전산 SMP 5주차

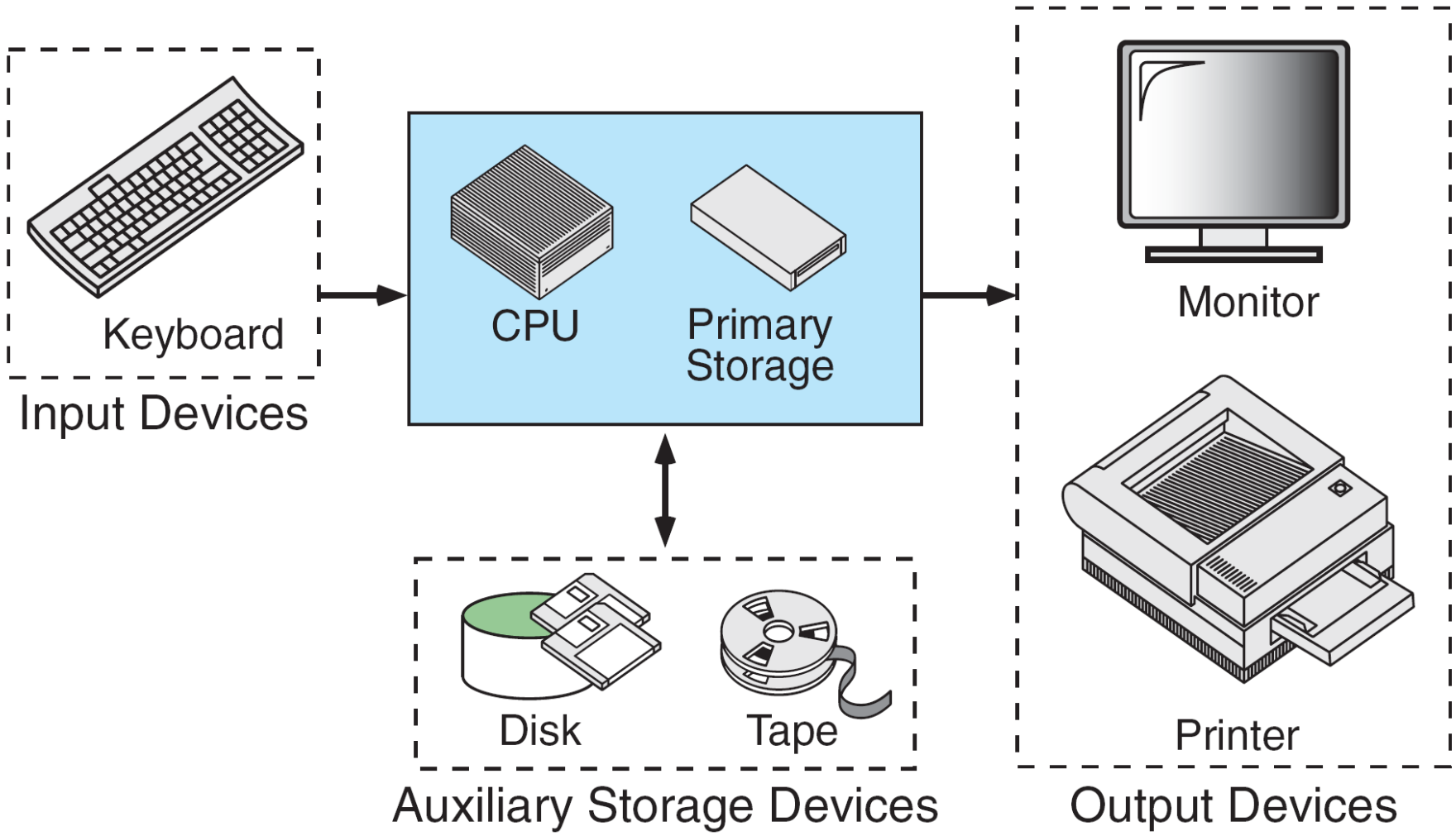
2014. 10. 18

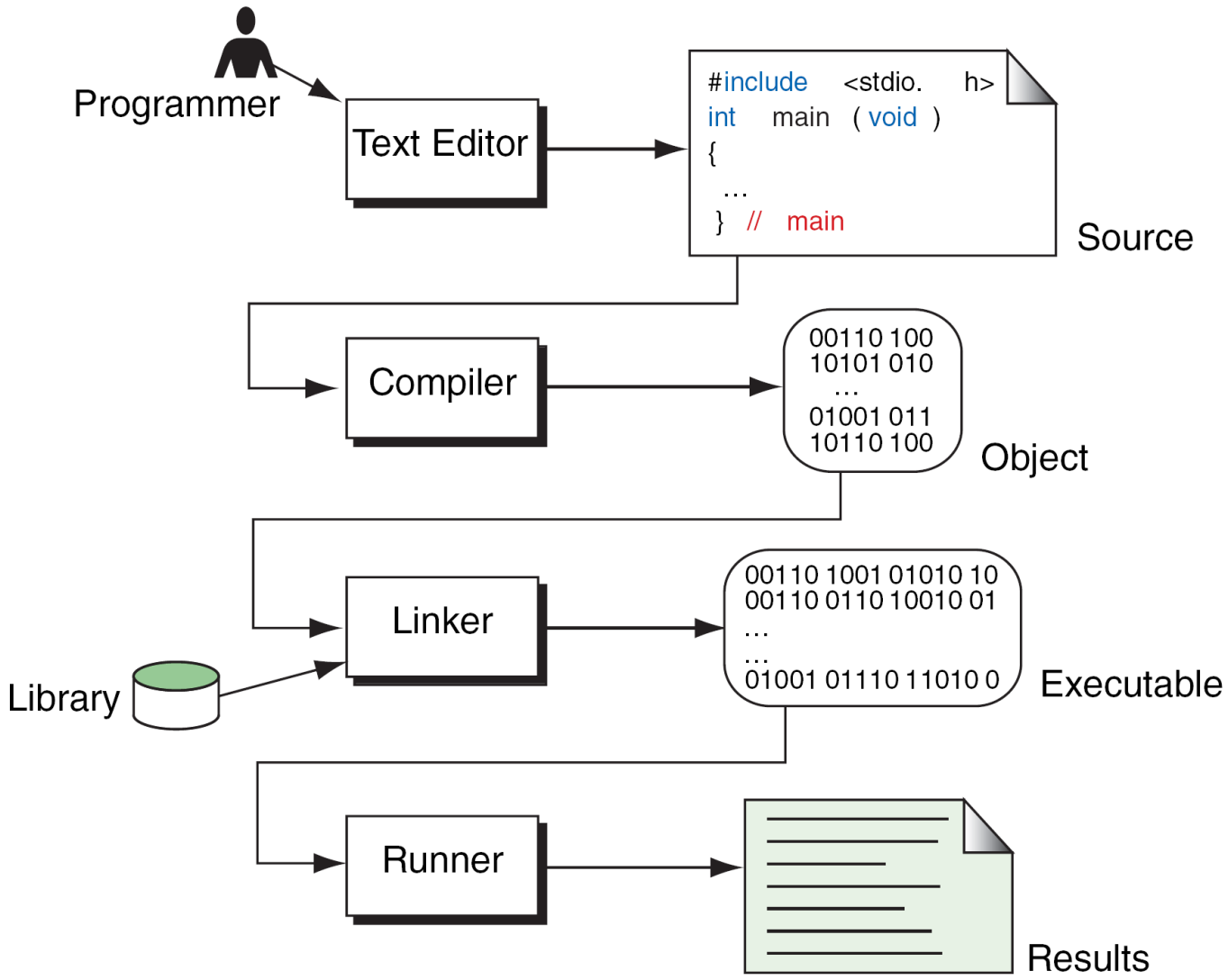
김범수
bskim45@gmail.com

Special thanks to 박기석 (kisuk0521@gmail.com)

