TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 : 이상훈
학생 : 황수정
2018. 02. 07
4일차

과제 1. 배운 내용 복습

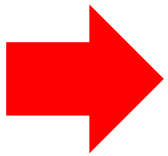
- 재귀 호출과 While문
- 배열
  -다중배열
  -함수와 배열
- 포인터
  -이중포인터
  -배열포인터와 포인터배열

재귀 호출과 while문

```
#include <stdio.h>

int fib(int num)
 {
     if(num == 1 || num == 2)
         return 1;
     else
         return fib(num – 1) +fib(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;
}
```



```
#include <stdio.h>

int fib(int num)
 {
     int first = 1;
     int second = 1;
     int tmp = 0;

    if(num == 1 || num == 2)
         return 1;

    while(num-- > 2)
     {
         tmp = first + second;
         first = second;
         second = tmp;
     }
     return tmp;
 }
```

Int main(void) 부분은 옆의 예시와 같다.

[재귀 호출 함수보다 while루프가 좋은 이유]

재귀 호출 함수는 stack 만들고 해제를 해야 한다. 그 과정에서 파이프라인도 깨진다. 그러나 while 루프는 함수 호출이 없으므로, stack을 만들지 않으며 파이프라인이 깨지지도 않는다. 덕분에 시간 또한 단축된다.

즉, while 루프가 시간이나 성능면에서 더 효율적이다.

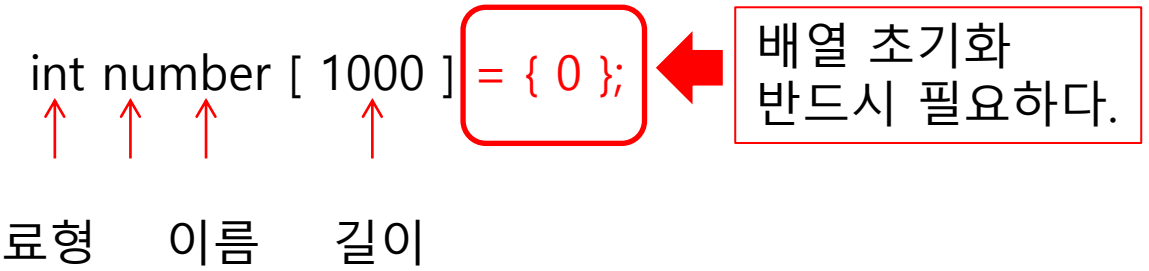
배열

- 같은 종류의 데이터를 메모리에 연속적으로 저장하기 위해 사용된다.

  예를 들어 1000개의 변수가 필요한 경우를 가정해보자. 1000개를 다 변수 선언을 할 수 없으므로 배열이 사용된다. 즉, 다수의 변수를 처리하기 위해 사용된다.

- 배열 선언

  변수 선언과 동일하다. 자료형과 이름이 필요하고, 추가적으로 '[길이]'를 붙인다.



배열 이름은 자유이나 가급적 무엇의 배열인지 알 수 있게끔 명시형으로 만드는 것이 좋다.

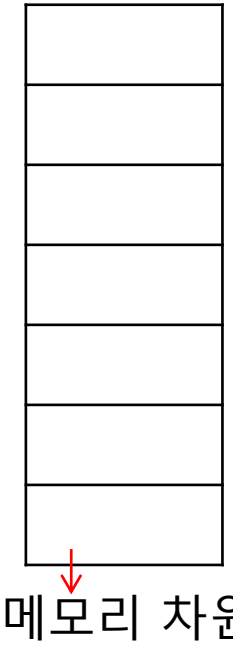
배열의 시작은 0부터임으로 1000개면 1~999까지로 Data에 마지막은 999고 시작은 0이다.

배열

- 메모리 상에 **순차적**으로 쌓이므로 인덱스를 달 수 있다. 덕분에 for문 등 제어문으로 데이터를 효율적으로 정리 및 처리 할 수 있다.

Number [0]	Number [1]	Number [2]	Number [3]	Number [4]	Number [5]	...

C언어 차원



메모리 차원

[ ]사이의 수를 index(인덱스)라고 하며
배열의 첫 번째 요소는 [0]에서 시작한다.
위와 같이 [0]부터 시작함으로, 마지막은 [n-1] 이다.

