TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

학생 – 이우석
colre99@naver.com

(14 회차) 오늘 복습한 내용

for 문, for 문(무한루프), goto 문, Register keyword, 재귀함수 호출,
배열, 배열 선언, 문자형 배열, NULL 문자 배열.

1. for 문

: while 문 보다 보기 편하다. 구성: ex) for(초기화; 조건식; 증감식) 구성.

```c
#include <stdio.h>

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

    for(i = 0; i < 10; i++)
    {
        printf("%c\n", result);
        result++;
    }

    return 0;
}
```

(초기화;조건식;증감식)을 확인 할 수 있다. 조건이 10 미만까지 이며 마지막 때 result 값이 증가되며 값이 저장되어 A 부터 J 까지 출력되는걸 확인할수있다.

결과값:

```
wooseok91@air:~/practice$ ./a.out
A
B
C
D
E
F
G
H
I
J
```

2. for 문(무한루프)

: for 문에서 조건문가 비어있거나,며 0 이 아닌 수가 있다면 무한루프가 생성된다.

```c
#include <stdio.h>

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

    for( ; ; )
    {
        printf("%c\n", result);
    }


    return 0;
}
```

조건식이 비어있으며,
이로써 무한루프가 생성됨.
결과값은 A 가 멈추지 않는한
끊임없이 출력된다.

결과값:

```
A
A
A
A
^C
wooseok91@air:~/practice$ vi 239.c
```

'Ctrl + C' 를 함으로써 실행을 멈추었다.

3. goto 문

: goto 문을 활용하면 특정한 상황을 아주 간결하게 해결할 수 있는 경우가 많다.

```c
#include <stdio.h>

int main(void)
{
    int i, j, number = 0;

    for(i = 1; i < 4; i++)
    {
        for(j = 1; j < 4; j++)
        {
            if(i == 2)
            goto error_handler;
            printf("i * j = %d\n", i * j);
        }
    }

    error_handler:
        printf("Error를 처리중입니다\n");

    return 0;
}
```

i 의 값이 1 까지는 진행 되었지만,
2 가 되자(if 문의 조건만족) goto 문으로
이동하여 goto 에 해당하는 결과값이
출력된걸 확인할수 있다.

결과값:

```
wooseok91@air:~/practice$ ./a.out
i * j = 1
i * j = 2
i * j = 3
Error를 처리중입니다
```

4. Register keyword

: 사용이유는 가장 접근속도가 빠른 register 를 이용하기위해.
함수의 input 과 지역변수에만 붙일 수 있다.

```c
#include <stdio.h>

int inc(register int num)
{
    return num + 1;
}

int main(void)
{
    int i = 0;
    register int number = 7;

    for(i = 0; i < 3; i++)
    {
        number = inc(number);
        printf("number = %d\n", number);
    }

    return 0;
}
```

for 문의 조건이 3 미만이기에 0,1,2 3 번의 동작이 있다. 조건이 불릴때마다 register number 가 + 1 씩 되기에 7 부터 시작해서 9 까지 +1 씩 붙는다.
결과값으로 7+1, 8+1, 9+1 으로 나오는걸 확인할 수 있다.

결과값:

```
wooseok91@air:~/practice$ ./a.out
number = 8
number = 9
number = 10
```

5. 재귀함수 호출 (Recursive Function Call)

: 사용한 함수를 다시 호출하는 방식. Program 구현상 반드시 필요한 경우가 있음.

```c
#include <stdio.h>

int fib(int num)
{
    if(num == 1 || num == 2)
        return 1;
    else
        return fib(num -1) * (num -2);
}

int main(void)
{
    int result, final_val;
    printf("피보나치 수열의 항의 개수를 입력하시오:");
    scanf("%d",&final_val);
    result = fib(final_val);
    printf("%d번째 항의 수는 = %d\n", final_val, result);
    return 0;
}
```

fib 함수에서 따로 계산식을 만들어 주고, main 함수에서 result 의 값을 fib 함수를 불러오는 것을 알수 있다. 이것이 재귀함수 호출인데, main 함수 외의 공간에서 따로 함수를 만들어 호출하여 사용하는 것을 알 수 있다.

