C Programming - Day 5

2017.09.11.

JunGu Kang
Dept. of Cyber Security

아주대학교 Whois

표준 입출력

printf / scanf

Formatting된 문자열을 입출력한다.

아주대학교 Whois

printf

```
main() {
    printf(“hello, world!”);
}
```


아주대학교 Whois

printf

```
main() {
    int a = 10;
    printf(a);  // 이렇게 쓸 수도 있음
                // 그러나 FSB 취약점이 존재하기 때문에 사용해서는 안 됨
}
```


아주대학교 Whois

printf

```
main() {
    int a = 10;
    float b = 7.32;
    printf(“a는 %d, b는 %f”, a, b);
}
```


아주대학교 Whols

Format String

Format string	Meaning
%d, %i	10진수 정수(Decimal)
%u	부호 없는 10진수 정수(Unsigned Decimal)
%o	부호 없는 8진수 정수(Unsigned Octal)
%x, %X	부호 없는 16진수 정수(Unsigned Hexadecimal)
%c	문자 하나(Single Character)
%s	문자열(String)
%f, %F	10진수 실수(Double)
%e, %E	부동소수점 표현으로 나타낸 10진수 실수(Double)
%g, %G	%f 또는 %g중에서 알아서 결정(Double)
%a, %A	16진수로 실수(Double)
%p	포인터(Pointer)
%n	지금까지 출력된 글자 수를 메모리 공간에 저장한다.

아주대학교 WhoIs

Escape Sequence

Character	Escape Sequence
비프음(Alert)	\a
백스페이스(Backspace)	\b
폼피드(Formfeed)	\f
줄 바꿈(New Line)	\r
줄 바꿈(Carriage Return)	\n
수평 탭(Horizontal Tab)	\t
수직 탭(Vertical Tab)	\v
백슬래쉬(Backslash)	\\
물음표(Question Mark)	\?
작은따옴표(Single Quote)	\’
큰따옴표(Double Quote)	\”
8진수(Octal Number)	\ooo
16진수(Hexadecimal Number)	\xhh

아주대학교 AJOU UNIVERSITY 1973
WhoIs

scanf

```
main() {
    int a;
    float b;
    scanf(“%d”, &a);  // 10진수 정수로 입력 받기
    scanf(“%f”, &b);  // 10진수 실수로 입력 받기
}
```


아주대학교 Whols

왜 변수 앞에 &가 붙지?

~~포인터를 배우면 알게 된다.~~

포인터를 배웠으니 알 수 있다.

printf

변수에 저장된 내용만 출력하면 된다. → Call by Value

아주대학교 Whois

scanf

변수에 값을 직접 저장해줘야 한다. → Call by Reference

아주대학교 WhoIs

스트림

프로그램으로 입출력을 할 수 있도록 연결해주는 것

putchar / getchar

하나의 문자를 표준 스트림으로 입출력한다.

아주대학교 Whois

putchar / getchar

```c
#include <stdio.h>

int putchar(int);
// 성공하면 출력한 문자, 실패하면 EOF

int getchar(void);
// 성공하면 입력받은 문자, 파일 끝에 도달하거나 실패하면 EOF
```


아주대학교 Whols

putchar / getchar

```
#include <stdio.h>

main() {
     int ch;
     ch = getchar();
     putchar(ch);
}
```


아주대학교 WhoIs

fputc / fgetc

하나의 문자를 지정한 스트림으로 입출력한다.

아주대학교 WhoIs

fputc / fgetc

```
#include <stdio.h>

int fputc(int, FILE *);
// 성공하면 출력한 문자, 실패하면 EOF

int fgetc(FILE *);
// 성공하면 입력받은 문자, 파일 끝에 도달하거나 실패하면 EOF
```

fputc / fgetc

```
#include <stdio.h>

main() {
    int ch;
    ch = fgetc(stdin);
    fputc(ch, stdout);
}
```


아주대학교 WhoIs

puts / gets

하나의 문자열을 표준 스트림으로 입출력한다.

