

6장. 함수



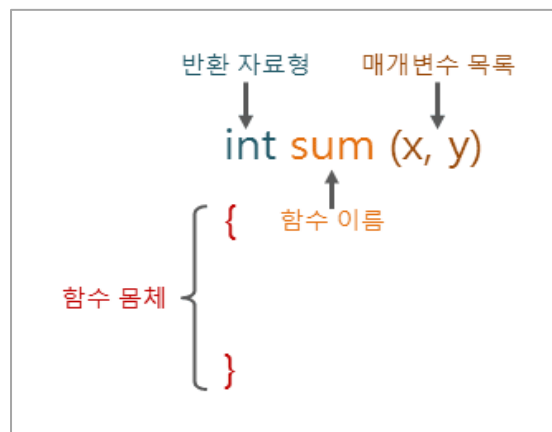
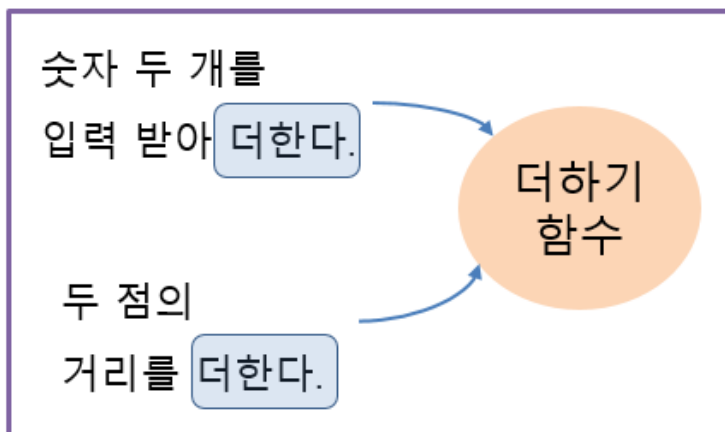
Visualstudio 2019



함수(function)

❖ 함수란?

