Test

임베디드스쿨2기

Lv1과정

2021. 04. 28

박태인

1. Q1)

1) C언어는 함수를 호출 할 때 마다 무엇을 생성하는가?
어째서 재귀호출이 일반적인 Loop 보다 성능이 떨어지는 것인가?
이에 대해 상세히 기술하시오.

나의답 :

C언어는 함수를 호출 할 때마다, 이전 함수 복귀 주소를 Main 또는 이전 함수의의 rsp-8 위치에 복귀 주소를 push 하고 호출 하는 함수의 새로운 스택으로 jump 이동하게 된다.

재귀 호출이 일반적인 Loop 보다 성능이 떨어지는 이유?

시간이 많이 걸리고 공간을 많이 차지 하기 때문 입니다.
재귀함수를 사용 했을 경우 호출한 함수에서 다시 또 자신의 함수를 호출하는 구조 입니다.

함수는 기본적으로 새로운 스택을 만들어 내야 하고 계속해서 새로운 메모리 공간이 필요하게 됩니다.
물론 결론이 나는 함수(= 더이상 재귀하지 않는) 메모리의 경우 결론을 도출하고 사라지지만
기본적으로 한번에 많은 함수가 호출이 되어 메모리 공간이 더욱 많이 필요하게 됩니다.

이러한 원리로 재귀 호출이 일반적인 Loop 보다 성능이 떨어 질 수 있다고 생각합니다.

Answer :

C언어는 함수를 호출 할 때마다 Stack을 생성한다.
함수 호출의 경우 어셈블리어를 보면 Stack Frame을 만들고 해제하는 코드가 들어간다.
이 부분이 쓸대 없이 지속적으로 반복 되며 call이 발생하므로
오히려 mov cmp jmp인 Loop 보다 효율이 좋지 못하다.
(파이프라인은 둘 다 깨진다)

종합 :

나의 답에서는 **함수 호출이 반복 됨으로써 사용하는 메모리 공간이 늘어남으로써** 재귀 호출이 성능이 떨어 질 수 있다는 것을 설명 하였고,
답에서 이 부분이 어**셈블리 상에서 stack을 생성하고 해제하는 코드가 추가로 들어감에 따라 이것이 반복**되는
재귀호출이 일반적인 루프인 **mov cmp jmp 보다 성능이 떨어 질 수 있다는 판단**이 됩니다.

1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

2) 회원이름, 나이, 전화번호, 거주지를 입력 받도록 한다. (나의 답)

```c
int main(void)
{
        char name[10];
        char address[20];
        int age;
        int phone;

        printf("이름을 입력 하세요. : ");
        scanf("%s", name);

        printf("거주지를 입력 하세요. : ");
        scanf("%s", address);

        printf("나이를 입력 하세요. : ");
        scanf("%d", &age);

        printf("전화번호를 입력 하세요. ");
        scanf("%d", &phone);

//      printf("당신의 이름은 %s \n 거주지는 %s \n 나이는 %d \n 전화번호는 0%d 입니다.\n",name, address, age, phone);
```

```
이름을 입력 하세요. : 박태인
거주지를 입력 하세요. :  수원
나이를 입력 하세요. : 33
전화번호를 입력 하세요. 01090761365
당신의 이름은 박태인
 거주지는 수원
 나이는 33
 전화번호는 01090761365 입니다.
```

나의 답 :
- 이름, 주소를 정보를 받을 수 있게끔 캐릭터형 배열 생성.
- 나이, 전화번호를 받기 위해 int 형 배열 생성.
- sacnf 와 printf 를 통해 각각의 배열에 정보를 입력 받음.
- 문제점 : 함수로 표현하진 못함.

1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

2) 회원이름, 나이, 전화번호, 거주지를 입력 받도록 한다. (Answer)

```
int main(void)
{
        int age;
        char name[32] = { 0 };
        char phone[32] = { 0 };
        char city[32] = { 0 };
        char street[128] = { 0 };
        char detail[128] = { 0 };

        input_info(name, &age, phone, city, street, detail);

        printf("이름: %s, 나이: %d, 전화번호: %s\n거주지: %s시 %s %s\n",
                name, age, phone, city, street, detail);
}
```

우선 main 문을 살펴 보자.
- 나이를 제외한 이름, 전화번호, 도시, 거리, 상세를 전부 char형 배열로 { 0 }; 초기화 하였습니다.
- input_info 라는 함수를 통해서 위의 정보들을 받습니다.
  (여기서 는 &age로 인자를 받습니다. 나머지는 배열이고 **배열 이름이 곧 배열의 첫 주소를 가르키기 때문**입니다.)

- 그리고 호출 된 함수가 완성 되면 각각의 변수의정보를 print 합니다.
- 자 이제 상세히 iput_info 함수로 넘어가 보자.