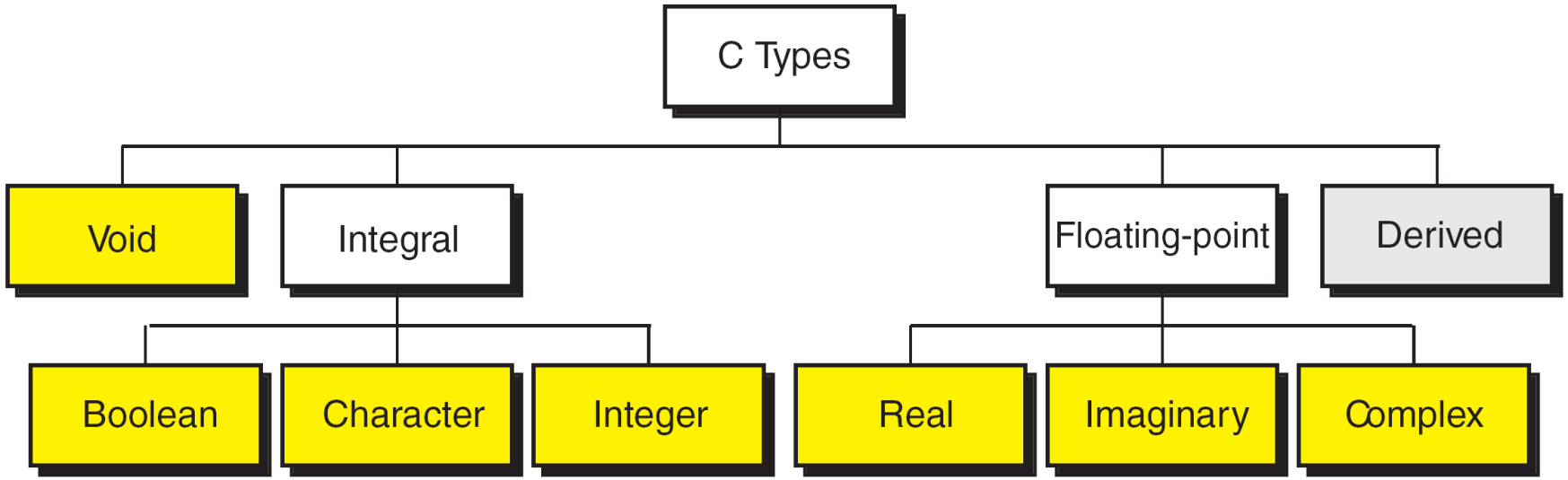
지난시간 복습

1주차~중간고사 범위


Keyboard
Input Devices
CPU
Primary Storage
Disk
Tape
Auxiliary Storage Devices
Monitor
Printer
Output Devices


Programmer
Text Editor
#include <stdio. h>
int main (void)
{
…
} // main
Source
Compiler
00110 100
10101 010
…
01001 011
10110 100
Object
Linker
00110 1001 01010 10
00110 0110 10010 01
…
…
01001 01110 11010 0
Executable
Library
Runner
Results

자료형과 형변환


C Types
Void
Integral
Floating-point
Derived
Boolean
Character
Integer
Real
Imaginary
Complex

연산자, 우선순위, 결합순서

- 산술연산자 : +, −, *, /, %
- 대입연산자 : =, +=, −=, *=, /=, %=, 〈〈=, 〉〉=,
  &=, ^=, |=
- 관계(비교)연산자 : 〉, 〈, 〉=, 〈=, ,!=
- 논리연산자 : &&, ||, !
- 증가/감소 연산자 : ++, −−
- 비트연산자 : &, |, ^, ~
- 시프트 연산자 : 〈〈, 〉〉
- 주소 연산자 : &

조건문 (Selection Statement)

- 조건 (Condition)에 따라서 선택적으로 프로그램을 진행
- 프로그램의 Flow를 조종

2-Way Selection
  - if~else

Multi-Way Selection
  - if~else if~else, switch

- 조건문 안에 얼마든지 또 조건문을 넣을 수 있다. (nested)

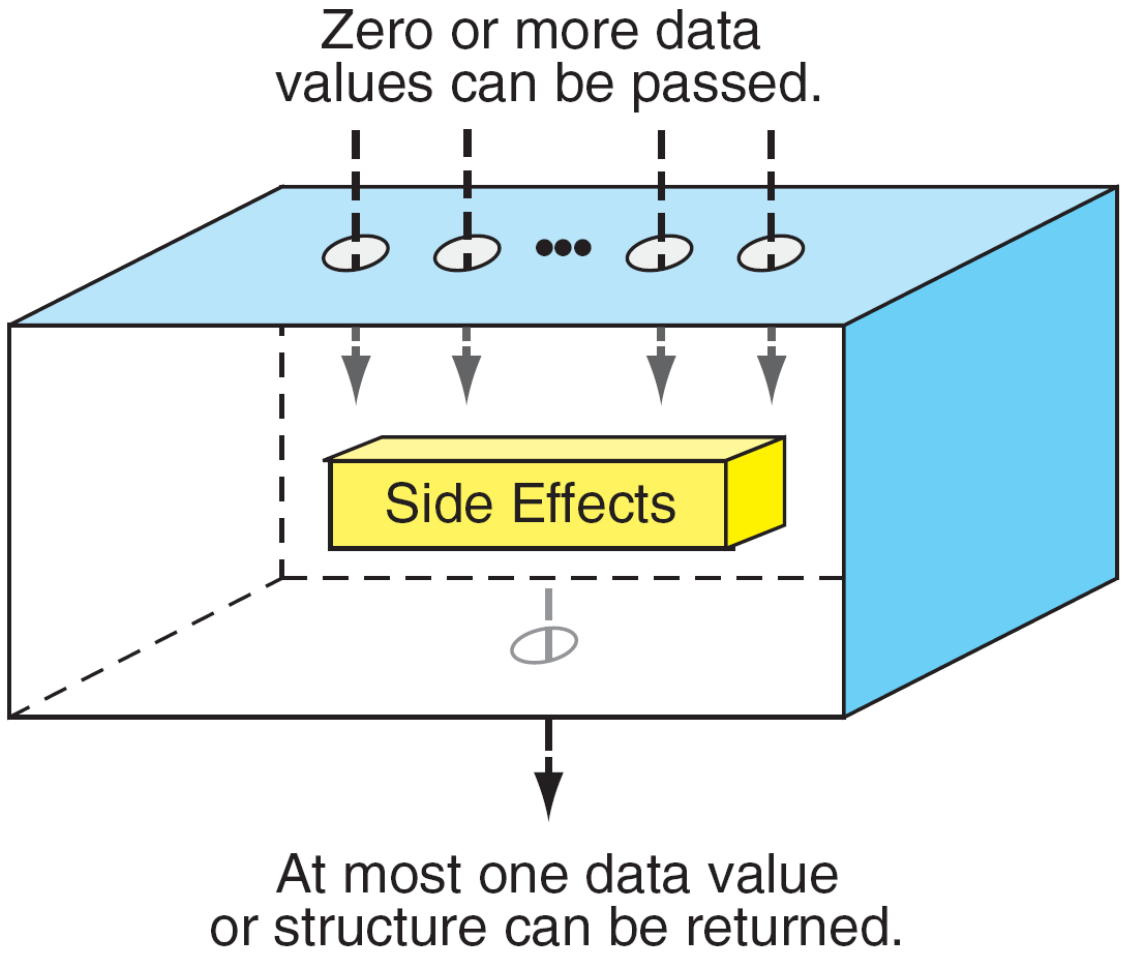
Loop (고리)

- pre-test loop: 검사하고 실행하기
  while 문, for 문
- post-test loop: 실행하고 검사하기
  do~while 문

