

C Programming Assignment / Week 9

정리노트 #6

- 함수

장점

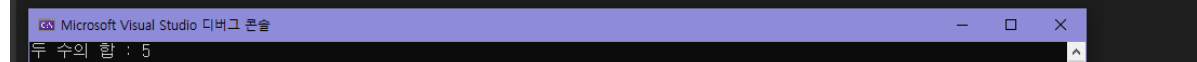
1. 반복적으로 실행해야 할 내용을 함수로 만들어 필요할 때마다 호출해서 사용한다.
2. 프로그램이 모듈화되므로 읽기 쉽고, 디버깅과 편집이 수월하다.
3. 프로그램의 기능과 구조를 한눈에 알아볼 수 있다.
4. 다른 프로그램에서도 재사용 가능하다.

코드를 빌드하는데 있어서 여러번 재사용되는 함수나, 좀 깔끔하게 만들고 싶을 때 사용할 수 있다.

함수를 선언하고, 그 함수가 정의까지 되었다면, 호출로 그 함수를 사용할 수 있다.

함수 선언을 하지 않고도 함수 정의 부분을 맨 앞으로 도출 시켜도 함수를 사용할 수 있다.

```
1  #include <stdio.h>
2
3  int Sum2(int a, int b); // 1. 함수를 선언한다
4
5  int main(void)
6  {
7      int x = 2, y = 3;
8      int value;
9
10     value = Sum2(x, y); // 3. 함수를 호출한다, 정의된 함수를 기반으로 계산 후 그 값을 value에 대입
11
12     printf("두 수의 합 : %d\n", value); // value 출력
13
14     return 0;
15 }
16
17 int Sum2(int a, int b) // 2. 함수를 정의한다
18 {
19     int result;
20     result = a + b; // result 계산
21     return result; // result 반환
22 }
```



함수의 유용성 (예제 8-3 활용)

```

#include <stdio.h>

int integral(int start, int end); //integral이란 함수 선언

int main(void)
{
    int result = 0, i; // result = 첫 값부터 끝 값까지 합을 저장, i = for 반복문에 쓰일 정수형 변수

    printf("함수를 사용하지 않고 합 구하기\n");
    for (i = 1; i <= 10; i++) // for 반복문을 이용해서 1~10까지의 합 구하기
        result += i;

    printf("1부터 10까지 합은 %d다.\n", result);

    result = 0;

    for (i = 7; i <= 17; i++) // for 반복문을 이용해 7~17까지의 합 구하기
        result += i;

    printf("7부터 17까지의 합은 %d다.\n", result);
    printf("*****\n");
    printf("함수를 사용하여 합을 구함\n");
    printf("1부터 10까지 합은 %d다.\n", integral(1, 10)); //함수 호출
    printf("7부터 17까지 합은 %d다.\n", integral(7, 17)); // 함수 호출

    return 0;
}

int integral(int start, int end) //함수 정의 (start, end값을 이용)
{
    int sum = 0, i; //sum = 첫 값부터 끝 값까지의 합을 저장하는 용도, i = 반복문에 쓰일 정수형 변수
    for (i = start; i <= end; i++) // i가 첫 값부터 끝 값과 작거나 같아질 때 까지 1씩 증가
        sum += i; // sum에 i + (i+1) + (i+2)...값을 저장

    return sum; //sum 값을 반환
}

```

Microsoft Visual Studio 디버그 × + ▾

```

함수를 사용하지 않고 합 구하기
1부터 10까지 합은 55다.
7부터 17까지의 합은 132다.
*****
함수를 사용하여 합을 구함
1부터 10까지 합은 55다.
7부터 17까지 합은 132다.

```


여기서 볼 수 있듯이 함수를 사용하면 프로그램이 더 간결해진다. 또한 사용자가 원하는 것을 수시로 반영할 수 있는 형태로도 함수 작성이 가능하다.

함수

앞에는 함수를 선언하고, 정의하고, 호출하여 사용했지만, 함수의 선언을 생략하고 바로 정의로

넘어갈 수도 있다.

```
1 #include <stdio.h>
2
3 int SumTwo(int a, int b) // 함수의 선언 대신 바로 정의 작성
4 {
5     int result;
6     result = a + b;
7     return result;
8 }
9
10 int main(void)
11 {
12     int x = 10, y = 5;
13     int value;
14
15     value = SumTwo(x, y); // 정의된 함수를 호출하여 사용
16
17     printf("두 수의 합: %d\n", value);
18
19     return 0;
20 }
```



위와 같이 정의를 맨 위에 쓴 다음 호출하여 사용할 수도 있다.

