C - HW3

임베디드스쿨1기

Lv1과정

2020. 08. 06

김인겸

7. C Programming Function

1) 함수란?
함수는 프로그램의 반복을 피하기위해 만든 기능이다.
여러 개 의 함수로 인해 모듈화가 가능하다.

2) 용어 정도는 알아두자
return type
fuction name
parameters
fuction body

7. C Programming Function

3) 함수 사용법

맨 위에 글로벌 변수로 선언된 변수 a와 main함수 내에서 선언된 변수 a는 변수 이름은 같지만 서로 다른 메모리 주소를 의미한다.

main함수 내에서 선언된 변수 a가 우선순위가 더 높기 때문에 main함수 내에서 print된 a 값은 10이 된다.

```c
#include <stdio.h>

int a = 20;

int sum(int a, int b);

int main(void)
{
    int a = 10;
    int b = 20;
    int c = 0;

    printf("value of a in main() =  %d\n", a);
    c = sum(a, b);
    printf("value of c in main()( = %d\n", c);

    return 0;
}

int sum(int a, int b)
{
    printf("value of a in sum() = %d\n", a);
    printf("value of b in sum() = %d\n", b);

    return a + b;
}

```

7. C Programming Function

왜 값이 바뀌지 않지? 1

13번째 줄에서
함수 swap(a,b)를 쓸 때에는
a와 b의 값만 복사되기 때문에
swap함수를 사용한 뒤에
a와 b의 값은 변하지 않는다

```c
#include <stdio.h>

void swap(int x, int y);

int main(void)
{
    int a = 100;
    int b = 200;

    printf("Before swap, value of a : %d\n", a);
    printf("Before swap, value of b : %d\n", b);

    swap(a,b);

    printf("After swap, value of a : %d\n", a);
    printf("After swap, value of b : %d\n", b);

    return 0;
}

void swap(int x, int y)
{
    int temp;

    temp = x;
    x = y;
    y = temp;

}

```

7. C Programming Function

왜 값이 바뀌지않지? 2

다음 예제를 통해
a와 b의 값은 변하지 않지만
swap함수를 통해
x와 y의 값만 바뀌었음을 알 수 있다.

함수를 쓸 때에는 값만 복사된다는 것에 유의하자!

```
Before swap, value of a : 100
Before swap, value of b : 200
Before swap, value of x : 100
Before swap, value of y : 200
After swap, value of x : 200
After swap, value of y : 100
After swap, value of a : 100
After swap, value of b : 200
```

```c
#include <stdio.h>

void swap(int x, int y);

int main(void)
{
    int a = 100;
    int b = 200;

    printf("Before swap, value of a : %d\n", a);
    printf("Before swap, value of b : %d\n", b);

    swap(a,b);

    printf("After swap, value of a : %d\n", a);
    printf("After swap, value of b : %d\n", b);

    return 0;
}

void swap(int x, int y)
{
    int temp;

    printf("Before swap, value of x : %d\n", x);
    printf("Before swap, value of y : %d\n", y);

    temp = x;
    x = y;
    y = temp;

    printf("After swap, value of x : %d\n", x);
    printf("After swap, value of y : %d\n", y);

}

```

7. C Programming Function

이건 왜 값이 바뀔까?

a와 b의 주솟값을
포인터변수 x와 y에 넣어주고

*x(a의 주소가 가르키는 값)과
*y(b의 주소가 가르키는 값)을
바꿔주는 작업을 진행하면

a값과 b값이 서로 바뀌게 된다.

이전 예제와의 차이점:
변수는 함수 내에서만 유효하지만
주솟값은 코드 전체에서 유효하다

```c
#include <stdio.h>

void swap(int *x, int *y);

int main(void)
{
    int a = 100;
    int b = 200;

    printf("Before swap, value of a : %d\n", a);
    printf("Before swap, value of b : %d\n", b);

    swap(&a, &b);

    printf("After swap, value of a : %d\n", a);
    printf("After swap, value of b : %d\n", b);

    return 0;
}

void swap(int *x, int *y)
{
    int temp;

    temp = *x;
    *x = *y;
    *y = temp;

}
```

8. C Programming Array

배열이란 한 가지의 자료형을 나열한 것이다.
배열을 선언하면 손쉽게 원하는 메모리 공간을 확보할 수 있다.