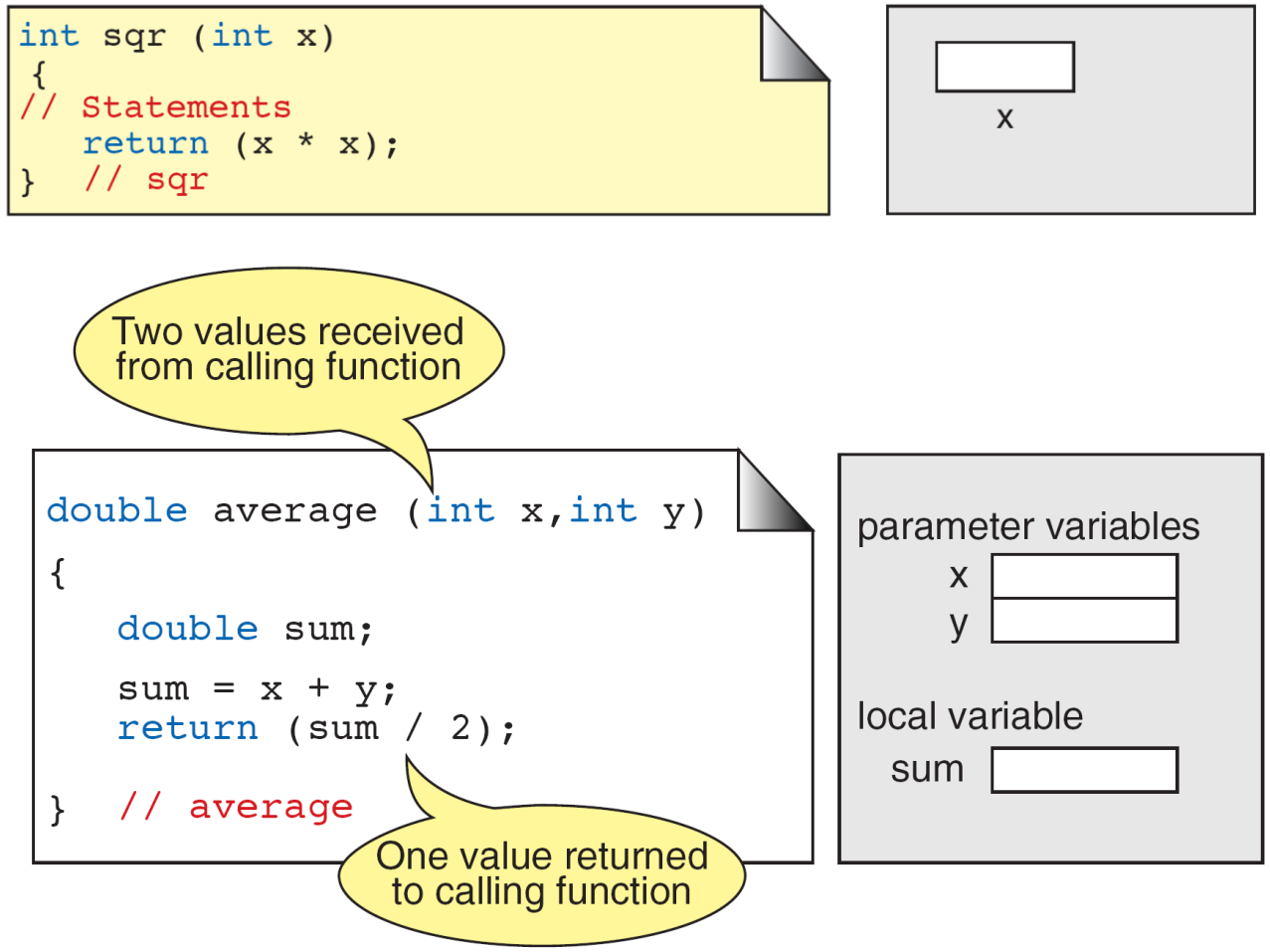
- Requirement
  - Initialization
  - Update
  - Condition Check

- 언제 무엇을 쓰나요 – “주로”
  for는 반복 횟수를 알 때, while, do~while은 모를 때

함수


Zero or more data
values can be passed.
Side Effects
At most one data value
or structure can be returned.

메모리 공간은 함수마다 따로따로

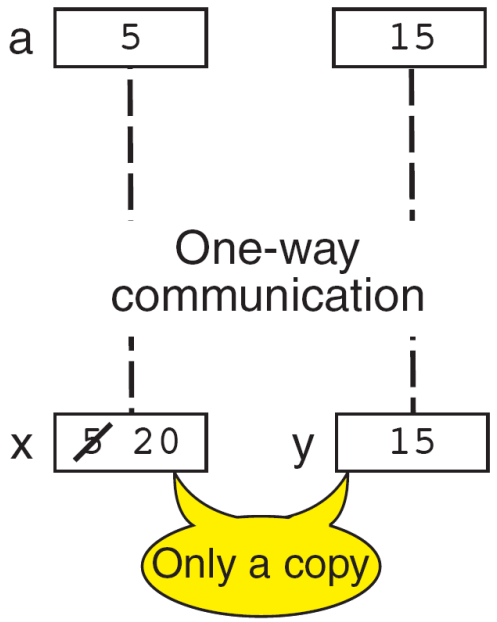


Call-by-Value

```
// Function Declaration
void downFun (int x, int y);
int main (void)
{
// Local Definitions
   int a = 5;
// Statements
   downFun (a, 15);
   printf("%d\n", a);
   return 0;
}  // main
```

prints 5

```
   void downFun (int x, int y)
   {
   // Statements
      x = x + y;
      return;
   }  // downFun
```



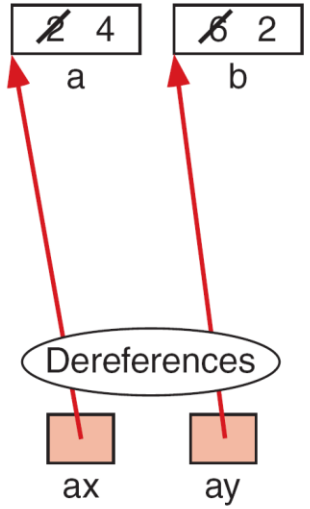
Call-by-Reference

```
// Function Declaration
void biFun (int* ax, int* ay);

int main (void)
{
// Local Definitions
   int  a = 2;
   int  b = 6;

// Statements
   …
   biFun (&a, &b);
   …
 return 0;
}  // main
```

```
void biFun (int* ax, int* ay)
{
   *ax = *ax + 2;
   *ay = *ay / *ax;
   return;
}  // biFun
```



SWAP

```
// Function Declarations
void exchange (int* num1, int* num2);

int main (void)
{
// Local Definitions
   int  a;
   int  b;

// Statements
   …
   exchange (&a, &b);
   …
 return 0;
}  // main
```

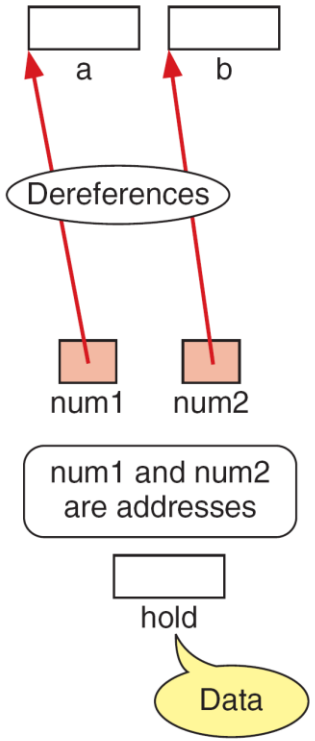
Note that the type includes an asterisk.

Address operators

```
void exchange (int* num1, int* num2)
{
// Local Definitions
   int  hold;

// Statements
   hold   =  *num1;
   *num1  =  *num2;
   *num2  =   hold;
   return;
}  // exchange
```

Note the indirection operator is used for dereferencing.



Recursion

필수조건

- Base Case (n=1)
- n=k → n=k+1?

왜 쓰나요?

- 장점: 문제를 단순하게 풀 수 있다
- 단점: 메모리 소모가 많다

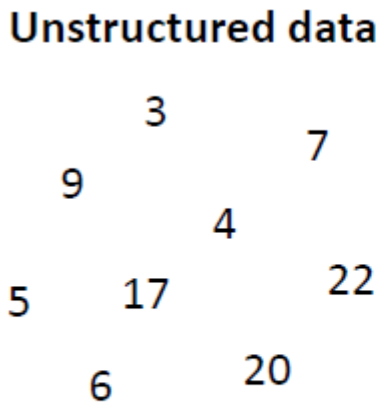
오늘 할 것

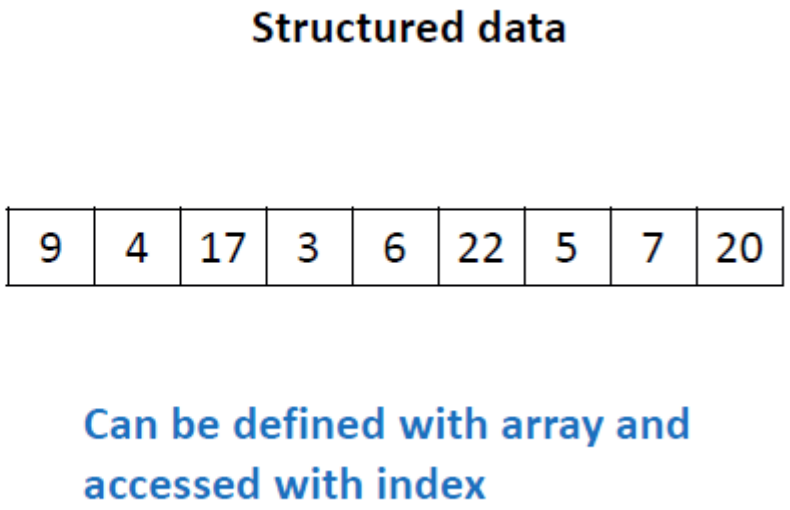
- Array + Search(Sequential)
- File I/O

자료구조 Data Structure

- A particular way of **storing and organizing data** in a computer so that it can be **used efficiently**.
- 많은 양의 데이터를 효과적으로 관리, 사용하기 위해 (ex. 인터넷 검색)
- 좋은 자료구조는 데이터를 빠르게 검색할 수 있다.

