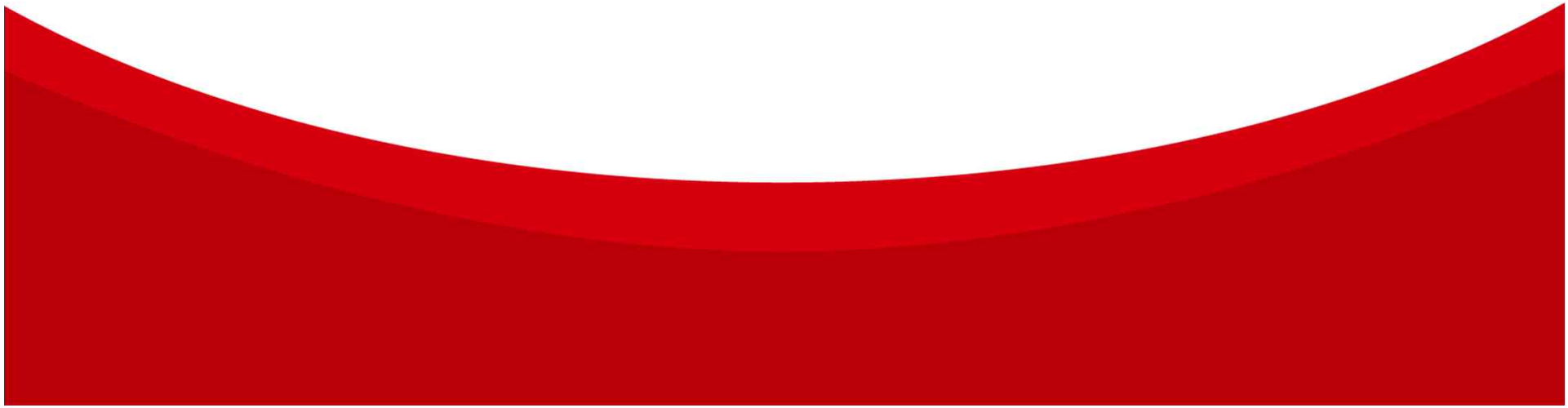
Chapter 11
함수 1



01 Question

사용자 정의 함수를 만들어 선을 그리는 프로그램을 작성하시오.

출력 예

```
==============================
line 함수를 호출하였습니다.
line 함수를 다시 호출합니다.
==============================
```

소스1

```c
#include <stdio.h>

void line()
{
	puts("================================");
}

int main()
{
	line();

	puts("line 함수를 호출하였습니다.");
	puts("line 함수를 다시 호출합니다.");

	line();

	return 0;
}
```

설명

- 함수란, 프로그램 내에서 특정한 작업을 수행하기 위해서 독립적으로 만들어진 프로그램의 단위를 말한다. C언어는 함수의 집합으로 구성되어 있고 프로그램을 실행하면 처음으로 실행되는 함수가 바로 지금까지 작성해 온 main() 함수이다.

- 함수의 종류
  함수는 Compiler를 만든 회사에서 미리 만들어 제공하는 표준라이브러리 함수와 사용자가 직접 정의해서 사용하는 사용자 정의 함수가 있다. 표준라이브러리 함수는 printf(), scanf()와 같이 C언어에 내장되어 있는 함수로 해당 함수가 포함된 헤더파일을 #include 하면 바로 사용이 가능한 것이고 사용자 정의 함수는 위의 프로그램에서 line() 함수와 같이 사용자가 직접 정의해서 사용하는 함수를 말한다.

- void line()
  {
  puts("==========");
  }
  함수의 내용이 기록된 것을 함수의 정의라고 한다. line이라는 함수를 생성하여 정의 한다. 호출을 하면 실행만 하고 끝내는 가장 단순한 형식의 함수이다. 함수도 변수와 같이 리턴값에 따라 자료형을 정의해 주어야 한다. 이 함수와 같이 리턴값이 없는 경우에는 void로 정의한다.

- line();
  "함수의 호출"이라고 하며 함수가 호출되면 실행순서가 호출된 함수로 이동하여 함수의 내용을 실행하고 다시 호출한 곳으로 돌아온다. 위의 예제는 함수에서 선을 출력하는 단순한 작업만 하는 것으로 인수나 리턴값이 없다.

소스2

```c
#include <stdio.h>

void line();

int main()
{
        line();

        puts("line 함수를 호출하였습니다.");
        puts("line 함수를 다시 호출합니다.");

        line();

        return 0;
}
```

```c
void line()
{
        puts("================================");
}
```

설명

- void line();

함수를 호출하기 위해서는 반드시 호출하려는 함수가 현재의 위치보다 앞에 정의되어 있어야 한다. main()함수에서 line() 함수를 호출하는데 line()함수는 main() 함수보다 아래에 정의되어 있어서 문법적으로 에러가 발생한다. 이 때 함수의 위치를 그대로 유지하기 위해서는 main()함수 앞에 호출할 함수의 형태만 미리 선언해 주면 된다. 이것을 프로토타입(함수원형) 선언이라고 한다.