8. C Programming Array

HW1. 두 개의 문자열 바꾸기

```
Before swap, str1 = angel
Before swap, str2 = devil
After swap, str1 = devil
After swap, str2 = angel
```

```c
#include <stdio.h>

int main(void)
{
    char str1[] = "angel";
    char str2[] = "devil";
    char temp[5];
    int i, j;

    printf("Before swap, str1 = %s\n", str1);
    printf("Before swap, str2 = %s\n", str2);

    for(i = 0 ; i < 5 ; i ++){
        temp[i] = str1[i];
        str1[i] = str2[i];
        str2[i] = temp[i];
    }

    printf("After swap, str1 = %s\n", str1);
    printf("After swap, str2 = %s\n", str2);

    return 0;
}

```

8. C Programming Array

HW2. 2By2 행렬의 곱셈게산

```c
#include <stdio.h>

int main(void)
{
    int a[2][2] = { {1, 2}, {3, 4} };
    int b[2][2] = { {5, 6}, {7, 8} };
    int sum[2][2];
    int i, j;

    sum[0][0] = a[0][0] * b[0][0] + a[0][1] * b[1][0];
    sum[0][1] = a[0][0] * b[0][1] + a[0][1] * b[1][1];
    sum[1][0] = a[1][0] * b[0][0] + a[1][1] * b[1][0];
    sum[1][1] = a[1][0] * b[0][1] + a[1][1] * b[1][1];

    for(i = 0 ; i < 2 ; i++){
        for(j = 0 ; j < 2 ; j++){
            printf("%d\n", sum[i][j]);
        }
    }

    return 0;
}
```

```
19
22
43
50
```

8. C Programming Array

HW3. 최댓값 찾는 함수 만들기

```c
#include <stdio.h>

int max_function(int arr[][3]);

int main(void)
{
    int arr[3][3] = { {1, 15, 4}, {8, 10, 16}, {2, 7, 20} };
    int max ;

    max = max_function(arr);

    printf("arr의 최댓값은 %d 입니다.\n", max);

    return 0;
}

```

(밑으로)

```c
int max_function(int arr[][3])
{
    int max[3];
    int result;
    int i;

    for(i = 0 ; i < 3 ; i++){                                   // arr[0][0~2]의 최댓값을 max[0]에 넣고
                                                                // arr[1][0~2]의 최댓값을 max[1]에 넣고
    if((arr[i][0] > arr[i][1]) && (arr[i][0] > arr[i][2]))      // arr[2][0~2]의 최댓값을 max[2]에 넣는 반복문
        max[i] = arr[i][0];
    else if(arr[i][1] > arr[i][2])
        max[i] = arr[i][1];
    else
        max[i] = arr[i][2];
    }

    if((max[0] > max[1]) && (max[0] > max[2]))                  // max[0~2]중에서 최댓값을 비교하는 조건문.
        result = max[0];
    else if(max[1] > max[2])
        result = max[1];
    else
        result = max[2];

    return result;
}
```

```
(base) thgyeomkim@thgyeomk
arr의 최댓값은 20 입니다.
```

8. C Programming Array

HW4. 오름차순 함수 만들기

```
2
3
17
50
1000
```

```c
#include <stdio.h>

int* ascending_order(int num[]);

int main(void)
{
    int balance[] = {1000, 2, 3, 17, 50};
    int i;

    int* balance_ptr = ascending_order(balance);

    for(i = 0 ; i < 5 ; i++)
        printf("%d\n", *(balance_ptr + i) );

    return 0;
}
int* ascending_order(int num[]) // 배열을 리턴하는 함수
{
    int i, j;
    int temp;

    for(i = 0 ; i < 4 ; i++){
        for(j = 1 ; j <= 4-i ; j++){
            if(num[i] > num[i+j]){
                temp = num[i];
                num[i] = num[i+j];
                num[i+j] = temp;
            }
        }
    }
    return num;
}
```

9. C Programming Pointer

포인터란 주솟값을 담는 변수다.
int자료형 변수가 4byte 정수를 담을 수 있는 변수인 듯이
포인터도 주솟값을 담을 수 있는 변수이다. 이때의 주소는 가상의 메모리 주소이다.