아주대학교 WhoIs

puts / gets

```
#include <stdio.h>

int puts(const char *);
// 성공하면 음수가 아닌 값, 실패하면 EOF

char * gets(char *);
// 파일의 끝에 도달하거나 실패하면 NULL
```

puts / gets

```
#include <stdio.h>

main() {
    char str[1024];
    gets(str);
    puts(str);
}
```


아주대학교 WhoIs

fputs / fgets

하나의 문자열을 일정 길이만큼 입출력한다.

아주대학교 Whois

fputs / fgets

```
#include <stdio.h>

int fputs(const char *, FILE *);
// 성공하면 음수가 아닌 값, 실패하면 EOF

char * gets(char *, int, FILE *);
// 파일의 끝에 도달하거나 실패하면 NULL
```

fputs / fgets

```
#include <stdio.h>

main() {
    char str[1024];
    fgets(str, sizeof str, stdin);
    puts(str, stdout);
}
```


아주대학교 WhoIs

fgets

\n을 만날 때까지 입력받는다. 단, \n도 입력받는다.

아주대학교 Whois

EOF

```
// stdio.h

#define EOF (-1)
```


아주대학교 Whois

EOF

EOF를 반환한다 == -1을 반환한다

아주대학교 WhoIs

Buffer

데이터는 **버퍼를 거쳐서** 입출력된다.(Buffering)

아주대학교 Whois

Buffer

입출력은 매우 느린 작업이기 때문에
실시간으로 처리하면 비효율적이다.

fflush

지정한 스트림의 버퍼를 비운다.

아주대학교 Whois

fflush

```
#include <stdio.h>

int fflush(FILE *)
// 성공하면 0, 실패하면 EOF
```


아주대학교 Whois

파일 입출력

파일 스트림

파일과 프로그램을 연결하는 스트림.

아주대학교 Whois

fopen / fclose

파일을 열고 닫는다.

아주대학교 WhoIs

fopen / fclose

파일 스트림을 열고 닫는다.

아주대학교 WhoIs

fopen / fclose

```
#include <stdio.h>

FILE * fopen(const char *, const char *);
// 성공하면 해당 파일의 FILE 구조체 포인터, 실패하면 NULL

FILE * fclose(FILE *);
// 성공하면 0, 실패하면 EOF
```

fopen / fclose

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “rt”);  // 읽기모드
    if(fp == NULL) return -1;
    fclose(fp);
}
```

fopen

파일은 한 가지 모드로만 열 수 있다.

아주대학교 Whois

fopen

Mode	Meaning
rt	텍스트로 읽기(파일이 없으면 오류)
wt	텍스트로 쓰기
at	텍스트로 이어서 쓰기
rt+	텍스트로 읽고 쓰기(파일이 없으면 오류)
wt+	텍스트로 읽고 쓰기
at+	텍스트로 읽고 이어서 쓰기
rb	바이너리로 읽기(파일이 없으면 오류)
wb	바이너리로 쓰기
ab	바이너리로 이어서 쓰기
rb+	바이너리로 읽고 쓰기(파일이 없으면 오류)
wb+	바이너리로 읽고 쓰기
ab+	바이너리로 읽고 이어서 쓰기

아주대학교 Whols

fopen

가급적 r, w, a를 쓰자.

아주대학교 WhoIs

fclose

반드시 fclose를 사용해 파일 스트림을 닫아주어야 한다.

아주대학교 Whois

개행

개행이 항상 \n으로 처리되지는 않는다.

아주대학교 Whois

개행

OS	New Line
Microsoft Windows	\r\n
MacOS	\r
Unix / Linux	\n

아주대학교 WhoIs

fputc / fgetc

하나의 문자를 지정한 스트림으로 입출력한다.

아주대학교 WhoIs

fputc / fgetc

