [2378 kB]  
Get:22 http://archive.ubuntu.com/ubuntu bionic/main amd64 gcc-7 amd64 7.3.0-16ub  
untu3 [7445 kB]  
Get:23 http://archive.ubuntu.com/ubuntu bionic/main amd64 gcc amd64 4:7.3.0-3ubu  
ntu2 [5192 B]  
Get:24 http://archive.ubuntu.com/ubuntu bionic/main amd64 libc-dev-bin amd64 2.2  
7-3ubuntu1 [71.8 kB]  
Ign:25 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 linux-libc-dev  
amd64 4.15.0-29.31  
Err:25 http://security.ubuntu.com/ubuntu bionic-updates/main amd64 linux-libc-de  
v amd64 4.15.0-29.31  
404 Not Found [IP: 91.189.88.161 80]  
Get:26 http://archive.ubuntu.com/ubuntu bionic/main amd64 libc6-dev amd64 2.27-3  
ubuntu1 [2587 kB]  
Get:27 http://archive.ubuntu.com/ubuntu bionic/main amd64 manpages-dev all 4.15-  
1 [2217 kB]  
Fetched 25.9 MB in 2min 31s (171 kB/s)  
E: Failed to fetch http://security.ubuntu.com/ubuntu/pool/main/l/linux/libc-  
dev_4.15.0-29.31_amd64.deb 404 Not Found [IP: 91.189.88.161 80]  
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-mis  
sing?  
musby@LoveGrace:~$ gcc
```

어디서 해당 gcc 패키지를 받아야 하는지
모르기 때문!

→ Repository update 필수
→ “sudo apt update”



2. 기본 자료형과 형변환의 이해

Contents

1. 실습 환경 구축

2. 자료형과 형변환

3. 배열

4. 배열과 포인터

기본 자료형 종류와 데이터의 표현 범위

- **자료형(data type)**
 - "선언할 변수의 특징을 나타내기 위한 키워드"
- **기본 자료형**
 - 기본적으로 제공이 되는 자료형
- **사용자 정의 자료형**
 - 사용자가 정의하는 자료형 : 구조체, 공용체

```
int val;
```


2. 자료형과 형변환

Contents

1. 실습 환경 구축

2. 자료형과 형변환

3. 배열

4. 배열과 포인터

기본 자료형 종류와 데이터의 표현 범위

자료형(data type)		할당되는 메모리 크기	표현 가능한 데이터의 범위
정수형	char	1 바이트	-128 ~ +127
	short	2 바이트	-32768 ~ +32767
	int	4 바이트	-2147483648 ~ +2147483647
	long	4 바이트	-2147483648 ~ +2147483647
실수형	float	4 바이트	$3.4 \times 10^{-37} \sim 3.4 \times 10^{+38}$
	double	8 바이트	$1.7 \times 10^{-307} \sim 1.7 \times 10^{+308}$
	long double	8 바이트 혹은 그 이상	차이를 많이 보임

Contents

1. 실습 환경 구축

2. 자료형과 형변환

3. 배열

4. 배열과 포인터

기본 자료형 종류와 데이터의 표현 범위

- 다양한 자료형이 제공되는 이유
 - 데이터의 표현 방식이 다르기 때문
 - 정수형 데이터를 표현하는 방식
 - 실수형 데이터를 표현하는 방식
 - 메모리 공간을 적절히 사용하기 위해서
 - 데이터의 표현 범위를 고려해서 자료형 선택
 - 작은 메모리 공간에 큰 데이터를 저장하는 경우
데이터 손실이 발생할 수 있음

기본 자료형 종류와 데이터의 표현 범위

- 자료형 선택의 기준

- 정수형 데이터를 처리하는 경우

- 컴퓨터는 내부적으로 int형 연산을 가장 빠르게 처리,
따라서 정수형 변수는 int형으로 선언
- 범위가 int형 변수를 넘어가는 경우 long형으로 선언
- 값의 범위가 $-128 \sim +127$ 사이라 할지라도 int형으로 선언

기본 자료형 종류와 데이터의 표현 범위

- 자료형 선택의 기준

- 실수형 데이터를 처리하는 경우

- 선택의 지표는 정밀도
- 정밀도란 오차 없이 표현 가능한 정도를 의미함
- 오늘날의 일반적 선택은 double!

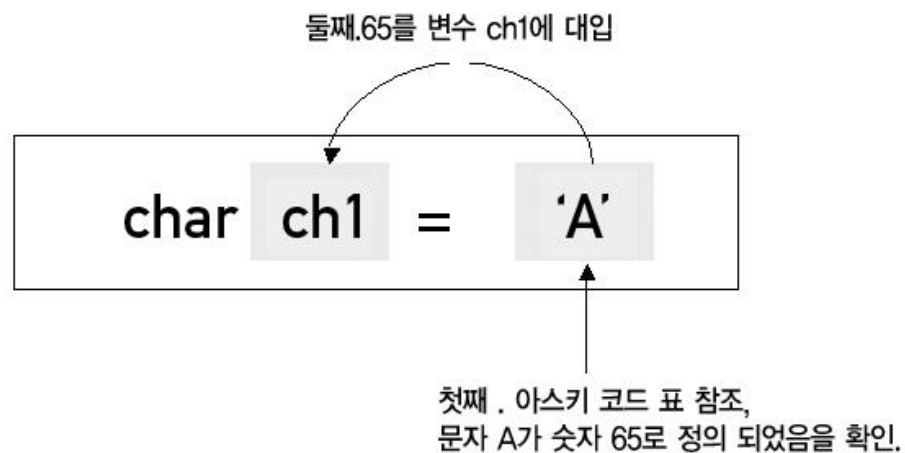
자료형	정밀도
float	소수 이하 6자리
double	소수 이하 15자리
long double	double의 정밀도와 같거나 크다.

Contents

- 1. 실습 환경 구축
- 2. 자료형과 형변환
- 3. 배열
- 4. 배열과 포인터

기본 자료형 종류와 데이터의 표현 범위

- **ASCII 코드의 범위**
 - 0이상 127이하, char형 변수로 처리 가능
 - char형으로 처리하는 것이 합리적
- **문자의 표현**
 - 따옴표(' ')를 이용해서 표현



Contents

1. 실습 환경 구축

2. 자료형과 형변환

3. 배열

4. 배열과 포인터

자료형 변환에 관한 이야기

- **자동 형 변환이 발생하는 상황 1**
 - 대입 연산 시