재귀함수

재귀함수란, 함수에서 그 함수를 다시 호출하는 것을 말한다. 데이터가 N개로 이루어진 문제가 N-1의 문제를 해결하면 간단하게 해결될 때 사용한다.

대표적으로 팩토리얼(!)이 있다.

재귀함수를 활용한 팩토리얼 구하기

```
#include <stdio.h>
```

```
int factorial(int n); // 함수의 선언 부분
```

```
int main(void)
```

```
{
```

```
int fact_num; // 변수 fact_num 추가
```

```
fact_num = factorial(9); // factorial(9) 를 대입하여 계산된 값을 fact_num에 넣음
```

```
printf("10 팩토리얼 : %d\n", fact_num); /* fact_num에는 factorial(10)이 들어가 있음, 아래의 정의된 부분에서 factorial(10)을 계산한 값이 나옴*/
```

```
return 0;
```

```
}
```

```
int factorial(int n) // 함수의 정의 부분
```

```
{
```

```
if (n <= 1) //n이 1이거나 그보다 작을 경우에는
```

```
return(1); // 1 출력
```

```
else // 그 외의 숫자들은
```

```
return(n * factorial(n - 1)); /*factorial(10)을 기준으로, 10 * factorial(9) 라는 값//을 반환, factorial(9)
는 다시한번 정의에 기반하여 계산, 이렇게 함수안에 함수가 없는 if문의 return(1)이 나올때 까지
반복*/
```

```
}
```

```
1 #include <stdio.h>
2
3 int factorial(int n); // 함수의 선언 부분
4 int main(void)
5 {
6     int fact_num; // 변수 fact_num 추가
7     fact_num = factorial(9); // fact_num에 선언한 함수 factorial에 n=9를 대입하여 계산된 값을 fact_num에 넣음
8
9     printf("9 팩토리얼 : %d\n", fact_num); // fact_num에는 factorial(9)이 들어가 있음, 아래의 정의된 부분에서 factorial(9)을 계산한 값이 나옴.
10
11     return 0;
12 }
13
14 int factorial(int n) // 함수의 정의 부분
15 {
16     if (n <= 1) //n이 1이거나 그보다 작을 경우에는
17         return(1); // 1 출력
18     else // 그 외의 숫자들은
19         return(n * factorial(n - 1)); /*factorial(9)을 기준으로, 9 * factorial(8) 라는 값을 반환, factorial(9)는 다시한번 정의에 기반하여 계산,
20         이렇게 함수안에 함수가 없는 if문의 return(1)이 나올때 까지 반복*/
21 }
```

scanf를 활용하여 사용자가 입력한 값을 기반으로 한 재귀함수도 활용 가능하다.

사용자 정의 함수 (헤더파일)

사용자가 만든 함수의 선언은 일반적으로 사용자가 만든 헤더파일에 포함한다.

우측 솔루션 탐색기에서 헤더파일 폴더에, 추가 > 새항목 > 헤더 파일을 만든 후, 임의의 사용자

지정 함수를 생성할 수 있다.

배열을 인자로 사용하는 함수

이 함수같은 경우에는 수업 마지막에 265p 예제 11번으로 정리했다.

페어 프로그래밍을 활용하여 같이 완성하였다.

문제. 정수형 배열 `b[] = { 20, 34, 12, 24, 54, 91, 9, 40, 81, 10 }`의 최댓값, 최솟값, 평균, 표준편차를 구하고 출력하는 프로그램 만들기

프로그래밍 방향 지시: 정민수

코드 작성자: 정재현

정민수: 일단 오늘 배운 함수의 선언, 정의, 호출을 이용하여 프로그램을 작성할 것 입니다.

최대, 최소, 평균, 표준편차를 구하기 위한 함수를 선언해 주세요. 그리고 후에 제곱근을 사용하기 위한 `math.h`도 추가해주세요.

정재현:

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int MAXX(int b[]); // 함수 선언
5  int MINN(int b[]); // 함수 선언
6  int AVERAGE(int b[]); // 함수 선언
7  int PYO(int b[]); // 함수 선언
8
```

정민수: 그 다음으로는, `int main()`으로 시작하고, 배열 `b[]`와, 위에 상응하는 함수들을 호출한 다음, 대입할 변수들을 선언해주세요. 그리고, 출력화면도 미리 만들어주세요.

정재현:

```

10 int main()
11 {
12     int b[] = { 20,34,12,24,54,91,9,40,81,10 }; // 정수형 배열 b[]
13     int maxValue = MAXX(b); // 최댓값을 구하기 위한 MAXX(b)와, 그 값을 maxValue에 대입
14     int minValue = MINN(b); // 최솟값을 구하기 위한 MINN(b)와, 그 값을 minValue에 대입
15     int average = AVERAGE(b); // 평균값을 구하기 위한 AVERAGE(b)와, 그 값을 average에 대입
16     int pyojunpyeoncha = PYO(b); // 표준편차를 구하기 위한 PYO(b)와, 그 값을 pyojunpyeoncha에 대입
17
18     printf("최댓값은: %d\n", maxValue);
19     printf("최솟값은: %d\n", minValue);
20     printf("평균은: %d\n", average);
21     printf("표준편차는: %d", pyojunpyeoncha);
22 }

```

정민수:

이제 최댓값을 구하는 MAXX함수를 정의해봅시다. 예전 수업시간에 사용했던 배열 내의 변수들을 for문을 사용하여 하나하나씩 비교하고, if문을 추가하여 비교를 이용한 전 수업시간에 배운 최댓값 구하기를 해봅시다.

정재현:

```

23 int MAXX(int b[]) // 함수 정의
24 {
25     int max = 0, i; // 최댓값 max = 0으로 변수 초기화 선언, i변수 선언
26     for (i = 0; i < 10; i++) {
27         if (b[i] > max) { // i를 0부터 9까지 넘어서, 배열의 원소 하나하나씩 비교하여 최댓값 출력
28             max = b[i];
29         }
30     }
31     return max;
32 }
33

```

for문을 이용하여 배열 b[i]를 0부터 9까지, 전부 비교할 수 있게 만들었고, if문을 사용하여 저번 수업시간에 배운 최댓값 찾기를 활용했습니다.

정민수: 그러면 MINN도 비슷한 맥락으로 작성해주세요.

정재현:

```

34 int MINN(int b[]) // 함수 정의
35 {
36     int min = 100, i; // 최솟값 min = 100으로 변수 선언, i변수 선언
37     for (i = 0; i < 10; i++) {
38         if (b[i] < min) { // i를 0부터 9까지 넘어서, 배열의 원소 하나하나씩 비교하여 최솟값 출력
39             min = b[i];
40         }
41     }
42     return min;
43 }

```

정민수: 잘하셨습니다! 평균은 이제 배열 원소의 모든 합을 for문을 이용하여 구하고 sum변수에 저장한 후, sum 변수를 배열 원소의 개수만큼 나누면 나올 것 같네요.

정재현:

```

45 int AVERAGE(int b[]) // 함수 정의
46 {
47     int sum = 0; // sum 변수 0으로 초기화 선언
48     int end; // end 변수 선언
49     int i; // i 변수 선언
50     for (i = 0; i < 10; i++) {
51         sum = sum + b[i]; // 배열의 총합 구하기
52     }
53     end = sum / 10; // 평균 구하기
54     return end;
55 }
56

```

정민수: end 변수를 평균값을 저장할 변수로 설정하고, for문을 이용하여 배열의 총합까지 잘 구했네요. 이제 표준편차 구하기만 남았어요.

정민수: 표준편차의 값을 구하고 저장할 변수 하나랑, 평균값을 구하고 저장할 변수 하나를 선언하고, 교과서 236페이지에 있는 표준편차 구하기 식을 활용해서 최종값을 저장할 변수에다 넣고 그 값으로 return을 하면 될 것 같아요.

표준편차 구하기 식 = $\sqrt{(\text{원소} - \text{평균})^2 / \text{for문을 이용하여 원소 10개 다} * / (\text{원소의개수} - 1)}$

정재현:

```

57 int PYO(int b[]) // 함수 정의
58 {
59     int wtf; // wtf변수 선언
60     int avg; // avg변수 선언
61     int i; // i변수 선언
62     int sum = 0; // sum은 평균 구하기용도
63     int x = 0;
64     int end = 0;
65
66     for (i = 0; i < 10; i++) {
67         sum = sum + b[i];
68     }
69     avg = sum / 10;
70
71     for (i = 0; i < 10; i++) {
72         x = x + pow((b[i] - avg), 2);
73     }
74
75     wtf = sqrt(x / 9); // 표준편차 구하는 식, #include <math.h>의 제곱근 구하는 함수 활용
76
77     return wtf;
78 }
79

```

전체의 합과, 평균을 구한 다음, for문을 이용하여 원소 10개 다 구하여 변수 x에 다 더한값을 대입한 후, 제곱근 함수를 활용하여 완성했습니다.



// 이 문제와같이 함수의 인자로 날개의 변수 뿐 아니라, 배열도 함수 인자로 사용할 수 있다.