1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

2) 회원이름, 나이, 전화번호, 거주지를 입력 받도록 한다. (Answer)

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

void input_info(char *name, int *age, char *phone, char *city, char *street, char *detail)
{
        printf("회원 정보를 입력해주세요.\n이름: ");
        scanf("%s", name);

        printf("나이: ");
        scanf("%d", age);

        printf("전화 번호: ");
        scanf("%s", phone);

        printf("도시: ");
        scanf("%s", city);

        printf("도로명: ");
        scanf("%s", street);

        printf("상세 주소: ");
        scanf("%s", detail);
}
```

- input info 함수를 분석해 보자.
  - ㄴ 우선 함수의 인자 변수는 메인 함수의 변수와는 다른 것이다.
  - ㄴ 이는 어셈블리 분석을 통해서도 알 수 있었다.
  - ㄴ 여기서 모든 인자는 주소값으로 받기 위해 인자에 *를 붙여 준다.
    (scanf를 활용하기 위함 이다. scanf는 두 번째 값에 주소 값을 넣어야 한다.

- 이 함수의 경우 이렇듯 함수의 정보를 인자로 받고,
  각 변수를 통해 받은 값이 main으로 전달 된다.

```
회원 정보를 입력해주세요.
이름: 박태인
나이: 33
전화 번호: 01090761365
도시: 수원
도로명: 평동로
상세 주소: 114-1
이름: 박태인, 나이: 33, 전화번호: 01090761365
거주지: 수원시 평동로 114-1
```

1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

3) 적당히 여러 회원을 입력한 이후 거주지가 같은 사람들만 출력해 본다. (Answer)

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

typedef struct _member_info member_info;

struct _member_info
{
        int age;
        char name[32];
        char phone[32];
        char city[32];
        char street[128];
        char detail[128];
};

int total_member;

member_info *init_member_info(void)
{
        member_info *tmp = (member_info *)malloc(sizeof(member_info) * 64);
}
```

구조체 선언을 한 부분이다.

Typedef 을 통해서 struct member_info를 **member_info로 단축 정의** 합니다.

기본적으로 member_info구조체는
Int 형 나이를 제외하고는
이름, 전화번호, 도시, 거리, 상세는 모두 캐릭터형 배열로 선언 되어 있습니다.

그리고 int형 total_member;라는 전역 변수를 선언 하였다.
ㄴ 추후에 사용 될 것이다.

그리고 회원들의 정보를 저장하기 위한 메모리를 동적할당 하기 위해
malloc함수를 사용하게 됩니다. **Malloc 함수는 성공하면 메모리 주소를 반환**하기 때문에
member_info **구조체에 포인터(*) 반환형**의 init_member_info 함수를 생성 합니다.

이 때, 반환형 포인터(*) malloc(크기) 가 malloc의 원형이므로,

(member_info *)malloc(sizeof(member_info) * 64);로 사용이 됩니다.
ㄴ 구조체 포인터 ㄴ구조체 크기 x 64

이렇게 생성된 메모리는 member_info *tmp
ㄴ 구조체 포인터형인 tmp에 저장됩니다.
ㄴ **즉, 회원 정보가 들어 있는 구조체의 주소가 저장됨.**

추가 조사

- 동적할당 malloc 사용

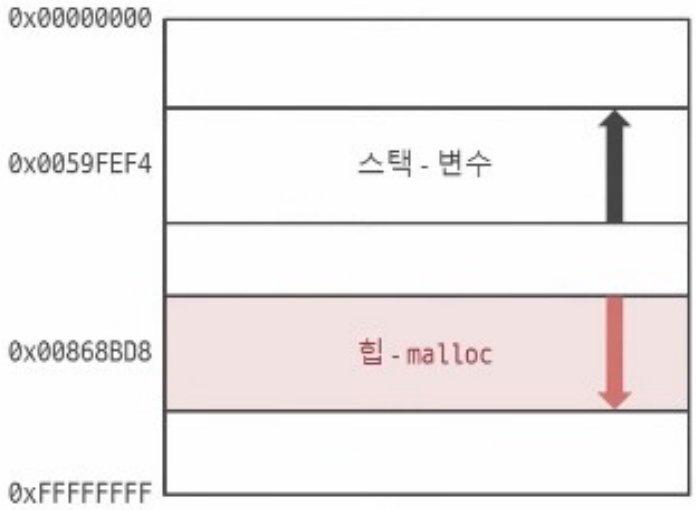
→ malloc 사용법에 대해 알아 보자.
메모리를 사용하기 위해 Malloc 함수로 사용할 메모리의 공간을 확보해야 합니다.
- 헤더는 stdlib.h 에 선언되어 있다.
- 원하는 시점에 원하는 메모리를 할당 할 수 있다고 해서 '동적 할당' 방법이라고도 한다.

- 포인터 = malloc(크기);
  Void *malloc(size_t_Size);
  성공하면 메모리 주소를 반환, 실패하면 NULL을 반환함.

