TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

[1 차 시험(재시험)]

강사 - 이상훈
gcccompil3r@gmail.com
학생 – 이우석
colre99@naver.com

[자료구조]

1. 이번 시험 결과와 관련하여 자료구조 과목에 대한 자기 성찰을 해보자!
자기 성찰과 앞으로의 포부를 기술하시오.
(자기 성찰을 자기 성찰 답게 대충적지 말고 진심과 성심을 다해 적도록 한다)
답:

자기성찰 부분: 자기성찰을 해보자면..우선, 코딩연습이 너무나도 부족하였고, C 언어 학습에 대한 공부가 많이 부족하였다. 그렇다 보니 코딩연습시 구현도 잘 안될뿐더러 응용에서는 더더욱 부족하였다. 연습이 굉장히 부족하다고 느꼈으며 이론 부분도 부족하니 다양한 시도에서 제한적 임을 알게되었다. 현재 수업진행에 따라가기에 많은 어려움이 있지만, 내가 이 과정을 신청한 이상, 다른 분들이 이 수업에 들이는 시간의 배로는 공부해야 그마나 따라갈까 하고 생각한다. 나에게 현재 직면한 문제는 매일매일 수업나가는 부분을 하루하루 익히고 그 전의 이해못했던 수업과정을 내 스스로 시간을 내어 익히고 연습할 수 밖에 없다. 사실, 현재로서 설상가상 공업수학,

기초전자회로에 대한 지식이 없어 이 부분들도 익혀야 하는데 잠 다잤다 생각하고 매일 밤새며 기초공부를 할 수밖에 없다. 심지어 이미 전공자 분들도 나보다 더 열심히 하고 있어서 깜짝놀랬다. 임베디드 과정이 시작되고 나서 한달동안 학원에 남으면서 까지 하는 노력이 없었는데 너무나 큰 후회를 하는중이다. 이제 막 학원에 남으면서 하는데 다른분들은 매일매일 열심히 하시는걸 보니 내가 한달동안 노력하지 않은 부분이 크게 다가옴을 느꼈다. 마음가짐이 나약했고 안일하게 생각했다는 스스로의 반성이 들고있다. 더이상의 큰 후회를 또 하지 않도록 마음과 행동부터 잡아야 겠다는 생각을 하였다.

앞으로의 포부: 앞으로의 포부는 처음 이 과정을 신청한 이유가 나의 꿈이자 나의 진로인 드론을 시작하기 위해서 였는데, 이 과정을 열심히, 그리고 제대로 익혀서 나의 첫작품 드론을 만들 수 있도록 수업받은 이 전의 과정, 그리고 앞으로 남은 과정을 들을 하루하루 성실히 익히고 연습하며 오로지 드론완성을 위해 많은시간을 투자할 것이다.
강사님께 더이상 더이상 시키고 싶지도 않고, 팀원들에게 앞으로 단순히 폐가 되지 않게를 떠나, 같이 더욱 힘이 낼 수 있도록 개인의 역량을 늘려갈 계획이다. 확실히 자료구조 부분이든 어느 부분이든 그 부분들에 대한 연습이 많이 필요하다는걸 느꼈기에 먼저,

수업을 따라갈수 있도록 한다음, 차차 수업때 연습도 가능하게 연습하고 , 그 이후는 안정적으로 습득한 것을 자유로운 표현으로 할 수있게 할 것이다. 당장의 앞으로의 포부는 강사님께서 말씀하신 것처럼 수업이후 나머지에 남아서 하루하루의 수업을 따라갈수 있도록 보충해 나가는것이며, 수업과 별개로 출결 및 수업참여에도 지장이 없도록 지각과 결석을 최대한 안하도록 하는것이 포부에 대한 실천의 첫걸음이라고 생각된다.

2. 연결리스트에 대한 문제.

선입선출을 따르는 구조를 무엇이라 하는가?

답: Queue (큐)

3. 연결리스트 문제다.

후입선출 기능을 가진 자료구조를 무엇이라 하는가?