배열

```
#include <stdio.h>

int main(void)
 {      int i;
      int num1_arr[ ] = {1, 2, 3, 4, 5};
      int num2_arr[3] = {1, 2, 3};

     int len1 = sizeof(num1_arr)/sizeof(int);
     int len2 = sizeof(num2_arr)/sizeof(int);

      printf("num1_arr length = %d₩n", len1);
      printf("num2_arr length = %d₩n", len2);

   for(i = 0; i < len1; i++)
    {          printf("num1_arr[%d] = %d₩n", i, num1_arr[i]);      }

   for(i = 0; i < len2; i++)
    {          printf("num2_arr[%d] = %d₩n", i, num2_arr[i]);      }

  return 0;
 }
```

- 몇 개의 원소를 가지고 있는가?

길이 값을 알아야 할 때,
sizeof 가 사용 된다.

일반 상수 값이 아니라 배열에 맞추어서
(for 문 부분) 소스를 짰기 때문에
방대한 값이 들어와도 정상작동 한다.

배열

```
#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; }
```

Example

- char형 배열이 필요한 이유

string인 문자열은 변경이 불가하다.

char형 배열은 내부 데이터 변경이 가능하다.

단, 마지막 data에 Null Character(₩0)가 필요하다.

여기서, Null Character는 문자열의 마지막을 의미한다.

배열에 값을 1개씩 직접 설정할 경우 '₩0'으로 어느 부분이 배열의 마지막인지를 명시해주는 것이 좋다. 만약 들어가지 않는다면, 어디까지가 끝인지 알 수 없어지기 때문이다.

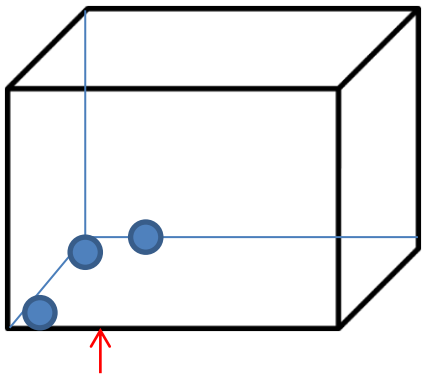
예전에는 null이 안 들어갈 경우, 오류가 발생했는데 요즘은 컴파일러가 최적화시키면서 대신 대입해준다.

문자열 사용시에는 이 null이 들어갈 자리를 생각해서 '공간 1'만큼을 더 넣어주어야 한다.

다중배열

- 2차원, 3차원 배열이 아닌 이중배열, 삼중 배열처럼 다중배열이 옳은 표현이다.
  메모리의 구조를 살펴보았을 때, 일직선으로 순차적으로 되므로 차원이 존재하지 않는다.

  지금 3D 메모리라는 것이 화자가 되고 있다.이 것이 나온다면 그 때 부터는 차원 배열이 생성 될 것이다. 메모리가 큐브 형태로 배열 위치에 차원이 생길 수 있기 때문이다.



차원 구조일 경우

현재 메모리 차원

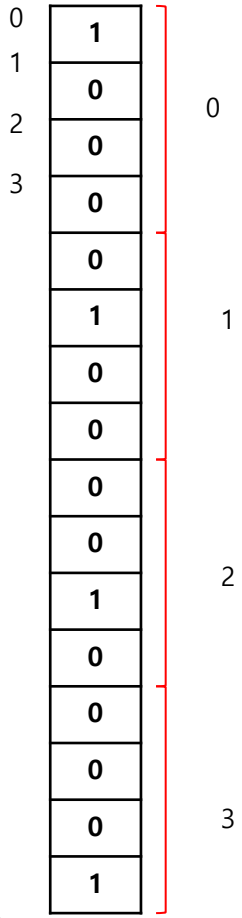
다중배열 -이중배열

```
#include <stdio.h>

int main(void)
{     int arr[4][4];
     int i, j;

    for(i =0; i < 4; i++)
    {
        for(j = 0; j < 4; j++)
        {
             if(i == j)
                 arr[i][j] = 1;
            else
                 arr[i][j] = 0;
         }
    }
 for(i =0; i < 4; i++)
    {
         for(j = 0; j < 4; j++)
         {
              printf("%d ", arr[i][j]);
         }
         printf("₩n");
    }

    return 0;
}
```



- 배열에 [인덱스]를 하나 더 추가한 것이라고 생각하면 된다. 이중배열은 행렬 표현에 용이하여 사용된다.
- 옆의 그림처럼 이중배열은 배열이 배열 형태로 모인 것이다. 따라서, 2차원 배열이 아닌 이중 배열인 것이다.
- 이중 배열 선언 : int arr[4][4];
- arr[4][4]
  여기서 맨 끝이 최소 단위가 된다.
- 변수가 메모리에 있을 때, 주소인 것처럼 배열도 마찬가지이다. 주소를 갖는다. 배열은 **배열 이름 자체가 주소**다.

