함수(Functions)

함수

- 함수는 호출될 때만 실행되는 코드 블록입니다.
- 매개변수라고 하는 데이터를 함수에 전달할 수 있습니다.
- 함수는 특정 작업을 수행하는 데 사용되며 코드 재사용에 중요합니다. 코드를 한 번 정의하고 여러 번 사용하세요.

사전 정의된 함수 내장함수

- main() 코드를 실행하는 데 사용되는 함수입니다.
- printf() 화면에 텍스트를 출력/인쇄하는 데 사용됩니다.

```
#include <stdio.h>

int main() {
  printf("Hello World!");
  return 0;
}
결과
Hello World!
```

함수 생성

> 자신만의 함수를 생성(선언)하려면 함수 이름을 지정하고 그 뒤에 괄호() 와 중괄호{}를 지정하세요 .

```
void myFunction() {  // 함수생성
  // 함수가 수행해야 하는 작업(코드)
}
```

함수 호출

- 선언된 함수는 즉시 실행되지 않습니다.
- 나중에 사용하기 위해 저장되고 호출될 때 실행됩니다.
- 함수를 호출하려면 함수 이름 뒤에 두 개의 괄호 () 와 세미콜론을 입력하세요.;

```
리턴 X
void myFunction() {  // 함수생성
  printf("함수실행");
}

int main() {
  myFunction(); // 함수호출
```

```
  return 0;
}

결과 : 함수실행
```

> 함수는 여러 번 호출될 수 있습니다.

```
#include <stdio.h>

void myFunction() {
  printf("함수실행!\n");
}

int main() {
  myFunction(); // 함수호출
  myFunction(); // 함수호출
  myFunction(); // 함수호출
  return 0;
}

결과 :
함수실행!
함수실행!
함수실행!
```

함수 매개변수

매개변수 및 인수

- 정보는 매개변수로 함수에 전달될 수 있습니다.
- 매개변수는 함수 내에서 변수 역할을 합니다.
- 매개변수는 함수 이름 뒤의 괄호 안에 지정됩니다.
- 원하는 만큼 매개변수를 추가할 수 있습니다.
- 쉼표로 구분하면 됩니다.

```
반환유형 함수이름(매개변수1, 매개변수2, 매개변수3) {
  // 실행코드
}
```

- 다음은 이름이 있는 문자열을 매개변수로 사용하는 함수입니다 .
- 함수가 호출되면 함수 내에서 "Hello"와 각 사람의 이름을 인쇄하는 데 사용되는 이름을 전달합니다.

```
#include <stdio.h>

void myFunction(char name[]) {
  printf("안녕하세요 %s 님\n", name);
```

```
}

int main() {
  myFunction("강호동");
  myFunction("유재석");
  myFunction("신동엽");
  return 0;
}

결과 :
안녕하세요 강호동 님
안녕하세요 유재석 님
안녕하세요 신동엽 님
```

- 매개변수가 함수에 전달되면 이를 인수라고 합니다 .
- 매개변수는 name 입니다
- 인수는 강호동 유재석 신동엽 입니다

다중 매개변수

> 함수 내에서 원하는 만큼 매개변수를 추가할 수 있습니다.

```
#include <stdio.h>

void myFunction(char name[], int age) {
  printf("안녕하세요 %s. 님은 %d 살입니다~\n", name, age);
}

int main() {
  myFunction("강호동", 43);
  myFunction("유재석", 40);
  myFunction("신동엽", 41);
  return 0;
}
결과 :
안녕하세요 강호동. 님은 43 살입니다~
안녕하세요 유재석. 님은 40 살입니다~
안녕하세요 신동엽. 님은 41 살입니다~
```

> 여러 매개변수로 작업하는 경우 함수 호출에는 매개변수와 동일한 수의 인수가 있어야 하며 인수는 동일한 순서로 전달되어야 합니다.

배열을 함수 매개변수로 전달하기

> 함수에 배열을 전달할 수도 있습니다 .

```
#include <stdio.h>

void myFunction(int myNumbers[5]) {
```

```
  for (int i = 0; i < 5; i++) {
    printf("%d\n", myNumbers[i]);
  }
}

int main() {
  int myNumbers[5] = {10, 20, 30, 40, 50};
  myFunction(myNumbers);
  return 0;
}
결과 :
10
20
30
40
50
```

myFunction( )함수는 배열을 매개변수(int myNumbers[5])로 사용하고 for루프를 사용하여 배열 요소를 반복합니다.

myNumbers 함수가 main() 내부에서 호출되면 배열 요소를 전달합니다 .

함수를 호출할 때 배열을 인수로 전달할 때 myFunction(myNumbers) 배열 이름만 사용해야 한다는 점에 유의하세요. 그러나 함수 매개변수( )에는 int myNumbers[5] 배열의 전체 선언이 필요합니다 .