답: Stack(스택)

4. 트리에 관련한 개념 문제

트리 자료구조는 연결리스트에 비해 어떠한 이점을 가질 수 있는가?

답: 접근이 용이하여 자료의 삽입, 삭제, 검색이 빨라진다.

5. AVL 트리 개념

AVL 트리와 일반 트리의 차이점이 무엇인가?

답: 일반트리는 한쪽으로 쏠려, 원하는 자료를 찾으려면 루트부터 끝까지 찾아봐야 하는데 반면에 AVL 트리는 이진검색 트리이면서 동시에 균형을 유지하는 트리이다.

AVL 트리는 모든 노드의 왼쪽과 오른쪽 서브트리의 높이차이가 1 이하이기에, 이진 검색 시에 효율성을 보장할 수 있다. (삽입과 삭제 할때 트리의 높이가 달라지게 되는데 만약 어떠한 노드라도 균형치가 2 이상 일경우 RR, LL, RL, LR 의 4 가지 방식을 이용하여 회전을 시킨다.)

6. 레드 블랙 트리는 왜 사용하는가?

기존 트리들과 레드 블랙 트리의 차이점은 무엇인가?

답: 이진 탐색 트리 중에 값이 편향되게 들어오는 경우가 있는데, 그럴경우 이진 탐색 트리의 검색 효율을 나쁘게 하므로, 균형을 바로잡기 위해 레드 블랙 트리를 사용한다.

레드블랙 트리는 기존 트리들의 특성을 모두 가지면서 그외 다섯가지 특성을 추가적으로 가지고 있다.

1. 트리의 모든 노드는 검정색 아니면 빨간색이다.
2. 루트 노드는 무조건 검정색이다.
3. 모든 잎 노드는 검정색이다.
4. 빨간색의 노드 자식들은 모두 검정색이지만, 검정색 노드 자식들은 어느 색깔이든 상관없다.
5. 루트 노드에서 모든 잎 노드 사이에 있는 검정색 노드의 수는 모두 동일하다.

[임베디드 애플리케이션 분석]

1. 한달간 C 언어 등등 많은 것들을 학습했는데 이것들에 대해 자기 성찰을 수행 해보자! 해당 과목에 대한 진심과 성의가 담긴 자기 성찰을 수행하도록 한다. (앞으로의 포부 포함)
답: 자료구조 1 번에 작성해 놓은것과 같이, 자기성찰을 해보면 절대적으로 내 스스로 수업에 대한 열의와 노력 실천이 정말 부족하였다는 것을 알았다. 해당과목이 사실 기초자들을 위함이 아닌 기본적으로 전공자라는 전제하에 진행되는 수업이기에 수업에 대한 기초지식이 부족하면 내 스스로가 더더욱 공부하고 찾아보고 했어야 했는데 첫 한달간은 너무 안일하게 한 것 같아서 크나큰 갭의 차이를 느꼈고, 나의 행동에 따른 결과에 대해 후회가 너무나 컸다. 앞으로도 많이 남아있는 과목들이 있는데, 그 과정들은 절대 안일하게 보내지 않을것이다. 아직도 C 언어 부분은 다른분들이 이미 학습한 부분을 난 아직도 모르는데 더이상 이런 바보같은 상황이 오지않도록 출결에도 꼭 신경쓰며, 하루하루의 학습을 놓치지 않고 따라갈 수 있도록 노력할 것이다. 이제 시작이라 생각하고 마음 다잡고 공부할 계획이다. 앞으로는 평일엔 그날그날의 학습을 이해하며, 자기전 기초지식들을 채우고 주말엔, 앞으로 시작될 과정을 맞기전 내가 모르는 부분을 학습할 계획이다. 아무래도

그 과정이 내 마음보다 빨리 올텐데 과정에 못따라가는 일은 절대 없도록 할 것이다.