결과값:

```
wooseok91@air:~/practice$ ./a.out
피보나치 수열의 항의 개수를 입력하시오:5
5번째 항의 수는 = 6
```

6. 배열

: 배열은 인덱스 안에 수를 넣음으로써 각 공간을 나열하여 정리할 수 있다.

```c
#include <stdio.h>

int main(void)
{
    int i;
    int num[7];

    for(i = 0; i < 7; i++)
    {
        num[i] = i;
        printf("num[%d] = %d\n", i, num[i]);
    }

    return 0;
}
```

Num 에 7 을 넣음으로서 7 개의 각 공간이 생겼으며 시작은 0 부터 시작한다. 그러므로, 0 부터 6 까지 7 개의 각각의 공간에 데이터를 넣을수 있다. 조건부는 7 미만이기에 0 부터 6 까지 배열 공간에 각각 들어가게 된다.

결과값:

```
wooseok91@air:~/practice$ ./a.out
num[0] = 0
num[1] = 1
num[2] = 2
num[3] = 3
num[4] = 4
num[5] = 5
num[6] = 6
```

7. 배열선언

: 변수 선언과 동일하게 자료형과 이름을 써넣어주면 된다. 인덱스에 숫자를 넣음으로서 '길이'를 정해주게 된다.

```c
#include <stdio.h>

int main(void)
{
    int i;
    int num1_arr[7] = {1, 2, 3};

    for(i = 0; i < 7; i++)
    {
        printf("num1_arr[%d] = %d\n", i, num1_arr[i]);
    }


    return 0;
}
```

함수 num1_arr 에 7 이라는 길이를 설정하며 데이터 값은 { } 사이에 1, 2, 3 을 넣어주었다. for 문의 조건식이 7 미만이기에 배열은 0 부터 6 까지 진행되며, 배열선언된 1, 2, 3 이 차례로 들어가게 된다.

결과값:

```
wooseok91@air:~/practice$ ./a.out
num1_arr[0] = 1
num1_arr[1] = 2
num1_arr[2] = 3
num1_arr[3] = 0
num1_arr[4] = 0
num1_arr[5] = 0
num1_arr[6] = 0
```

8. 문자열 배열

: 문자열 배열은 말그대로 문자를 출력한 배열이다.

```c
#include <stdio.h>

int main(void)
{
    char str1[5] = "AAA";
    char str2[] = "BBB";
    char str3[] = {'A', 'B', 'C'};
    char str4[] = {'A', 'B', 'C', '\0'};


    printf("str1 = %s\n", str1);
    printf("str2 = %s\n", str2);
    printf("str3 = %s\n", str3);
    printf("str4 = %s\n", str4);


    str1[0] = 'E';
    str2[1] = 'H';
    printf("str1 = %s\n", str1);


    return 0;
}
```

main 함수 안에 문자형(char) 배열이 4 개가 선언되었으며, 각 배열마다 길이가 정해진 것과 정해지지 않은것들이 있다. 인덱스 안에 길이가 정해지지 않았다면, 그 안에 들어있는 data 만큼 길이가 정해진다. 코드 밑부분 출력하나는 str1 의 배열의 데이터값을 새로 초기화 해주었다. 그러므로 str1 의 데이터 시작값은 E 로 시작하되 AA 는 그래도 출력되어 EAA 가 되는것이다. 인덱스 길이만큼 문자가 없다면 빈공간으로 인식하여 출력이 없다.

결과값:

```
wooseok91@air:~/practice$ ./a.out
str1 = AAA
str2 = BBB
str3 = ABC
str4 = ABC
str1 = EAA
```

9. NULL 문자 배열

:NULL 문자는 문자열의 마지막을 의미. 배열에 값을 1 개씩 직접 설정할 경우, '\0' 으로 어느부분이 배열의 마지막인지를 명시해 주는것이 좋다.

```c
#include <stdio.h>

int main(void)
{
    char str1[5] = "AAA";
    char str2[] = "BBB";
    char str3[] = {'A', 'B', 'C'};
    char str4[] = {'A', 'B', 'C', '\0'};


    printf("str1 = %s\n", str1);
    printf("str2 = %s\n", str2);
    printf("str3 = %s\n", str3);
    printf("str4 = %s\n", str4);


    str1[0] = 'E';
    str2[1] = 'H';
    printf("str1 = %s\n", str1);


    return 0;
}
```

Char str4[]의 데이터 값을 보면 네번째 값이 '\n' 으로 표현되었다. 즉, 배열의 마지막임을 나타냄으로서 A, B, C 3 개가 전부임을 알수 있다.

결과값:

```
wooseok91@air:~/practice$ ./a.out
str1 = AAA
str2 = BBB
str3 = ABC
str4 = ABC
str1 = EAA
```