- 변수는 Stack에 메모리를 생성하지만 malloc 함수를 Heap 영역에 메모를 생성한다.
  ㄴ 영역의 예시로 우측의 그림을 참고 하면 되지만 시스템 마다 다를 수 있다.
- Stack 과 Heap의 가장 큰 차이는 Stack 함수를 사용한 뒤에 따로 처리를 하지 않아도 되지만,
  Malloc 함수로 heap에 할당한 메모리는 반드시 메모리 해를 해주어야 한다.(필수사항이다)

- 후처리(free)
  ㄴ free(포인터)
    → void free(void *_Block);

35-2 스택과 힙(Windows x86/32비트)



1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

3) 적당히 여러 회원을 입력한 이후 거주지가 같은 사람들만 출력해 본다. (Answer)

```
void input_info(member_info *mi)
{
        bool run = true;
        int num;

        while (run)
        {
                printf("회원 정보를 계속 입력하시겠습니까 ? 1(yes), 0(no)\n");
                scanf("%d", &num);

                switch (num)
                {
                        case 1:
                                printf("회원 정보를 입력해주세요.\n이름: ");
                                scanf("%s", mi[total_member].name);

                                printf("나이: ");
                                scanf("%d", &mi[total_member].age);

                                printf("전화 번호: ");
                                scanf("%s", mi[total_member].phone);

                                printf("도시: ");
                                scanf("%s", mi[total_member].city);

                                printf("도로명: ");
                                scanf("%s", mi[total_member].street);

                                printf("상세 주소: ");
                                scanf("%s", mi[total_member].detail);

                                total_member++;

                                break;

                        case 0:
                                printf("회원 입력을 종료합니다.\n");
                                run = false;
                                break;
                }
        }
}
```

input_info 함수는 구조체 포인터형 mi를 인자로 받아 정보를 기입하는 함수이다.

여기서 while문 조건에 **Bool 변수형을 사용**하는데,
Bool 변수형을 사용하기 위해서는
#include <stdbool.h> 를 선언하고,이것은 **bool, true, false가 정의된 것**이다.

While문 조건이 run 인데, run이 초기 값이 true 이므로 함수가 실행되면 초기부터 실행이 된다.

회원정보를 입력 하겠느냐 안하겠느냐의 조건은

Switch 문으로 선택을 한다 1 (yes) 0 (no)
여기서 회원들의 정보는 **구조체 mi[멤버 번호].name** 등 의 방식으로 받는다.
여기서 age의 경우 구조체에서 배열이 아닌 int 형 이므로
Scanf 사용시 & 를 붙여 줘야 한다.

이렇게 회원정보는 받고, 만약 사용자가 더이상의 회원정보 입력을 원하지 않으면 Case 0 이 되어 switch 문을 빠져 나오게 된다.

1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

3) 적당히 여러 회원을 입력한 이후 거주지가 같은 사람들만 출력해 본다. (Answer)

```
void print_member_info(member_info *mi)
{
        int i;

        if (mi->name)
        {
                for (i = 0; i < total_member; i++)
                {
                        printf("이름: %s, 나이: %d, 전화번호: %s\n거주지: %s시 %s %s\n",
                                mi[i].name, mi[i].age, mi[i].phone,
                                mi[i].city, mi[i].street, mi[i].detail);
                }
        }
}
```

ㄴ 입력 받은 멤버들의 정보를 출력하는 함수 이다.
여기에서 **if 조건문은 값이 있으면을 의미 하는 것이다!!** 따라서, Mi→age, mi→phone 등의 조건을 써도 마찬가지 인 것이다.

```
int main(void)
{
        member_info *mi;

        mi = init_member_info();
        input_info(mi);

        print_member_info(mi);

        free(mi);
}
```

→ 최종적으로 main 문 실행 절차를 살펴 보자.
- member_info * 구조체 포인터 형태인 mi 를 생성한다.
- Mi = init_member_info()를 통해서 구조체에 동적할당을 실시한다.
- input_info(mi) 구조체 인자를 받아서 scanf로 회원들의 정보를 입력한다.
- 입력 받은 구조체의 정보들을 출력한다.
- 동적 할당 하였던 구조체의 메모리들을 free 시켜 준다.