- 종류
  - Matrices
  - Linked lists
  - Priority queues
    - stack, queue, etc.
  - Hash tables
  - Trees and graphs

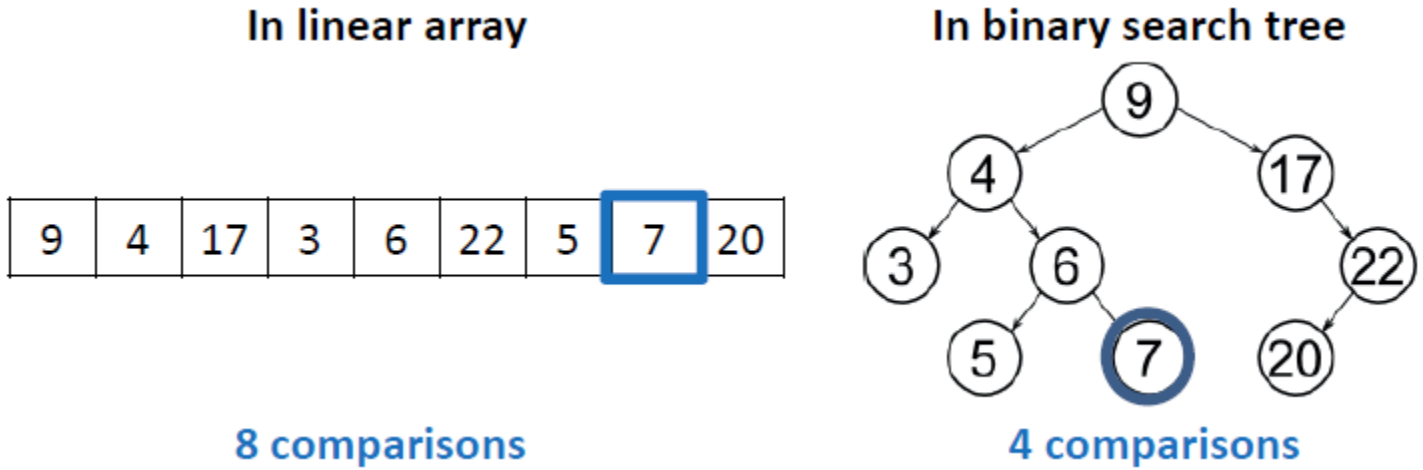




왜 쓰나요

- 많은 데이터를 예쁘게 잘 저장해서
- 쉽고 빠르게 꺼내 쓰려고
- 데이터가 많아지면 찾는데도 오래 걸린다.

- Example: finding a number from a set of numbers
  - How many comparisons do we need to retrieve 7?



Array

Array가 뭔가요

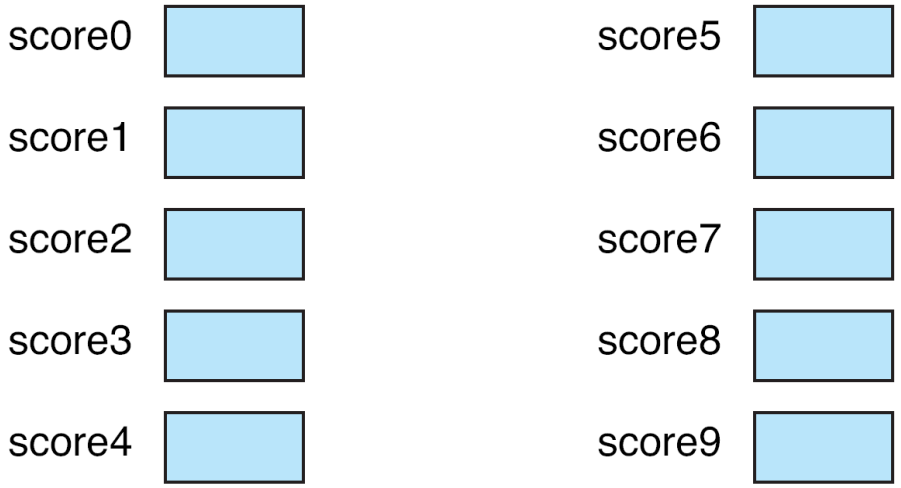
- 연관성 있는 변수들을 하나로 묶어서 보관하는 데이터 타입

Array를 왜 쓰나요?

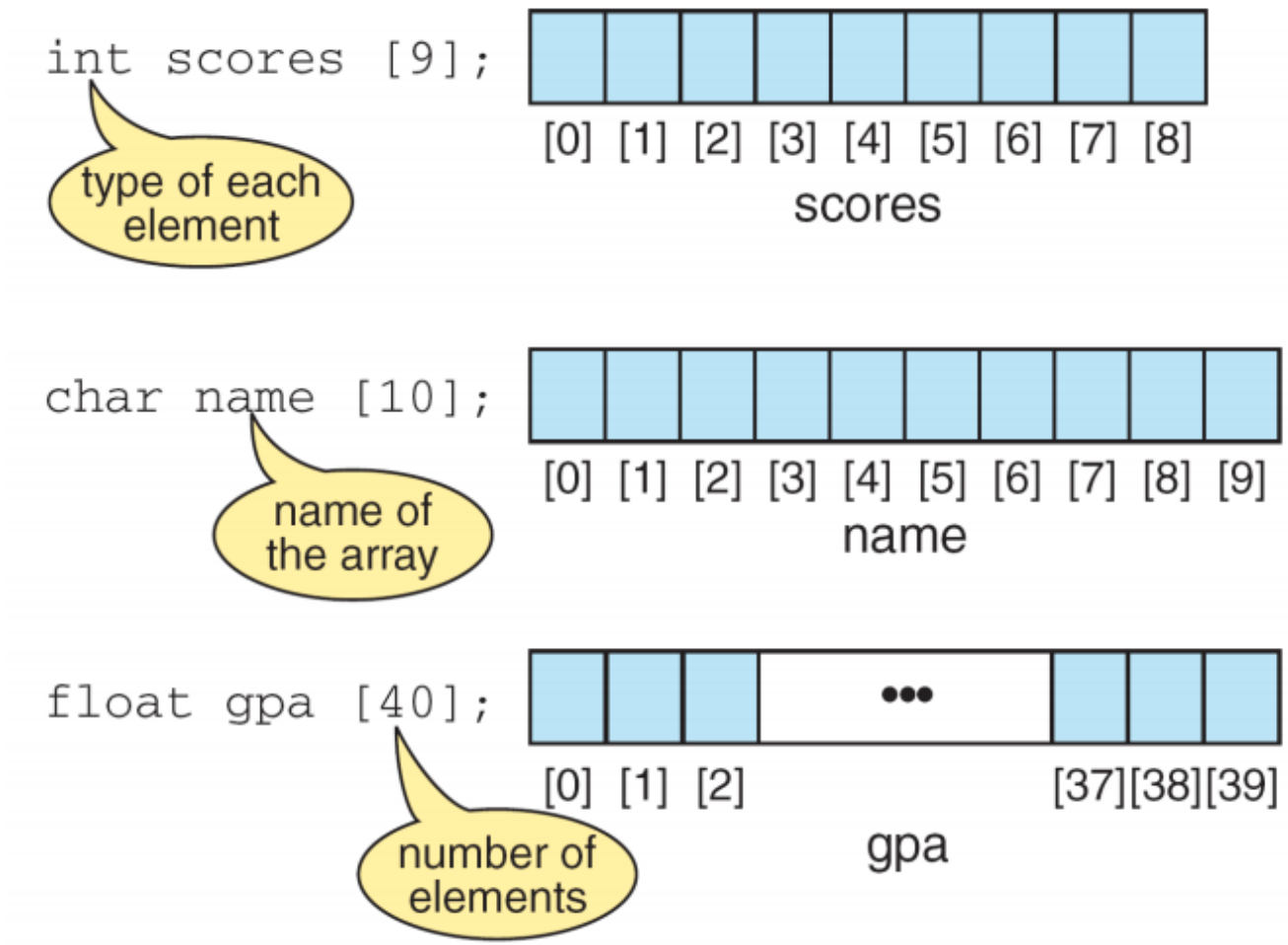
- 여러개의 데이터를 저장해야 할때!
  - 변수를 일일이 만들건가?