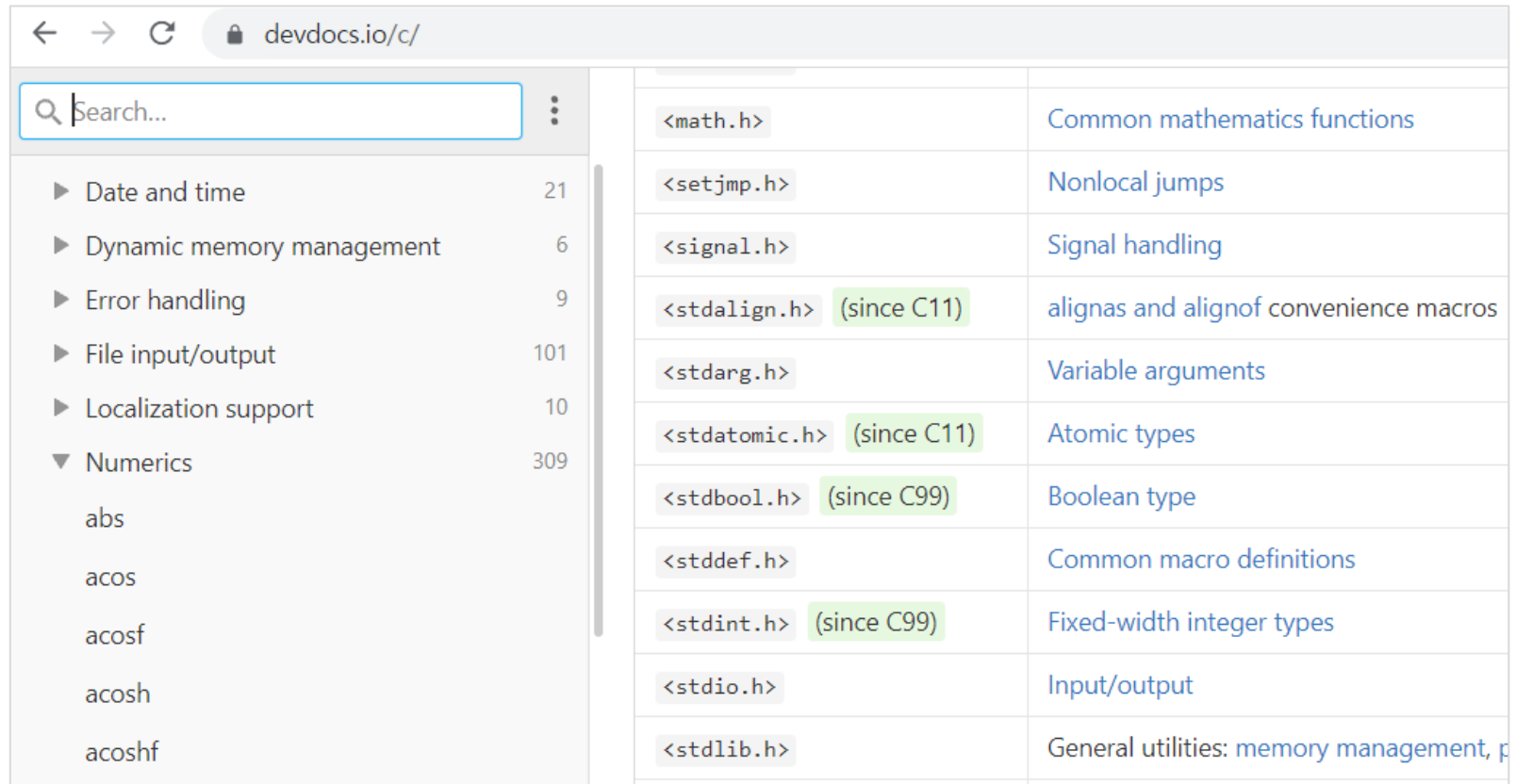
- 하나의 기능을 수행하는 일련의 코드이다.
- 함수는 이름이 있고, 반환값과 매개변수가 있다.(없는 경우도 있음)
- 하나의 큰 프로그램을 작은 부분들로 분리하여 코드의 중복을 최소화하고, 개발은 물론 코드의 수정이나 유지보수를 쉽게 한다.



표준 라이브러리 함수(function)

❖ 내장 함수 – 표준 라이브러리 함수

C언어 Document: <https://devdocs.io/c>



← → ↺ 🔒 devdocs.io/c/	
<input type="text" value="Search..."/>	
▶ Date and time 21	<math.h> Common mathematics functions
▶ Dynamic memory management 6	<setjmp.h> Nonlocal jumps
▶ Error handling 9	<signal.h> Signal handling
▶ File input/output 101	<stdalign.h> (since C11) alignas and alignof convenience macros
▶ Localization support 10	<stdarg.h> Variable arguments
▼ Numerics 309	<stdatomic.h> (since C11) Atomic types
abs	<stdbool.h> (since C99) Boolean type
acos	<stddef.h> Common macro definitions
acosf	<stdint.h> (since C99) Fixed-width integer types
acosh	<stdio.h> Input/output
acoshf	<stdlib.h> General utilities: memory management, p

표준 라이브러리 함수(function)

❖ 헤더 파일 위치

로컬 디스크 (C:) > Program Files (x86) > Microsoft Visual Studio > 2019 > Community > VC > Tools > MSVC > 14.24.28314 > include

```
#pragma once
#ifndef _CSTDIO_
#define _CSTDIO_
#include <yvals_core.h>
#if _STL_COMPILER_PREPROCESSOR

#include <stdio.h>

#pragma pack(push, _CRT_PACKING)
#pragma warning(push, _STL_WARNING_LEVEL)
#pragma warning(disable : _STL_DISABLED_WARNINGS)
_STL_DISABLE_CLANG_WARNINGS
#pragma push_macro("new")
#undef new

// undef common macro overrides
#undef clearerr
#undef feof
#undef ferror
#undef fgetc
```



표준 라이브러리 함수(function)