```
int main(void)
{
    int n=5.25;    // 소수부의 손실
    double d=3;    // 값의 표현이 넓은 범위로의 변환
    char c=129;    // 상위 비트의 손실
```

Contents

1. 실습 환경 구축

2. 자료형과 형변환

3. 배열

4. 배열과 포인터

자료형 변환에 관한 이야기

- 자동 형 변환이 발생하는 상황 2
 - 정수의 승격에 의해(int형 연산이 빠른 이유)
 - 정수형 연산 자체를 단일화시킨 결과

```
int main(void)
{
    char c1=10, c2=20;
    char c3=c1+c2;
    . . . . .
```


Contents

1. 실습 환경 구축

2. 자료형과 형변환

3. 배열

4. 배열과 포인터

자료형 변환에 관한 이야기

- **자동 형 변환이 발생하는 상황 3**
 - 산술 연산 과정에 의해

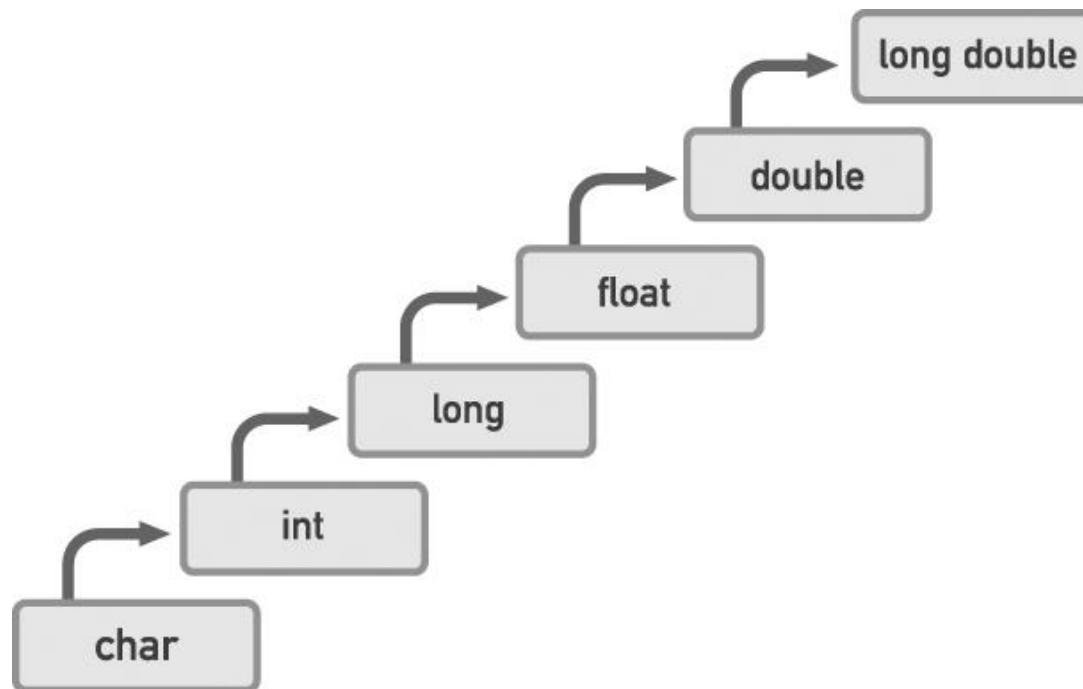
```
int main(void)
{
    double e1 = 5.5 + 7;           // double + int
    double e2 = 3.14f+5.25; // float + double
    . . . . .
```

Contents

- 1. 실습 환경 구축
- 2. 자료형과 형변환
- 3. 배열
- 4. 배열과 포인터

자료형 변환에 관한 이야기

- 산술 연산 형 변환 규칙
 - 데이터의 손실이 최소화되는 방향으로...



Contents

1. 실습 환경 구축

2. 자료형과 형변환

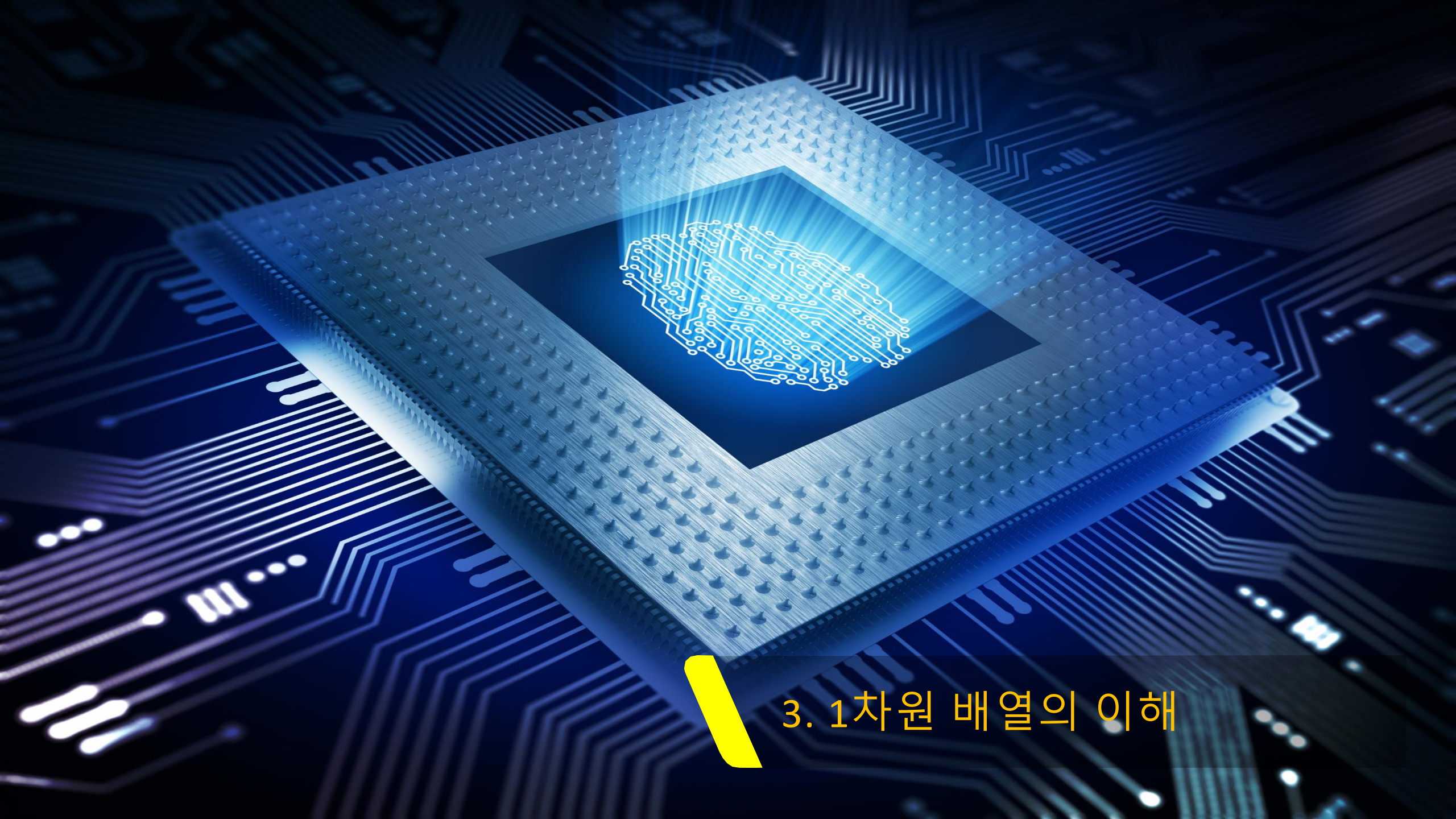
3. 배열

4. 배열과 포인터

자료형 변환에 관한 이야기

- **강제 형 변환**
 - **프로그래머의 요청에 의한 형 변환**