```
#include <stdio.h>

int fputc(int, FILE *);
// 성공하면 출력한 문자, 실패하면 EOF

int fgetc(FILE *);
// 성공하면 입력받은 문자, 파일 끝에 도달하거나 실패하면 EOF
```


아주대학교 Whols

fputc

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “wt”);
    if(fp == NULL) return -1;

    fputc(‘A’, fp);

    fclose(fp);
}
```


아주대학교 Whois

fgetc

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “rt”);
    int ch;
    if(fp == NULL) return -1;

    ch = fgetc(fp);
    printf(“%c\n”, ch);

    fclose(fp);
}
```


아주대학교 Whois

fputs / fgets

하나의 문자열을 지정한 스트림으로 입출력한다.

아주대학교 Whois

fputs / fgets

```
#include <stdio.h>

int fputc(int, FILE *);
// 성공하면 출력한 문자, 실패하면 EOF

int fgetc(FILE *);
// 성공하면 입력받은 문자, 파일 끝에 도달하거나 실패하면 EOF
```


아주대학교 Whols

fputs

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “wt”);
    if(fp == NULL) return -1;

    fputs(“Hello, World!\n”, fp);

    fclose(fp);
}
```

fgets

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “rt”);
    char str[1024];
    if(fp == NULL) return -1;

    fgets(str, sizeof str, fp);
    printf(“%s”, str);

    fclose(fp);
}
```


아주대학교 Whois

feof

EOF는 파일의 끝에서도 발생하고, 오류에서도 발생한다.

아주대학교 Whois

feof

EOF가 발생했을때 파일의 끝에 도달한 것인지,
오류가 발생한 것인지 구별할 필요가 있다.

아주대학교 Whois

feof

```
#include <stdio.h>

int feof(FILE *);
// 파일의 끝에 도달했다면 0이 아닌 값
// 파일의 끝이 아니라면(오류) 0
```

feof

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “rt”);
    char str[1024];
    if(fp == NULL) return -1;

    while(fgets(str, sizeof str, fp) != NULL) printf(“%s, str);

    if(feof(fp) != 0) printf(“success”);
    else printf(“failed”);

    fclose(fp);
}
```


아주대학교 WhoIs

fread / fwrite

지정한 크기만큼 바이너리로 입출력한다.

아주대학교 Whois

fread / fwrite

```
#include <stdio.h>

size_t fread(void *, size_t, size_t, FILE *);
// 성공하면 갯수, 실패하면 갯수보다 작은 값

size_t fwrite(const void *, size_t, size_t, FILE *);
// 성공하면 갯수, 실패하면 갯수보다 작은 값
```

fread

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “rb”);
    int data[10];
    if(fp == NULL) return -1;

    fread(data, sizeof (int), 10, fp);

    fclose(fp);
}
```


아주대학교 WhoIs

fwrite

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “wb”);
    int data[10] = { /* some data */ };
    if(fp == NULL) return -1;

    fwrite(data, sizeof (int), 10, fp);

    fclose(fp);
}
```


아주대학교 Whols

fprintf / fscanf

Formatting해서 입출력.

아주대학교 Whois

fprintf

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “rb”);
    int num1 = 123, num2 = 234, num3 = 345;
    if(fp == NULL) return -1;

    fprintf(fp, “%d %d %d”, num1, num2, num3);  // 123 234 345

    fclose(fp);
}
```


아주대학교 WhoIs

fscanf

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “rb”);
    int num1 = 123, num2 = 234, num3 = 345;
    if(fp == NULL) return -1;

    fscanf(fp, “%d %d %d”, &num1, &num2, &num3);  // 123 234 345

    fclose(fp);
}
```


아주대학교 WhoIs

fseek

파일을 중간부터 읽고싶다면?

fseek

```
#include <stdio.h>

int fseek(FILE *, long, int);
// 성공하면 0, 실패하면 0이 아닌 값
```


