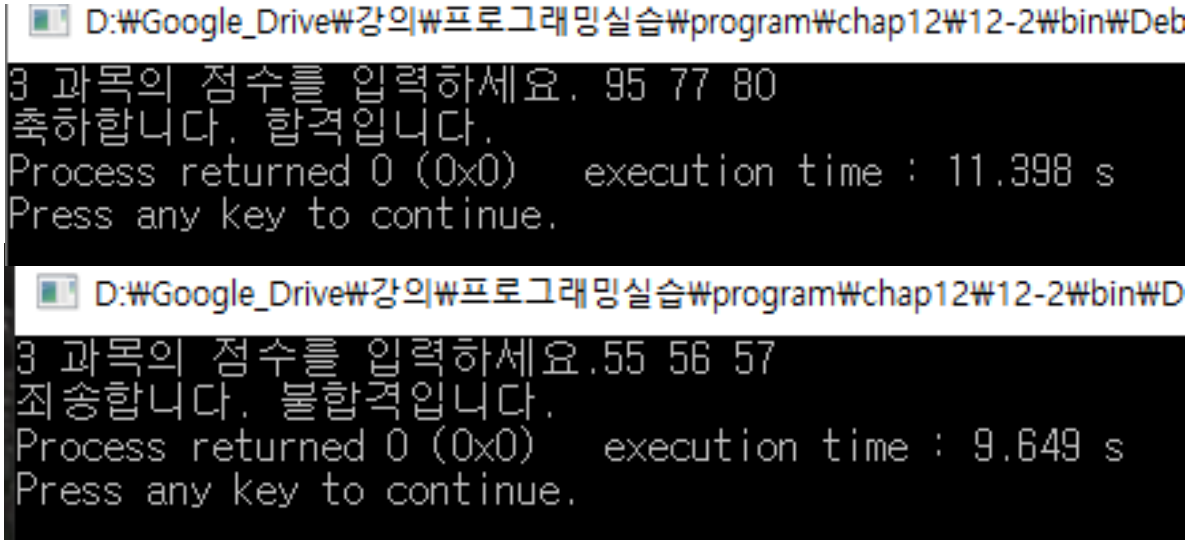
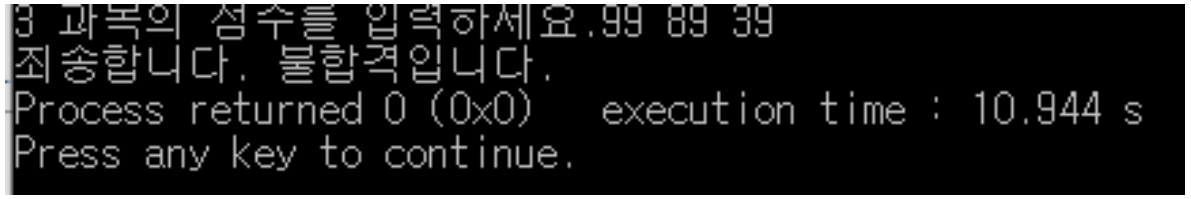
Chapter 12
함수 2

02번
3과목의 정수를 입력받아서 평균이 60 미만이거나
1과목이라도 40점 미만이 있으면 "불합격", 아니면 "합격 " 이라는
메시지를 아래와 같이 출력하는 프로그램을 작성하시오.





```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int sum=0, num;

    printf("3 과목의 점수를 입력하세요.");
    scanf("%d", &num);
    if (num < 40) {
        printf("죄송합니다. 불합격입니다.");
        return 0;
    }
    sum += num;
    scanf("%d", &num);
    if (num < 40) {
        printf("죄송합니다. 불합격입니다.");
        return 0;
    }
    sum += num;
    scanf("%d", &num);
    if (num < 40) {
        printf("죄송합니다. 불합격입니다.");
        return 0;
    }
    sum += num;
    if (sum/3 < 60) {
        printf("죄송합니다. 불합격입니다.");
        return 0;
    }
    else {
        printf("축하합니다. 합격입니다.");
        return 0;
    }
}
```