자가진단 1

문자열 "~!@#$^&*()_+|" 를 출력하는 함수를 작성하고 정수를 입력받아 입력받은 수만큼 함수를 호출하는 프로그램을 작성하시오.

입력 예

3

출력 예

~!@#$^&*()_+|
~!@#$^&*()_+|
~!@#$^&*()_+|

02 Question

정수를 입력 받아 10큰 수와 10작은 수를 출력하는 프로그램을 작성하시오.

입력 예 50

출력 예 10큰수 : 60
10작은수 : 40

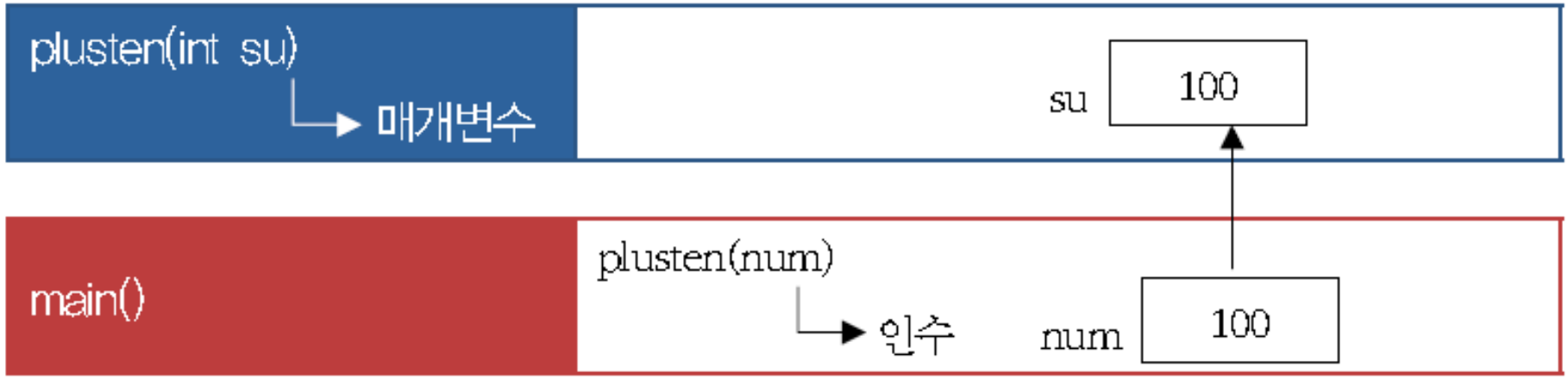
소스

```
#include <stdio.h>

void plusten(int su)
{
	printf("10큰수 : %d \n", su + 10);
}

void minusten(int su)
{
	printf("10작은수 : %d \n", su - 10);
}

int main()
{
	int num;

	scanf("%d", &num);

	plusten(num);
	minusten(num);

	return 0;
}
```

설명

- void plusten(int su) {...}
  함수를 선언하고 정수 한 개를 매개변수로 받아서 su라는 변수에 저장한다.
  이 함수를 호출할 때는 반드시 한 개의 정수를 인수로 전달해야 한다.

- plusten(num);
  num을 인수로 전달하여 plusten 함수를 호출한다.
  만약 num으로 입력받은 값이 100이라면 함수를 호출할 때 그 값을 함께 전달하는데 plusten 함수에서는 넘겨받은 num값 100을 새로 선언한 su 변수에 저장하고 블록 내의 문장들을 실행한다. 위와 같이 함수를 호출하면서 전달하는 값(num)을 인수라고 하고 함수에서 전달받은 값을 저장하는 변수(su)를 매개변수라고 한다. 인수와 매개변수는 위와 같이 명확하게 구분되지만 일반적으로는 같은 의미로 쓰인다. 위의 프로그램에서 num은 main 함수에서 선언된 변수이므로 main 함수 이외의 곳에서는 사용할 수 없으며, su는 plusten 함수에서 선언된 것이므로 그 함수 내에서만 사용이 가능하다. minusten 함수에 선언된 변수 su는 plusten에 같은 이름으로 선언된 것이 있지만 그것과는 전혀 다른 것으로 인식하여 작동하게 된다.


plusten(int su)
매개변수
su
100
main()
plusten(num)
인수
num
100

자가진단 2

반지름의 길이를 전달받아 넓이를 출력하는 함수를 작성하고 반지름의 길이를 입력받아 함수를 호출하여 넓이를 출력하는 프로그램을 작성하시오. (원주율은 3.14로 하고 반올림하여 소수 둘째자리까지 출력한다. 원의 넓이 = 반지름 * 반지름 * 원주율이다.)

