C언어 - HW7

임베디드스쿨2기
Lv1과정
2021. 04. 29
차현호

시험문제 풀이

1. C언어는 함수를 홀출할 때마다 무엇을 생성하는가?
어째서 재귀호출이 일반적인 Loop 보다 성능이 떨어지는 것인가?
이에 대해 상세히 기술하시오

나의 풀이

c언어 함수를 호출할 때마다 콜스택에다가 함수가 종료 된 후 리턴될 주소를 저장한다.
어셈블리어로 하면 아래와 같은 연산이 추가된다.

(인텔 cpu 기준)

```
push %rbp
mov %rsp, %rbp
```

이렇게 계속해서 콜스택에다가 데이터를 쌓아주는 연산이 진행 되므로 일반적인 loop문 보다 성능이 저하된다.

정답

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)

시험문제 풀이

1. 오답정리 및 요약

c언어에서는 함수를 호출할때마다 Stack Fram을 만들고 해제하는 코드가 들어가므로 Loop보다 효율이 안좋아진다.

실제로 그런지 테스트해보자.
아래 코드는 똑같이 넘을 199999번 +1씩하는 코드이다.

```c
void recursive_add_num(int *num)
{
    if ((*num) > 199999)    return;

    (*num)++;

    recursive_add_num(num);
}


int main()
{
    int num;

    recursive_add_num(&num);

    return 0;
}
```

```c
int main()
{
    int i;
    int num = 0;

    for (i = 0; i < 199999; i++)
    {
        num++;
    }

    return 0;
}
```

시험문제 풀이

1. 오답정리 및 요약

두코드의 실행시간을 비교해보면 실제 재귀함수 호출하는 코드가 2배이상 느린것을 확인할 수 있다.

```
chh@chh-15ZD970-GX7SK:~/proj/es02/Lv01-02/HyunhoCha/homework/c/07$ time ./recur
ive

real    0m0.015s
user    0m0.010s
sys     0m0.005s
```

```
chh@chh-15ZD970-GX7SK:~/proj/es02/Lv01-02/HyunhoCha/homework/c/07$ time ./loop

real    0m0.005s
user    0m0.001s
sys     0m0.004s
```

시험문제 풀이

2. 콘솔창에서 회원가입을 시키고자 한다.

회원 이름, 나이, 전화번호, 거주지를 입력 받도록 만든다.

정답

```c
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);
}
```

```c
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);
}
```

시험문제 풀이

2.

정답

```
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);
}
```

시험문제 풀이

2. 오답정리 및 요약

먼저 main문을 살펴보면 입력 데이터인 사용자의 나이, 이름, 전화번호, 거주지(도시이름, 상세주소)들의 데이터를 각각 따로 선언한것을 볼 수 있으며

Input_info함수의 인자로 위의 데이터들을 넘겨주어 scanf로 입력받아 데이터를 채우는 것을 확인 할 수 있다.

시험문제 풀이

3. 콘솔창에서 회원가입을 시키고자 한다.

적당히 여러 회원을 입력한 이후 거자주가 같은 사람들만 출력해본다.

정답

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

        mi = init_member_info();
        input_info(mi);

        print_member_info(mi);

        free(mi);
}
```

```c
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;
```

시험문제 풀이

3. 콘솔창에서 회원가입을 시키고자 한다.

적당히 여러 회원을 입력한 이후 거자주가 같은 사람들만 출력해본다.

정답

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

```
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);
                }
        }
}
```

시험문제 풀이

3. 오답정리 및 요약

2번문제와의 큰차이는 3번풀이에서는 각각 따로 선언된 변수들을 구조체로 묶어서 _member_info 타입으로 정의한 것이다.
그리고 typef를 사용해 _member_info 변수를 member_info로 재정의 하였다.

main문을 살펴보면 init_member_info 함수를 먼저 불러오는데 이 함수는 member_info데이터 타입을 가지는 메모리 공간을 64개 동적할당해준다.

그리고 input_info 함수에서 사용자 정보를 입력받고 print_member_info 에서 회원 정보를 출력해준다.

시험문제 풀이

4. 콘솔창에서 회원가입을 시키고자 한다.

10대 20대 30대 별로 출력해보도록 한다.