❖ 내장 함수 – 표준 라이브러리 함수

- 수학 관련 함수 – math.h를 include 해야 함

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

int main() {
    //반올림
    printf("%.2f\n", round(2.54));
    printf("%.2f\n", round(-2.54)); //작은쪽 정수로 결과 출력
    printf("%.21f\n", round(2.54));

    //내림
    printf("%.2f\n", floor(11.3));
    printf("%.2f\n", floor(-11.3));
    printf("%.21f\n", floor(11.3));

    //절대값
    printf("%d\n", abs(8));
    printf("%d\n", abs(-8));

    return 0;
}
```



표준 라이브러리 함수(function)

- 시간 관련 함수 – time.h를 include 해야 함

```
#include <stdio.h>
#include <time.h>
#include <Windows.h>
int main() {
    //time_t 자료형
    //time_t now = time(NULL);
    long now = time(NULL);

    printf("1970년 1월 1일 이후 : %ld초 \n", now);
    //%ld - long decimal 자료형 서식문자*/

    //1부터 10까지 출력하기 - 대기 시간 사용하기
    int i;
    for (i = 1; i <= 10; i++) {
        printf("%d\n", i);
        Sleep(500); //0.01초 Window.h 포함
    }
}
```



표준 라이브러리 함수(function)

- 시간 관련 함수 – time.h를 include 해야 함

```
//실행 시간 측정하기
long start, end;
start = time(NULL); //시작 시간

for (int i = 0; i < 100; i++) {
    printf("%d번째 for문\n", i);
    Sleep(20);
}

end = time(NULL); //종료 시간

double timer = difftime(end, start);

printf("for문 수행 시간(difftime): %lf초\n", timer);

return 0;
}
```



표준 라이브러리 함수(function)

- rand() 함수 – srand() 함수 필요함

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

int main(void)
{
    int dice, i;

    //srand(10);    //seed 배정
    srand(time(NULL)); //time(NULL) - seed가 계속 변함

    printf("%d\n", rand());

    //주사위 10번 던지기
    for (i = 0; i < 10; i++) {
        dice = rand() % 6 + 1;
        printf("%d\n", dice);
    }

    return 0;
}
```



표준 라이브러리 함수(function)

❖ 2개의 주사위 던지기

- ✓ 주사위 눈의 합이 11이 나오면 "Eleven thrown!!" 출력
- ✓ 주사위 눈이 같으면 "Double thrown!!" 출력

```
Microsoft Visual Studio 디버그 콘솔
6
6
Double Thrown!
10
Double Thrown!
5
11
Eleven Thrown!
10
2
Double Thrown!
7
10
9
```

```
srand(time(NULL));
int i;
for (i = 1; i <= 10; i++)
{
    int dice1 = rand() % 6 + 1;
    int dice2 = rand() % 6 + 1;
    int total = dice1 + dice2;
    printf("%d \n", total);
    if (total == 11)
        printf("Eleven Thrown! \n");
    if (dice1 == dice2)
        printf("Double Thrown!\n");
}
```



함수(function)

❖ 함수의 정의와 호출

1. return값이 없고, 매개변수 없는 경우

```
#include <stdio.h>
void hello(); ← 프로토타입
int main()
{
    hello(); ← 함수 호출
    return 0;
}

void hello() ← 함수 정의
{
    printf("Hello C!!");
}
```

```
#include <stdio.h>
void hello()
{
    printf("Hello~\n");
}

int main()
{
    hello();
    hello();

    return 0;
}
```

정의한 함수가 main() 위에 위치하면 프로토타입 사용안함

⚠ C4013 'hello'이(가) 정의되지 않았습니다. extern은 int형을 반환하는 것으로 간주합니다.
✖ C2371 'hello': 재정의. 기본 형식이 다릅니다.