입력 예	출력 예
10	314.00

03 Question

정수를 전달받아 출력 예와 같이 '*' 로 이루어진 직각삼각형을 출력하는 함수를 작성하고 입력받은 정수를 전달하여 출력하는 프로그램을 작성하시오.

입력 예

```
5
```

출력 예

```
*
**
***
****
*****
```

소스

```
#include <stdio.h>

void star(int n)
{
	int i, j;

	for (i = 1; i <= n; i++ ) {
		for (j = 1; j <= i; j++) {
			printf("*");
		}
		printf("\n");
	}
}

int main()
{
	int n;

	scanf("%d", &n);
	star(n);

	return 0;
}
```

설명

- void star(int n) {...}
  정수를 넘겨받아 변수 n에 저장하고 n행에 해당하는 삼각형을 출력한다.
  이 때 메인에서 넘겨받아 이 함수 내에 저장된 n은 메인에서 입력받은 n과는 다른 변수이다.

자가진단 3

정수를 전달받아 다음과 같이 숫자 정사각형을 출력하는 함수를 작성하고 함수를 호출하여 입력받은 정수를 함수로 전달하여 출력하는 프로그램을 작성하시오.

입력 예

```
4
```

출력 예

```
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

04 Question

합과 차를 각각 리턴하는 함수를 작성한 후 두 정수를 입력받아 함수를 호출하여 두 수의 합과 차를 출력하는 프로그램을 작성하시오.

입력 예 30 50

출력 예 두 수의 합 = 80
두 수의 차 = 20

소스

```c
#include <stdio.h>

int add(int x, int y)
{
	return x + y;
}

int sub(int x, int y)
{
	int cha = x - y;
	if (cha < 0) cha *= -1;
	return cha;
}

int main()
{
	int a, b, sum;

	scanf("%d %d", &a, &b);

	sum = add(a, b);
	printf("두 수의 합 = %d \n", sum);
	printf("두 수의 차 = %d \n", sub(a,b));

	return 0;
}
```

설명

- int add(int x, int y)
  {
  return x + y;
  }
  두 정수를 x와 y로 받아서 저장했다가 두수의 합을 리턴한다. 이 때 리턴값을 함수의 결과값이라고 하며 함수의 이름앞에 표시되는 자료형은 바로 이 리턴값의 자료형과 일치해야 한다. 위에서 리턴되는 x + y값이 정수이므로 int add(...) 라고 선언한 것이다. 리턴값이 없을 때는 void를 사용하게 된다.

- if (cha < 0) cha *= -1;
  변수 cha의 값이 음수인 경우 -1을 곱하여 양수로 바꾼다.

- sum = add(a, b);
  a와 b를 전달하여 add함수를 실행하고 리턴되는 결과값을 sum에 저장한다.

- printf("두 수의 차 = %d ₩n", sub(a, b));
  sub(a, b)함수를 실행하고 리턴되는 값을 바로 출력한다.

자가진단 4

세 개의 정수를 전달받아 최대값을 구하여 리턴하는 함수를 작성하고 세 정수를 입력받아 최대값을 출력하는 프로그램을 작성하시오.

입력 예	출력 예
-10 115 33	115

- hint

3개의 매개변수가 a, b, c라고 할 때 a가 b보다 크고 c보다도 크다면 a를 리턴하고, 그렇지 않을 경우에는 b와 c 중 큰 것을 리턴하면 된다.

05 Question

평균을 구하는 함수를 작성한 후 세과목의 점수를 입력받아 평균을 구하여 소수 둘째자리까지 반올림하여 출력하는 프로그램을 작성하시오.

입력 예 *세과목의 점수를 입력하세요.* 80 65 95

출력 예 평균 : 80.00

소스1

```c
#include <stdio.h>

double pyung(int a, int b, int c)
{
	int sum = a + b + c;
	return sum / 3.0;
}

int main()
{
	int kor, eng, mat;
	double avg;

	printf("세과목의 점수를 입력하세요. ");
	scanf("%d %d %d", &kor, &eng, &mat);

	avg = pyung(kor, eng, mat);

	printf("평균 : %.2lf \n", avg);

	return 0;
}
```

설명

- double pyung(int a, int b, int c)
  {
  int sum = a + b + c;
  return sum / 3.0;
  }
  정수 3개를 전달 받아 평균을 구하여 리턴한다. 리턴값이 실수이므로 함수를 double로 정의하였다.

- avg = pyung(kor, eng, mat);
  함수를 호출하여 세 값을 보내서 리턴되는 값을 avg에 저장한다.

소스2