아주대학교 WhoIs

fprintf

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “rb”);
    int num1 = 123, num2 = 234, num3 = 345;
    if(fp == NULL) return -1;

    fseek(fp, 100, SEEK_SET);  // 처음 위치부터 100바이트 뒤로
    fseek(fp, 200, SEEK_CUR);  // 현재 위치부터 200바이트 뒤로
    fseek(fp, -100, SEEK_END);  // EOF에서부터 100바이트 앞으로

    fclose(fp);
}
```


아주대학교 WhoIs

fseek

현재 커서의 위치를 알려준다.

아주대학교 WhoIs

fseek

```
#include <stdio.h>

long ftell(FILE *)
```


아주대학교 WhoIs

fprintf

```
#include <stdio.h>

main() {
    FILE * fp = fopen(“file_name.txt”, “rb”);
    int num1 = 123, num2 = 234, num3 = 345;
    if(fp == NULL) return -1;

    fseek(fp, 100, SEEK_SET);  // 처음 위치부터 100바이트 뒤로
    ftell(fp);  // 100

    fclose(fp);
}
```


아주대학교 Whols

문자열 다루기

strlen

NUL을 제외한 문자열의 길이를 구한다.

아주대학교 Whois

strlen

```
#include <string.h>

size_t strlen(const char *);
// 문자열의 길이
```

strlen

```
#include <string.h>

main() {
     char * ptr = “Hello, World!”;
     printf(“%d\n”, strlen(ptr));
}
```


아주대학교 WhoIs

strcpy

문자열을 복사한다.

아주대학교 WhoIs

strcpy

```
#include <string.h>

char * strcpy(char *, const char *);
// 복사된 문자열의 주소
```


아주대학교 WhoIs

strcpy

```
#include <string.h>

main() {
    char src[1024] = “Hello, World!”;
    char dst[1024];
    strcpy(dst, src);
}
```

strncpy

문자열을 복사한다.

아주대학교 Whois

strncpy

```
#include <string.h>

char * strncpy(char *, const char *, size_t);
// 복사된 문자열의 주소
```

strncpy

```
#include <string.h>

main() {
    char src[1024] = “Hello, World!”;
    char dst[8];
    strcpy(dst, src, sizeof dst - 1);  // 7바이트만 복사
    dst[sizeof dst - 1] = 0;  // 마지막 바이트는 NULL로 고정
}
```

strcat

두 문자열을 연결한다.

아주대학교 WhoIs

strcat

```
#include <string.h>

char * strcat(char *, const char *);
// 덧붙여진 문자열의 주소
```


아주대학교 WhoIs

strcat

```
#include <string.h>

main() {
    char src[512] = “World!”;
    char dst[1024] = “Hello ”,
    strcat(dst, src);
}
```


아주대학교 WhoIs

strncat

지정한 길이만큼 가져와 두 문자열을 연결한다.

아주대학교 Whois

strncat

```
#include <string.h>

char * strncat(char *, const char *, size_t);
// 덧붙여진 문자열의 주소
```


아주대학교 WhoIs

strncat

```
#include <string.h>

main() {
    char src[512] = “World!”;
    char dst[1024] = “Hello ”,
    strncat(dst, src, 5);  // 5바이트만 복사
}
```


아주대학교 WhoIs

strcmp

두 문자열을 비교한다.

아주대학교 WhoIs

strcmp

```
#include <string.h>

int * strcmp(const char *, const char *);
// 두 문자열이 같으면 0, 아니면 0이 아닌 값
// 첫 문자열이 더 크면(뒤에 있으면) 0보다 큰 값
// 나중 문자열이 더 크면(뒤에 있으면) 0보다 작은 값
```


아주대학교 Whols

strcmp

