함수와 기억부류 (Storage Class)

Jee-Hwan Ryu

School of Mechanical Engineering
Korea University of Technology and Education

함수의 원형

- 되돌림 자료형 함수이름(매개변수 리스트)
  { 변수선언 및 초기화
    여러 C 구문 }

함수간의 데이터 전달

- 값에 의한 호출 (call by value)
- 참조에 의한 함수 호출 (call by reference): 본질적으로 데이터가 저장되어 있는 시작번지를 전달한다는 점에서 그 원리는 모두 같다.
  - 1차원, 2차원 배열의 전달
  - 포인터의 전달
  - 포인터 배열의 전달

1차원 배열의 전달

```
#include<stdio.h>

int GetSum(int *iptr);

void main()
{   int arr[10] = { 10,20,30,40,50,60,70,80,90,100 };
    printf("%p₩n", arr);    /* Array Start Address */

    printf("Total Sum is %d₩n", GetSum(arr));
}

int GetSum( int *iptr )
{   int sum = 0 ,i;
    printf("%p₩n", iptr);     /* Pointer Value */
    for(i = 0; i < 10; i++)
        sum += *iptr++ ;
    return(sum);
}
```

2차원 배열의 전달

```
#include<stdio.h>
#define M 5
#define N 8

void  SetArr( int brr[][N], int dat);   //열의 크기는 반드시 선언 필요
void main()
{    int arr[M][N], j,k;
     SetArr(arr ,1);
     for(j = 0; j < M; j++){
        for(k = 0; k < N; k++)
                printf("%2d", arr[j][k]);
        printf("₩n");
     }
}
void SetArr( int brr[][N] , int dat )
{    int j , k;
     for(j = 0; j < M; j++)
         for(k = 0; k < N; k++)
                          brr[j][k] = dat;
}
```


포인터의 전달

```
#include<stdio.h>

void Exchg1(int x, int y);
void Exchg2(int *x, int *y);

void main()
{    int a = 2, b = 3;  /* Initial Value */
     Exchg1(a,b);    //값에 의한 호출
     printf("a = %d, b = %d₩n", a,b);
     Exchg2(&a, &b);  //참조에 의한 호출
     printf("a = %d, b = %d₩n", a,b);
}

void Exchg1(int x, int y)
{    int temp;
     temp = x; x = y; y = temp;
}

void Exchg2(int *x, int *y)
{    int temp;
     temp = *x; *x = *y; *y = temp;
}
```

포인터 배열의 전달

```
#include<stdio.h>

void Display( char *sptr[] );
void main()
{
    static char *name[] = { "CHA   IN PYOU"  ,
                                                "KANG  HO DONG"  ,
                                                "CHAE  SI LA"    ,
                                                "GO    SO YOUNG" ,
                                                "/"              };
    Display(name);
}
void Display( char *sptr[] )
{
    int i = 0;
    while( *sptr[i] != '/' ){
            printf("%s", sptr[i]);
            printf("₩n");
            i++;
    }
}
```


기억부류 (Storage Class)

- 변수와 함수의 통용범위에 관한 속성
  - 형(type): 데이터의 크기
  - 기억부류: 범위
- 기억부류 지정자
  - auto
  - static
  - register
  - extern
  - 변수선언 예제: static int i_var;

전역변수와 지역변수

- 변수의 성격은 변수를 선언하는 기억부류 지정자와 선언장소(함수의 내부이냐 외부이냐)에 의해 결정된다.
- 하나의 블럭은 '{' 로 시작해서 '}'로 끝이 나며, 이 내부에서 선언된 변수는 이 블록 안에서만 사용 가능
- 모든 블록의 외부에서 선언되면 다른 블록에서도 사용 가능
- 전역변수는 초기화 시키지 않아도 0의 값으로 정해지고, 지역변수는 임의의 값을 가지게 된다.

global.c

```
#include<stdio.h>

void OutPut();

int i_var = 1;

void main()
{
    printf("i_var = %d₩n", i_var);
    printf("c_var = %c₩n", c_var);   //c_var is not defined
    OutPut();
}
void OutPut()
{    char c_var = 's';
    printf("i_var = %d₩n", i_var);
    printf("c_var = %c₩n", c_var);
}
```

자동변수 (auto)

- 지금까지 사용한 변수들은 자동변수들이 대부분
  - auto int i; = int i;
- 메모리 내의 스택 영역에 위치
- 함수시작에서 생성, 함수가 끝날 때 없어짐
- 임시적으로 이용하는 변수
- 전역변수로 사용할 수 없다.

- 모든 블록 바깥에 “int i;” 라고 하면 static(정적) 변수로 내정되어 있다.

정적변수와 정적함수 (static)

- 스택이 아닌 메모리의 정적 구간에 생성되고 프로그램 자체가 끝날 때까지 보존되는 정적인 의미

```
#include<stdio.h>
void count();
void main()
{   int i;
    for(i = 0; i < 10; i++)
            if(getch()) count();
}
void count()
{   auto int aui = 0;
    static int sta = 0;
    static int stb;
    stb = 0;
    printf("aui = %d  ", aui++);
    printf("sta = %d  ", sta++);
    printf("stb = %d₩n", stb++);
}
```

레지스터 변수

- 스택이나 정적영역은 RAM에 존재
- CPU내부의 레지스터는 RAM 보다 입출력 속도가 빠르다.
- 자주쓰는 변수는 레지스터 변수로 사용하는 것이 좋다
- 레지스터 개수에 한계가 있다.
- register int i;

외부변수와 외부함수(extern) 그리고 분할 컴파일

- 하나의 프로그램을 짜기위해 여러 사람이 각기 다른 파일에서 작업을 하는 경우
- 다른 사람이 작성한 변수나 함수를 사용해야 하는 경우가 있다.
- 변수나 함수를 extern 형으로 선언하면, 그 변수나 함수가 다른 화일내에 존재한다는 것을 알려줌

```
#include <stdio.h>
int s_count; //외부변수 정의
void display()
{

}
```

```
#include <stdio.h>
extern int s_count; //외부변수 선언
extern void display();
main()
{.....
display();
printf("%d", s_count);
.....
}
```

분명히 등록되어 있는 것이니 딴 파일에서 찾아봐라!

헤더화일의 외부변수 선언

- 각 파일마다 extern을 선언해 주어야 하나?
- Header 파일에 함수와 변수에 대한 원형을 선언

```
#include <stdio.h>
int s_count; //외부변수 정의
void display()
{

}
```

```
#include <stdio.h>
#include “sub.h”
main()
{.....
display();
printf(“%d”, s_count);
.....
}
```

stu.h

```
extern int s_count;
extern void display();
```


프로젝트 컴파일