- 그렇게 일일이 만들었다고 쳐!
  - 그 여러개에 저장된거를 어떻게 가져올 것인가?

Array!



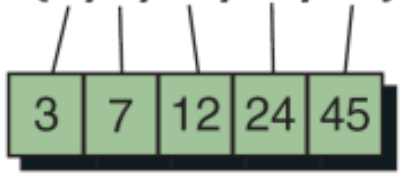
선언


int scores [9];
type of each element
[0] [1] [2] [3] [4] [5] [6] [7] [8]
scores
char name [10];
name of the array
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
name
float gpa [40];
number of elements
[0] [1] [2]
[37][38][39]
gpa

초기화

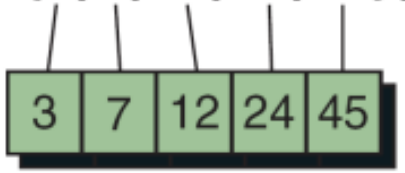
(a) Basic Initialization

```
int numbers[5] = {3,7,12,24,45};
```



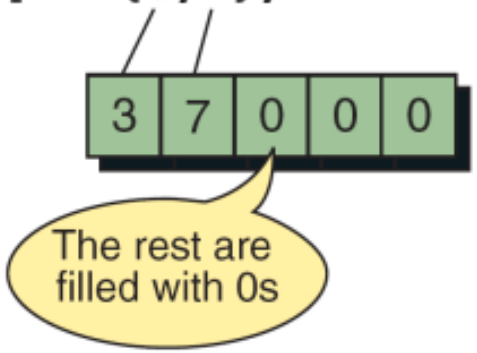
(b) Initialization without Size

```
int numbers[ ] = {3,7,12,24,45};
```



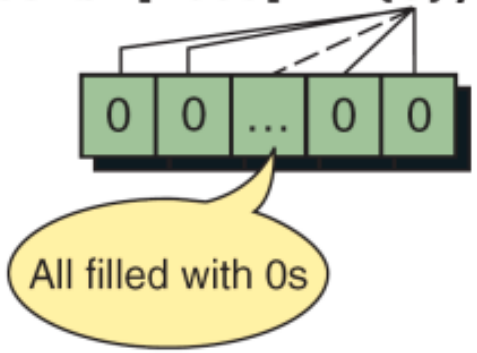
(c) Partial Initialization

```
int numbers[5] = {3,7};
```



(d) Initialization to All Zeros

```
int lotsOfNumbers [1000] = {0};
```



접근

name[index]
배열은 0부터 시작하는 것 주의

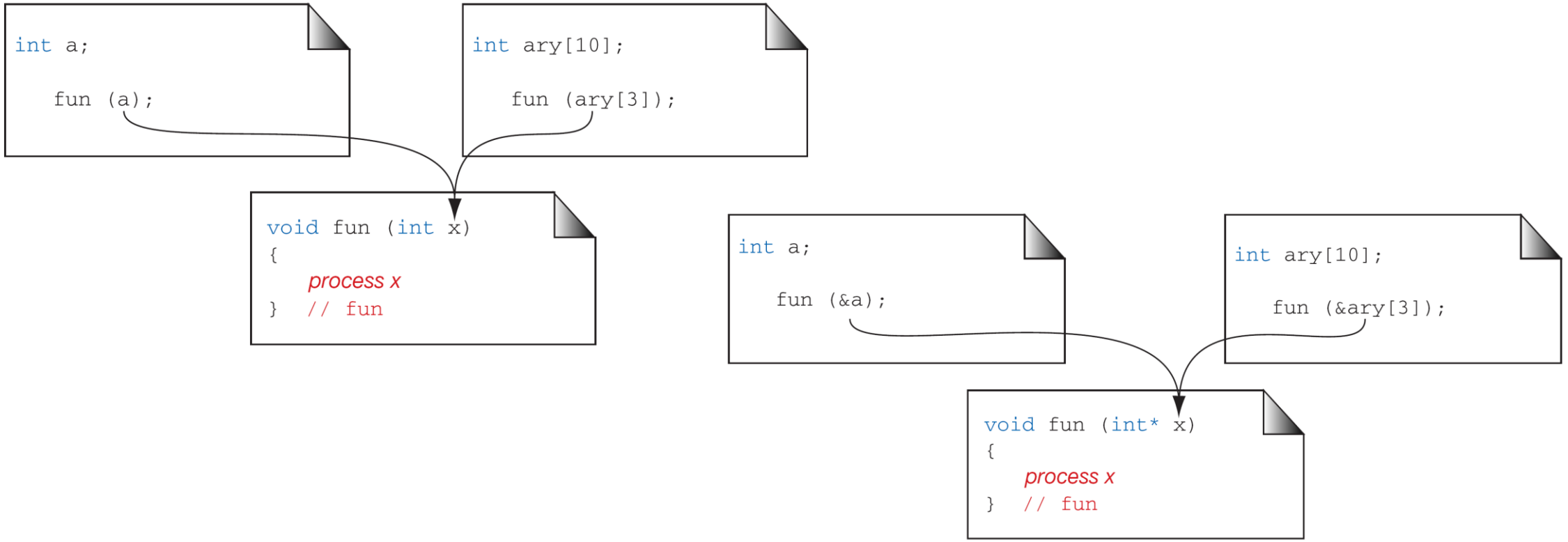
```
for (i = 0; i < 9; i++)
    scanf ("%d", &scores[i]);

scores[4] = 23;
printf("%f",gpa[11]);     //배열 gpa의 '12번째' 실수 출력
```

Element address = array address + (sizeof (element) * index)

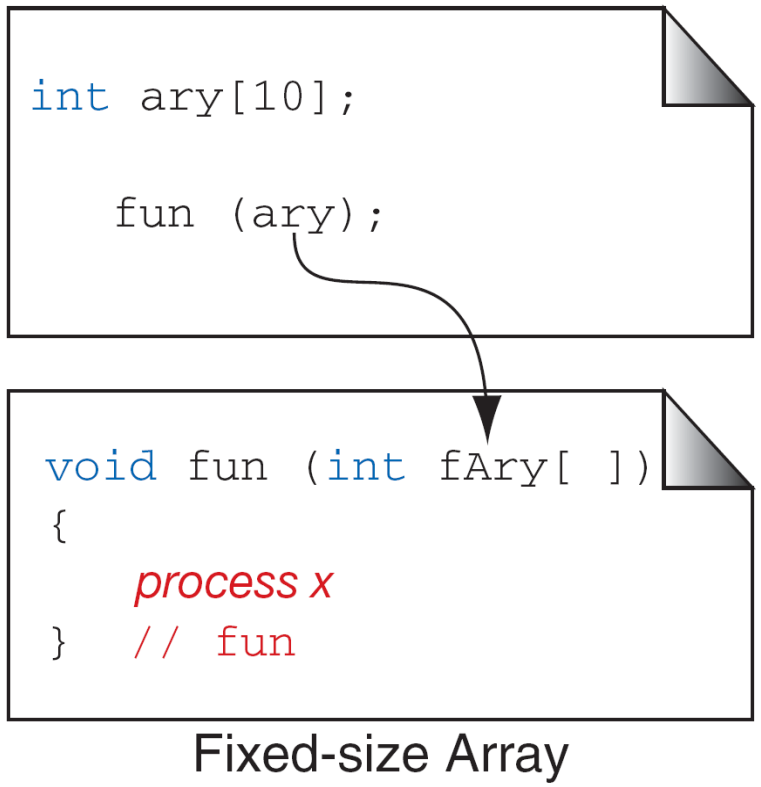
함수의 인자로 배열 넘기기

- 똑같이 하면 됩니다.