2. 프로그램을 작성하시오.
아무런 함수 3 개를 작성하여 프로그램이 동작하도록 만드시오.
(본인이 만들고 싶은대로 만든다 - 함수 3 개이상 쓰면 됨)
답:

```
#include <stdio.h>

int ccc(void)
{
	printf("1234");
}

int ddd(void)
{
```

```
	printf("4567");
}

float eee(void)
{
	printf("123.456");
}

int main(void)
{
		printf("숫자 1~4\n");
		ccc();
		printf("\n");

		printf("숫자 4~7\n");
		ddd();
		printf("\n");


		printf("소수표현\n");
```

```
            eee();
            printf("\n");

    return 0;

}
```

출력은

숫자 1~4
1234
숫자 4~7
4567
소수표현
123.456

3. C 언어 메모리 레이아웃을 기술하시오.

답: 메모리 구성은 Stack, Heap, Static & Global Data, Program Code 로 구성된다.
메모리는 낮은 주소로부터 순서대로 쌓여진다. 예외적으로는 스택만, 높은 주소로
부터 낮은 주소 순으로 쌓여진다. 여기에는 Stack 영역과 Heap 영역을 분할 하기 위함.
동적바인딩과 정적바인딩을 구별을 위해서 서로 떼어놓기 위해 한쪽은 아래에서 위로,
한쪽은 위에서 아래로 시작하는 한다.

<Program Code>
프로그램 코드 영역은 3 가지로 나뉘게 된다. init(const), text, rodata.

-Text 영역-
코드 그 자체가 기록되는 영역. (코딩시 선언한 함수들 역시 여기에 기록됨.)
컴파일러가 목적 프로그램으로 바꾸고 링커가 묶어서 실행프로그램으로 변경이 되는데
이 실행 프로그램 자체를 모조리 RAM 에 기록. 기록된 부분이 text 영역.

-init(const)영역-
const 는 상수라는 뜻이고 init 은 initialize(초기화)의 준말.

이 영역은 Read Only 이듯, 저장되는 값은 절대 변경될 수 없는 상수 그자체가 기록됨.
정확히는 상수들 중에서도 ‘선언과 동시에 초기화된 변수의 상우’ 가 기록된다.

-rodata 영역-
read only data 영역인데, 이 부분은 init 과 같이 리터럴 타입만 저장되지만 이 데이터 영역은 init 영역에서 쓰이지 않았던 나머지 부분들만 모두 저장된다.
즉, 초기화 구분을 제외하고 모조리 기록해 두는것.

<Static & Global Data>
전역, 정적 자료. 전역변수와 정적 변수들의 특징은 프로그램이 열릴때 시작해서 프로그램이 닫힐때까지 생존. 따라서 이것들은 단 하나밖에 존재하지 않고 둘 이상을 만들 순 없다. = 새로 만들어지는 경우는 없음. / 이 영역에는 static 키워드가 붙은 모든 것들과 전역에 선언하 변수들이 기록된다는 점. 또한, 읽기 자유로우며 새로운 변수생성은 불가능해서 자료량이 늘지는 않지만, 자료를 수정 할 수는 있다. 여기에는 RO(Read Only)가 아니라 RW(Read Write)이다.

- data 영역 -

정적변수와 전역변수가 저장되는데 모든 전역변수와 모든 정적변수가 여기에 저장되는 것은 아니다. 여기에는 이미 초기화된, 이니셜라이징잉 완료된 변수들만 여기에 저장됨

- bss 영역 -
bss 는 Block Started Symbol 의 줄임말. 이영역은 초기화 되지 않은 전역변수와 정적변수를 저장한다. Data 영역은 시작하자마자 바로 init 영역의 값을 찾아서 자신의 짝의 값을 가져가도록 되어있는데, bss 영역은 자신의 짝이 없기 떄문에 init 영역을 확인하지 않고 바로 0 으로 초기화 시켜버린다. bss 에 저장되어 시작하자마자 main 을 생성하기 전에 바로 초기화 하기때문.