문제점

3이 큰 수로 바뀔 경우 코드 복사 필요

```
scanf("%d", &num);
if (num < 40) {
    printf("죄송합니다. 불합격입니다.");
    return 0;
}
sum += num;
```

부분 개수 만큼 복사! => 매우 비 효율적 방법

3이라는 상수를 #define으로 정의
반복되는 소스를 반복문 구현 방법

```
#define S_NUM 3
int main()
{
    int sum=0, num;
    int i;

    printf("%d 과목의 점수를 입력하세요.", S_NUM);
    for (i=0; i<S_NUM;i++) {
        scanf("%d", &num);
        if (num < 40) {
            printf("죄송합니다. 불합격입니다.");
            return 0;
        }
        sum += num;
    }
    if (sum/S_NUM < 60) {
        printf("죄송합니다. 불합격입니다.");
        return 0;
    }
    else {
        printf("축하합니다. 합격입니다.");
        return 0;
    }
}
```

설명

- #define S_NUM 3
  매크로 정의를 통해 프로그램 내의 모든 S_NUM을 3로 바꾸어 컴파일한다.

  이렇게 사용하는 주된 목적은 프로그램 작성 중 자료의 개수를 변경하려면 본문 내에서 여 러 곳을 찾아서 변경해야 하므로 매우 불편할 뿐 아니라 실수로 누락시킬 우려가 많기 때문이다.

  이렇게 매크로를 사용하여 숫자 하나만 수정함으로써 프로그램내에서 자료의 개수로 사용된 모든 곳을 찾아내어 수정하는 불편을 덜 수 있다.

  또한 자료의 개수가 너무 커서 디버깅을 하기가 어려울 때 디버깅하는 동안 숫자를 작게 하였다가 최종 컴파일할 때는 원래의 숫자로 바꾸어주는 방법으로도 자주 사용된다.

문제점

전체 입력에 대한 다양한 처리를 추가로 해야할 경우, 소스코드 변화 필요
예) 전체의 합, 짝수 번째 입력값의 합, …

```
for (i=0; i<S_NUM;i++) {
        scanf("%d", &num);
        if (num < 40) {
            printf("죄송합니다. 불합격입니다.");
            return 0;
        }
        sum += num;
    }
```

단일 for 문 안에서 처리 불가

해결책

입력 파트와 처리 파트를 분리

분리 처리를 위해서는 배열 필요

```
#define S_NUM 3

int main()
{
    int sum=0, num;
    int i;
    int score[S_NUM];

    printf("%d 과목의 점수를 입력하세요.", S_NUM);
    for (i=0; i<S_NUM;i++) {
        scanf("%d", &score[i]);
    }

    for (i=0; i<S_NUM;i++) {
        if (score[i] < 40) {
            printf("죄송합니다. 불합격입니다.");
            return 0;
        }
        sum += score[i];
    }

    if (sum/S_NUM < 60) {
        printf("죄송합니다. 불합격입니다.");
        return 0;
    }
    else {
        printf("축하합니다. 합격입니다.");
        return 0;
    }
}
```

문제점

-main 함수에 개발을 계속하면
전체 소스코드를 보고 이해하기 어려움

=>구조화(=함수로 나누기)를 통해

프로그램의 이해가 쉬움
개발과 수정이 편함
여러 사람이 나누어 개발하기 좋음

```
#define S_NUM 3

int main()
{
    int sum=0, num;
    int i;
    int score[S_NUM];

    printf("%d 과목의 점수를 입력하세요.", S_NUM);
    for (i=0; i<S_NUM;i++) {
        scanf("%d", &score[i]);
    }

    for (i=0; i<S_NUM;i++) {
        if (score[i] < 40) {
            printf("죄송합니다. 불합격입니다.");
            return 0;
        }
        sum += score[i];
    }

    if (sum/S_NUM < 60) {
        printf("죄송합니다. 불합격입니다.");
        return 0;
    }
    else {
        printf("축하합니다. 합격입니다.");
        return 0;
    }
}
```

input

pass

구조화 선택시 고려 포인트 => 함수간 자료 전달 방법
함수에게 어떤 정보를 넘겨줄 것인가?
함수로부터 어떤 정보를 넘겨받을 것인가?