함수(function)의 유형

2. return값이 없는 함수 – 매개변수는 있는 경우

```
void hello(char[]);  
int main() {  
    hello("kimsan\n");  
    hello("Elsa\n");  
    return 0;  
}  
  
void hello(char name[]) {  
    printf("Hello, %s", name);  
}
```



함수(function)의 유형

3. return값이 있는 함수 – 매개변수는 1개 있는 경우

```
#include <stdio.h>
int abs(int n) {
    if (n < 0)
        return -n;
    else
        return n;
}

int main()
{
    int value;
    value = abs(-5);
    printf("절대값은 : %d", value);
    return 0;
}
```



함수(function)의 유형

3. return값이 있는 함수 – 매개변수는 2개 있는 경우

```
#include <stdio.h>
int add(int x, int y)
{
    return x + y;
}

int main()
{
    int result;
    result = add(10, 20);
    printf("두 수의 합 : %d\n", result);

    return 0;
}
```

함수 정의

함수 호출



함수(function) 예제

- 정사각형과 삼각형의 넓이 구하는 프로그램

- 정사각형

- 한 변의 길이 : 4cm

- ▷ 삼각형

- 밑변 : 3cm, 높이 : 4cm

```
#include <stdio.h>
int square(int);
int triangle(int, int);
int main()
{
    int rec_area;
    int tri_area;
    rec_area = square(4);
    tri_area = triangle(3, 4);
    printf("정사각형의 넓이 : %d\n", rec_area);
    printf("삼각형의 넓이 : %d\n", tri_area);
}

int square(int n) {
    return n * n;
}

int triangle(int b, int h) {
    return b * h / 2;
}
```



함수(function) 예제

- 배열에서 최대값 구하기

```
int findMax(int[], int);

int main() {
    int arr[] = { 21, 35, 71, 2, 66, 97 };

    int max1 = findMax(arr, 6);

    printf("최대값 max1 = %d\n", max1);

    return 0;
}
```

```
int findMax(int arr[], int len) {

    int maxVal = arr[0];

    for (int i = 0; i < len; i++) {
        if (maxVal < arr[i])
            maxVal = arr[i];
    }

    return maxVal;
}
```



변수의 메모리 영역

- **코드 영역** : 프로그램의 **실행 코드** 또는 **함수**들이 저장되는 영역
- **스택 영역** : **매개 변수 및 종괄호(블록)** 내부에 **정의된 변수**들이 저장되는 영역
- **데이터 영역** : **전역 변수**와 **정적 변수**들이 저장되는 영역
- **힙 영역** : **동적으로 메모리 할당하는 변수**들이 저장되는 영역



코드 영역
(실행 코드, 함수)



스택 영역
(지역 변수, 매개 변수)



데이터 영역
(전역 변수, 정적 변수)



힙 영역
(동적 메모리 할당)



변수의 적용 범위 - 지역변수

➤ 지역 변수(local variable)

- 하나의 코드 블록에서만 정의되어 사용되는 변수
- 함수 또는 제어문의 중괄호{ } 내부에서 사용

지역 변수의 메모리 생성 시점

- 중괄호 내에서 초기화할 때

지역 변수의 메모리 소멸 시점:

- 중괄호를 벗어났을 때

```
void ab10(int a, int b) {  
    a = a * 10;  
    b = b * 10;  
}  
  
int main(void)  
{  
    int a, b;  
  
    a = 1;  
    b = 2;  
  
    ab10(a, b);  
  
    printf("a의 값은 %d이고, b의 값은 %d입니다.\n", a, b);  
  
    return 0;  
}
```



변수의 적용 범위 – 전역 변수

➤ 전역 변수(global variable)

- 전체 소스 코드를 범위로 적용되는 변수
- 소스 파일 내의 어디서든지 사용 가능한 변수

전역 변수의 메모리 생성 시점

- 프로그램이 시작되었을 때

전역 변수의 메모리 소멸 시점:

- 프로그램이 종료되었을 때

```
int quantity = 2; //전역 변수
void price();

int main(void)
{
    printf("%d개에\n", quantity);
    price();

    return 0;
}

void price() {
    int price = 20000 * quantity;
    printf("%d원입니다.\n", price);
}
```



변수의 적용 범위 – 정적 변수

➤ 정적 변수(static variable)