<Stack >
레지스터의 임시 저장 장소, 서브루틴 사용시 복귀 주소저장, 서브루틴에 인자 전달 등에 사용된다. 스택은 메모리의 상위 주소에서 하위 주소 방향으로 사용하며, 후입선출(LIFO: Last In First Out)원칙에 따라 나중에 저장된 값을 먼저 사용한다.

<Heap>
프로그램잉 실행 될때까지 알 수 없는 가변적인 양의 데이터를 저장하기 위해 프로그램의

프로세스가 사용할 수 있도록 미리 예약되어 있는 메인 메모리의 영역으로, 프로그램들에 의해 할당되었다가 회수되는 작용이 되풀이된다. 스택영역이 엄격하게 후입선출(LIFO) 방식으로 운영되는데 반해, 힙 영역은 프로그램들이 요구하는 블록의 크기나 요구/횟수 순서에 일정한 규칙이 없다. 힙의 기억 장소는 대게 포인터 변수를 통해 동적으로 할당받고 돌려주며, 이는 연결 목록이나 나무, 그래프 등의 동적인 데이터 구조를 만드는데 반드시 필요하다. 만약 프로그램 실행중 해당 힙 영역이 없어지면 메모리 부족으로 이상 종료를 하게 된다.

4. 함수 포인터는 왜 쓰는가? 알고있는대로 기술하시오.

답: 함수포인터는 우선, 함수의 주소를 저장하는 변수.

간단히 말하자며는, 사용 용도는 함수포인터의 문법은 메인보다 비교적 쉽게 이해할 수 있다. 함수를 한단계 추상화 하여 개발자에게 인터페이스만 제공하고 용도에 맞게 사용하게 만드는게 목표. 자세하게는 함수의 동적바인딩과 정적바인딩이 때문.

함수 포인터 없이 그냥, 함수이름을 직접써서 호출하는 것은 정적 바인딩. 해당 함수의 위치가 컴파일 당시에 파악이 가능하므로 컴파일러가 해당 위치를 바이너리에 포함시키고

해당 파일이 실행될때는 향상 같은 부분을 가르키게 되는것. 하지만, 어떤 함수를 호출할지 컴파일 타입이 아니라, 런타임에 동적으로 결정해야 할 경우들이 생긴다.
정리하자면, 정적바인딩(컴파일타입) 뿐만 아니라 동적바인딩(런타임)도 가능하게 하기위해 필요하다.

5. 함수를 여러개 만드는 이유는 무엇인가? (서술형 기술)

답: 함수를 여러개로 만드는 이유는 모듈화를 할수 있기때문.
간단히 표현하자면, 함수 재상용의 극대화와 디버깅의 용이함, 객체지향으로의 접근때문. 메인함수에 속하지 않기 때문에 에러 발생시 찾기에도 쉽고 디버깅 시에도 그 해당함수의 부분만 찾아볼 수 있기때문에 시간절약도 할수 있다. 게다가, 나 혼자만이 아닌, 다른 사람들이 코딩파일을 볼 시 에도 작성자 말고도 다른사람들도 코딩된걸 보았을때 분리되어 있는 함수를 쉽게 파악할수 있어서 모듈화가 편하다는 것이다.

6. 포인터가 없는 언어들과의 차이점

(포인터 때문에 할수 있는 기능이 무엇인지 기술하시오)

답: 포인터는 주소를 저장하는 변수인데, 기능은 함수를 코딩을 간결하고 효율적인 표현과 처리가 가능하다. 이는 더 빠른 기계어 코드생성을 한다. 복잡한 자료구조와 함수에 쉬운접근이 가능하다. 포인터를 사용하면 메모리에 직접 접근이 가능하며 동적 기억장소에 할당이 가능하다. 또한 call by reference 에 의한 전역변수 사용을 억제한다.

간단히 표현하자면, 포인터가 없는 변수는 직접 값을 입력하며 호출해야 하지만,

포인터를 사용할 경우, 간접호출로 변수사용에 용이하고 빠르고 효율적 사용이 가능하다.