TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

학생 – 이우석
colre99@naver.com

(11 회차) 오늘 복습한 내용:
scope 문, 전역변수, 정적변수, while 문,
무한루프 with while 문, continue 문, do while 문.

1. scope 문
: ‘{‘ 으로 시작해서 ‘}’ 으로 끝나는 영역을 가르킨다.

```c
#include <stdio.h>

int main(void)
{
        int global_area = 1;

        {
                int local_area1 = 2;
                printf("global_area = %d\n", global_area);
                printf("local_area1 = %d\n", local_area1);
        }

        {
                int local_area2 = 3;
                printf("global_area = %d\n", global_area);
                printf("local_area2 = %d\n", local_area2);
        }

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

return 0;
}
```

local_area1 은 지역변수(범위한정)

local_area2 도 지역변수(범위한정)

global_area 는 전역변수(한정없음)

2. 전역변수 (global variable)

: 변수의 위치에 관계없이 data 값을 참조 할수있다.

Memory 의 data 영역에 load 된다.

```c
#include <stdio.h>

int global_number = 100;

int main(void)
{

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

return 0;
}
```

: 출력값은 main 밖에서 변수를 선언해준 100 이라는 숫자가 출력된다.

3. 정적변수 (static keyword)

: static 변수는 전역변수와 마찬가지로 data 영역에 load 된다.

지역변수를 static 으로 선언했다면 이 변수를 선언한 함수 내에서만 접근 가능하다.

```c
#include <stdio.h>

void count_static_value(void)
{
        static int count = 1;
        printf("count = %d\n", count);
        count++;
}

int main(void)
{
        int i;
        for(i = 0; i < 7; i++)
                count_static_value();

}
```

: Count_static_value 라는 지역변수에 static 으로 선언해줬다.

: main 에서는 7 전까지 진행하며 count_static_value 함수를 호출한다. Count 의 결과값은 7 까지 출력.

4. while 문

: 반복문(~하는동안). for 문 하고 비슷하다.

반복을 위한 조건이 필요하고, 조건이 맞을때까지만 반복.

```c
#include <stdio.h>

int main(void)
{
        int i = 0, result = 'A';

        while(i < 10)
        {
                printf("%c\n", result);
                result++;
                i++;

        }

return 0;
}
```

: 변수 0 부터 시작해서 10 번째 전까지 반복.
변수는 1 단계씩 증가. 최종출력을 10 번째까지 됨.
즉, 0 을 A 로 시작해서 9 번째 J 까지 출력됨.

5. 무한루프 with while 문 (Infinite loop with while)

: 조건을 무조건 참=1(True)으로 만들면 무한루프 생성. 거짓(False)은 0.

(ex) while(1) ← 여기서 조건식이 0 이 아닌 수를 넣으면 '무한' 의미)

```c
#include <stdio.h>

int main(void)
{
        while(1)
        {
                printf("Hello Infinite Loop\n");
        }

return 0;
}
```

:while(1) = 괄호 안 조건식을 1 을 넣음으로써 무한루프 생성.
출력값은 Hello ~ Loop 가 무한생성.
실행하고 빠져나가려면 Ctrl + C.

6. continue 문

: continue 에 걸리는 해당 case 는 넘어가고, 반복은 계속 해주고 싶을때 사용.
Continue 부분에 걸리는 조건값은 패스가 되어 넘어가고, 다음조건들로 넘어감.

```c
#include <stdio.h>

int main(void)
{
        int number = 0;
        while(1)
        {
                number++;
                if(number == 5)
                        continue;
                printf("%d\n", number);

                if(number == 10)
                        break;
        }


        return 0;
}
```

:while 문을 사용하며 무한루프 생성.
숫자는 0 부터 시작하며 1 씩 계속증가.
숫자 5 가 되었을때 continue 문에 걸려
패스가 되고 다시 위로 올라가 계속진행.
6 부터 다시 출력이 되고 10 까지 출력
되었을때 if(number == 10) 에 종료.
출력값은 5 를 제외한 1 ~ 10 까지 출력.

7. do while 문

```c
#include <stdio.h>

int main(void)
{
        int number = 0;

        do
        {
                number++;
                if(number == 5)
                        continue;
                printf("%d\n", number);

        }while(number < 10); // 책에는 i 라고 표시됨.

        return 0;
}
```

: 일단 조건이 만족하지 않더라도 한번은 수행한다. 궁극적으로 do while 을 사용하는 이유는 매크로 확장때문에. (kernel 의 매크로에 자주 사용 된다.)

: 5 를 제외하고 1 ~ 10 까지 출력된다.