프로그램 실행 예시 →

```
회원 정보를 계속 입력하시겠습니까 ? 1(yes), 0(no)
1
회원 정보를 입력해주세요.
이름: 박태인
나이: 33
전화 번호: 010
도시: 수원
도로명: 평동로
상세 주소: 114-1
회원 정보를 계속 입력하시겠습니까 ? 1(yes), 0(no)
1
회원 정보를 입력해주세요.
이름: 박찬석
나이: 30
전화 번호: 010
도시: 부산
도로명: 하신중앙로
상세 주소: 60
회원 정보를 계속 입력하시겠습니까 ? 1(yes), 0(no)
1
회원 정보를 입력해주세요.
이름: 노정숙
나이: 58
전화 번호: 010
도시: 부산
도로명: 하신중앙로
상세 주소: 60
회원 정보를 계속 입력하시겠습니까 ? 1(yes), 0(no)
0
회원 입력을 종료합니다.
이름: 박태인, 나이: 33, 전화번호: 010
거주지: 수원시 평동로 114-1
이름: 박찬석, 나이: 30, 전화번호: 010
거주지: 부산시 하신중앙로 60
이름: 노정숙, 나이: 58, 전화번호: 010
거주지: 부산시 하신중앙로 60
```

1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

4) 10대, 20대, 30대 별로 출력해보도록 한다. (Answer)

```
typedef struct _member_info member_info;

struct _member_info
{
        int age;
        char name[32];
        char phone[32];
        char city[32];
        char street[128];
        char detail[128];
};

int total_member;

int age[64] = { 10, 11, 12, 15, 16,
                                18, 20, 21, 24, 27,
                                33, 34, 32, 36, 38, 39 };
char name[64][32] = { "김윤환", "김택용", "이영호", "조기석", "임요환",
                                "마주작", "진영수", "염보성", "김명운", "임홍구",
                                "박경수", "이영수", "김성환", "이재호", "김창수", "도재욱" };
char phone[64][32] = { "010-1111-1234", "010-1111-1234", "010-1111-1234", "010-1111-1234", "010-1111-1234",
                                "010-1111-1234", "010-1111-1234", "010-1111-1234", "010-1111-1234", "010-1111-1234",
                                "010-1111-1234", "010-1111-1234", "010-1111-1234", "010-1111-1234", "010-1111-1234", "010-1111-1234" };
char city[64][32] = { "서울", "서울", "서울", "부산", "서울",
                                "대전", "대전", "대구", "전주", "대구",
                                "전주", "대전", "서울", "서울", "대전", "서울" };
char street[64][128] = { 0 };
char detail[64][128] = { 0 };
```

구조체 member info를 member info로 typedef 정의 합니다.
구조체 내용은 앞선 내용과 동일 합니다.

앞서 방식과 다른점은 회원의 정보를 이중배열을 통해 나타 냅니다.
이중 배열 구조의 예를 들면 아래와 같은 구조이다.

하나더 하면 int **arr[2][4]**

[0]				[1]			
[0][0]	[0][1]	[0][2]	[0][3]	[1][0]	[1][1]	[1][2]	[1][3]

따라서 아래 것 중 하나를 예를 들어 이름 정보의 name[64][32] 는
[64] : 전체 종류 , [32] : 각각의 개수 즉, 64개의 정보를 32byte씩.

1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

4) 10대, 20대, 30대 별로 출력해보도록 한다. (Answer)

```
member_info *init_member_info(void)
{
        member_info *tmp = (member_info *)malloc(sizeof(member_info) * 64);
}

void input_info(member_info *mi)
{
        bool run = true;
        int num;
        int i;

        for (i = 0; i < 16; i++)
        {
                mi[i].age = age[i];

                strcpy(mi[i].name, name[i]);
                strcpy(mi[i].phone, phone[i]);
                strcpy(mi[i].city, city[i]);
                strcpy(mi[i].street, street[i]);
                strcpy(mi[i].detail, detail[i]);
        }

        total_member = 16;
}
```

→ malloc 함수를 통해서 구조체의 동적 할당을 실행하는 함수

→ input_info 함수를 통해 회원들의 정보를 정리한다.
16명 회원의 정보를 동적 할당된 구조체 메모리에 할당합니다.

for문을 통해서

mi[i].age : I 번째 구조체의 age에 age[i] 정보를 입력.
ㄴ **나이의 경우 숫자 값 이기에 바로 대입.**

문자열 값의 경우 strcpy 함수를 사용하여 값을 복사한다.
아래는 strcpy의 함수 원형이며
예를 들어 **Strcpy(s1, s2) →s2의 문자열을 s1로 복사 한다는 의미**.

- **strcpy(대상문자열, 원본문자열);**
  - **char *strcpy(char *_Dest, char const *_Source);**
  - 대상문자열의 포인터를 반환

즉, 하나 문장을 예로 들어보면
strcpy(mi[i].phone, phone[i]);
ㄴ **phone배열의 값을 mi 구조체 I 번째의 phone 배열에 복사.**

1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

4) 10대, 20대, 30대 별로 출력해보도록 한다. (Answer)

```
void print_condition_member_info(member_info *mi, int num)
{
        int i;

        if (mi->name)
        {
                for (i = 0; i < total_member; i++)
                {
                        if ((mi[i].age / num) == 1)
                        {
                                printf("이름: %s, 나이: %d, 전화번호: %s\n거주지: %s시 %s %s\n",
                                        mi[i].name, mi[i].age, mi[i].phone,
                                        mi[i].city, mi[i].street, mi[i].detail);
                        }
                        else
                        {
                                continue;
                        }
                }
        }
}

void print_member_info(member_info *mi)
{
        int i;

        if (mi->name)
        {
                for (i = 0; i < total_member; i++)
                {
                        printf("이름: %s, 나이: %d, 전화번호: %s\n거주지: %s시 %s %s\n",
                                mi[i].name, mi[i].age, mi[i].phone,
                                mi[i].city, mi[i].street, mi[i].detail);
                }
        }
}
```

→ print condition 함수는 구조체 정보와 숫자 num을 받아
원하는 나이대의 정보를 출력하기 위한 함수 입니다.