```
float f= (float)3.14;           // 3.14를 float 형으로 형 변환
double e1 = 3 + 3.14;          //정수 3이 double 형으로 자동 형 변환
double e2 = 3 + (int)3.14;      // 3.14가 int형으로 강제 형 변환
```

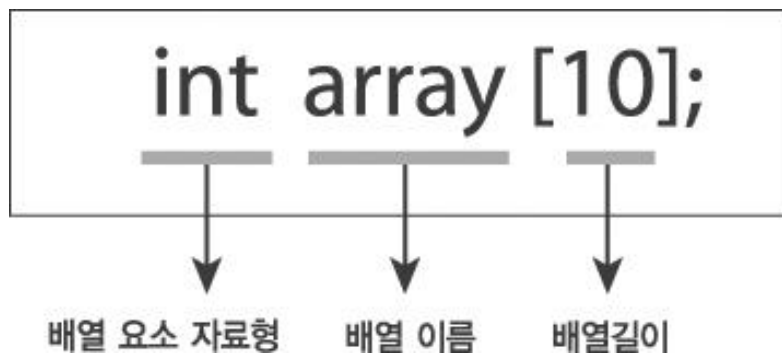
3. 1차원 배열의 이해

Contents

- 1. 실습 환경 구축
- 2. 자료형과 형변환
- 3. 배열
- 4. 배열과 포인터

1차원 배열의 선언 및 초기화

- 배열 선언에 있어서 필요한 것 세 가지
 - 배열 길이 - 배열을 구성하는 변수의 개수 (반드시 상수를 사용)
 - 배열 요소 자료형 : 배열을 구성하는 변수의 자료형
 - 배열 이름 : 배열에 접근할 때 사용되는 이름

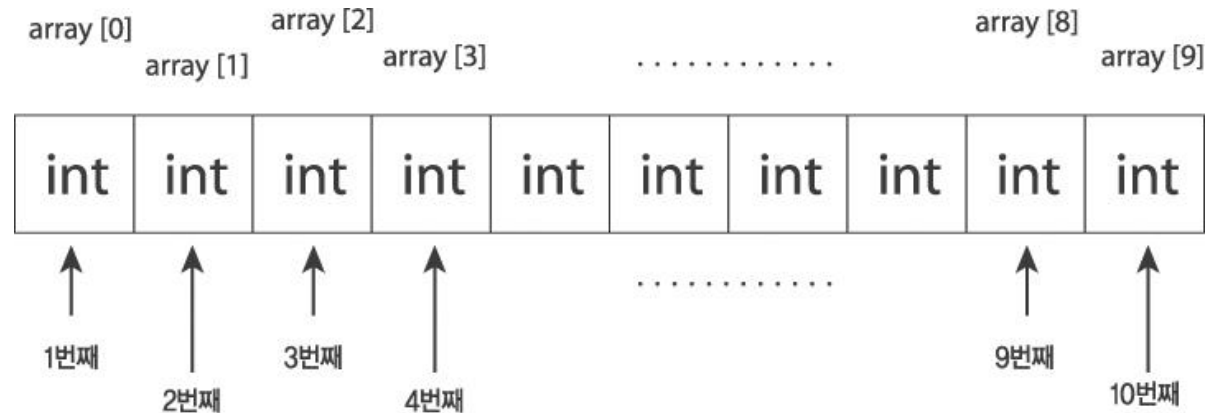


Contents

1. 실습 환경 구축
2. 자료형과 형변환
3. 배열
4. 배열과 포인터

1차원 배열의 선언 및 초기화

- 1차원 배열의 접근
 - 배열 요소의 위치를 표현 : 인덱스(index)
 - 인덱스는 0에서부터 시작



`int` : int형 정수 저장을 위한 4바이트 메모리 블록

Contents

1. 실습 환경 구축
2. 자료형과 형변환
3. 배열
4. 배열과 포인터

1차원 배열의 선언 및 초기화

- 배열 선언과 접근의 예