함수와 배열

- 함수의 인자로 배열을 전달

  배열을 통째로 전달 할 수 없다(포인터가 더 깔끔하게 전달 된다.)

  함수 호출 시 **배열의 주소값을 전달**한다. 주소를 알고 있으면 stack의 배리어('{ }'을 말한다.)를 뛰어 넘을 수 있다. 코드에서 밑의 'main' 함수를 불러올 수 있는 것이다.

  즉, '주소를 알고 있다.' 는
  함수 어디서든 변수를 제어 가능하다. 는
  말이 된다.

  함수 자체에서 배열의 index를 생략하고 [ ]만 전달해도 된다. 되도록이면 index를 모두 정확하게 적는 것이 편하다. 다중 배열도 모두 동일한 방식을 띈다.

포인터(Pointer)

- 포인터는 **주소를 저장할 수 있는 변수**를 말한다.
- 포인터를 쓰는 이유는 주소값을 지닌 장비에 값을 넣을 때 쓰기 위해서이다. Ex : 메모리
- 포인터의 크기는 HW가 몇 bit를 지원하는지에 따라 달라진다. 이는 포인터는 이 메모리의 어떤 주소값이든 접근할 수 있어야 하기 때문이다. 작은 사이즈에 맞추면 그 값이 변동(보통 버려진다.)되므로 **가장 큰 주소에 맞추어야** 하기 때문에 달라진다. 보통 가장 큰 값에 맞추므로 > Pointer의 크기는 4byte이다.
- NULL Pointer

  지금 당장 표시할 것이 없을 경우 사용한다. 엉뚱한 값을 가지지 않도록 초기화하기 위함이다. Ex : int *ptr = NULL;
- 상수형 Pointer : '&' 뒤에 붙은 변수명의 주소값을 가져온다. Ex: *(&num) += 30;
- 포인터 선언

  가리키고 싶은 것의 자료형을 적는다. 주소값을 저장할 변수명(이름)을 적고, 변수명 앞에 '*'을 붙인다. 자료형 없이 변수명 앞에 '*'만 붙은 경우, 해당 변수가 가리키는 것의 값을 의미한다.

  ex) int num1= 7;

  int *p = &num1; > 일반적인 int 가 아니고 데이터 타입을 선언

  int*형이기도 하고 *p 주소 나타내기도 한다.

  변수 이름 = 주소

이중포인터(Pointer to Pointer)

- Pointer도 Stack에 할당되는 지역변수이다. 즉, Pointer에 대한 주소값도 존재한다. 주소값을 얻는법은 동일하다('&'를 이용). 그 주소값은 Pointer에 대한 Pointer로 '**' 형태를 띈다.

  ex : int num1;

  int *ptr1 = &num1;

  int **ptr2 = &ptr1; >주소에 주소를 저장하겠습니다.

  '*' 하나당 주소 1개

  *의 횟수에 대한 제한은 없다.

- 자료구조에서 많이 사용된다. < 포인터를 잘 쓸 줄 알아야 한다.
- * > 주소값에 접근한다.

  & > 주소값을 받아 온다. >> 둘을 함께 쓰면 상쇄 된다.

배열포인터와 포인터배열

- int *p[2] > 포인터배열 : 포인터에 대한 배열을 말한다.
  포인터를 저장하는 배열이다.

  int(*p)[2] = int(*)[2]p > 배열포인터 : 일단 두 개는 같은 형태이다.
  int형 2개 짜리 포인터를 말한다.

- 포인터배열
  포인터로 이루어진 배열을 의미한다.