여기서 int num 인자에 10을 받으면 10대 사람만 출력하게 되는 것 입니다.

여기서도 if 문 조건에 mi → name 이라는 값이 있으면이라는 의미가
적용 되었고, toal member의 숫자 만큼

for문을 반복한다.

두번째 if 문의 조건은
(mi[i].age / num) == 1 은 구조체 나이 값을 10으로 나누었을 때
몫이 1 인경우만 10대 이므로 **10대의 조건을 구하기 위함**이다.

→ print_member_info 함수는 10대의 조건 뿐만 아닌
구조체에 있는 전체의 값을 출력하는 함수이다.

1. Q2) 2 ~ 4

- 콘솔창에서 회원가입을 시키고자 한다.

4) 10대, 20대, 30대 별로 출력해보도록 한다. (Answer)

```c
int main(void)
{

        member_info *mi;

        mi = init_member_info();
        input_info(mi);

        printf("10대만 출력\n");
        print_condition_member_info(mi, 10);

        free(mi);
}
```

→ 최종적으로 main 문 동작을 다시 한번 살펴 보자.

- member_info * mi 구조체 포인터형 mi 를 선언
- mi 구조체에 동적메모리 할당
- input_info를 통해 구조체에 정보 입력
- 10대만 출력하는 출력 함수 사용
- 동적 할당 한 함수 free

```
10대만 출력
이름: 김윤환, 나이: 10, 전화번호: 010-1111-1234
거주지: 서울시
이름: 김택용, 나이: 11, 전화번호: 010-1111-1234
거주지: 서울시
이름: 이영호, 나이: 12, 전화번호: 010-1111-1234
거주지: 서울시
이름: 조기석, 나이: 15, 전화번호: 010-1111-1234
거주지: 부산시
이름: 임요환, 나이: 16, 전화번호: 010-1111-1234
거주지: 서울시
이름: 마주작, 나이: 18, 전화번호: 010-1111-1234
거주지: 대전시
```

→ 결과물 예시

```c
#include <stdio.h>

int arr[30] = { 1, 6, 2 };

int find_series(int num)
{
        int i;

        for (i = 3; i < num; i++)
        {
                if (i % 2)
                {
                        arr[i] = arr[i - 1] + arr[i - 3];
                        printf("홀 ");
                }
                else
                {
                        arr[i] = arr[i - 1] - arr[i - 3];
                        printf("짝 ");
                }

                printf("검토용 arr[%d] = %d\n", i, arr[i]);
        }

        return arr[num - 1];
}

int main(void)
{
        int res = find_series(20);

        printf("res = %d\n", res);
}
```

```
홀 검토용 arr[3] = 3
짝 검토용 arr[4] = -3
홀 검토용 arr[5] = -1
짝 검토용 arr[6] = -4
홀 검토용 arr[7] = -7
짝 검토용 arr[8] = -6
홀 검토용 arr[9] = -10
짝 검토용 arr[10] = -3
홀 검토용 arr[11] = -9
짝 검토용 arr[12] = 1
홀 검토용 arr[13] = -2
짝 검토용 arr[14] = 7
홀 검토용 arr[15] = 8
짝 검토용 arr[16] = 10
홀 검토용 arr[17] = 17
짝 검토용 arr[18] = 9
홀 검토용 arr[19] = 19
res = 19
```

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

#define MAX                     30

int rand_arr[MAX];

void init_rand_num(void)
{
        int i;

        for (i = 0; i < MAX; i++)
        {
                rand_arr[i] = rand() % 10 + 1;
        }
}

void print_rand_arr(void)
{
        int i;

        for (i = 0; i < MAX; i++)
        {
                printf("rand_arr[%d] = %d\n", i, rand_arr[i]);
        }
}

int main(void)
{
        srand(time(NULL));

        init_rand_num();

        print_rand_arr();