- 일반적인 변수에 &(ampersand)를 붙이면 주소를 의미한다
  ex) int a;
  &a: a라는 변수의 주솟값

- 포인터를 선언하는 방법 : 변수를 선언할 때 *(asterisk)를 붙여준다
  ex) int *a; (int형 변수의 주솟값을 담을 수 있는 포인터a라는 뜻)
  a는 주소를 담을 수 있는 변수가 되었다.

- 포인터가 가르키는 주소의 값을 확인하는 방법 : *(asterisk)를 붙여준다
  ex) int a = 40;
  int *ptr = &a;
  printf("%d", *ptr);

9. C Programming Pointer

- 배열의 이름은 배열[0]이 가르키는 주소를 의미한다.
  ex) int array[5];
  int *ptr1 = array;
  int *ptr2 = &array[0];
  이때 ptr1과 ptr2는 같은 값이다.

- 포인터에 1을 증가시키면 자료형의 크기만큼 주소가 더해진다.(배열과 자주쓰임)
  ex) int array[] = {10, 20, 14, 9, 8};
  int *ptr = array;
  ptr++; (이때 ptr값은 array[1]의 주소를 나타낸다)
  ptr++; (이때 ptr값은 array[2]의 주소를 나타낸다)

- 함수에서 포인터를 리턴하는 방법
  1. 반환자료형에 *(asterisk)를 붙여준다
  2. 리턴된 값은 포인터변수에 넣어주면 끝

- 포인터를 가르키는 주솟값을 담는 변수인 이중 포인터도 있다.

9. C Programming Pointer

- rand()함수 : 사전에 정의된 여러 개의 난수 리스트들 중에서 하나의 리스트를 랜덤으로 선택.
  출력하게 되면 계속 똑같은 수가 나옴

- srand()함수 : 괄호 안에 임의의 숫자를 직접 입력하면 그 숫자에 해당하는 난수 리스트를 정할 수 있음
  ex)srand(21), srand(22) 등등...

- time(NULL) : 1970년 1월 1일 00시부터 흐른 시간을 초 단위로 알려주는 함수.

- srand((unsigned) time(NULL)) : srand의 괄호 안의 값이 계속해서 바뀌므로 매초마다 난수들의
  리스트가 바뀜(무작위)

- rand()와 srand()는 <stdlib.h>라이브러리를 포함해야 된다
- time()은<time.h>라이브러리를 포함해야 된다

9. C Programming Pointer

HW1.

```
Int main(void)
{
        char c = ‘a’;
        int n = 7;
        double d = 3.14;
}
```

0x1000	a(97)
0x1001	7
0x1002	
0x1003	
0x1004	
0x1005	3.14
0x1006	
0x1007	
0x1008	
0x1009	
0x100a	
0x100b	
0x100c	

9. C Programming Pointer

HW2.

```
Int main(void)
{
	int a[2][3] = { {0, 1, 2}, {3, 4, 5} };
}
```

0x1000	0
0x1004	1
0x1008	2
0x1012	3
0x1016	4
0x1020	5

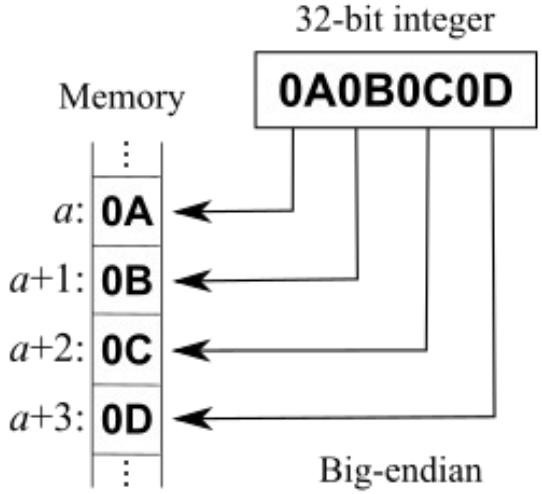
9. C Programming Pointer

HW3은 잘모르겠습니다..
다차원배열이랑 포인터 공부 좀만 더 하고 가겠습니다

Big endian, Little endian

Big endian :

주소값이 작은 메모리에 큰 값부터 저장되는 방식
0A > 0B > 0C > 0D이므로
0A가 가장작은 메모리에 할당된다



Little endian :

주소값이 작은 메모리에 작은 값부터 저장되는 방식
0A > 0B > 0C > 0D이므로
0D가 가장 작은 메모리에 할당된다