```c
#include <stdio.h>

double pyung(int a, int b, int c)
{
	return (a + b + c) / 3.0;
}

int main()
{
	int kor, eng, mat;

	printf("세과목의 점수를 입력하세요. ");
	scanf("%d %d %d", &kor, &eng, &mat);

	printf("평균 : %.2lf \n", pyung(kor, eng, mat));

	return 0;
}
```

설명

- 앞의 소스와 같으나 함수에서 중간 계산과정을 거치지 않고 직접 리턴하였고 main에서도 리턴값을 저장하지 않고 즉시 출력하였다.

자가진단 5

10 이하의 두 정수 m과 n을 입력 받아서 m을 n만큼 거듭제곱하여 나온 값(m^n)을 리턴하는 함수를 작성하여 다음과 같이 출력하는 프로그램을 작성하시오.

입력 예

2 10

출력 예

1024

- hint

$2^{10} = 1024$

06 Question

정수의 연산식을 입력받아 연산을 위한 함수를 호출하여 사칙연산의 결과를 출력하는 프로그램을 작성하시오. ('/'의 경우는 정수 부분만 출력하고 사칙연산 이외의 연산 결과는 0으로 한다.)

입력 예 10 + 20

출력 예 10 + 20 = 30

소스

```
#include <stdio.h>

int gesan(int x, int y, char buho)
{
        switch(buho){
        case '+' :
                return x + y;
        case '-' :
                return x - y;
        case '*' :
                return x * y;
        case '/' :
                return x / y;
        }
        return 0;
```

```
}

int main()
{
        int a, b;
        char c;

        scanf("%d %c %d", &a, &c, &b);
        printf("%d %c %d = %d \n", a, c, b, gesan(a, b, c));

        return 0;
}
```

설명

- int gesan(int x, int y, char buho)
  연산을 위한 두 개의 정수와 연산자 한 개를 문자로 전달 받는 함수를 정의한다.

- switch (buho)
  buho 변수의 연산자에 따라 위치를 분기한다.

- case '+':
  return x+y;
  buho 연산자가 '+'이면 두 정수의 합을 리턴한다.
  return 문장을 만나면 함수를 종료하고 호출한 위치로 돌아가므로 break; 문은 필요가 없다.

- return 0;
  switch 문장에서 case의 어느 하나를 만나게 되면 리턴이 된다. 이 문장을 만나기 위해서는 case의 어느 것도 만나지 않아야 한다. 즉, 위 4가지 연산자 이외의 문자일 경우에 0을 리턴하는 것이다.

자가진단 6

위 소스에서 함수 내의 switch 문을 if~ else if~ else 문으로 바꾸어 실행해보자.

08 Question

두 정수를 입력받아 합과 곱을 출력하는 프로그램을 구조화하여 작성하시오.

입력 예 *두 수를 입력하세요. 35 26*

출력 예 합 : 61
곱 : 910

소스

```c
#include <stdio.h>

int a, b;
int hap, gop;

void input()
{
        printf("두 수를 입력하세요. ");
        scanf("%d %d", &a, &b);
}

void gesan()
{
        hap = a + b;
        gop = a * b;
}

void output()
{
        printf("합 : %d \n", hap);
        printf("곱 : %d \n", gop);
}

int main()
{
        input();
        gesan();
        output();
        return 0;
}
```

설명

- int a, b;
  int hap, gop;
  변수를 선언한 위치가 어떤 함수에도 속하지 않고 있다. 특정한 함수 내에 선언된 변수는 그 함수 내에서만 사용이 가능하며 그 함수가 종료되면 그 안에서 선언된 변수도 모두 자동적으로 소멸된다. 이렇게 함수 내에서 선언된 변수를 지역변수 (local variable) 라고 한다. 위와 같이 어떤 함수에도 속하지 않고 독립적으로 선언된 변수를 전역변수(global variable) 라고 한다.
  전역변수는 변수가 선언된 위치 이후의 어떤 함수에서도 접근과 사용이 가능하다.

- int main()
  {
  input();
  gesan();
  output();
  return 0;
  }

  위의 문제를 잘 분석하여 프로그램에서 해야 할 일을 순서대로 나열해보면 입력 → 계산 → 출력 이렇게 3가지 작업을 하게 된다. 따라서 이 3가지 작업을 각각 함수로 작성한 후 메인에서는 순서대로 함수만 호출하면 된다. 이러한 방법으로 프로그램을 작성하는 것을 **구조화**한다고 하며 이렇게 작성된 프로그램은 분석이 쉽고 작성이 간단하며 디버깅을 통해 잘못된 점을 찾아내기도 훨씬 수월해진다.

Thank You!!!