```
int main(void)
{
    int array[10];    // 배열 선언
    array[0]=10;      // 첫 번째 요소 접근
    array[1]=20;      // 두 번째 요소 접근
    array[2]=30;      // 세 번째 요소 접근
    .....
    return 0;
}
```

array[s] = 10;



S+1번째 요소에 10을 대입하라.

Contents

- 1. 실습 환경 구축
- 2. 자료형과 형변환
- 3. 배열
- 4. 배열과 포인터

1차원 배열의 선언 및 초기화

```
/* array1.c */
#include <stdio.h>

int main(void)
{
    double total;
    double val[5];

    val[0]=1.01;
    val[1]=2.02;
    val[2]=3.03;
    val[3]=4.04;
    val[4]=5.05;

    total=val[0]+val[1]+val[2]+val[3]+val[4];
    printf("평균 : %lf \n", total/5);

    return 0;
}
```

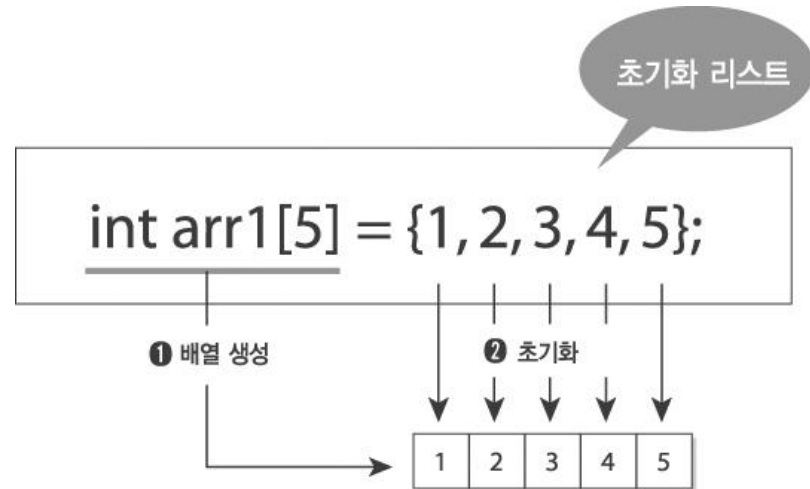
