14. 함수와 포인터 활용

12주차

call by value, call by reference

- 값에 의한 호출(call by value)
  - C 언어의 기본적인 인자 전달 방식
  - 함수 호출 시 실인자의 값이 형식인자에 복사되므로 함수 내에서 형식인자를 수정해도 실인자에는 영향이 없음.

- 참조에 의한 호출 (call by reference)
  - 포인터를 매개변수로 사용하면 실인자와 형식인자가 같은 주소를 갖게 되어 변수의 변화된 값을 공유할 수 있음.

Call by value 사용
(callbyvalue.c)

```
#include <stdio.h>

void increase(int origin, int increment);

int main(void) {
    int amount = 10;
    //amount가 20 증가하지 않음
    increase(amount, 20);
    printf("%d₩n", amount);

    return 0;
}

void  increase(int origin, int increment) {
    origin += increment;
}
```

[결과]
10

amount를 전달 받은 origin 값을 수정했지만 amount에는 영향이 없음.

Call by reference 사용
(callbyreference.c)

```
#include <stdio.h>

void increase(int *origin, int increment);

int main(void) {
    int amount = 10;
    increase( &amount, 20 );
    printf("%d₩n", amount);
    return 0;
}

void increase( int *origin, int increment) {
    *origin += increment;
}
```

[결과]
30

함수로 주소를 전달
-> 함수 내에서 주소를 간접참조하여 값을 수정
-> 호출한 함수에서 변경된 값 확인 가능.

배열을 매개변수로 전달 (arrayparameter.c)

```
#include <stdio.h>
#define ARYSIZE 5
double sum(double g[], int n);
int main(void) {
    double data[] = { 2.3, 3.4, 4.5, 6.7, 9.2 };
    for (int i = 0; i < ARYSIZE; i++)
        printf("%5.1f", data[i]);
    puts("");
    printf("합: %5.1f₩n", sum(data, ARYSIZE));
    return 0;
}
double  sum( double ary[], int n ) {
    double total = 0.0;
    for (int i = 0; i < n; i++)
        total += ary[i];
    return total;
}
```

[결과]
2.3 3.4 4.5 6.7 9.2
합: 26.1

배열 원소 값을 모두 더하는 함수정의 :
- 배열이름은 주소이므로 call by reference 와 같은 의미.
- 배열원소수를 별도의 인자로 전달하기.

다양한 배열원소 참조 방법 (arrayparam.c)

```
#include <stdio.h>
int sumary(int *ary, int SIZE); //int sumary(int ary[], int SIZE)도 가능
int main(void) {
    int point[] = { 95, 88, 76, 54, 85, 33, 65, 78, 99, 82 };
    int aryLength = sizeof(point) / sizeof(int);
    int *address = point;
    int sum = 0;
    for (int i = 0; i < aryLength; i++)
        sum += *(point + i);
    printf("메인에서 구한 합은 %d₩n", sum);
    printf("함수sumary() 호출로 구한 합은 %d₩n", sumary(point, aryLength));
    printf("함수sumary() 호출로 구한 합은 %d₩n", sumary(&point[0], aryLength
    printf("함수sumary() 호출로 구한 합은 %d₩n", sumary(address, aryLength))
    return 0;
}
```

```
//sum += *(point++);      //오류발생
//sum += *(address++);    //가능
```

[결과]
메인에서 구한 합은 755
함수sumary() 호출로 구한 합은 755
함수sumary() 호출로 구한 합은 755

다양한 배열원소 참조 방법 (arrayparam.c)

```
int sumary(int *ary, int SIZE)
{
    int sum = 0;
    for (int i = 0; i < SIZE; i++) {
        //sum += ary[i];
        //sum += *(ary + i);
        sum += *ary++;
        //sum += *(ary++);
    }
    return sum;
}
```

int sumary(int ary[], int SIZE)
도 가능

4가지가 모두 가능!

[결과]
메인에서 구한 합은 755
함수sumary() 호출로 구한 합은 755
함수sumary() 호출로 구한 합은 755
함수sumary() 호출로 구한 합은 755

배열 원소 수 계산 1/2
(arrayfunction.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
void readarray(double[], int);
void printarray(double[], int);
double sum(double[], int);
int main(void) {
    double data[5];
    int arraysize = sizeof(data) / sizeof(data[0]);
    printf("실수 %d개의 값을 입력하세요. ₩n", arraysize);
    readarray(data, arraysize);
    printf("₩n입력한 자료값은 다음과 같습니다.₩n");
    printarray(data, arraysize);
    printf("함수에서 구한 합은 %.3f 입니다.₩n", sum(data, arraysize));
    return 0;
}
```

[결과]