```
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;
                        }
                }
        }
}
```

이전슬라이드까지 작성된 코드에서 회원정보를 입력받은뒤 입력받은 구조체 배열의 데이터중 나이 변수와 num(10 or 20 or 30) 값을 나누어 주어 몫이 1이면 출력하는 코드가 추가되었다.

몫이 1이라는것은 num이 10이면 10~19 20이면 20~29 30이면 30~39를 나타내므로
올바른 판별이라는 것을 알 수 있다.

시험문제 풀이

5. 구조체를 사용하는 이유는 무엇인가?

나의 풀이

구조체를 사용하는 이유

서로다른 데이터타입을 패키지처럼 한번에 묶어서 사용할때 편리하기 때문에 사용합니다.
가령 2~4번 문제에서처럼 사용자 정보를 저장하는데 있어서 구조체를 사용하지 않으면
각각 따로 데이터 타입변수를 선언해주고 관리해줘야해서 큰 불편함이 발생합니다.

정답

구조체를 사용하는 이유는 앞선 문제에서와 같이
여러개의 데이터를 한 번에 묶어서 처리하고자 할 때 유용하다.
일일히 데이터를 모두 파라미터로 넘기는 것은 너무 불편하고 관리하기에도 힘들다.
반면 구조체로 관리하면 어떤 목적으로 사용하는지가 보다 명시적이 된다.

시험문제 풀이

6. 1 6 2 3 -3 -1 -4 -7 -6 -10
위 숫자 나열에서 20번째 숫자를 구하시오

나의 풀이

```
int find_number(int num1, int num2, int num3)
{
        int i;
        int n1 = num1;
        int n2 = num2;
        int n3 = num3;

        int number = 0;

        for (i = 1; i <= 17; i++)
        {
                if(i % 2 == 0)
                {
                        number = n3 + (n1 * -1);
                }

                else
                {
                        number = n3 + n1;
                }

                n1 = n2;
                n2 = n3;
                n3 = number;
        }

        return number;
}
```

정답

```
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];
}
```

시험문제 풀이

6. 오답정리 및 요약

6번문제는 피보나치 수열의 변형 문제이다. 규칙을 찾으면 바로 풀리는데 규칙은
처음 3개의 항 a1,a2,a3가 있을 때 a4 = a1 + a3 a5 = a2 – a4 와같이

짝수항일때는 바로 전항과 전전전항과 더하는것이고 홀수항일때는 빼는것이 규칙이다.

시험문제 풀이

7. 1 ~ 10 사이의 숫자 30개를 생성하시오.

나의 풀이

```c
void make_random_numbers(int *numbers, int len)
{
	int i;

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

void print_random_numbers(int *numbers, int len)
{
	int i;

	for (i = 0; i < len; i++)
	{
		printf("%3d", numbers[i]);
	}

	printf("\n");
}

int main(void)
{
	int numbers[30] = { 0 };
	int len = sizeof(numbers) / sizeof(int);

	srand(time(NULL));

	make_random_numbers(numbers, len);

	print_random_numbers(numbers, len);

	return 0;
}
```

정답

```c
#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;
}
```

시험문제 풀이

7. 오답정리 및 요약

Rand함수를 이용해서 1 ~ 10 사이의 숫자30개를 만들어 낸다.

rand() % 10 + 1 을하면 rand() % 10 모듈러 연산을 통해 0~9사이의 값이 나오고 더하기 1을 하여 1 ~ 10 사이의 값을 구한다.

시험문제 풀이

8. 1 ~ 10 사이의 숫자를 중복되지 않게 10개 생성하시오

나의 풀이