        return 0;
}
```

```
rand_arr[0] = 3
rand_arr[1] = 9
rand_arr[2] = 6
rand_arr[3] = 5
rand_arr[4] = 10
rand_arr[5] = 4
rand_arr[6] = 8
rand_arr[7] = 5
rand_arr[8] = 4
rand_arr[9] = 6
rand_arr[10] = 8
rand_arr[11] = 2
rand_arr[12] = 3
rand_arr[13] = 1
rand_arr[14] = 1
rand_arr[15] = 10
rand_arr[16] = 1
rand_arr[17] = 10
rand_arr[18] = 3
rand_arr[19] = 6
rand_arr[20] = 4
rand_arr[21] = 1
rand_arr[22] = 3
rand_arr[23] = 8
rand_arr[24] = 2
rand_arr[25] = 3
rand_arr[26] = 2
rand_arr[27] = 5
rand_arr[28] = 7
rand_arr[29] = 10
```

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

#define MAX                     10

int rand_arr[MAX];

bool dup_check(int num)
{
        int i;

        for (i = 0; i < num; i++)
        {
                if (rand_arr[i] == rand_arr[num])
                {
                        return true;
                }
        }

        return false;
}

void init_rand_num(void)
{
        int i;

        for (i = 0; i < MAX; i++)
        {
redo:
                rand_arr[i] = rand() % 10 + 1;

                if (dup_check(i))
                {
                        goto redo;
                }
        }
}
```

```c
void print_rand_arr(void)
{
        int i;

        for (i = 0; i < MAX; i++)
        {
                printf("rand_arr[%d] = %d\n", i, rand_arr[i]);
        }
}

int main(void)
{
        srand(time(NULL));

        init_rand_num();

        print_rand_arr();

        return 0;
}
```

```
rand_arr[0] = 10
rand_arr[1] = 9
rand_arr[2] = 6
rand_arr[3] = 7
rand_arr[4] = 3
rand_arr[5] = 4
rand_arr[6] = 8
rand_arr[7] = 1
rand_arr[8] = 2
rand_arr[9] = 5
```

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

int run_dice(void)
{
        return rand() % 6 + 1;
}

int main(void)
{
        srand(time(NULL));

        int dice = run_dice();

        printf("dice = %d\n", dice);

        return 0;
}
```

```
dice = 5
taein@tae
dice = 2
```

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

/* 물류센터에 짐이 들어온다.
이 물류 센터 3000평 정도의 면적을 가지고 있다.
여기에 각 물건을 박스화하여 배치한다.
박스의 크기는 28평정도이며 박스간의 이격 간격은 4평이다.
물류를 배치하는 가장 효율적인 방법을 프로그래밍하여 구현하시오. */

float calc_line(int area)
{
        return sqrt(area);
}

int get_num(float total_line, float sub_line)
{
        return total_line / sub_line;
}

int main(void)
{
        // 먼저 정육면체의 길이를 구하도록 한다.
        float row, col, box_row, box_col;
        int num;

        row = col = calc_line(3000);
        box_row = box_col = calc_line(32);

        num = get_num(row, box_row);

        printf("row = col = %f\n", row);
        printf("box_row = box_col = %f\n", box_row);
        printf("3000평에 배치된 박스는 모두 %d개다.\n", num * num);

        return 0;
}
```

```
taein@taein-Lenovo-ideapad-700-15ISK:~/proj/es02/Lv01-02/TaeinPark/first$ gcc -o an10 an10.c -lm
taein@taein-Lenovo-ideapad-700-15ISK:~/proj/es02/Lv01-02/TaeinPark/first$ ./an10
row = col = 54.772255
box_row = box_col = 5.656854
3000평에 배치된 박스는 모두 81개다.
```

```c
#include <stdio.h>

void conversion_big_to_small(char *str)
{
        int i;

        for (i = 0; str[i]; i++)
        {
                if (str[i] > 64 && str[i] < 92)
                {
                        str[i] ^= 0x20;
                }
        }
}

int main(void)
{
        char str[] = "WhErE ArE YOu FROM ?";

        printf("default: %s\n", str);

        conversion_big_to_small(str);

        printf("conversion: %s\n", str);
}
```

```
taein@taein-Lenovo-ideapad-700-15IS
default: WhErE ArE YOu FROM ?
conversion: where are you from ?
```

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

typedef struct _employee emp;

struct _employee
{
        char name[32];
        int pay;
};

char name[5][32] = { "이경수", "박지완", "강지훈", "김지환", "조민현" };

emp *init_employee(void)
{
        int i;

        emp *tmp = (emp *)malloc(sizeof(emp) * 5);

        for (i = 0; i < 5; i++)
        {
                strcpy(tmp[i].name, name[i]);
                tmp[i].pay = rand() % 501 + 3000;
        }

        return tmp;
}
```

```c
void print_employee(emp *e)
{
        int i;

        for (i = 0; i < 5; i++)
        {
                printf("name = %s, pay = %d\n", e[i].name, e[i].pay);
        }
}

void years_ago(emp *e, int year)
{
        int i, j;

        for (i = 0; i < year; i++)
        {
                printf("%d 년차\n", i + 1);

                for (j = 0; j < 5; j++)
                {
                        e[j].pay = e[j].pay + e[j].pay * (((rand() % 10) + 1) / 100.0);
                }