```
#include <string.h>

main() {
    char src[1024] = “Hello, World!”;
    char dst[1024] = “Hello, World!”,
    int result = strcmp(dst, src);
}
```


아주대학교 WhoIs

strncat

두 문자열을 지정한 길이만큼 비교한다.

아주대학교 Whois

strncat

```
#include <string.h>

int strncat(const char *, const char *, size_t);
// 두 문자열이 같으면 0, 아니면 0이 아닌 값
// 첫 문자열이 더 크면(뒤에 있으면) 0보다 큰 값
// 나중 문자열이 더 크면(뒤에 있으면) 0보다 작은 값
```

strncat

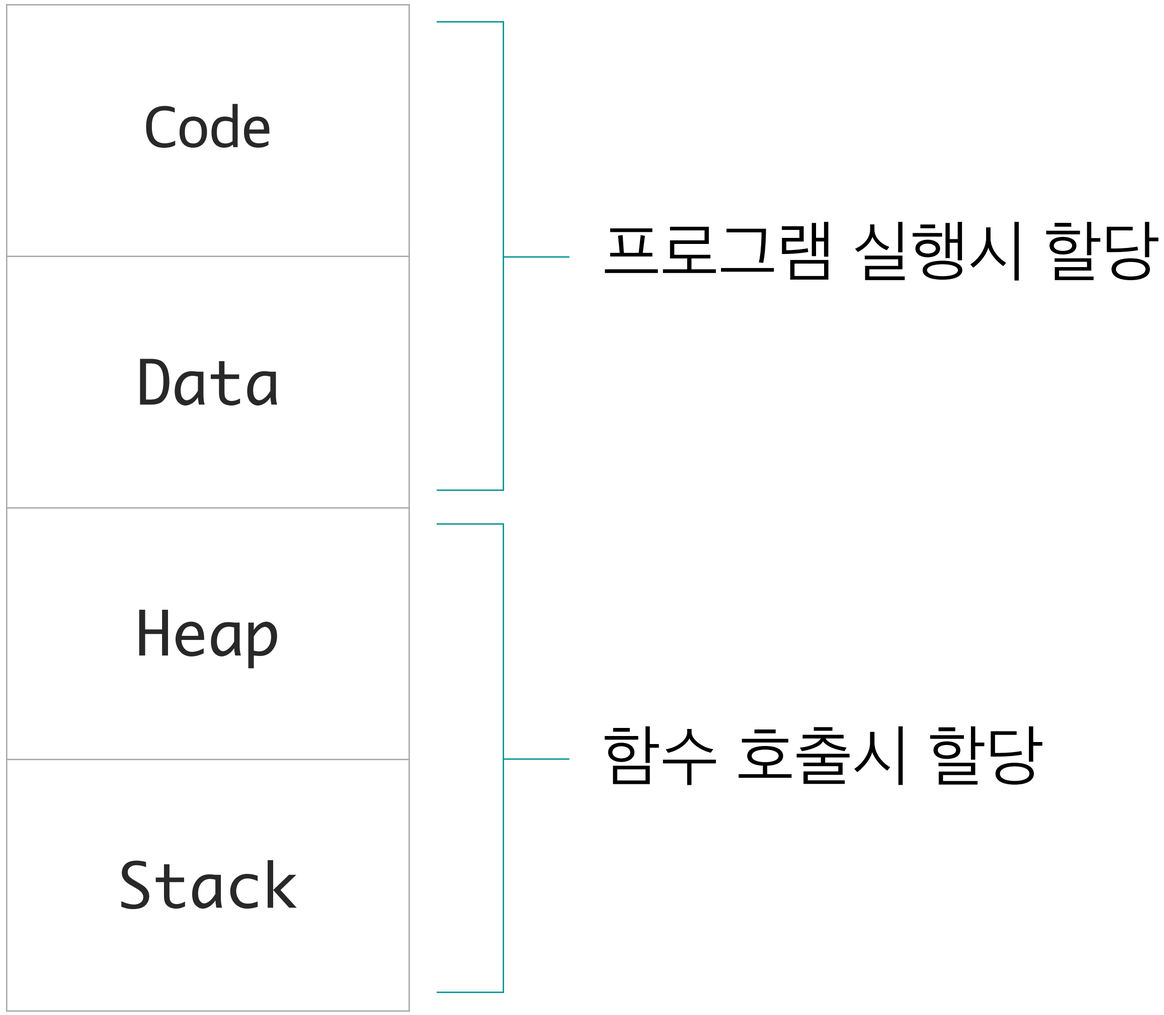
```
#include <string.h>