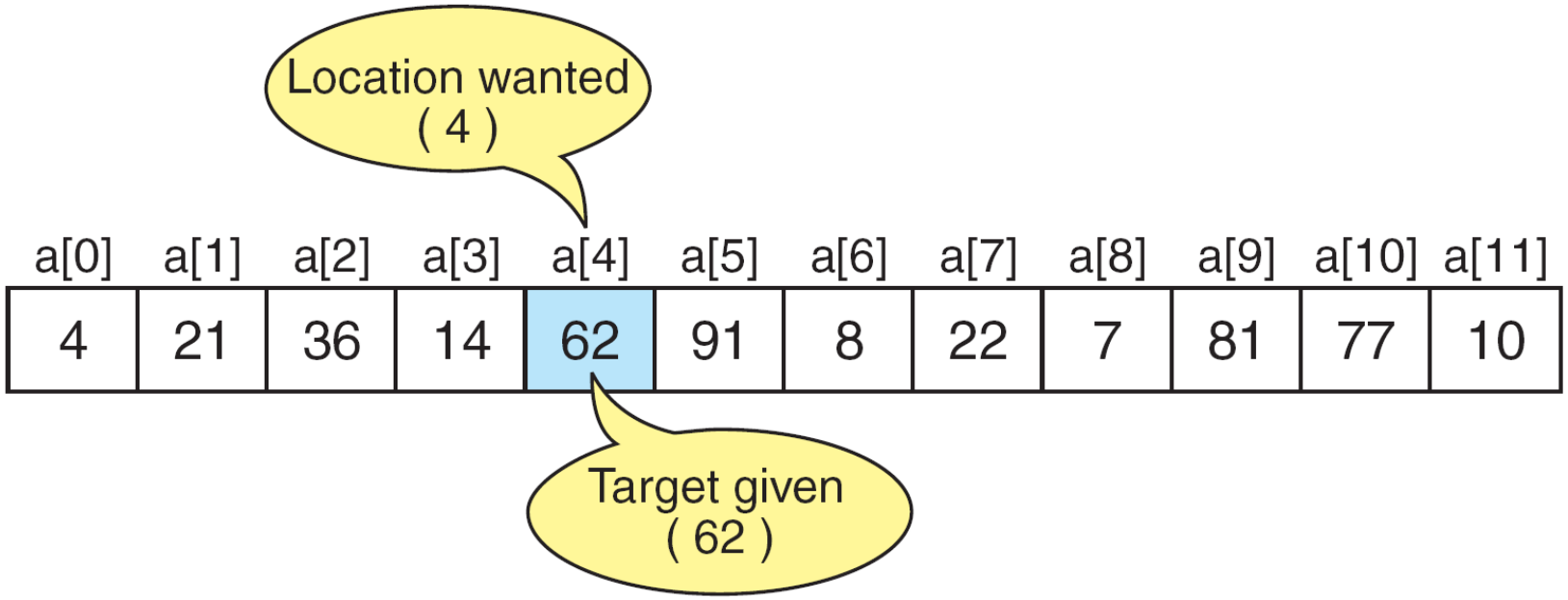
함수의 인자로 배열 넘기기 #2

- 배열 전체를 넘길때

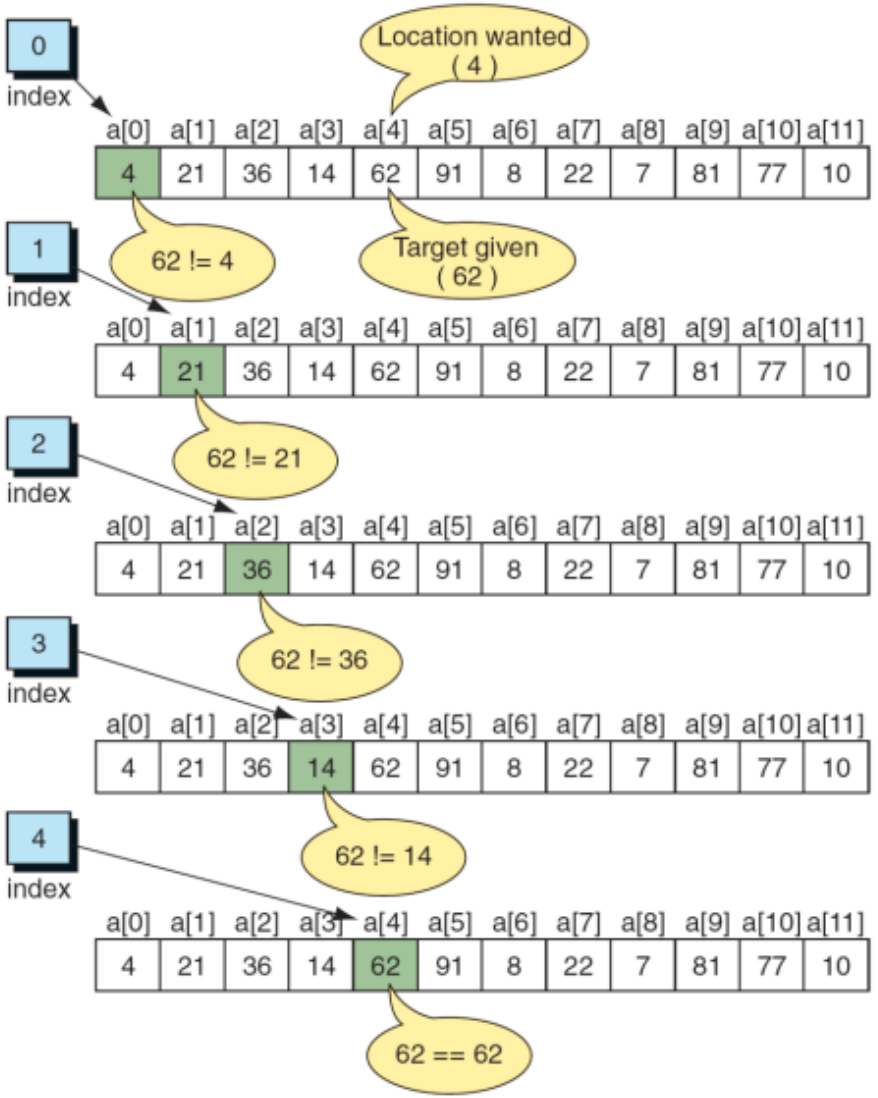


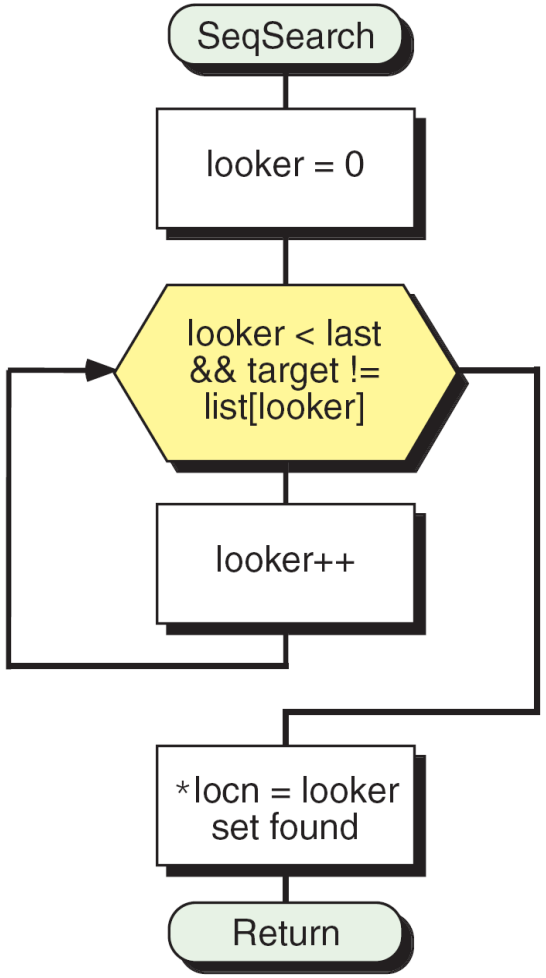
Search

Searching on Array



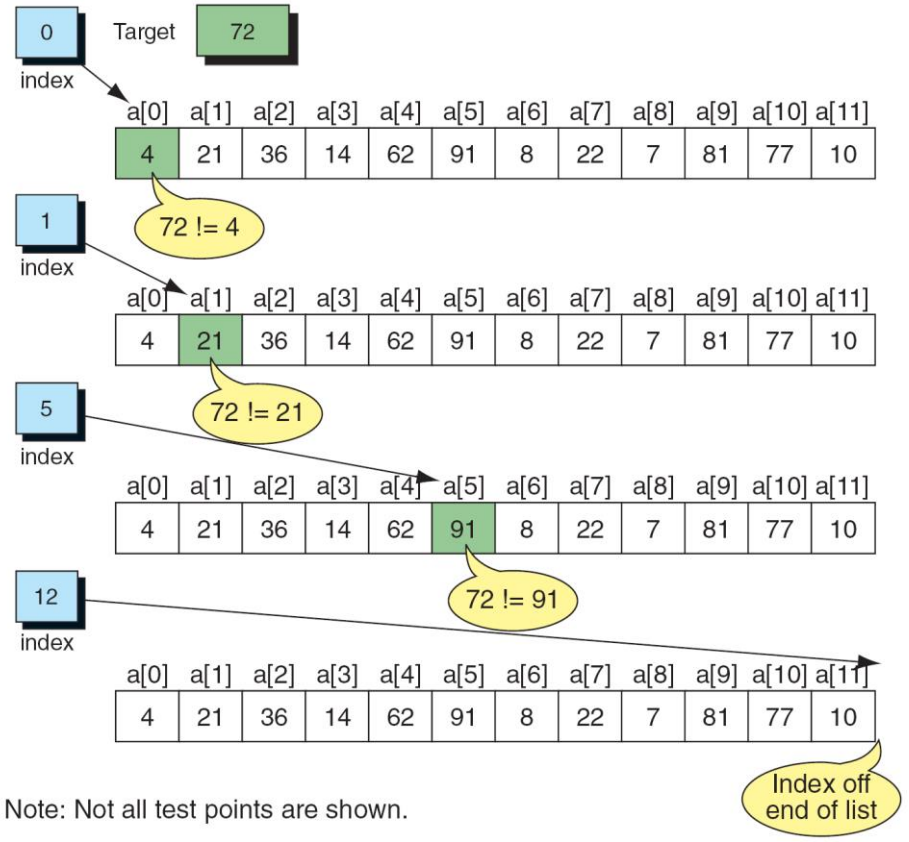
Sequential Search Design





Sequential Search 특징

- Unsorted 배열에 대해서는 처음부터 쭉 봐야하기 때문에 비효율적이다.
- Worst Case: O(n)



Code

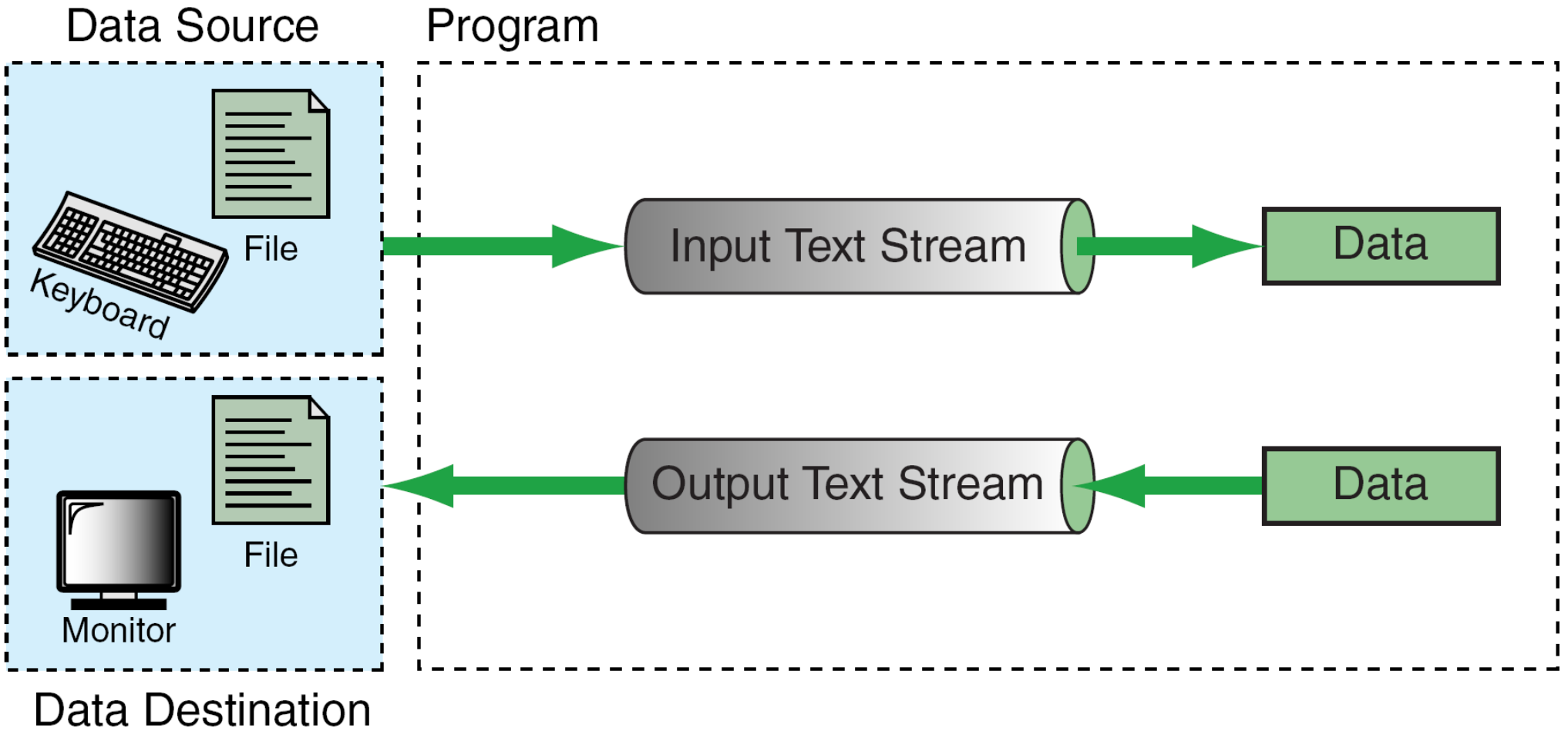
```
1  /* ==================== seqSearch ====================
2     Locate target in an unordered list of size elements.
3        Pre   list must contain at least one item
4              last is index to last element in list
5              target contains the data to be located
6              locn is address for located target index
7        Post  Found: matching index stored in locn
8                     return true (found)
9              Not Found: last stored in locn
10                           return false (not found)