```
void make_random_numbers(int *numbers, int* number_check, int len)
{
        int i;
        int random;

        for(i = 0; i < len; i++)
        {

                random = rand() % 10 + 1;

                while(number_check[random - 1] != 0)
                {
                        random = rand() % 10 + 1;
                }

                numbers[i] = random;
                number_check[random - 1] = 1;
        }
}
```

정답

```
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;
                }
        }
}
```

시험문제 풀이

8. 오답정리 및 요약

나의 풀이 방법 : number_check 배열을 한개 더할 당하여 받아온 랜덤숫자를 인덱스로 활용하여 해당인덱스가 false면 true로 변환 만약 true면 다시한번 랜덤숫자를 받아와서 체크

정답 : 랜덤숫자를 받아온 후 현재 저장되어있는 숫자들과 비교하여 중복되면 다시 생성

시험문제 풀이

9. 주사위를 굴려서 나온 숫자를 출력해보자.

나의 풀이

```
void dice_game()
{
        int number = rand() % 6 + 1;

        printf("주사위 결과 : %d \n", number);
}
```

정답

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

시험문제 풀이

9. 오답정리 및 요약

rand() % 6 + 1 을 통해 1 ~ 6 사이의 랜덤숫자를 출력한다.

시험문제 풀이

11. If문 어셈블리의 특성을 상세히 서술하시오.

나의 풀이

if문 어셈블리 설명

if문을 컴파일하여 어셈블리를 살펴보면
아래와 같은 연산을 반복하는것을 확인 할 수 있다.

cmp(레지스터1, 레지스터2)
jle,je,ja (분기할 주소)

이를 분석해보면 먼저 cmp 명령어를 통해 두개의 레지스터를 뺀다 뺀 결과 값을 토대로
eflags 라는 레지스터의 비트(ZF, CF)를 업데이트한다.

eflags 레지스터를 보고 jle,je 등의 조건을 확인한뒤 정해진 주소로 분기한다.

정답

if 문의 경우엔 mov로 비교할 대상을 가져와서 cmp를 통해 비교를 수행한다.
이후 EFLAGS 레지스터나 비교 연산의 결과를 가지고 분기를 결정하는 형식으로 구동된다.
결국 mov, cmp, jmp 형식으로 구성된다.

시험문제 풀이

12. 배열 작성시 나타나는 어셈블리의 특성을 상세히 서술하시오.

나의 풀이

변수의 정의는 특정한 데이터타입을 가지는 데이터를 저장하는 메모리 공간이다.
이와 관련하여 배열은 "연속된" 특정한 데이터타입을 가지는 데이터를 저장하는 메모리 공간이다.
여기서 중요한 부분은 "연속된"인데 실제 배열을 선언하여 메모리 공간 할당 주소들을 보면 데이터 타입의 크기차이만큼 연속되게 할당된것을 확인할 수 있다.

가령
크기가 2인 int 형 배열이 있다고 가정할때

array[0]의 주소가 = 0x0 이라면
array[1]의 주소는 = 0x4 이다.

또한 배열의 변수명은 그배열의 시작주소를 가리키고 있다.

정답