- 선언된 함수가 종료하더라도 그 값을 계속 유지하는 변수
- **static** 키워드를 붙임

전역 변수의 메모리 생성 시점

- 중괄호 내에서 초기화될때

전역 변수의 메모리 소멸 시점:

- 프로그램이 종료되었을 때

```
void count();
int main()
{
    count();
    count();
    return 0;
}

void count() {
    int x = 0;
    static int y = 0;
    x = x + 1;
    y = y + 1;
    printf("x : %d, y : %d\n", x, y);
}
```



변수의 적용 범위 – 정적 변수

➤ 정적 변수(static variable)

```
void bell();

int main(void)
{
    bell();
    bell();
    bell();

    return 0;
}

void bell() {
    static int order = 0;
    order++;

    printf("현재 주문번호는 %d번 입니다.\n", order);
}
```



헤더 파일 사용하기

❖ 헤더파일 사용하기

- 다른 소스 파일에서 함수를 사용하는 방법이다.(공동작업 시 활용)
- 헤더파일에서는 함수의 프로토타입을 선언한다.

add.c

```
int add(int x, int y) {  
    return x + y;  
}
```

add.h

```
int add(int, int);
```

```
#include <stdio.h>  
#include "add.h"
```

```
int main() {  
  
    int value = add(10, 20);  
    printf("합계 : %d\n", value);  
  
    return 0;  
}
```

usingadd.c



헤더 파일 사용하기

전처리기 지시자

#if ~ #elif ~ #else ~ #endif

defined EN

정의된 식별자

```
#if defined EN
    #define HELLO_MESSAGE "Hello"
#else defined KO
    #define HELLO_MESSAGE "안녕하세요"
#endif
```

```
#include <stdio.h>
#define KO
#include "message.h"

int main() {
    printf("%s\n", HELLO_MESSAGE);

    return 0;
}
```



변수의 적용 범위 – 외부 변수

➤ 외부 변수(Extern Variable)

- 다른 소스 파일에 있는 변수를 사용할 때는 **extern** 키워드를 사용한다.

Calculator.c

```
int count = 0;
int add(int a, int b)
{
    int sum;
    sum = a + b;
    return sum;
}
```

CalcMain.c

```
#include <stdio.h>
extern int count;
extern int add(int, int);
void main()
{
    int x = 3, y = 4, z;
    count++;
    z = add(x, y);
    printf("z=%d\n", z);
    printf("count = %d", count);
}
```

다른 파일의 함수
사용시 extern 사용
(생략가능)

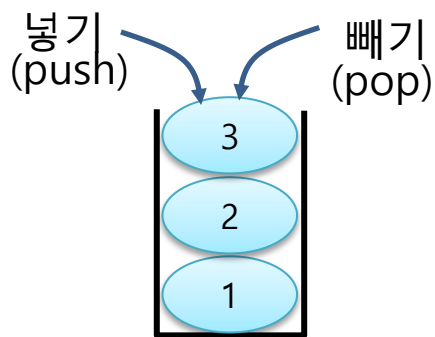


스택(Stack)과 큐(Queue)

❖ Stack

- 후입선출(LIFO : Last in First Out) 구조
 - 나중에 들어간 자료를 먼저 꺼냄
- (응용 예: 스택 메모리, 접시 닦이, 게임 무르기)

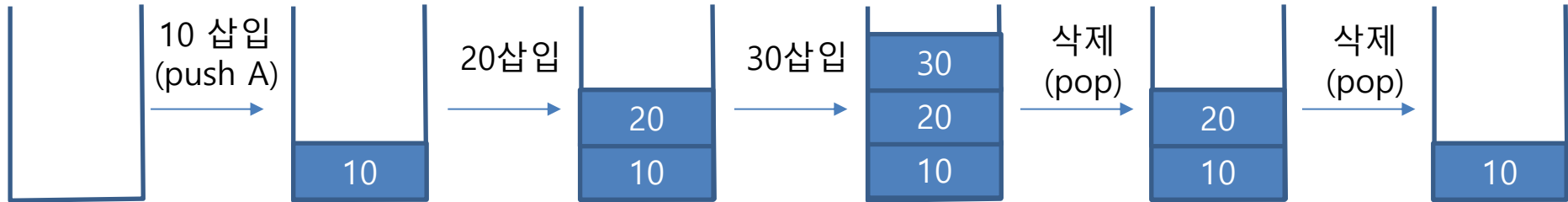
메소드명	설명
push()	원소(자료)를 스택에 넣는다.
pop()	스택의 맨 위 원소를 가져온다.