반환 값

이전 예에서 사용된 void 키워드는 함수가 값을 반환해서는 안 된다는 것을 나타냅니다. 함수가 값을 반환하도록 하려면 대신 데이터 유형(예: int or float void return등)을 사용 하고 함수 내부에 키워드를 사용할 수 있습니다.

```
#include <stdio.h>

int myFunction(int x) {
  return 5 + x;
}

int main() {
  printf("Result is: %d", myFunction(3));
  return 0;
}

결과 : Result is: 8
```

> 다음 예에서는 두 개의 매개변수가 있는 함수의 합계를 반환합니다 .

```
#include <stdio.h>

int myFunction(int x, int y) {
```

```
  return x + y;
}

int main() {
  printf("Result is: %d", myFunction(5, 3));
  return 0;
}

결과 :
Result is: 8
```

> 결과를 변수에 저장할 수도 있습니다.

```
#include <stdio.h>

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

int main() {
  int result = myFunction(5, 3);
  printf("Result is = %d", result);
  return 0;
}
결과 : Result is: 8
```

함수 선언 및 정의

함수 선언 및 정의

함수는 두 부분으로 구성됩니다

- 선언 : 함수이름, 반환 유형 및 매개변수 (있는 경우)
- 정의 : 함수 본문(실행할 코드)

코드 최적화를 위해서는 함수 선언과 정의를 분리하는 것이 좋습니다
main()위에 함수 선언이 있고 main()아래에 함수 정의가 있는 C 프로그램을 자주 볼 수 있습니다
이렇게 하면 코드가 더 잘 구성되고 읽기 쉬워집니다.

```
// 함수 선언
void myFunction();

// main 실행함수
int main() {
  myFunction();  // 함수호출
  return 0;
}

// 함수 정의
```

```
void myFunction() {
  printf("I just got executed!");
}
```

> 함수 매개변수와 반환값에 관해 이전 장의 예를 사용한다면

```
#include <stdio.h>

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

int main() {
  int result = myFunction(5, 3);
  printf("Result is = %d", result);
  return 0;
}
결과 : Result is: 8
```

> 대신 다음과 같이 작성하는 것이 좋습니다.

```
// 함수 선언
int myFunction(int, int);

// main 실행함수
int main() {
  int result = myFunction(5, 3); // 함수호출
  printf("Result is = %d", result);
  return 0;
}

// 함수 정의
int myFunction(int x, int y) {
  return x + y;
}

결과:
Result is: 8
```

재귀 다시 나를 호출

재귀함수

- 재귀는 함수 자체를 호출하는 기술입니다.
- 이 기술은 복잡한 문제를 해결하기 더 쉬운 간단한 문제로 나누는 방법을 제공합니다.
- 재귀는 이해하기 조금 어려울 수 있습니다.
- 그것이 어떻게 작동하는지 알아내는 가장 좋은 방법은 직접 실험해 보는 것입니다.

다음예제

- 두 숫자를 합치는 것은 쉽지만 숫자 범위를 더하는 것은 더 복잡합니다.
- 다음 예에서는 재귀를 사용하여 두 숫자를 추가하는 간단한 작업으로 나누어 숫자 범위를 함께 추가합니다.

```c
int sum(int k);

int main() {
  int result = sum(10);  // 함수 호출
  printf("%d", result);
  return 0;
}

int sum(int k) {  // k = 10
  if (k > 0) {
    return k + sum(k - 1);  // 재귀호출
  } else {
    return 0;
  }
}
```

> 프로그램은 다음 단계를 따릅니다.

```
10 + sum(9)
10+(9 + sum(8))
10 + (9 + (8 + sum(7)))
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0
```

개발자는 재귀에 대해 매우 주의해야 합니다. 종료되지 않는 함수나 과도한 양의 메모리나 프로세서 성능을 사용하는 함수를 작성하는 것이 매우 쉬울 수 있기 때문입니다. 그러나 올바르게 작성되면 재귀는 프로그래밍에 대한 매우 효율적이고 수학적으로 우아한 접근 방식이 될 수 있습니다.

수학 함수

수학 함수

이를 사용하려면 프로그램에 math.h 헤더 파일을 포함해야 합니다.

```c
#include <math.h>
```

제곱근

숫자의 제곱근을 찾으려면 다음 sqrt() 함수를 사용하십시오.

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

int main() {
  printf("%f", sqrt(16));
  return 0;
}
```

결과 :

```
4.000000
```

숫자 반올림

- 함수 round()는 숫자를 반올림합니다
- 함수 ceil()는 숫자를 가장 가까운 정수로 올림합니다
- 함수 floor()는 숫자를 가장 가까운 정수로 내림합니다.

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

int main() {
  printf("%f\n", round(1.4));
  printf("%f\n", round(1.5));
  printf("%f\n", ceil(1.4));
  printf("%f\n", floor(1.4));
  return 0;
}
```

결과 :

```
1.000000
2.000000
2.000000
1.000000
```

제곱근

pow(x, y) x의 값을 y의 거듭제곱으로 반환합니다.

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

int main() {
  printf("%f", pow(4, 3));  // 4 x 4 x 4
  return 0;
}
```

```
결과:
64.000000
```