main() {
    char src[1024] = “Hello, World!”;
    char dst[1024] = “Hello, C!”,
    int result = strncmp(dst, src, 7);  // 7바이트만 비교
}
```


아주대학교 WhoIs

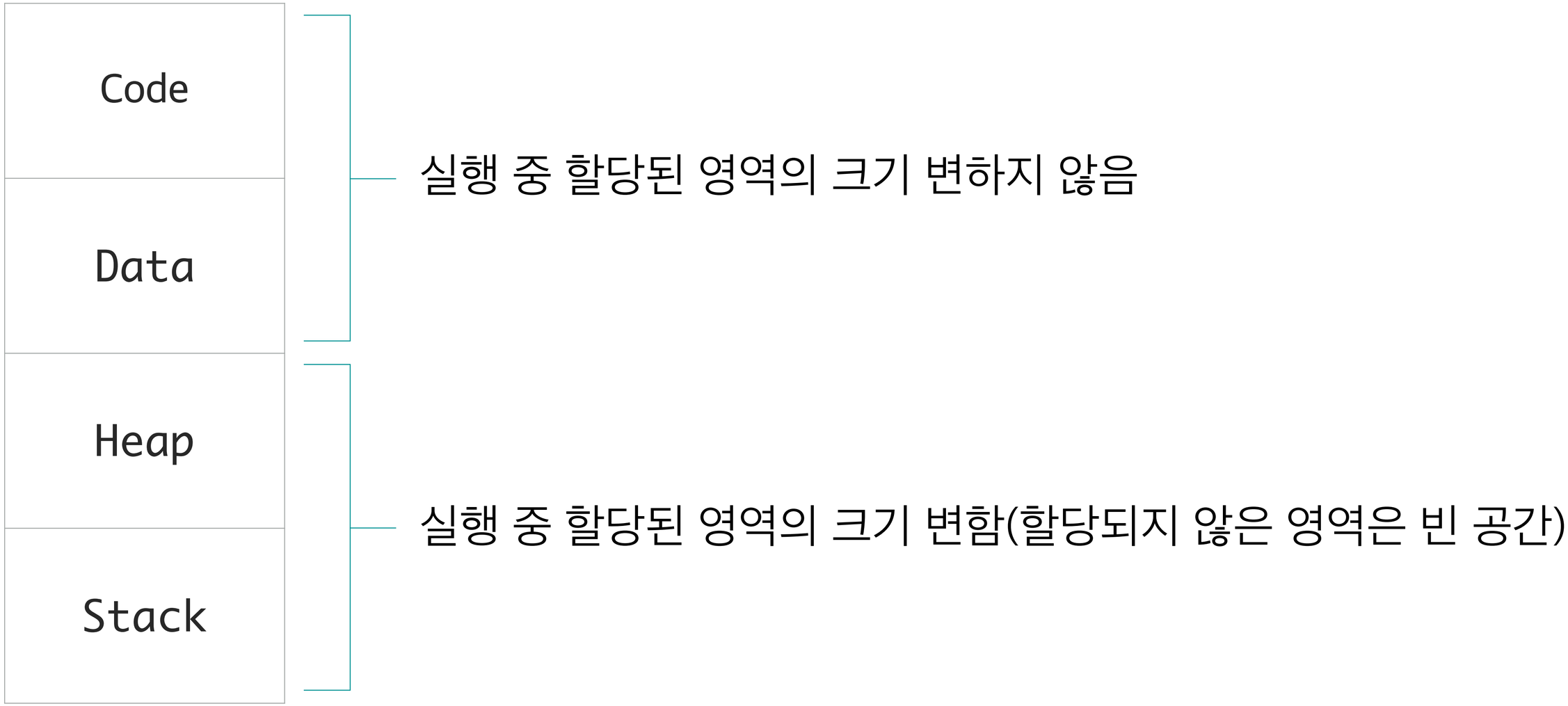