스택(Stack)



배열 - 스택(Stack)



```
#include <stdio.h>
#define MAX 10

int top = -1;
int stack[MAX];

void push(int n) {
    top++;
    stack[top] = n;
    printf("%d\n", stack[top]);
}

int pop() {
    if (top < 0) {
        printf("스택이 비었습니다.\n");
        return 0;
    }
    else {
        return stack[top--];
    }
}
```

```
int main(void)
{
    printf("== 스택에 자료 삽입 ==\n");
    push(10);
    push(20);
    push(30);

    printf("== 스택에서 자료 삭제 ==\n");
    printf("%d\n", pop());
    printf("%d\n", pop());
    printf("%d\n", pop());
    printf("%d\n", pop());

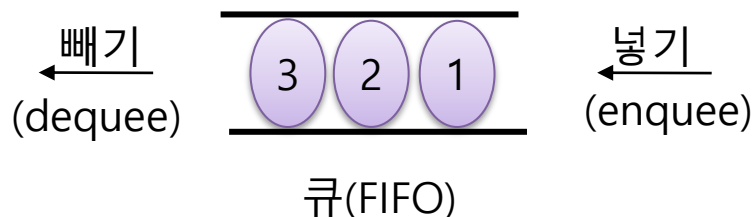
    return 0;
}
```

스택(Stack)과 큐(Queue)

❖ Queue

- 선입선출(FIFO : First in First Out) 구조
 - 먼저 들어간 자료를 먼저 꺼냄(응용 예: 택시정류장 줄서기, 운영체제 작업큐)

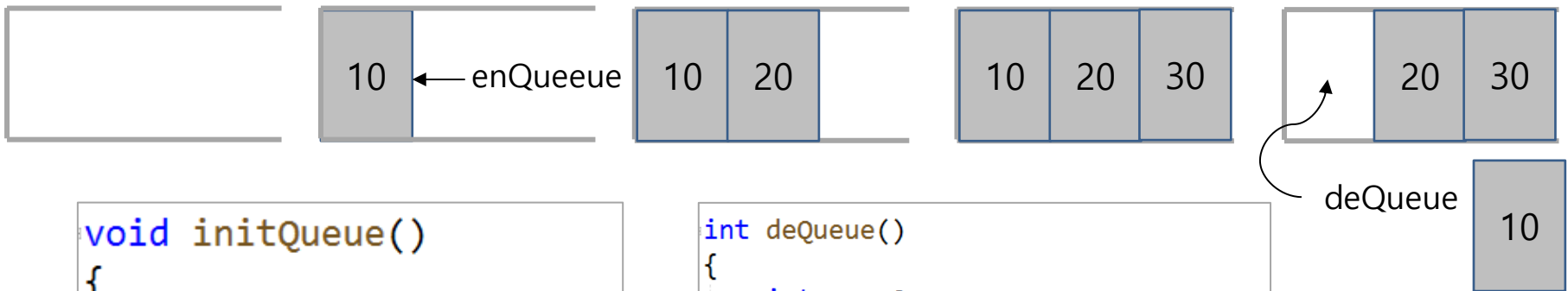
메소드명	설명
enqueue()	주어진 객체를 스택에 넣는다.
dequeue()	객체 하나를 가져온다. 객체를 큐에서 제거한다.