Contents

1. 실습 환경 구축
2. 자료형과 형변환
3. 배열
4. 배열과 포인터

1차원 배열의 선언 및 초기화

- 선언과 동시에 초기화

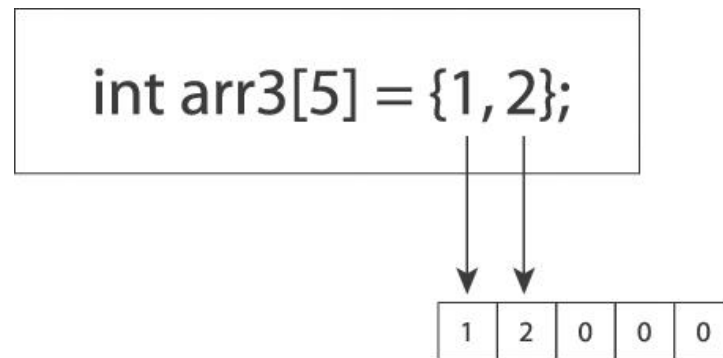
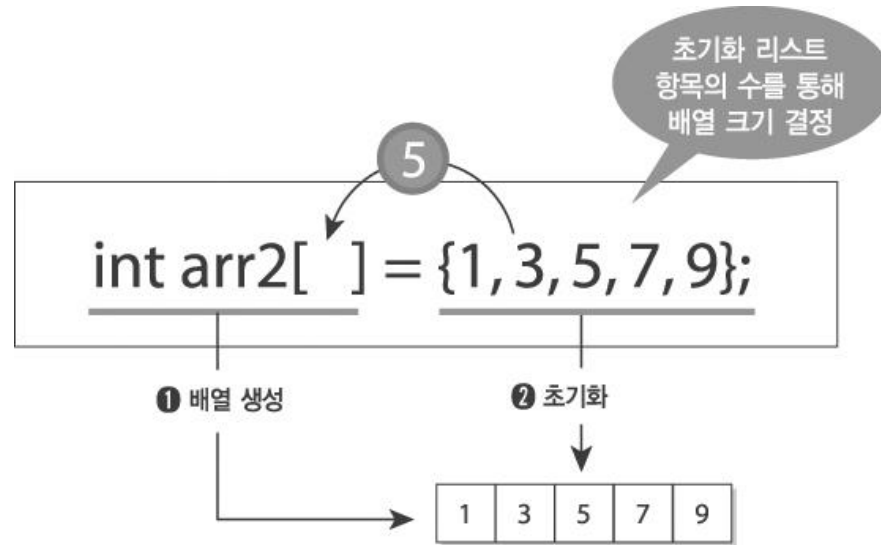
```
int main(void)
{
    int arr1[5]={1, 2, 3, 4, 5};
    int arr2[ ]={1, 3, 5, 7, 9};
    int arr3[5]={1, 2}
}
```

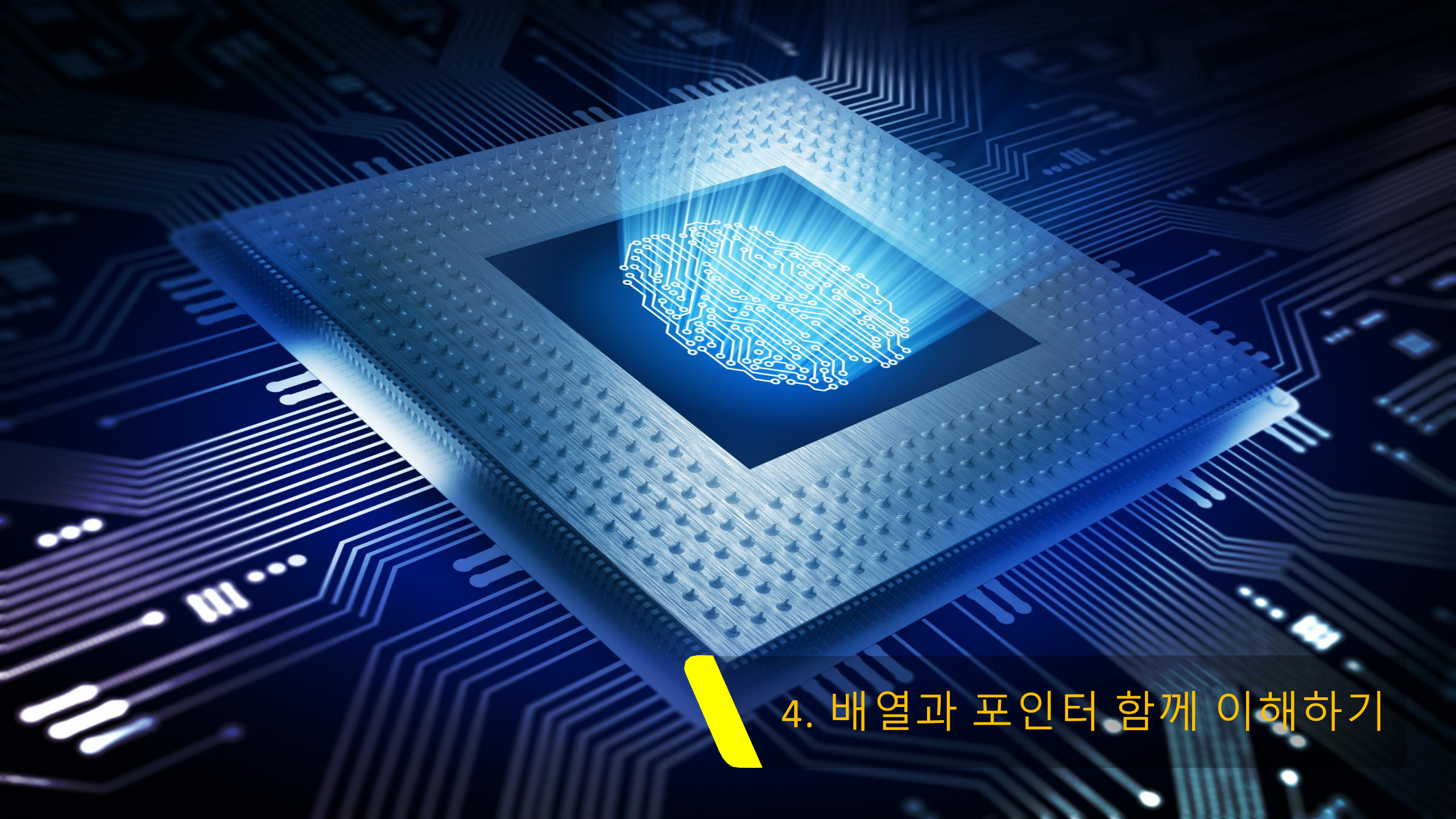


Contents

1. 실습 환경 구축
2. 자료형과 형변환
3. 배열
4. 배열과 포인터

1차원 배열의 선언 및 초기화





4. 배열과 포인터 함께 이해하기

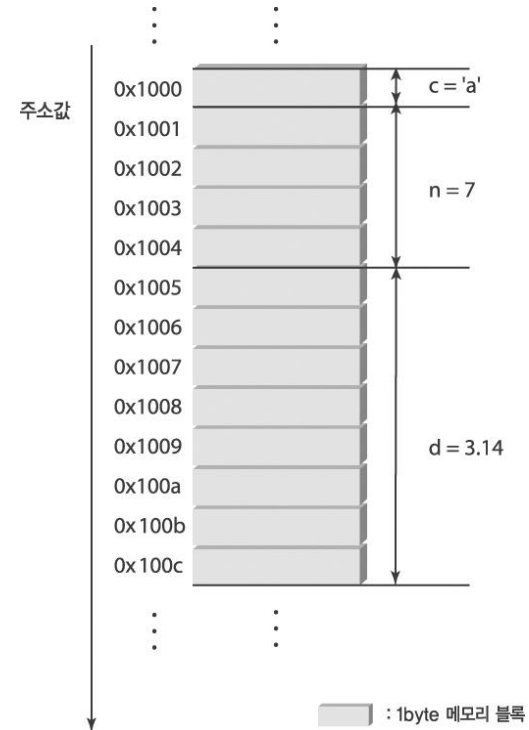
Contents

1. 실습 환경 구축
2. 자료형과 형변환
3. 배열
4. 배열과 포인터

포인터란

- **포인터와 포인터 변수**
 - 메모리의 주소 값을 저장하기 위한 변수
 - "포인터"를 흔히 "포인터 변수"
 - 주소 값과 포인터는 상이