```
실수 5개의 값을 입력하세요.
data[0] = 4
data[1] = 3
data[2] = 7
data[3] = 8
data[4] = 9

입력한 자료값은 다음과 같습니다.
data[0] = 4.00 data[1] = 3.00 data
함수에서 구한 합은 31.000 입니다
```

배열 원소 수 계산 2/2
(arrayfunction.c)

```
void readarray(double data[], int n) {
    for (int i = 0; i < n; i++) {
        printf("data[%d] = ", i);
        scanf("%lf", &data[i]);
    }
}
void printarray(double data[], int n) {
    for (int i = 0; i < n; i++)
        printf("data[%d] = %.2lf ", i, *(data + i));
    printf("₩n");
}
double sum(double data[], int n) {
    double total = 0;
    for (int i = 0; i < n; i++)
        total += data[i];
    return total;
}
```

(data + i)
도 가능

*(data + i)
도 가능

2차원 배열을 함수 인자로 사용 1/2 (twodarrayfunction.c)

```
#include <stdio.h>

double sum(double data[][3], int, int);
void printarray(double data[][3], int, int);

int main(void) {
    double x[][3] = { { 1, 2, 3 },{ 7, 8, 9 },{ 4, 5, 6 },{ 10, 11, 12 } };

    int rowsize = sizeof(x) / sizeof(x[0]);
    int colsize = sizeof(x[0]) / sizeof(x[0][0]);
    printf("2차원 배열의 자료값은 다음과 같습니다.₩n");
    printarray(x, rowsize, colsize);
    printf("2차원 배열 원소합은 %.3lf 입니다.₩n", sum(x, rowsize, colsize));

    return 0;
}
```

[결과]

```
2차원 배열의 자료값은 다음과 같습니
 1행원소: x[0][0] =  1.00   x[0][1] =
 2행원소: x[1][0] =  7.00   x[1][1] =
 3행원소: x[2][0] =  4.00   x[2][1] =
 4행원소: x[3][0] = 10.00   x[3][1] =

2차원 배열 원소합은 78.000 입니다.
```

2차원 배열의 행/열 개수를 알아내어 함수에 전달.

2차원 배열을 함수 인자로 사용 1/2 (twodarrayfunction.c)

```
void printarray(double data[][3], int rowsize, int colsize) {
    for (int i = 0; i < rowsize; i++) {
        printf("% d행원소: ", i + 1);
        for (int j = 0; j < colsize; j++)
            printf("x[%d][%d] = %5.2lf    ", i, j, data[i][j]);
        printf("₩n");
    }
    printf("₩n");
}
double sum(double data[][3], int rowsize, int colsize) {
    double total = 0;
    for (int i = 0; i < rowsize; i++)
        for (int j = 0; j < colsize; j++)
            total += data[i][j];
    return total;
}
```

가변 인자(variable argument)

- 인자의 수와 자료형이 결정되지 않은 함수 인자 방식
- 인자들 중 맨 뒤에 위치 : (예) printf, scanf
- 함수 정의 시 … 으로 기술

  void vatest(int numargs, …)
  - 가변인자 … 시작 전의 고정 매개변수에서 가변인자 처리에 필요한 정보를 줘야 함. (numargs가 가변인자의 개수)
- 함수에서의 가변 인자 처리 과정

  #include <stdarg.h>
- 1. 가변인자 선언 : va_list argp;
  - 마치 변수선언처럼 가변인자로 처리할 변수를 하나 만듬
- 2. 가변인자 처리 시작 : va_start(argp, numargs);
  - 선언된 변수에서 마지막 고정 인자를 지정해 가변 인자의 시작 위치를 알리는 방법
- 3 가변인자 얻기 : va_arg(argp, int);
  - 가변인자 각각의 자료형을 지정하여 가변인자를 반환 받는 절차
- 4 가변인자 처리 종료 : va_end(argp);
  - 가변 인자에 대한 처리를 끝내는 단계

가변인자 처리 함수
(vararg.c)

```
#include <stdio.h>
#include <stdarg.h>
double avg(int count, ...);
int main(void) {
    printf("평균 %.2f₩n", avg(5, 1.2, 2.1, 3.6, 4.3, 5.8));
    return 0;
}
double avg(int numagrs, ...) {
    va_list argp;
    va_start(argp, numagrs);
    double total = 0; //합이 저장될 변수
    for (int i = 0; i < numagrs; i++)
        total += va_arg(argp, double);
    va_end(argp);
    return  total / numagrs;
}
```

[결과]
평균 3.40