```
0x0000555555555184 <+27>:    movl   $0x1,-0x14(%rbp)
0x000055555555518b <+34>:    movl   $0x2,-0x10(%rbp)
0x0000555555555192 <+41>:    mov    -0x10(%rbp),%edx
0x0000555555555195 <+44>:    mov    -0x14(%rbp),%eax
0x0000555555555198 <+47>:    mov    %edx,%r8d
0x000055555555519b <+50>:    mov    $0x1,%ecx
0x00005555555551a0 <+55>:    mov    %eax,%edx
0x00005555555551a2 <+57>:    mov    $0x0,%esi
0x00005555555551a7 <+62>:    lea    0xe56(%rip),%rdi        # 0x555555556004
0x00005555555551ae <+69>:    mov    $0x0,%eax
0x00005555555551b3 <+74>:    callq  0x555555555070 <printf@plt>
```

배열의 시작 주소를 edx에 배치한다.
r8d의 경우엔 r8 레지스터의 32비트 표현이다.
int형 자료이므로 순차적으로 4바이트씩 공간에 배치하는 것을 볼 수 있다.
데이터 자체에 접근할 때도 edx를 기준으로 활용함을 볼 수 있다.
즉, 배열 사용시에는 dx 레지스터가 활용됨을 볼 수 있다.

시험문제 풀이

12. 오답정리 및 요약

배열 사용시 edx레지스터가 사용된다는 것을 확인 할 수 있다.

시험문제 풀이

13. 반복문 작성시 나타나는 어셈블리의 특성을 상세히 서술하시오.

나의 풀이

반복문 작성시 나타나는 어셈블리의 특성은 이전 문제에 나왔던 if문의 동작과 유사하다.

```
int i;

for(i=0; i < 10; i++)
```

위의 코드를 예로들면 먼저 for문을 진입할때

cmp(10, i) 같이 i 값과 10을 cmp문을 통해 비교한다.

그다음

jne(같지 않으면 분기)같은 분기문을 통해 분기한후 for문안의 로직을 실행한다.

그 후 i값을 한개 더하여 다시한번 cmp(10, i)와 같은 비교를 하고 조건분기문을 실행한다.

반복문은 위와같이 반복한다.

정답

반복문 작성시 if 문과 마찬가지로 mov, cmp, jmp로 구성된다.
다만 반복을 하기 위해 다시 초기 위치로 되돌리는 jmp와
반복을 탈출하기 위한 jmp로 구성된다.
if와의 차이점이라면 이와 같이 jmp가 두 개 존재한다는 것이다.

시험문제 풀이

14. 함수를 호출할 때 발생하는 어셈블리의 특성을 상세히 서술하시오.

나의 풀이

함수 호출시 아래와 같은 어셈블리가 반복된다.

```
push %rbp
mov %rsp, %rbp
```

분석해보면 먼저 호출하는 함수의 base 스택 주소를 스택에 저장한후에 함수를 분기한 후 base 스택레지스터 값을 현재 스택 최상단 주소로 바꿔 준다.

정답

함수를 호출할 때 나타나는 특성은 call이다.
이 call의 경우엔 push + jmp를 수행하므로
push에서 Stack에 복귀 주소를 저장하며 jmp에서는 특정 주소로 이동을 하게 된다.
뿐만 아니라 함수 호출시에는 반드시 스택 프레임을 형성하기 위해
push와 mov가 함께 세트로 움직인다.
또한 마지막에 스택을 해제하기 위해 pop과 ret가 동작하게 되어 있다.

시험문제 풀이

14. 오답정리 및 요약

1번문제에 설명이 나와있듯이 c언어는 함수를 호출하면 Stak Fram을 만들었다가 리턴할때 해제한다.

Stack Fram을 만드는 부분은 push %rbp mov %rsp,%rbp 이고 해제하는 부분은 pop, ret이다.

시험문제 풀이

15. 프로그래머가 반드시 알아야 하는 가상메모리 섹션 4가지를 서술하고 각각의 섹션 4가지에 대해 상세히 기술하시오.

나의 풀이

가상메모리 섹션의 4가지는 다음과 같다.

stack

---

heap

---

data

---

text

---

stack : 함수안에서 선언하는 지역 변수들(동적할당제외), 함수 호출시 돌아올 주소등을 저장하는 메모리공간이며 컴파일 실행시 크기가 결정된다.

heap : 동적할당(malloc)시 할당시 할당되는 공간이며 프로그램이 실행중에 할당된다. 따라서 동적할당을 많이 사용할시 성능 저하의 우려가 있다.

data : 초기화된 전역 변수, static 변수등의 할당시 사용되는 메모리 공간이다. 초기화가 안된 변수는 bss 라는 공간에 할당된다.

text : 우리가 작성한 함수의 주소 저장된 공간이다.

정답

스택(Stack): 지역 변수가 배치된다.
힙(Heap): malloc, calloc등의 동적할당된 데이터가 배치된다.
데이터(Data): 전역 변수 및 static 변수가 배치된다.
텍스트(Text): 기계어 및 함수가 배치된다.

시험문제 풀이

16. 배열에 대문자로 작성된 문장을 소문자로 변경하여 출력해보세요

정답