- 배열포인터
  자료형으로 생각하면 이해가 좀 더 쉽다. 위의 예시를 보면 int형 2개 짜리 포인터를 말한다. 따라서 int(*p)[2]는 8byte에 대한 포인터를 나타낸다.
  앞의 자료형이 변경되면, 자료형에 따라 포인터의 단위가 달라진다.

과제 2. c언어 코드 복습 – 1번 문제

```c
#include <stdio.h>

	// input: first - day, second - 37500
	int borrow_equip(int day, double money)
	{
		int i = 0, res = 0;
		double rate = 1.0;
		double tmp = 0;

		if(day >= 3)
		{
			rate = 0.8;
			tmp = money * rate;
		}

		while(i++ != 7)
		{
			res += tmp;
			printf("res = %d₩n", res);
		}

		return res;
	}

	int main(void)
	{
		printf("res = %d₩n", borrow_equip(7,
	37500));
		return 0;
	}
```

```c
#include <stdio.h>

int rent_equip(int day, double money)
{
	int i, sum = 0;
	double rate = 1.0;

	if(day > 2)
		rate = 0.8;

	for(i = 0; i < day; i++)
		sum += money * rate;

	return sum;
}

int main(void)
{
	printf("money = %d₩n", rent_equip(7, 37500));
	return 0;
}
```

과제 2. c언어 코드 복습

```
#include <stdio.h>
int mul(int num1, int b)
{
   return(num1*b);
}

int main(void)
{
   int num1;
   printf("정수를 입력하세요 :");
   scanf("%d", &num1);

   int b = 37500;

      if (num1< 3)
         printf("%d * %d
      = %d₩n",num1,b,mul(num1,b));
   else
      printf("%d * %d
      =%d₩n",num1,b,0.8*mul(num1,b));

      return 0;
}
```

위의 답안의 2개는 선생님이 작성하신 것으로 각각 while 문과 for 문으로 작성된 것이다. 옆의 코드는 처음에 작성한 것으로 오류가 생긴 코드이다.
답이 소수점에 영향 받지 않을 것이라 생각하고 코드를 작성했기 때문이다. 37500*0.8의 답은 소수점이 아니기 때문이다.
그러나 실제로 코드가 실행 되었을 때, 0.8은 정수가 아니기 때문에 0이 되어 코드가 실행되었고 3일 이후부터는 값이 0원으로 나왔다.
이를 고치기 위해서는 double로 실수값을 설정해주는 작업이 필요하다. 날짜와 돈이 변수가 된다.

과제 2. c언어 코드 복습

```
#include <stdio.h>

        int f1(int num1)
{
         int i=0, b=37500;
         double rate = 1.0;

         if(3<=num1)
         {
           rate=0.8;
           i= num1*b*rate;
         }
        else
         {
                 i= num1*b;
         }
         return i;
}
int main(void)
{
   int num1;
   printf("대여하는 일수를 입력해주세요 :");
   scanf("%d", &num1);

printf("금액 = %d₩n",f1(num1));

       return 0;
}
```

과제 2. c언어 코드 복습 – 3번 문제

```
#include <stdio.h>

    // synthesis : first – start, second – end,
    three - times
    int  syn(int start, int end, int times)
    {
        int res = 0, i = start;

        while(i < end + 1)
        {
            if(! ( i  % 3))
            {
                res += i;
            }
            ++i;
        }
        return res;
    }


    int main(void)
    {
        printf("tot series sum
    = %d₩n",  syn(1, 1000, 3));
        return 0;
    }
```

```
#include <stdio.h>

int sum(int s, int e, int t)
{
    int i, sum = 0;

    for(i = s; i <= e; i++)
        if(!(i % 3))
            sum += i;

    return sum;
}

int main(void)
{
    printf("sum of 3 series = %d₩n", sum(1, 1000, 3));
    return 0;
}
```

과제 2. c언어 코드 복습

```
#include <stdio.h>

void main(void)
{
	int i;

while (i = 0; i <= 10000; i++)
{
	if (i / 2 == 0)
	printf("%d", i)
i++;
}
```

위의 답안의 2개는 선생님이 작성하신 것으로 각각 while 문과 for 문으로 작성된 것이다. 옆의 코드는 처음에 작성한 것으로 통메인을 함수를 따로 하지 않고 통메인으로 만든 것이다.

통메인보다는 함수를 따로 잡는 연습이 필요하므로 다른 식으로 수정이 필요하다.