배열 - 큐(Queue)

◆ 큐(Queue) 방식

선입선출(First-In-First-Out) : 먼저 들어간 데이터를 먼저 빼는 자료구조



```
void initQueue()
{
    front = rear = 0;
}

void enqueue(int n)
{
    element[rear] = n;
    rear = ++rear % MAX;
}
```

```
int dequeue()
{
    int n = 0;
    if (front == rear)
    {
        printf(" 큐가 비었습니다. ");
    }
    else
    {
        n = element[front];
        front = ++front % MAX;
    }
    return n;
}
```



재귀 호출

➤ 재귀 호출(Recursive Call)

- 어떤 함수의 내부에서 다시 자신을 호출하는 기능

```
#include <stdio.h>
void HelpMe();
int main()
{
    int num;
    for (num = 0; num < 10; num++) {
        HelpMe();
    }
    return 0;
}

void HelpMe()
{
    printf("살려주세요!\n");
}
```

반복문

```
#include <stdio.h>
int HelpMe(int);
int main()
{
    int num = 10;
    HelpMe(num);
    return 0;
}

int HelpMe(int x)
{
    if (x <= 0)
        return 0;
    else
        printf("살려주세요!\n");
        return HelpMe(x - 1);
}
```

재귀 호출

재귀 호출

```
if 입력값이 충분히 작으면 : //종료 조건
    return 결과값;
else
    return 결과값;           // 더 작은 값으로 자기 자신 호출
```

```
#include <stdio.h>
int factorial(int);
int main() {
    printf("팩토리알은 : %d\n", factorial(1));
    printf("팩토리알은 : %d\n", factorial(5));
    printf("팩토리알은 : %d\n", factorial(10));

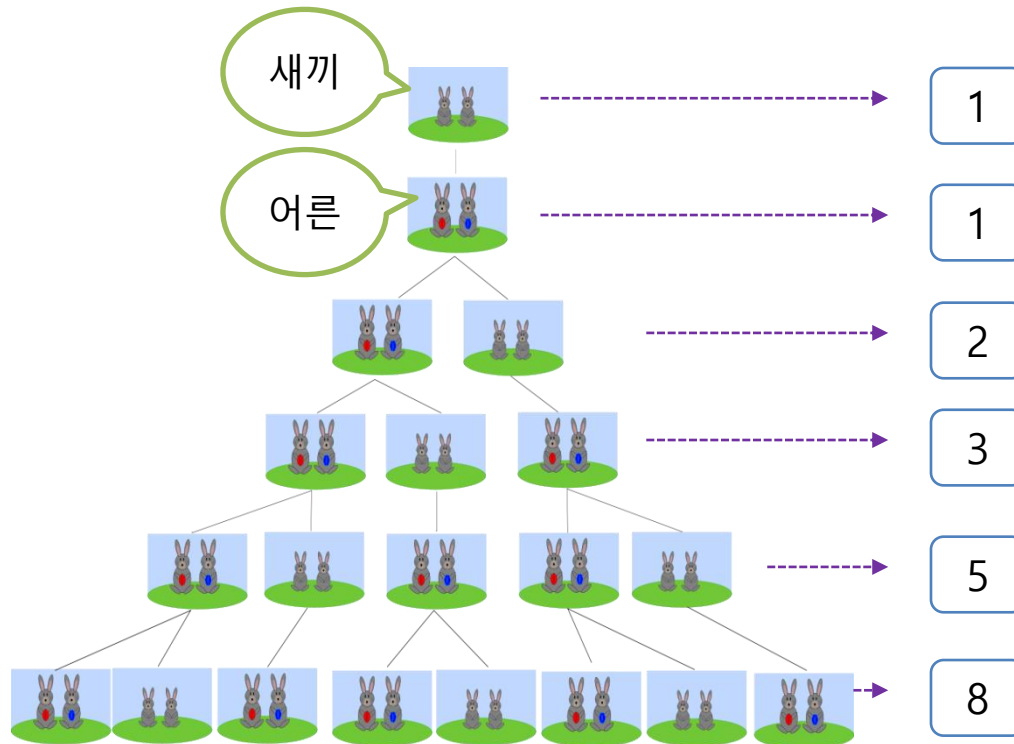
    return 0;
}

int factorial(int n) {
    if (n <= 1)
        return 1;
    else
        return n * factorial(n - 1);
}
```

```
팩토리알 : 1
팩토리알 : 120
팩토리알 : 3628800
```