```c
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);
}
```

시험문제 풀이

16. 오답정리 및 요약

입력 받은 문자 배열을 처음부터 끝까지 순회하면서 문자가 만약 문자값이 64 초과 92 미만이라면 0x20과 xor 연산을 해줘 소문자로 변환해준다.

값 65 ~ 91 은 아스키코드로 A ~ Z에 해당된다.

시험문제 풀이

17. 리눅스를 사용하며 알고 있는 명령어를 5가지 이상 작성하고 각각의 명령어 기능에 대해 기술 하세요.

나의 풀이

리눅스 명령어

ls : 현재 디렉토리안의 파일 및 디렉토리를 보여주는 명령어이다.
 -a : 숨김파일을 포함하여 모든 파일 및 디렉토리를 보여준다.
 -R : 현재 위치부터 하위 디렉토리들의 파일을 보여 준다.

cd : 디렉토리를 이동한다.
ex)
cd ~/ : home 디렉토리로 이동
cd .. : 상위 디렉토리로 이동

mkdir : 디렉토리를 만드는 명령어이다.
 -p : 하위디렉토리를 만들며 실제 이름에 해당하는 디렉토리가 없으면 새로 생성해준다.

pwd : 디렉토리의 위치를 보여준다.

cp : 복사명령어이다.
 -r : 디렉토리 복사를 할때 사용하는 옵션이다.

정답

ls: 현재 디렉토리 위치에서 파일 리스트를 보는 것
pwd: 현재 디렉토리 위치 보기
mkdir: 디렉토리 만들기
cp: 복사
mv: 이름 바꾸기 및 위치 옮기기
rm: 삭제
gcc: 컴파일러
vi: 편집기

시험문제 풀이

18. 사원이 5명 있다. 5명의 사원의 초봉은 3000 ~ 3500이다. 이들에게 연마다 1 ~10%의 랜덤한 임금상승폭을 적용한다. 10년후 가장 연봉이 높은 사원의 임금과 이름은?

나의 풀이

```
bool init_salary(int *employee, int len)
{
        int i;

        if (employee == NULL)
        {
                return false;
        }

        for (i = 0; i < len; i++)
        {
                employee[i] = START_SALARY_MIN + (rand() % ((START_SALARY_MAX - START_SALARY_MIN) + 1));
        }

        return true;
}
```

정답

```
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;
}
```

시험문제 풀이

18. 사원이 5명 있다. 5명의 사원의 초봉은 3000 ~ 3500이다. 이들에게 연마다 1 ~10%의 랜덤한 임금상승폭을 적용한다. 10년후 가장 연봉이 높은 사원의 임금과 이름은?

나의 풀이

```
bool increas_salary(int *employee, int len)
{
	int i,j;
	float increase_rate;

	if(employee == NULL)
	{
		return false;
	}

	for (i = 0; i < YEAR; i++)
	{
		for (j = 0; j < len; j++)
		{
			increase_rate = (float) ((rand() % INCREASE_RATE_MAX) + INCREASE_RATE_MIN) / 100;
			employee[j] = employee[j] + (employee[j] * increase_rate);
		}
	}

	return true;
}
```

정답

```
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);
	}
}
```

시험문제 풀이

18. 사원이 5명 있다. 5명의 사원의 초봉은 3000 ~ 3500이다. 이들에게 연마다 1 ~10%의 랜덤한 임금상승폭을 적용한다. 10년후 가장 연봉이 높은 사원의 임금과 이름은?

나의 풀이