매개변수 vs 전역 변수

```
int main()
{
    int score[S_NUM];
    input(score, S_NUM);
    if (pass(score, S_NUM))
        printf("축하합니다. 합격입니다.");
    else
        printf("죄송합니다. 불합격입니다.");
    return 0;
}
void input(int a[], int cnt) {..}
int pass(int a[], int cnt) {..}
```

매개변수

```
int score[S_NUM];
int main()
{
    input();
    if (pass())
        printf("축하합니다. 합격입니다.");
    else
        printf("죄송합니다. 불합격입니다.");
    return 0;
}
void input() {..}
int pass() {..}
```

전역 변수

```
#define S_NUM 3
void input(int a[], int cnt)
{
    int i;
    printf("%d 과목의 점수를 입력하세요.",
cnt);
    for (i=0; i<cnt; i++)
        scanf("%d", &a[i]);
}
int pass(int a[], int cnt)
{
    int i, sum=0, avg;
    for (i=0; i<cnt ; i++)
    {
        if (a[i] < 40) return 0;
        sum += a[i];
    }
    avg = sum /cnt;
    if (avg < 60) return 0;
    return 1;
}
```

```
int main()
{
    int score[S_NUM];
    input(score, S_NUM);
    if (pass(score,S_NUM))
        printf("축하합니다. 합격입니다.");
    else
        printf("죄송합니다. 불합격입니다.");
    return 0;
}
```

매개변수 이용

설명

- void input(**int a[], int cnt**) { ... }
  메인에서 전달받은 배열의 이름을 a라 정하고 a배열의 개수를 전달받아 cnt에 저장한다.

배열을 함수로 전달하는 방법은 호출하는 곳에서는 배열의 이름을, 함수를 선언하는 곳에서는 배열이름 뒤에 "[]"를 붙여주면 된다.

이렇게 하면 **새로운 배열을 생성하는 것이 아니라 배열의 시작위치를 전달**하게 되어 참조에 의한 전달과 똑같이 작동하게 된다. 즉 **함수 내에서 배열의 값에 변화를 주면 원래 배열의 값이 변하게 되는 것이다.**

배열을 인수로 전달하게 되면 그 배열의 위치만 전달이 되므로 전달받은 함수측에서는 배열의 크기를 알 수가 없다.

따라서 배열을 전달할 때에는 크기도 함께 전달하는 경우가 많은데 위 함수에서 cnt가 바로 배열의 크기를 나타내는 것이다.

```
#define S_NUM 3
int score[S_NUM];
void input()
{
    int i;
    printf("%d 과목의 점수를 입력하세요.",
S_NUM);
    for (i=0; i<S_NUM; i++)
        scanf("%d", &score[i]);
}
int pass()
{
    int i, sum=0, avg;
    for (i=0; i<S_NUM ; i++)
    {
        if (score[i] < 40) return 0;
        sum += score[i];
    }
    avg = sum /S_NUM;
    if (avg < 60) return 0;
    return 1;
}
```

```
int main()
{

    input();
    if (pass())
        printf("축하합니다. 합격입니다.");
    else
        printf("죄송합니다. 불합격입니다.");
    return 0;
}
```

전역 변수 이용

```
int score[S_NUM];
void input() {
        ...score[]...
}
```

int score[S_NUM]; 를 함수 밖에서 전역변수 로 선언
각 함수내에서 선언하지 않고 score[]를 사용, **score[]를 공유함**

* 전역변수 장점
1) 모든 함수에서 접근 가능함.
2) 함수내의 지역변수와 달리, 저장된 값이 사라지지 않음

*전역변수 단점
1) 자주 사용하게 되면 결과를 예측하기 어려움
2) 함수 하나를 분석해도 기능 이해 어려움
3) 지역변수, 전역변수의 이름이 겹치는 문제 발생 => 전역과 지역의 혼용
4) 함수 재사용 불가능

*프로그래밍 원칙
가급적 지역변수 활용, 꼭 필요할 때만 전역변수 사용