재귀 함수

- 재귀 함수(recursive function)란
  - 함수구현에서 자신을 호출하는 함수
  - 재귀적 특성을 표현하는 알고리즘에 유용
    - 재귀 함수를 이용하면 문제를 쉽게 해결할 수 있고 이해하기도 쉬움
- 수학 수식에서 재귀적 특성
  - n 계승(n factorial)을 나타내는 수식 n!
    - 1 * 2 * 3 * … * (n-2) * (n-1) * n을 의미
    - 즉 n!의 정의에서 보듯 계승은 재귀적 특성
  - n! = (n-1)! n
  - n!을 함수 factorial(n)로 구현
    - factorial(n) 구현에서 factorial(n-1)을 호출
- 재귀 함수의 단점
  - 함수의 호출이 계속되면 시간도 오래 걸리고 메모리 사용도 많다

재귀 함수 사용
(factorial.c)

```
#include <stdio.h>
int factorial(int); //함수원형

int main(void) {
    for (int i = 1; i <= 10; i++)
        printf("%2d! = %d₩n", i, factorial(i));
    return 0;
}

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

[결과]

```
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
```

라이브러리 함수
(rand() 사용)

- 함수 rand()
  - 연속적인 난수(임의의 수)를 생성해 주는 함수
  - 임의의 수
    - 어느 수가 반환될 지 예측할 수 없으며 모든 수에 대해 선택될 확률이 동일하다는 의미
- 함수원형은 헤더파일 stdlib.h에 정의
  - 함수 rand()를 사용하려면…
    #include <stdlib.h>
  - Visual C++에서는 함수 rand()
    - 0~ 32767(RAND_MAX=0x7fff)사이의 정수 중에서 임의로 수
- 매번 난수를 다르게 생성 하려면
  - 시드(seed)값(난수를 다르게 만들기 위해 처음에 지정하는 수)을 매번 다르게 줘야 함.
  - srand(time(NULL)); 1회 호출 -> rand(); N회 호출
- 매번 다른 seed 값을 지정하기 위해 함수 time() 이용
  - time(NULL) : 1970년 1월 1일 이후 현재까지 경과된 시간을 초 단위로 반환
  - #include <time.h>
- a에서 b까지의 난수를 발생시키는 방법
  - rand() % (b – a + 1) + a

난수 출력
(srand.c)

```
#include <stdlib.h> //rand(), srand()를 위한 헤더파일 포함
#include <time.h>   //time()을 위한 헤더파일 포함
#define MAX 100

int main(void) {
    long seconds = (long)time(NULL);
    srand(seconds);

    printf("1 ~ %5d 사이의 난수 5개:₩n", MAX);
    for (int i = 0; i < 5; i++)
        printf("%5d ", rand() % MAX + 1);
    puts("");

    return 0;
}
```

[결과]
1 ~ 100 사이의 난수 5개:
  92 19 38 67 30

수학 라이브러리 함수 (math.c)

```
#include <stdio.h>
#include <math.h> //수학 관련 다양한 함수헤더 포함 헤더파일
int main(void) {
    printf("  i   i제곱   i세제곱  제곱근(sqrt)₩n");
    printf("-----------------------------------₩n");
    for (int i = 3; i < 7; i++)
        printf("%3d %7.1f %9.1f %9.1f₩n", i, pow(i, 2), pow(i, 3), sqrt(i));
    printf("₩n");
    printf("exp(1.0) == %5.2f, ", exp(1.0));
    printf("pow(2.72, 1.0) == %5.2f, ", pow(2.72, 1.0));
    printf("sqrt(49) == %5.2f₩n", sqrt(49));
    printf("abs(-10) == %5d, ", abs(-10));
    printf("ceil(7.1) == %5.2f, ", ceil(7.1));
    printf("floor(6.9) == %5.2f₩n", floor(6.9));
    return 0;
}
```

[결과]