11 */
12 bool seqSearch (int list[], int  last,
13                  int target, int* locn)
14 {
15 // Local Declarations
16    int  looker;
17    bool found;
18
19 // Statements
20    looker = 0;
21    while (looker < last && target != list[looker])
22       looker++;
23
24    *locn = looker;
25    found = (target == list[looker]);
26    return found;
27 }  // seqSearch
```

File I/O

뭘 배우나요

- Stream?
- Standard Input, Output functions
- 파일로 부터 입력 받기/파일에 출력하기
  - FILE*, fopen, fclose, fscanf, fprintf
- FLUSH?
- Character I/O
  - getChar, putChar, fgetc, fputc, ungetc
- Practice) 1. Copy Text 2. 임의 파일의 단어, 줄 갯수 세기

다시보는 Steam 다이어그램



STanDard I/O (stdio.h)

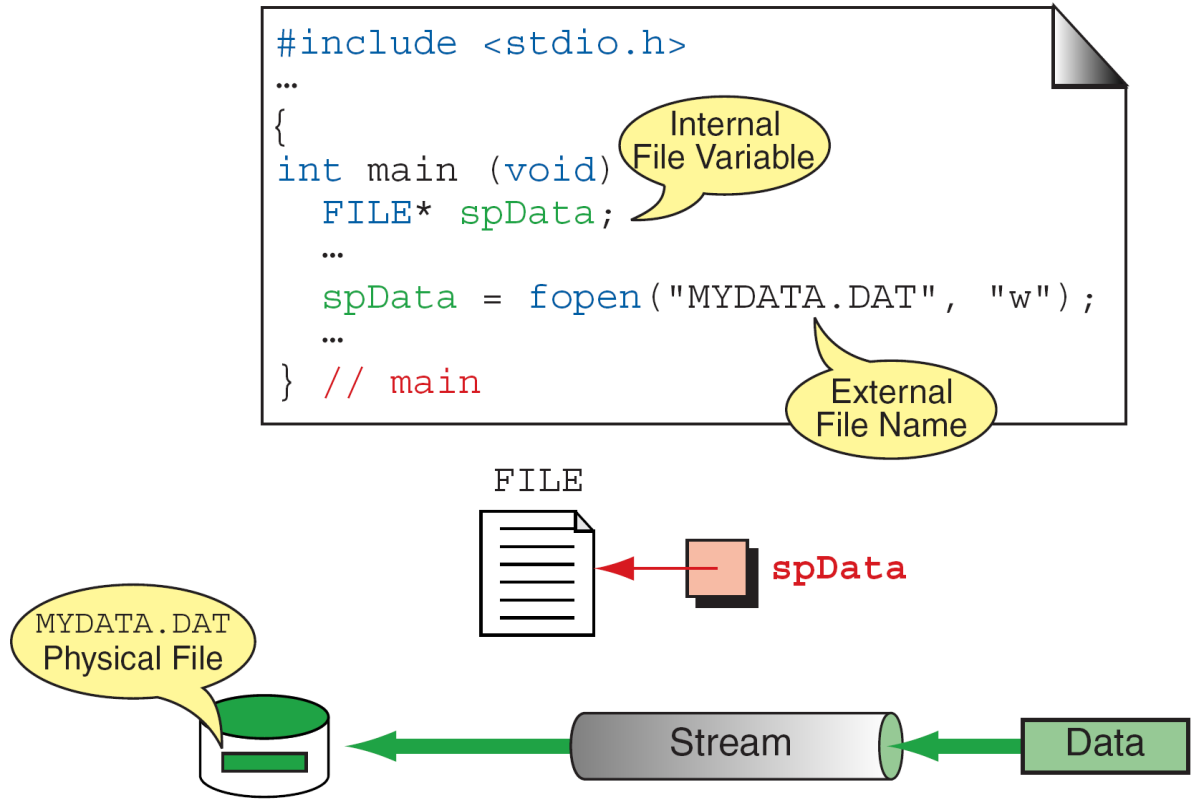
- Standard Input Stream : stdin
  - Scanf 함수가 사용하는 스트림
  - 콘솔(console) 화면에서 키보드를 통한 입력을 받음
- Standard Output Stream : stdout
  - Printf 함수가 사용하는 스트림
  - 콘솔 화면에 출력함

