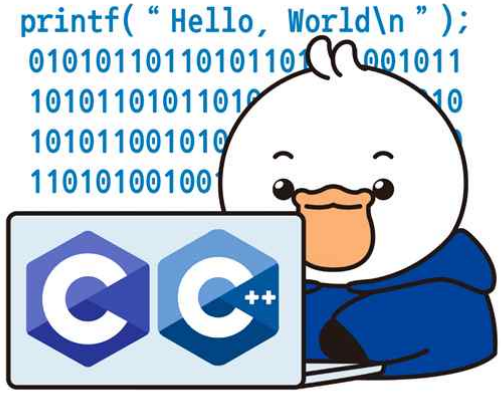
학번 : _____ 성명 : _____.

Part 8. 함수

목차

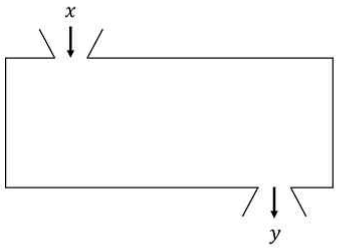


8.1 함수의 개요

- 함수란?
  - ✓ 소프트웨어 프로그램에서 [ ]을 수행하는 코드를 묶은 것
  - ✓ 언어에 따라 [ ]라고 부르기도 함
  - ✓ 프로그램을 [ ]로 나누어 작성하고 관리
    - ➢ 프로그램 생산성 향상에 도움
  - ✓ 일부 기능을 수정할 경우, 해당 함수만 수정
    - ➢ 효율적인 프로그래밍

8.1 함수의 개요

- C 언어에서의 함수
  - ✓ 수학의 함수 개념과 유사
  - ✓ 입력 값을 주면 결과 값을 출력하는 구조
  - ✓ 대표적인 함수
    - ➢ `printf()`, `scanf()`
    - ➢ 함수의 이름을 적어 실행하는 과정 : "호출(call)한다"



8.1 함수의 개요

- 함수의 장점
  - ✓ [ ]
    - ➢ 코드를 중복해서 작성하지 않아 간결하고 분석이 쉬운 코드
  - ✓ [ ]
    - ➢ 함수화한 코드는 재사용이 쉽고, 다른 프로그램에서도 같은 코드 사용 가능
  - ✓ [ ]
    - ➢ 함수의 기능에 따라 함수를 작성하기에 프로그램의 모듈화 증대
  - ✓ [ ]
    - ➢ 코드 수정 시, 함수 내부만 수정하여 코드 수정 사항을 적용

8.1 함수의 개요

- 함수의 종류
  - ✓ 3가지로 분류
    - ➢
  - ✓ `main()` 함수
    - ➢ 진입점(entry point) 함수
    - ➢ 프로그램을 시작할 때 가장 처음 실행되는 위치
    - ➢ 운영체제가 프로그램의 실행을 위해 호출하는 함수
    - ➢ 사용자는 호출 불가능

8.1 함수의 개요

- 함수의 종류
  - ✓ 라이브러리 함수
    - ➢ 컴파일러에서 기본적으로 제공하는 함수
    - ➢ 프로그램을 작성하는 도중 필요할 때마다 불러와 사용할 수 있는 함수
    - ➢ C 언어 컴파일러 : 표준 라이브러리 함수
    - ➢ 제공되는 기능
      - 매크로, 데이터 타입 정의, 문자열 처리, 수학적 연산, 입출력 프로세스, 메모리 할당 등

8.1 함수의 개요

- 함수의 종류
  - ✓ 라이브러리 함수
    - ➢ 기능에 따라 헤더 파일(header file) 형태로 저장

라이브러리 헤더	표준	상세 설명
<complex.h>	C99	복소수 조작 및 제어 함수 정의
<fenv.h>	C99	부동소수점 제어 함수 정의
<stdio.h>	-	C언어 프로그램 핵심 입력, 출력 함수 정의
<time.h>	-	시간 처리 관련 함수를 정의
<uchar.h>	C11	유니코드 문자 및 제어 함수 정의
<string.h>	-	문자열 처리 함수들을 정의
<math.h>	-	일반적인 수학 함수 정의

8.2 사용자 정의 함수

- 사용자 정의(user defined) 함수
  - ✓ 프로그래머가 직접 함수를 정의하고 호출하는 함수
  - ✓ 함수 사용하기
    - ➢
    - ➢
    - ➢