```
int main(void)
{
    char c='a';
    int n=7;
    double d=3.14;
    . . . . .
```

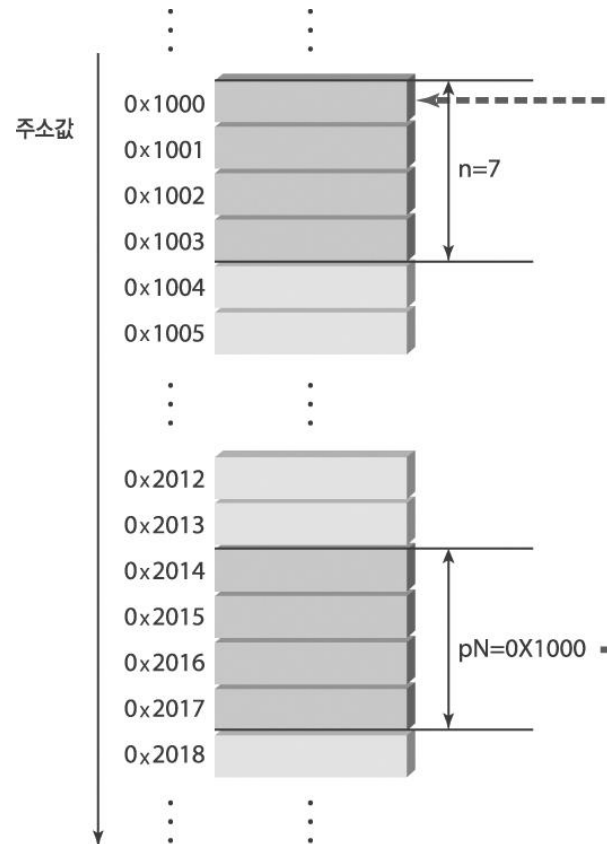


Contents

1. 실습 환경 구축
2. 자료형과 형변환
3. 배열
4. 배열과 포인터

포인터란

- 그림을 통한 포인터의 이해
 - 컴퓨터의 주소 체계에 따라 크기가 결정
 - 32비트 시스템 기반 : 4 바이트



Contents

- 1. 실습 환경 구축
- 2. 자료형과 형변환
- 3. 배열
- 4. 배열과 포인터

포인터란

- 포인터의 타입과 선언
 - 포인터 선언 시 사용되는 연산자 : *
 - A형 포인터(A*) : A형 변수의 주소 값을 저장