동적 메모리 할당

메모리 구조



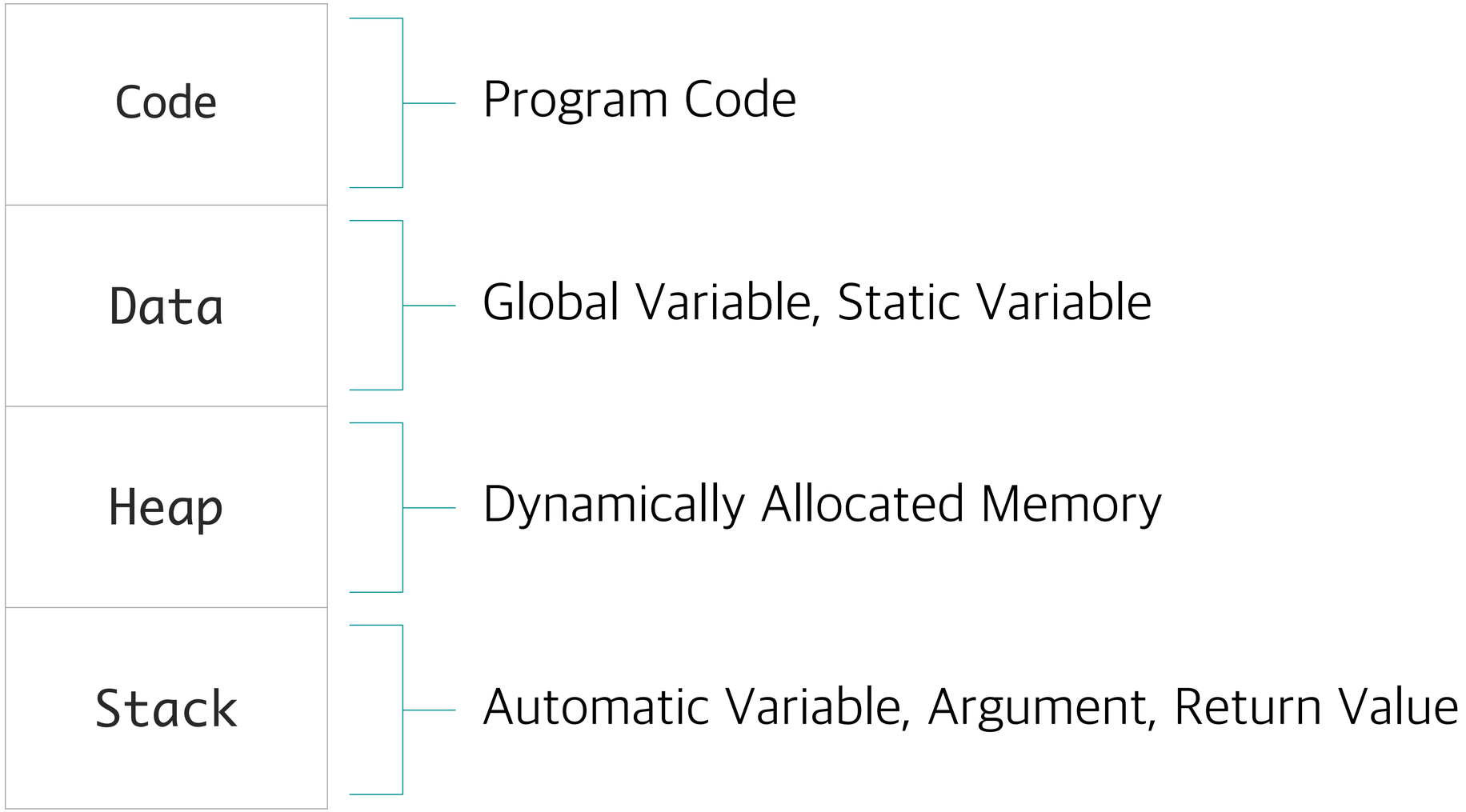
아주대학교 Whois

메모리 구조



아주대학교 WhoIs

메모리 구조



아주대학교 Whois

malloc

heap 영역에 지정된 크기의 메모리를 할당한다.

아주대학교 WhoIs

malloc