재귀 호출

피보나치 수열 - 재귀 함수



첫번째 달에 새로 태어난 토끼 한쌍이 있고, 둘째달에 토끼가 커서 그대로 어른토끼 한쌍, 세째달에는 새끼를 한쌍 낳아 두쌍, 네째달에는 또 한쌍을 낳아 모두 세쌍, 이렇게 계속 새끼를 낳고, 죽지 않는다는 가정을 세우면 피보나치의 수가 된다.



재귀 호출

피보나치 수열

```
int i = 1;      //반복 변수
int n = 12;     //12개월 후
int value1, value2, value_new;
printf("피보나치 수열 - 토끼의 번식\n");
value1 = value2 = 1; //1항, 2항
printf("%4d", value1);
printf("%4d", value2);
i = 3;         //3항
while (i <= n) {
    value_new = value1 + value2; //1항+2항
    value1 = value2;
    value2 = value_new;
    printf("%4d", value_new);
    i++;
}
```

반복문

```
#include <stdio.h>
int fivo(int);
int main()
{
    int i;
    for (i = 1; i <= 12; i++) {
        printf("%-3d", fivo(i));
    }
}

int fivo(int n)
{
    if (n <= 2)
        return 1;
    else
        return fivo(n - 2) + fivo(n - 1);
}
```

재귀 호출

rand() 함수

❖ 숫자를 추측해서 맞추는 게임

Microsoft Visual Studio 디버그 콘솔

```
맞춰 보세요(1~30) : 25
너무 커요!
맞춰 보세요(1~30) : 20
너무 커요!
맞춰 보세요(1~30) : 10
너무 작아요!
맞춰 보세요(1~30) : 15
너무 커요!
맞춰 보세요(1~30) : 13
너무 커요!
맞춰 보세요(1~30) : 12
정답입니다!
```

```
srand(time(NULL));
int com = rand() % 30 + 1;
int guess;
while (1)
{
    printf("맞춰보세요(1~30) : ");
    scanf_s("%d", &guess);
    if (com == guess)
    {
        printf("정답입니다! \n");
        break;
    }

    if (com > guess)
        printf("너무 작아요! \n");
    if (com < guess)
        printf("너무 커요! \n");
}
```



rand() 함수

❖ 동전던지기 게임

```
-----동전 던지기-----
1. 앞면 2. 뒷면 3. 종료는 다른 값
입력해주세요 : 2
컴퓨터 : 뒷면
당신 : 뒷면
맞았습

-----동전 던지기-----
1. 앞면 2. 뒷면 3. 종료는 다른 값
입력해주세요 : 1
컴퓨터 : 앞면
당신 : 앞면
맞았습

-----동전 던지기-----
1. 앞면 2. 뒷면 3. 종료는 다른 값
입력해주세요 : 1
컴퓨터 : 뒷면
당신 : 앞면
틀렸습

-----동전 던지기-----
1. 앞면 2. 뒷면 3. 종료는 다른 값
입력해주세요 : 3
게임을 종료합니다.
```

```
srand(time(NULL));
int com;
int you;
int run = 1;
while (run)
{
    com = rand() % 2 + 1;
    printf("-----동전 던지기-----\n");
    printf("1. 앞면 2. 뒷면 3. 종료는 다른 값 \n");
    printf("입력해주세요 : ");
    scanf_s("%d", &you);
    if (you < 1 || you > 2)
    {
        run = 0;
    }
    else
    {
        printf("컴퓨터 : %s\n", com == 1 ? "앞면" : "뒷면");
        printf("당신 : %s\n", you == 1 ? "앞면" : "뒷면");
        if (com == you)
            printf("맞았습 \n");
        else
            printf("틀렸습 \n");
    }
}
printf("게임을 종료합니다.\n");
```