```
int main(void)
{
    int *a;          // a라는 이름의 int형 포인터
    char *b;         // b라는 이름의 char형 포인터
    double *c;       // c라는 이름의 double형 포인터
    . . . . .
```

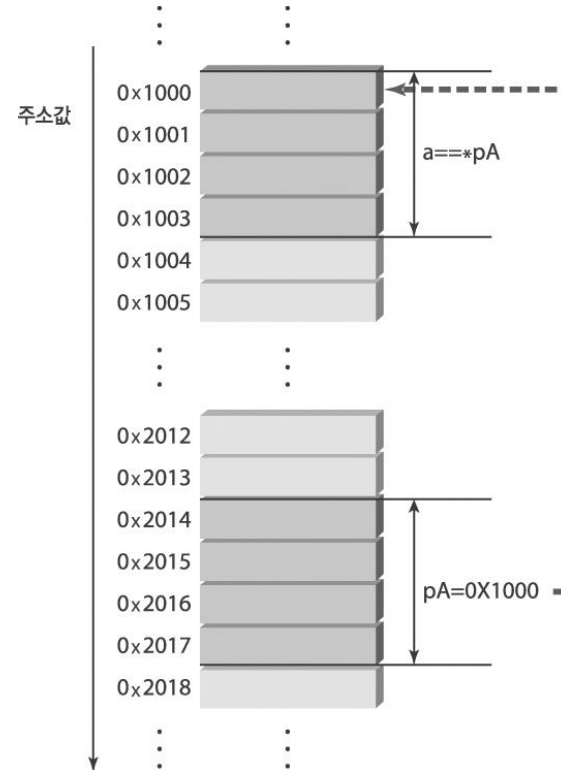
Contents

1. 실습 환경 구축
2. 자료형과 형변환
3. 배열
4. 배열과 포인터

포인터란

- 주소 관련 연산자
 - & 연산자 : 변수의 주소 값 반환
 - * 연산자 : 포인터가 가리키는 메모리 참조

```
int main(void)
{
    int a=2005;
    int *pA=&a;
    printf("%d", a); //직접 접근
    printf("%d", *pA); // 간접 접근
    . . . . .
}
```



Contents

- 1. 실습 환경 구축
- 2. 자료형과 형변환
- 3. 배열
- 4. 배열과 포인터

포인터란

```
/* pointer1.c */
#include <stdio.h>

int main(void)
{
    int a=2005;
    int* pA=&a;

    printf("pA : %d \n", pA);
    printf("&a : %d \n", &a);

    (*pA)++;          //a++와 같은 의미를 지닌다.

    printf("a   : %d \n", a);
    printf("*pA : %d \n", *pA);

    return 0;
}
```

Contents

1. 실습 환경 구축
2. 자료형과 형변환
3. 배열
4. 배열과 포인터

포인터란

- 포인터에 다양한 타입이 존재하는 이유
 - 포인터 타입은 참조할 메모리의 크기 정보를 제공

```
#include <stdio.h>
int main(void)
{
    int a=10;
    int *pA = &a;
    double e=3.14;
    double *pE=&e;

    printf("%d %f", *pA, *pE);
    return 0;
}
```


Q & A

