5 일차 . 진행...

과정 : (TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍)

▶ 배열

a) 배열은 메모리 상에 순차적으로 배열이 되어 있다 .

b) 배열에 항상 초기화를 해주자 .

→ Senser_data[1000] ={0} : 0~999/ 1000 개 배열 (단 , ={0} 을 넣어 항상 배열을 초기화 해주자 .)

→ Double data[77]={0}/ NULL

***char 형 배열이 필요한 이유?**

String 인 문자열 " I'm Marth Kim" 은 변경이 불가능 char 형 배열은 내부 데이터 변경이 가능하다 . 마지막 data 에 Null Character 가 필요함 .

*** Null character 른 무엇인가?**

NULL 문자는 문자열의 마지막을 의미한다 . 배열에 값을 1 개씩 직접 설정할 경우 ' \0' 로 배열의 마지막을 명시 해야함 .

ex) char str3[ ] = {'A', 'B', 'C'};
ex) char str2[ ] = "BBB";

* 다중배열

행렬을 만드려고 사용한다. 이중배열을 만들어보자.

*** Array Internals**

배열의 내부적으로 어떻게 ? 배열도 분명 주소를 가지고 접근할 것이다 . 그 주소는 바로 배열의 이름이다 .
(주소는 ? 배열 !)

ex) code

```
#include <stdio.h>

int main(void)
{
    int arr[4] = {10, 20, 30, 40};
    int i;

    printf("address = %p₩n", arr);

    for(i =0; i < 4; i++)
    {
        printf("address = %p, arr[%d] = %d₩n", &arr[i], i, arr[i]);
    }

    return 0;
}
```

%p : 주소값을 뿌릴 때, 사용한다.

run

```
address = 0x7ffe9befc420
address = 0x7ffe9befc420, arr[0] = 10
address = 0x7ffe9befc424, arr[1] = 20
address = 0x7ffe9befc428, arr[2] = 30
address = 0x7ffe9befc42c, arr[3] = 40
```

→ 배열은 int 형이므로 4byte 식 차지하고 있는 것을 볼 수 있다. 이때, 배열 전체 arr 의 주소값과 arr[0]의 주소값은 같다. 배열의 시작.

변수 : 무언가 저장하는 곳
포인터 : 주소를 저장할 수 있는 변수

→ 포인터는 주소를 저장하는 곳

ex) code

```c
#include <stdio.h>
int main(void)
{
int arr[3] = {1, 2, 3};
int *p= arr;
int i;

for(i=0; i<3; i++)

printf("p[%d] = %d\n", 1,p[i]);


return 0;
}
```

run

```
p[1] = 1
p[1] = 2
p[1] = 3
```

→

2 차원 배열과 1 차원 배열의 관계는 ?
[2][3] 의 경우 {{1,2,3,},{4,5,6}} 과 같은 방식으로 한다 2 차원 배열은 1 차원 배열 형태로 모인 것이다 .

ex) code

```c
#include <stdio.h>
int main(void)
{
        int arr[3][4];
        printf("arr address = %p\n", arr);
        printf("arr[0] address = %p\n", arr[0]);
        printf("arr[1] address = %p\n", arr[1]);
        printf("arr[2] address = %p\n", arr[2]);

        return 0;
}
```

run

```
arr address = 0x7ffc82d7d710
arr[0] address = 0x7ffc82d7d710
arr[1] address = 0x7ffc82d7d720
arr[2] address = 0x7ffc82d7d730
```

→ 2 차원배열 arr[3][4] 첫번째는 모두 int 형의 자리 4 열을 가지고 있으므로 16byte 씩 자리를 차지하는 것을 볼 수 있다.