```
int find_top_salary(int *employee, int len)
{
        int i;
        int max = 0;
        int max_index = 0;

        if (employee == NULL)
        {
                return -1;
        }

        for (i = 0; i < len; i++)
        {
                if (max < employee[i])
                {
                        max = employee[i];
                        max_index = i;
                }
        }

        return max_index;
}
```

정답

```
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];
}
```

시험문제 풀이

18. 오답정리 및 요약

코드는 크게 세가지 부분으로 나눠서 볼 수있었다.

1번째는 1년차 연봉을 init하는 부분이다. 나의 풀이와 답의 차이는 직원변수 배열의 동적 할당 유무에 일단 있으며 랜덤넘버 할당시 답처럼 rand() % 501 + 3000로 깔끔하게 처리하지 못해 아쉬웠다.

2번째 부분은 연봉을 인상하는 부분의 코드인데 한줄로 써도되는것을 두줄로써서 코드가 깔끔하지 못해다.

마지막으로 3번째는 가장 연봉이 높은 직원을 찾아내는 알고리즘으로 알고리즘 원리는 비슷하나 나의 코드의 경우 해당하는 인덱스를 리턴하고 답의 경우는 해당되는 구조체 포인터를 반환하는것을 확인할 수 있었다.

시험문제 풀이

19. 사원이 5명 있다. 5명의 사원의 초봉은 3000 ~ 3500이다. 이들에게 연마다 1 ~10%의 랜덤한 임금상승폭을 적용한다. 10년후 5명의 연도별 평균과 표준편차를 출력하시오.

정답

```
typedef struct _statistics stat;

struct _statistics
{
        float mean;
        float std_dev;
};
```

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

        return tmp;
}
```

시험문제 풀이

19. 사원이 5명 있다. 5명의 사원의 초봉은 3000 ~ 3500이다. 이들에게 연마다 1 ~10%의 랜덤한 임금상승폭을 적용한다. 10년후 5명의 연도별 평균과 표준편차를 출력하시오.

정답

```
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));
}
```

```
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;
}
```

시험문제 풀이

19. 오답정리 및 요약

이문제에서는 평균과 표준편차를 구해야하는데 일단 표준편차의 개념부터 살펴보면 다음과같다.

표준편차란 자료들이 전체적으로 어떻게 분포되 었는지 나타내는 수치로 분산의 양의 제곱근으로 정의된다. 즉 표준편차가 작을수록 정규분포그래프에서 그래프모양이 첨예해진다.

표준편차를 구하려면 먼저 분산을 구해야하는데 분산이란 주어진 자료의 데이터가 평균을 기준으로 어떻게 흩어져있는지 나타내는 지표이다.

분산을 구하는 방법은 다음과 같다.

1. 평균 구하기
2. 평균과 각각의 데이터를 뺀다.
3. 2번의 결과값을 제곱한다.
4. 제곱한 값들을 전부더한다.
5. 총합을 데이터갯수 - 1 한값으로 나눈다.

1~5를 통해 구해진 분산값에 루트를 취해주면 표준편차가 나온다.

시험문제 풀이

19. 오답정리 및 요약

앞의 과정을 코드로 보면 아래부분과 같다.

```
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));
}


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;
}
```

위의 코드는 표준편차를 구하는 코드이므로 calc_std_dev함수에서 구해진 분산값을 루트를 취해줘 표준편차 값을 얻는다.

아래코드는 분산을 얻는 연산이며 앞의 calc_mean함수를 통해 구한 평균값과 각 요소들의 값을 빼주고 제곱하여 모두 더한뒤 num으로 나눠준 것을 확인 할 수 있다.