                print_employee(e);
        }
}
```

```c
emp find_max(emp *e)
{
        int i, max_idx, max = 0;

        for (i = 0; i < 5; i++)
        {
                if (max < e[i].pay)
                {
                        max = e[i].pay;
                        max_idx = i;
                }
        }

        return e[max_idx];
}

int main(void)
{
        emp max;

        srand(time(NULL));

        emp *e = init_employee();

        print_employee(e);

        years_ago(e, 10);

        max = find_max(e);
        printf("10년후 가장 연봉이 높은 사원은 %s, %d\n", max.name, max.pay);

        return 0;
}
```

```
name = 이경수, pay = 3144
name = 박지완, pay = 3020
name = 강지훈, pay = 3314
name = 김지환, pay = 3050
name = 조민현, pay = 3004
1 년차
name = 이경수, pay = 3301
name = 박지완, pay = 3140
name = 강지훈, pay = 3479
name = 김지환, pay = 3355
name = 조민현, pay = 3274
2 년차
name = 이경수, pay = 3532
name = 박지완, pay = 3391
name = 강지훈, pay = 3826
name = 김지환, pay = 3455
name = 조민현, pay = 3568
3 년차
name = 이경수, pay = 3637
name = 박지완, pay = 3492
name = 강지훈, pay = 4093
name = 김지환, pay = 3731
name = 조민현, pay = 3603
4 년차
name = 이경수, pay = 3964
name = 박지완, pay = 3771
name = 강지훈, pay = 4502
name = 김지환, pay = 3954
name = 조민현, pay = 3963
5 년차
name = 이경수, pay = 4281
name = 박지완, pay = 3959
name = 강지훈, pay = 4952
name = 김지환, pay = 4270
name = 조민현, pay = 4319
6 년차
name = 이경수, pay = 4495
name = 박지완, pay = 4117
name = 강지훈, pay = 5100
name = 김지환, pay = 4697
name = 조민현, pay = 4491
7 년차
name = 이경수, pay = 4584
name = 박지완, pay = 4364
name = 강지훈, pay = 5559
name = 김지환, pay = 5072
name = 조민현, pay = 4715
```

```
8 년차
name = 이경수, pay = 5042
name = 박지완, pay = 4582
name = 강지훈, pay = 5781
name = 김지환, pay = 5528
name = 조민현, pay = 5045
9 년차
name = 이경수, pay = 5193
name = 박지완, pay = 4627
name = 강지훈, pay = 5896
name = 김지환, pay = 5583
name = 조민현, pay = 5549
10 년차
name = 이경수, pay = 5296
name = 박지완, pay = 4673
name = 강지훈, pay = 6308
name = 김지환, pay = 5638
name = 조민현, pay = 5937
10년후 가장 연봉이 높은 사원은 강지훈, 6308
```

```c
#include <math.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct _employee emp;

struct _employee
{
        char name[32];
        int pay;
};

typedef struct _statistics stat;

struct _statistics
{
        float mean;
        float std_dev;
};

char name[5][32] = { "이경수", "박지완", "강지훈", "김지환", "조민현" };

emp *init_employee(void)
{
        int i;

        emp *tmp = (emp *)malloc(sizeof(emp) * 5);

        for (i = 0; i < 5; i++)
        {
                strcpy(tmp[i].name, name[i]);
                tmp[i].pay = rand() % 501 + 3000;
        }

        return tmp;
}
```

```c
void print_employee(emp *e)
{
        int i;

        for (i = 0; i < 5; i++)
        {
                printf("name = %s, pay = %d\n", e[i].name, e[i].pay);
        }
}

float calc_mean(emp *e, int num)
{
        int i;
        float sum = 0;

        for (i = 0; i < num; i++)
        {
                sum += e[i].pay;
        }

        return sum / (float)num;
}

float calc_std_dev(emp *e, int num, float mean)
{
        int i;
        float sum = 0;

        for (i = 0; i < num; i++)
        {
                sum += pow((e[i].pay - mean), 2);
        }

        return sum / num;
}
```

```c
void record_statistics(emp *e, int idx, stat *s)
{
        int i;
        float mean = calc_mean(e, 5);

        s[idx].mean = mean;
        s[idx].std_dev = sqrt(calc_std_dev(e, 5, mean));
}

void print_statistics(stat *s, int idx)
{
        printf("평균 = %f, 표준편차 = %f\n", s[idx].mean, s[idx].std_dev);
}

void years_ago(emp *e, int year, stat *s)
{
        int i, j;

        for (i = 0; i < year; i++)
        {
                printf("%d 년차\n", i + 1);

                for (j = 0; j < 5; j++)
                {
                        e[j].pay = e[j].pay + e[j].pay * (((rand() % 10) + 1) / 100.0);
                }

                record_statistics(e, i, s);

                print_employee(e);
                print_statistics(s, i);
        }
}
```

```c
emp find_max(emp *e)
{
        int i, max_idx, max = 0;