ex) code

```c
#include <stdio.h>
int main(void)
{
        int arr[3][4];
        printf("arr size = %lu\n",sizeof(arr));
        printf("arr[0] size = %lu\n",sizeof(arr[0]));
        printf("arr[1] size = %lu\n", sizeof(arr[1]));
        printf("arr[2] size = %lu\n", sizeof(arr[2]));

        return 0;
}
```

run

```
arr size = 48
arr[0] size = 16
arr[1] size = 16
arr[2] size = 16
```

→

*** 함수의 인자로 배열을 전달하려면 ??**

1. 배열을 통째로 전달 할 수 없다 .
2. 함수 호출 시 배열의 주소 값을 전달한다 .
3. 함수 자체에서 배열의 index 를 생략하고 [] 만 전달해도 된다 .
4. 되도록이면 index 를 모두 정확하게 적는 것이 편한다 .
5. 2 차원 , 3 차원 배열도 모두 동일한 방식이다 .

ex) code

```c
#include <stdio.h>
void arr_print(int arr[])
{
        int i;
        for(i = 0; i < 3; i++)
                printf("%4d", arr[i]);
}
int main(void)
{
        int arr[3] = {3, 33, 333};
        arr_print(arr);
        printf("\n");
        return 0;
}
```

run

```
   3  33 333
```

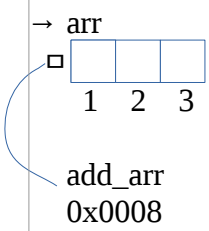
→ arr[3]은 main() 함수내의 지역 변수 arr_print 로 위 배열의 주소가 전달된다.
즉, arr_printf 함수에서는 main() 함수의 지역 변수인 arr[3]에 접근하여 값을 수정할 수 있음.
(주소값을 가지고 있기에 가능한 것.)

ex) code

```c
#include <stdio.h>

void add_arr(int *arr)
{
        int i;
        for(i = 0; i < 3; i++)
                arr[i] += 7;

}

void print_arr(int *arr)
{
        int i;
        for(i = 0; i < 3; i++)
                printf("arr[%d]= %d\n",i, arr[i]);
}
int main(void)
{
        int arr[3] = {1,2,3};
        add_arr(arr);
        print_arr(arr);
//      printf("\n");
        return 0;
}
```

run

```
arr[0]= 8
arr[1]= 9
arr[2]= 10
```



*** 포인터란 무엇일까?**

주소를 저장할 수 있는 변수 , 무언가를 가르키는 녀석 , (즉 포인터 = 주소)
pointer 의 크기는 HW 가 몇 비트를 지원하느냐에 따름 64bit 는 8byte 이다 .

*** 포인터 선언하는 방법**

1. 가르키고 싶은 녀석의 자료형을 적는다 .
2. 주소값을 저장할 변수명을 적는다 .
3. 변수명 앞에 ‘ *’ 을 붙인다 .

자료형 없이 변수명 앞에 ‘ *’ 만 붙은 경우 해당 변수가 가르키는 것의 값을 의미함 .

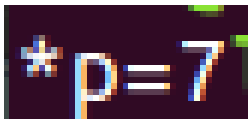
(int *) : int 형 주소 저장 가능한 데이터 타입 .

Int num=7;
int *p = #
printf(“*p=%d”,*p);

ex) code

```c
#include <stdio.h>

int main(void)
{
        int num=7;
        int *p = &num;
        printf("*p=%d",*p);
        return 0;
}
```

run



→

ex) code

```c
#include <stdio.h>

int main(void){
        int *ptr;
        printf("ptr = %p\n", ptr);
        printf("ptr value = %d\n", *ptr);

        *ptr = 27;

printf("ptr value = %d\n", *ptr);
return 0;
}
```

run

```
ptr = (nil)
Segmentation fault (core dumped)
```

→ Segmentation Fault 가 나는 이유?

우리가 기계어를 보면서 살펴봤던 주소값들이 사실은 전부 가짜 주소라고 말했었다 .
이 주소 값은 엄밀하게 가상 메모리 주소에 해당하고 , 운영체제의 paging 메커니즘을 통해서 실제 물리 메모리의 주소로 변환된다 . (윈도우도 가상 메모리 개념을 베껴서 사용한다 .) 그렇다면 당연히 맥 (유닉스) 도 사용함을 알 수 있다 .

가상 메모리는 리눅스의 경우 32 비트 버전과 64 비트 버전이 나뉜다 .