```
 i   i제곱   i세제곱  제곱근
---------------------------
 3     9.0     27.0      1.
 4    16.0     64.0      2
 5    25.0    125.0      2
 6    36.0    216.0      2

exp(1.0) ==  2.72, pow(2.
abs(-10) ==    10, ceil(7.1
```

문자 라이브러리 함수
(char.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <ctype.h>
void print2char(char);
int main(void) {
    char ch;
    printf("알파벳(종료x) 또는 다른 문자 입력하세요.₩n");
    do {
        printf("문자 입력 후 Enter: ");
        scanf("%c", &ch);
        getchar();              //enter 키 입력 받음
        if (isalpha(ch))
            print2char(ch);
        else
            printf("입력: %c₩n", ch);
    } while (ch != 'x' && ch != 'X'); //입력이 x 또는 X이면 종료
    return 0;
}
```

[결과]

```
알파벳(종료x) 또는 다른 문자 입력하
문자 입력 후 Enter: A
입력: A, 변환: a
문자 입력 후 Enter: x
입력: x, 변환: X
```

문자 라이브러리 함수 (char.c)

```
void print2char(char ch) {
    if (isupper(ch))
        printf("입력: %c, 변환: %c₩n", ch, tolower(ch));
    else
        printf("입력: %c, 변환: %c₩n", ch, toupper(ch));
    return;
}
```

```
[결과]
알파벳(종료x) 또는 다른 문자 입력하세요.
문자 입력 후 Enter: A
입력: A, 변환: a
문자 입력 후 Enter: x
입력: x, 변환: X
```

11. 문자와 문자열

6주차

문자열

- ◆ 문자
  - ● 한 글자를 작은 따옴표로 둘러싸서 표기(‘A’)
    - ❖ C 언어에서 : 1byte 자료형 char로 지원
  - ● 작은 따옴표에 의해 표기된 문자 = 문자 상수
- ◆ 문자열 상수
  - ● “ ” 안쪽에 문자들을 나열
    “C language!”
  - ● 일반문자, 제어문자
  - ● 맨 뒤에 눈에 보이지 않는 NULL문자(‘\0’)가 있음.
- ◆ 문자열 변수
  - ● C언어에서는 문자열 자료형이 따로 없음
  - ● 문자 자료형의 배열을 문자열 변수로 사용
  - ● 맨 끝에 NULL문자(‘\0’)를 붙여야 온전한 문자열

문자 배열(문자열)의 초기화

◆ 문자열 상수로써 초기화

```
char  a2[] =  “abc” ;
```

맨 뒤에 자동으로 NULL문자가 붙게 됨

◆ 문자의 나열로써 초기화

```
char  a1[] = {  ‘a’ ,  ‘b’ ,  ‘c’ ,  ‘\0’  };
```

NULL문자를 붙여줘야 됨

◆ 위 두 문자 배열이
정확히 같은 저장장소(4 byte씩)를 잡게 됨.

‘a’	‘b’	‘c’	0x00

문자와 문자열 출력 (chararray.c)

```
#include <stdio.h>
int main(void) {
    char ch = 'A';
    printf("%c %d₩n", ch, ch);

    char java[] = { 'J', 'A', 'V', 'A', '₩0' };
    printf("%s₩n", java);
    char c[] = "C language";
    printf("%s₩n", c);
    char csharp[5] = "C#";
    printf("%s₩n", csharp);

    printf("%c%c₩n", csharp[0], csharp[1]);
    return 0;
}
```

문자 선언과 출력

3가지 문자열 선언 방법

[결과]
A 65
JAVA
C language
C#
C#

문자열 상수를 문자 포인터에 저장하는 방식 (charpointer.c)

```
#include <stdio.h>
int main(void) {
    char *java = "java";
    printf("%s ", java);

    int i = 0;
    while (java[i]) //while (java[i] != '₩0')
        printf("%c", java[i++]);
    printf(" ");
    i = 0;
    while (*(java + i) != '₩0') //java[i]는 *(java + i)와 같음
        printf("%c", *(java + i++));
    printf("₩n");

    //java[0] = 'J';
    return 0;
}
```

[결과]
java java java

문자 포인터가 가리키는 문자부터 NULL문자까지 하나 하나 출력

java는 문자열 상수를 가리키고 있으므로 수정 불가능! 실행오류 발생!