```
#include <stdlib.h>

void * malloc(size_t);
// 성공하면 할당된 메모리 주소값, 실패하면 NULL
```


아주대학교 WhoIs

free

할당된 메모리 영역을 해제한다.

아주대학교 Whois

free

```
#include <stdlib.h>

void free(void * ptr);
```


아주대학교 WhoIs

malloc / free

```
#include <stdlib.h>

main() {
    void * ptr = malloc(4);
    free(ptr);
}
```


아주대학교 WhoIs

malloc

Void 포인터는 어떠한 연산도 할 수 없다.
단지 주소값을 저장할 수만 있다.

아주대학교 Whois

malloc

`malloc()`은 void 포인터를 반환한다.

아주대학교 Whois

malloc / free

```
#include <stdio.h>
#include <stdlib.h>

main() {
    int * ptr = (int *) malloc(4);  // int 포인터로 casting
    *ptr = 10;
    printf("%d\n", *ptr);
    free(ptr);
}
```


아주대학교 WhoIs

calloc

heap 영역에 지정된 크기의 메모리를 할당한다.
할당된 모든 메모리를 0으로 초기화한다.

calloc

```
#include <stdlib.h>

void * calloc(size_t, size_t);
// 성공하면 할당된 메모리 주소값, 실패하면 NULL
```

calloc / free

```
#include <stdio.h>
#include <stdlib.h>

main() {
    int * ptr = (int *) calloc(4, 1);  // 4바이트짜리 블록 1개
    *ptr = 10;
    printf("%d\n", *ptr);
    free(ptr);
}
```

realloc

할당된 메모리 영역의 크기를 변경해 재할당한다.

아주대학교 Whois

realloc

```
#include <stdlib.h>

void * realloc(void *, size_t);
// 성공하면 새로 할당된 메모리 주소값, 실패하면 NULL
```

realloc

```
#include <stdio.h>
#include <stdlib.h>

main() {
    int * ptr = (int *) malloc(4);
    int * ptr = (int *) realloc(ptr, 8);  // 8바이트로 재할당
    *ptr = 10;
    *(ptr + 1) = 20;
    printf("%d, %d\n", *ptr, *(ptr + 1));
    free(ptr);
}
```


아주대학교 Whols

memset

지정한 메모리 영역을 지정한 값으로 초기화.

아주대학교 Whois

memset

```
#include <string.h>  // 메모리 함수인데 string.h이다.

void * memset(void *, int, size_t);
// 성공하면 해당 메모리 주소, 실패하면 NULL
```


아주대학교 Whois

memset

```
#include <stdio.h>
#include <stdlib.h>

main() {
    int * ptr = (int *) malloc(4);
    memset(ptr, 0, 4);  // ptr부터 4바이트를 0으로 초기화
    printf("%d\n", *ptr);
    free(ptr);
}
```


아주대학교 Whols

C Programming - Day 5

2017.09.11.

JunGu Kang
Dept. of Cyber Security

아주대학교 Whois