32 비트 시스템은 2^32 = 4GB 의 가상 메모리 공간을 가짐 . 여기서 1:3 으로 1 을 커널이 3 을 유저가 가져간다 . 1 은 시스템 (HW,CPU,SW 각종 주요 자원들) 에 관련된 중요한 함수 루틴과 정보들을 관리하게 된다 . 3 은 사용자들이 사용하는 정보들로 문제가 생겨도 그다지 치명적이지 않은 정보들로 구성됨 .

64 비트 시스템은 1:1 로 2^63 승에 해당하는 가상메모리를 각각 가진다 . 문제는 변수를 초기화하지 않았을 경우 가지게 되는 쓰레기값이 0xccc..ccc 로 구성됨이다 .

32 비트의 경우에도 1:3 경계인 0xc0000000 을 넘어가게됨 . 64 비트의 경우에 시작이 c 이므로 이미 1:1 경계를 한참 넘어감 . 그러므로 접근하면 안되는 메모리 영역에 접근하였기에 엄밀하게는 Page Fault(물리 메모리 할당되지 않음) 가 발생하게 되고 , 원래는 interrupt 가 발생해서 Kernel 이 Page Handler(페이지 제어기) 가 동작해서 가상 메모리에 대한 Paging 처리를 해주고 , 실제 물리 메모리를 할당해주는데 , 문제는 이것이 User 쪽에서 들어온 요청이므로 Kernel 쪽에서 강제로 기각해버리면서 Ssegmentation Fault 가 발생하는 것이다 .

실제 Kernel 쪽에서 들어온 요청일 경우에는 위의 메커니즘에 따라서 물리 메모리를 할당해주게 된다 .

Int *ptr 을 하면 printf 의 ptr 의 주소를 불러오는건 ok
그러나 *ptr 을 하면 안에 0xcccccc 라는 쓰레기값이 들어가 있으므로 segmentation 오류가 난다 .
허나 *ptr 의 변수를 설정해주면 에러가 나지 않는다 .

ex) code

```c
#include <stdio.h>

int main(void){
        int num = 3;

        *(&num) += 30;
        printf("num = %d\n", num);
        return 0;
}
```

run

```
num = 33
```

→ 따로 p 라는 다른 포인터변수를 두지 않고도 셀프 참조를 통해 사용 가능하다.

& : 주소값을 받아온다 .
* : 주소값을 적는다

합치면 ? 상쇄된다 .
즉 , *(&num) = num

ex) code

```c
#include <stdio.h>
int main(void){
        char str1[33] = "Pointer is important!";
        char *str2 = "Pointer is important!";

        printf("str1 = %s\n", str1);
        printf("str2 = %s\n", str2);
        return 0;
}
```

run

```
str1 = Pointer is important!
str2 = Pointer is important!
```

→ 배열에 문자열이 저장되어 있다. 배열에 문자 하나씩 저장된다.

Pointer 에 대한 Pointer 는
**P : 주소의 주소를 받겠다. (int **p = &p)

ex) code

```c
#include <stdio.h>

int main(void)
{
int num =3;
int *p = &num;
int **pp = &p;

printf("num = %d\n",num);
printf(" *p = %d\n",*p);
printf("**p = %d\n",**pp);
return 0;


}
```

run

```
num = 3
 *p = 3
**p = 3
```

ex) code

```c
#include <stdio.h>

int main(void){
        int num1 = 3, num2 = 7;
        int *temp = NULL;
        int *num1_p = &num1;
        int *num2_p = &num2;
        int **num_p_p = &num1_p;

        printf("*num1_p = %d\n", *num1_p);
        printf("*num2_p = %d\n", *num2_p);

        temp = *num_p_p;
        *num_p_p = num2_p;
        num2_p = temp;

        printf("*num1_p = %d\n", *num1_p);
        printf("*num2_p = %d\n", *num2_p);

        return 0;
}
```

run

```
*num1_p = 3
*num2_p = 7
*num1_p = 7
*num2_p = 3
```

→

*** 포인터 배열**

포인터가 배열을 가진다는 것은 sizeof(자료형)* 배열 을 지정하는 것이다.

