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)

```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);
}
```

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

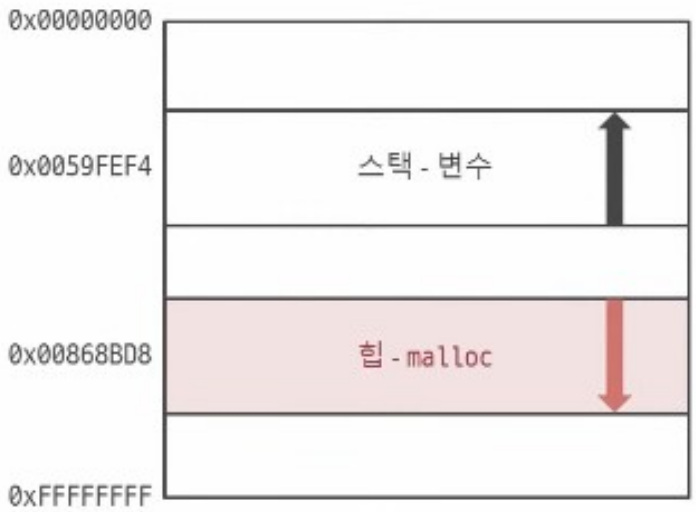
→ malloc 사용법에 대해 알아 보자.
메모리를 사용하기 위해 Malloc 함수로 사용할 메모리의 공간을 확보해야 합니다.
- 헤더는 stdlib.h 에 선언되어 있다.
- 원하는 시점에 원하는 메모리를 할당 할 수 있다고 해서 '동적 할당' 방법이라고도 한다.

- 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)



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??