- 일종의 콘솔 화면과 소통하기 위한 빨대(Stream)!

FILE I/O

- 콘솔(console) 화면과의 소통이 아닌 파일(.txt .c etc) 과 소통하기 위한 Steam

```
FILE *infile;
infile = fopen(“anyfile.txt”, “w”);
~~
fclose(infile);
```

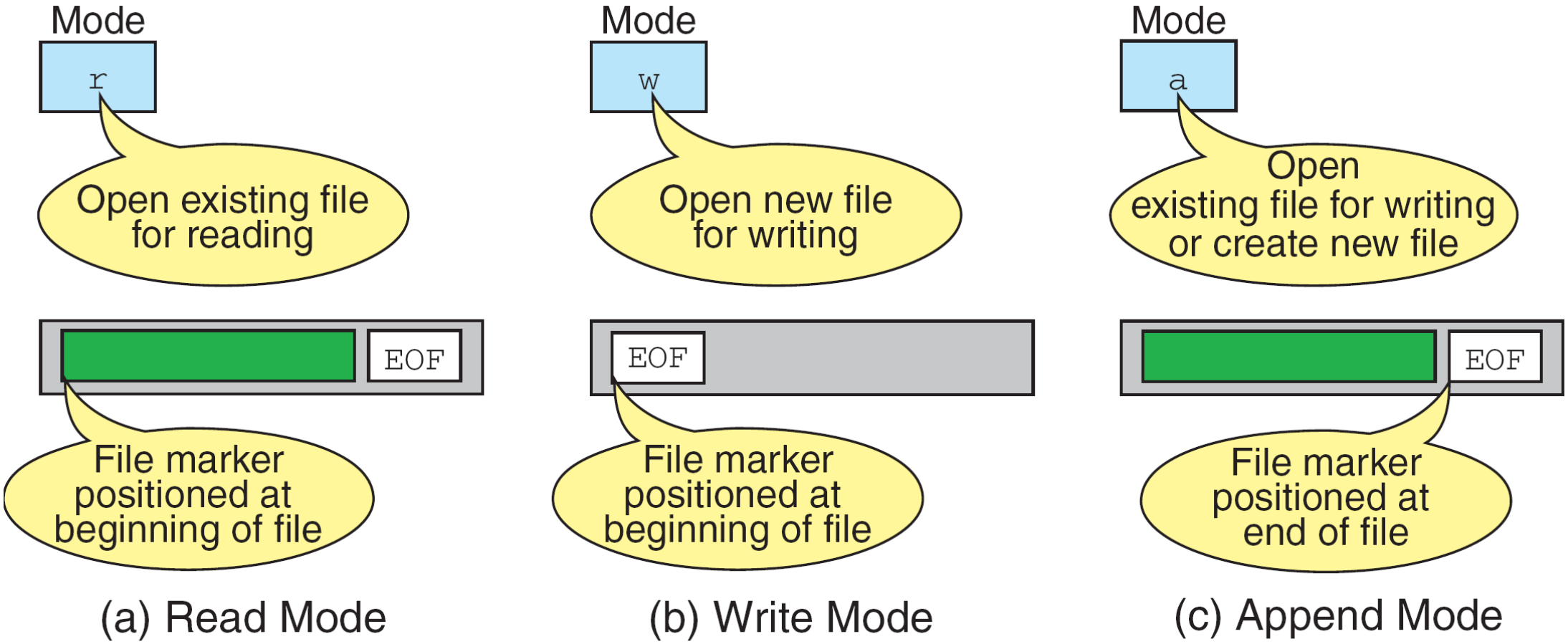


File 사용 options

- r : 읽기 모드 (텍스트 모드)
  - 없으면 널 리턴, 있으면 처음부터 읽기
- w : 쓰기 모드 (텍스트 모드)
  - 기존에 없으면 새로 만들기, 있으면 다 지우고 새로 쓰기
- a : 덧붙이기 모드 (텍스트 모드)
  - 기존에 없으면 새로 만들기, 있으면 제일 뒤에 더하기
- b : 바이너리 모드

```
FILE* infile = fopen(“indata.txt”, “r”);          //읽기로 열기
FILE* outfile = fopen(“outdata.txt”, “w”);        //쓰기로 열기
```

File-Opening Modes



FILE I/O 를 위해 필요한 것

```
//파일을 오픈하기 전에 반드시 파일 포인터 선언
FILE *fp;

//반드시 text.txt 파일이 폴더내에 있어야 한다. (코드있는 폴더)
fp = fopen(“test.txt”, “r”);

//파일이 제대로 열렸는지 반드시 체크
if (fp == NULL) { return -1; }

//사용이 끝나면 포인터를 닫아준다.
fclose(fp);
```

대표적 FILE I/O 관련 함수들

int fprintf(FILE * out, const char * format, …)

Filestream, out 으로 부터 format의 형태로 출력한다.

비교) int printf(const char * format, …)

int fscanf(FILE * in, const char * format, …)

File stream, in 으로 부터 format의 형태로 입력받는다.

비교) int scanf(const char * format, …)

그외 I/O 관련 함수들

int getchar(void)

int putchar(int out_char)

int fgetc(FILE* in)

int fputc (int oneChar, FILE* out)

int ungetc (int oneChar, FILE* Data)

Int fgets (char *str, int num, FILE * in)

Int fputs(const char *str, FILE * out)

fprintf

```
FILE * out;
out = fopen(“test.txt”, “w”);
fprintf(out, “%c %d\n”, ‘a’, 22);
…
fclose(out);
```

test.txt

fscanf

```
int a;
char b;
FILE * in;
in = fopen(“text.txt”, “r”);
fscanf(in, “%c %d”, &b, &a);
printf(“%c %d\n”, b, a);
fclose(in);
```

FLUSH

- 스트림 버퍼를 비우는 역할을 한다.
- Scanf 등 입력받는 함수 사용 시 주의!! 할 점!!
  - 근접한 scanf 두번 사이에는… 뭔가… 언짢은 일들이…

Test)

```
int a;  char b;
scanf(“%d”, &a);
scanf(“%c”,&b);
printf(“%d %c\n”, a, b);

fflush();
#define FLUSH while(getchar != ‘\n’)
```

예제

- input.txt에 있는 3쌍의 점수를 입력 받아 output.txt에 합계와 평균을 출력
- 점수의 개수는 정해져 있지 않음

- Input.txt

```
94 55 78
66 54 70
100 96 95
77 63 88
```

- Output.txt

```
234 77.534
210 65.431
294 97.236
223 74.549
```

```
#include <stdio.h>

int main(void) {
	FILE *inFile;
	FILE *outFile;
	int a, b, c;
	int sum;
	float avg;

	inFile = fopen("input.txt", "r");
	outFile = fopen("output.txt", "w");

	if(inFile == NULL) {
		printf(“Input file open error\n");
		return 0;
	}
	while(fscanf(inFile, "%d %d %d", &a, &b, &c) != EOF) {
		sum = a + b + c;
		avg = (float)(sum)/3.0;
		fprintf(outFile, "%d %.3f\n", sum, avg);
	}

	return 0;
}
```

다음시간

- 시험 잘 보세요!
- 질문 있으면 카톡으로 or 청암에서

- 중간고사 끝나고: Array 이어서 & Pointer