int a[2][2]=({10,20},{30,40});
int (*p)[2] = a; →p 가 가리키는 주소가 sizeof(int)*2 씩 증가하도록 지정을 한다
(즉, int 형 배열 2 개 , 한번 증가할 때 8byte 를 넘는다)

ex) code

```c
#include <stdio.h>

int main(void)
{
        int i, j, n1, n2, n3;
        int a[2][2] = {{10, 20}, {30, 40}};
        int *arr_ptr[3] = {&n1, &n2, &n3};
        int (*p)[2] = a;
printf("%d %d",a[1][1],a[0][0]);
        for(i = 0; i < 3; i++)
                *arr_ptr[i] = i;

        for(i = 0; i < 3; i++)
                printf("n%d = %d\n", i, *arr_ptr[i]);

        for(i = 0; i < 2; i++)
                printf("p[%d] = %d\n", i, *p[i]);

        return 0;
}
```

run

Quiz

1. 총 7 개의 통장을 만들어서
100 만원 단위로 최대 500 만원까지 입금하였다.
이자율이 연 4%라고 할 때, 3 년 후 각각의 총액을 구하시오.r

2. char str1[] = “devil”, char str2 = “angel”일 때
2 개의 문자열을 서로 바꿔보라
3. 2 by 2 행렬의 곱셈을 계산할 수 있는 프로그램을 만드시오.
4. int arr[3][3] = {{1, 13, 2}, {7, 3, 5}, {2, 9, 11}}에서
최대값을 찾는 함수를 만드시오

http://cafe.naver.com/hestit/79
1. 배열에 문자열을 입력 받고,
각 배열 요소가 짝수인 경우만을 출력하는 함수를 작성하라.

3. 아래와 같은 숫자들이 배열에 들어 있다고 가정한다.
3, 77, 10, 7, 4, 9, 1, 8, 21, 33
이 요소들을 배열에 거꾸로 집어넣어보자.

4. 위의 숫자 3, 77, 10, 7, 4, 9, 1, 8, 21, 33 에서
홀수 번째 요소의 합과 짝수 번째 요소의 합을 곱하시오.

6. 행렬의 곱셈, 덧셈, 나눗셈, 뺄셈에 대해 조사하시오.
숫자를 예로 들어서 계산도 해보시오.

http://cafe.naver.com/hestit/104 : 2,4 번만 푼다.

2. 정수 2004016 을 변수에 저장하고 이것을 char 형 포인터로 받는다.

그리고 정수형은 총 4byte 로 구성되므로 총 4 개의 byte 를 볼 수 있을것이다.

각 byte 에 숫자가 어떻게 배치되었는지 확인해보자.

4. 우리는 예제에서 주소값을 교환하여 값을 변경하는 것을 해보았다.

그렇다면 변수 3 개를 놓고, 이것에 대해서 무한 Loop 를 돌면서 저글링을 해보자!

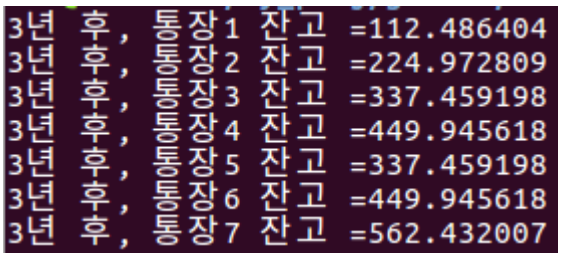
삼각형의 넓이 구하는 문제
case1) 밑변 높이
case2) 밑변, 밑변과 다른 변이 이루는 각도

Quiz 1.

code

```
// 총 7 개의 통장을 만들어서 100 만원 단위로 최대 500 만원까지 입금했다.
// 이자율 연 4%라고 할 때, 3 년 후 각각의 총액을 구하시오.

void arr_inc(float final_money[4][7])
{
	int i,j;
	for(j=0;j<3;j++)
	{
		for(i=0;i<7;i++)
		{
		  final_money[j+1][i]= final_money[j][i]*(1.04);
		}
	}
	for(i=1;i<8;i++)
	{
	   printf("3 년 후, 통장%d 잔고 =%f\n",i,final_money[3][i-1]);
	}


}

int main(void)
{

	float final_money[4][7]={{100,200,300,400,300,400,500},{0},{0},{0}};
	arr_inc(final_money);
	return 0;
}
```

run :