        for (i = 0; i < 5; i++)
        {
                if (max < e[i].pay)
                {
                        max = e[i].pay;
                        max_idx = i;
                }
        }

        return e[max_idx];
}

stat *init_statistics(void)
{
        stat *tmp = (stat *)malloc(sizeof(stat) * 10);

        return tmp;
}

int main(void)
{
        stat *s;
        emp max;

        srand(time(NULL));

        s = init_statistics();
        emp *e = init_employee();

        print_employee(e);

        years_ago(e, 10, s);

        max = find_max(e);
        printf("10년후 가장 연봉이 높은 사원은 %s, %d\n", max.name, max.pay);

        return 0;
}
```

```
$ gcc -o an19 an19.c -lm
$ ./an19
```

```
name = 이경수, pay = 3472
name = 박지완, pay = 3361
name = 강지훈, pay = 3496
name = 김지환, pay = 3477
name = 조민현, pay = 3341
1 년차
name = 이경수, pay = 3506
name = 박지완, pay = 3394
name = 강지훈, pay = 3530
name = 김지환, pay = 3824
name = 조민현, pay = 3608
평균 = 3572.399902, 표준편차 = 143.250259
2 년차
name = 이경수, pay = 3751
name = 박지완, pay = 3631
name = 강지훈, pay = 3777
name = 김지환, pay = 4015
name = 조민현, pay = 3896
평균 = 3814.000000, 표준편차 = 131.112167
3 년차
name = 이경수, pay = 3938
name = 박지완, pay = 3994
name = 강지훈, pay = 3965
name = 김지환, pay = 4336
name = 조민현, pay = 3934
평균 = 4033.399902, 표준편차 = 152.826172
4 년차
name = 이경수, pay = 3977
name = 박지완, pay = 4233
name = 강지훈, pay = 4361
name = 김지환, pay = 4596
name = 조민현, pay = 4170
평균 = 4267.399902, 표준편차 = 205.786880
5 년차
name = 이경수, pay = 4215
name = 박지완, pay = 4486
name = 강지훈, pay = 4491
name = 김지환, pay = 5055
name = 조민현, pay = 4587
평균 = 4566.799805, 표준편차 = 273.782684
6 년차
name = 이경수, pay = 4552
name = 박지완, pay = 4575
name = 강지훈, pay = 4625
name = 김지환, pay = 5105
name = 조민현, pay = 4770
평균 = 4725.399902, 표준편차 = 204.378662
7 년차
name = 이경수, pay = 4597
name = 박지완, pay = 4620
name = 강지훈, pay = 4856
name = 김지환, pay = 5156
name = 조민현, pay = 4817
평균 = 4809.200195, 표준편차 = 201.666458
```

```
8 년차
name = 이경수, pay = 4780
name = 박지완, pay = 5035
name = 강지훈, pay = 5293
name = 김지환, pay = 5671
name = 조민현, pay = 4961
평균 = 5148.000000, 표준편차 = 309.100647
9 년차
name = 이경수, pay = 5162
name = 박지완, pay = 5387
name = 강지훈, pay = 5451
name = 김지환, pay = 5784
name = 조민현, pay = 5159
평균 = 5388.600098, 표준편차 = 229.913559
10 년차
name = 이경수, pay = 5420
name = 박지완, pay = 5494
name = 강지훈, pay = 5505
name = 김지환, pay = 6015
name = 조민현, pay = 5623
평균 = 5611.399902, 표준편차 = 212.021317
10년후 가장 연봉이 높은 사원은 김지환, 6015
```

질문) 1

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

void input_info(char *name, int *age, char *phone, char *city, char *street, char *detail)
{
        printf("회원 정보를 입력해주세요.\n이름: ");
        scanf("%s", name);

        printf("나이: ");
        scanf("%d", age);

        printf("전화 번호: ");
        scanf("%s", phone);

        printf("도시: ");
        scanf("%s", city);

        printf("도로명: ");
        scanf("%s", street);

        printf("상세 주소: ");
        scanf("%s", detail);
}
```

여기에서 retrun name 이라던가
Return age 라던가를 하지 않아도
각각의 변수의 값이 main으로 넘어 갈 때
반환 되는 이유가 있나요??

질문) 2

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

typedef struct _member_info member_info;

struct _member_info
{
        int age;
        char name[32];
        char phone[32];
        char city[32];
        char street[128];
        char detail[128];
};

int total_member;

member_info *init_member_info(void)
{
        member_info *tmp = (member_info *)malloc(sizeof(member_info) * 64);
}
```

여기에서 malloc을 사용 할 때
tmp를 따로 선언하지 않은 이유는
Member info 포인터형인 형태인 tmp가 생성과 동시에 malloc이
실행 되기 때문 인가요??

그리고 sizeof 뒤에 x 64를 한 이유가 있을 까요??