'\0' 문자에 의한 문자열 분리 (string.c)

```
#include <stdio.h>

int main(void) {
    char c[] = "C C++ Java";
    printf("%s₩n", c);
    c[5] = '₩0';
    printf("%s₩n%s₩n", c, (c + 6));

    c[5] = ' '; //널 문자를 빈 문자로 바꾸어 문자열 복원
    char *p = c;
    while (*p != '₩0')
        printf("%c", *p++);
    printf("₩n");

    return 0;
}
```

[결과]
C C++ Java
C C++
Java
C C++ Java

문자열 2개로써 출력

문자 배열의 각 원소를 하나 하나 문자로 출력

문자입력 함수 비교

- getchar()
  - 라인 버퍼링(line buffering) 방식 사용
    - [enter] 키 입력까지 buffer에 저장하다가 [enter] 들어오면 buffer의 내용을 한꺼번에 프로그램 buffer로 줌
    - 즉각적(interactive)인 입력을 요구할 때는 사용 불가능
  - #include <stdio.h>
- _getche()
  - 버퍼를 사용하지 않으므로 문자 하나를 입력하면 바로 함수가 return됨.
  - 입력된 문자를 모니터에 표시
  - #include <conio.h>
- _getch()
  - 버퍼를 사용하지 않으므로 문자 하나를 입력하면 바로 함수가 return됨.
  - 입력된 문자를 모니터에 표시하지 않음
  - #include <conio.h>

다양한 문자 입력 (getche.c)

```
#include <stdio.h>
#include <conio.h>
int main(void) {
    char ch;
    printf("문자를 계속 입력하고 Enter를 누르면 >>₩n");
    while ((ch = getchar()) != 'q')
        putchar(ch);
    printf("₩n문자를 누를 때마다 두 번 출력 >>₩n");
    while ((ch = _getche()) != 'q')
        putchar(ch);
    printf("₩n문자를 누르면 한 번 출력 >>₩n");
    while ((ch = _getch()) != 'q')
        _putch(ch);
    printf("₩n");
    return 0;
}
```

[결과]

```
문자를 계속 입력하고 Enter를 누르면 >>
abc
abc
deq
de
문자를 누를 때마다 두 번 출력 >>
aabbccq
문자를 누르면 한 번 출력 >>
abc
```

문자열 입출력

- scanf(), printf()
  - 변환문자 %s 사용
  - 공백으로 구분되는 하나의 문자열 입력 받음
    - 이름과 성을 분리하여 입력한다면 성만 저장됨
  - 입력 받을 문자열은 충분한 공간이 준비되어 있어야 함
  - 단순히 문자 포인터로는 문자열 저장이 불가능
  - printf()에서 %10s : 폭이 10, 우측정렬
- gets()
  - #include <stdio.h>
  - [enter] 키를 누를 때까지 버퍼에 저장 한 후 입력처리
  - 마지막에 입력된 '\n'가 '\0'로 교체되어 저장
- puts()
  - 오류가 발생하면 EOF를 반환
    - EOF(End Of File) : #define EOF (-1)
  - 문자열의 마지막에 저장된 '\0'를 '\n'로 교체하여 버퍼에 전송
    -> puts()를 연속으로 사용하면 모니터에 한 행씩 보임

문자열 입출력 : scanf(), printf()
(stringput.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void) {
    char name[20], dept[30];

    printf("%s", "학과 입력 >> ");
    scanf("%s", dept);
    printf("%s", "이름 입력 >> ");
    scanf("%s", name);
    printf("출력: %10s %10s₩n", dept, name);

    return 0;
}
```

문자열 저장할 충분한 공간 확보!
char *name, *dept;
실행 오류 발생

공백 없이 입력해야 함

[결과]
학과 입력 >> computer
이름 입력 >> jychoi
출력: computer jychoi

문자열 입출력 : gets(), puts()
(gets.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void) {
    char line[101];

    printf("입력을 종료하려면 새로운 행에서 (ctrl + Z)를 누르십시요.₩n");
    while (gets(line))
        puts(line);
    printf("₩n");

    /*while (gets_s(line, 101))
        puts(line);
    printf("₩n");*/
    return 0;
}
```

문자열 저장할 충분한 공간 확보!
char *line 으로는 오류발생.

공백 없이 입력해야 함

```
[결과]
입력을 종료하려면 새로운 행에서 (ctrl + Z)를
abc
abc
de
de
^Z
```