→

quiz2.

```c
//  2. char str1[] = “devil”, char str2 = “angel”일 때
//    2개의 문자열을 서로 바꿔보라

void arr_inc(char str1[],char *str2)
{


   return str1[]="angel";


}

int main(void)
{
    char str1[] = "devil";
    char *str2="angel";
   // arr_inc(str1);

     printf("%s",arr_inc(str1,*str2));
     printf("%s",str2);
    return 0;
}
```

Run

error

→

quiz3.

```c
#include <stdio.h>

// 3. 2 by 2 행렬의 곱셈을 계산할 수 있는 프로그램을 만드시오.

void arr_inc(int a[2][2],int b[2][2])
{
   int x[2][2]={0};
   x[0][0]=((a[0][0]*b[0][0])+(a[0][1]*b[1][0]));
   x[0][1]=((a[0][0]*b[0][1])+(a[0][1]*b[1][1]));
   x[1][0]=((a[1][0]*b[0][0])+(a[1][1]*b[1][0]));
   x[1][1]=((a[1][0]*b[0][1])+(a[1][1]*b[1][1]));
   printf("2by2={%d,%d,%d,%d}",x[0][0],x[0][1],x[1][0],x[1][1]);

}

int main(void)
{
  int a[2][2]={{1,2},{3,4}};
  int b[2][2]={{10,20},{30,40}};
  arr_inc(a,b);
  return 0;
}
```

Run

```
2by2={70,100,150,220}
```

→
quiz4.

```
#include <stdio.h>
//int arr[3][3] = {{1, 13, 2}, {7, 3, 5}, {2, 9, 11}}에서
//최대값을 찾는 함수를 만드시오


int arr_a()
{
    int i,j;
    for(i=0;i<3;i++)
       {
  for(j=0;j<3;j++)
  {
     a[j+i*3]= arr[i][j];
  }
  if( a[i] <a[i+1])
    a[i]
}
}

int main(void)
```

{
int arr[3][3] = {{1, 13, 2}, {7, 3, 5}, {2, 9, 11}};

```
각 배열에 문자열을 입력받고
각배열의 배열의 요소가 짝수인것만 나열하라

int arr_add(int *arr )
{
        for(i;i<10;i++)
        {
                if(arr[i]/2==0)
                        printf("%d/t",arr[i]);
        }
}

int  main(void)
{
        int arr[10]={1,3,4,8,13,2,28,19,54,77};
        arr_add(arr);
}
```

```
3번 거꾸로 넣어
3 77 10 7 4  9 1  8 21  33

int arr_mirror(int *arr_m, int *arr )
{
        for(i=0;i<10;i--)
        {
                arr_m[i]=arr[9-i];
                printf("결과는 array[%d]=%d",i,arr_m[i]);
        }
}

int  main(void)
{
        int arr[10]={3 77 10 7 4 9 1 8 21 33};
        int arr_m[10]={0};
        arr_mirror(arr_m,arr);
}
```

```
4번 홀수번째 짝수번째 따로 더해
3 77 10 7 4 9 1 8 21  33

int arr_mirror(int *arr_m, int *arr )
{
        int a,b,i;
        for(i=0;i<10;i++)
        {
                if(arr[i]%2==0;)
                        a+=arr[i];
                else
                        b+=arr[i];
        }
        printf("짝수의 합=%d\n홀수의합=%d\n",a,b);
}

int  main(void)
{
        int arr[10]={3 77 10 7 4 9 1 8 21 33};
        int arr_m[10]={0};
        arr_mirror(arr_m,arr);
}
```

진행중..

메모:
기계어를 알면 포인터,배열, 변수등의 모든 것을 주소 관점에서 접근 할 수 있어서 매우 쉬워짐.

*goto 를 쓰면 좋은이유 찾아봐라.

*가상메모리와 페이징