8.2 사용자 정의 함수

소스 8-1

```
1  #include <stdio.h>
2  int add(int a, int b);        // 사용자 정의 함수 선언
3  int main(void) {
4     int sum = add(3, 4);       // 사용자 정의 함수 호출
5     printf("Sum of two integers : %d", sum);
6     return 0;
7  }
8  int add(int a, int b) {       // 사용자 정의 함수 add 정의
9     return a+b;
10 }
```

출력 예	Sum of two integers : 7

8.2 사용자 정의 함수

- 사용자 정의(user defined) 함수
  - ✓ 함수의 선언 및 정의, 호출의 내용 및 예시

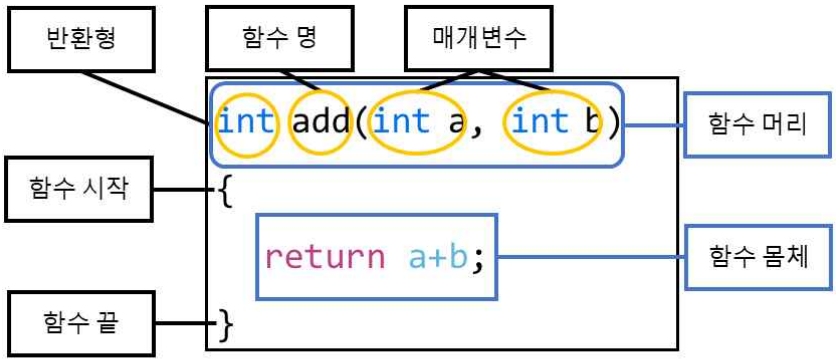
구분	내용	예시
	정의된 함수의 정보를 제공하는 것	`// 사용자 정의 함수 선언` `int add(int a, int b);`

	작성할 함수가 실제 동작 내용을 작성 및 정의하는 것	`// 사용자 정의 함수 add (덧셈)`
		`int add(int a, int b)`
		`{`
		`  return a+b;`
		`}`

	앞에서 선언되거나 정의된 함수를 이용하는 것	`// 사용자 정의 함수 호출` `int sum = add(3, 5);`

8.2 사용자 정의 함수

- 함수 정의 형식



8.2 사용자 정의 함수

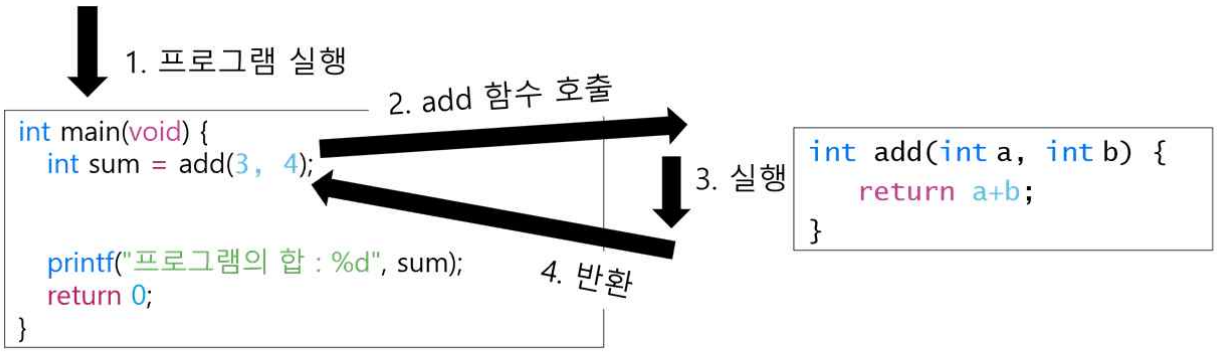
- 함수 선언과 함수 머리
  - ✓ 함수 선언
    - ➢ 세미콜론 선언 필요
    - ➢ 변수 이름 생략 가능 `int add(int, int);`
  - ✓ 함수 머리
    - ➢ 세미콜론 선언하지 않음
    - ➢ 변수 이름 생략 불가능 `int add(int a, int b)`

8.2 사용자 정의 함수

- 함수 정의
  - ✓ `main()` 함수 위에 정의하는 경우
    - ➢ 함수 선언 생략 가능
    - ➢ 정의와 함께 선언한 것으로 컴파일러가 간주

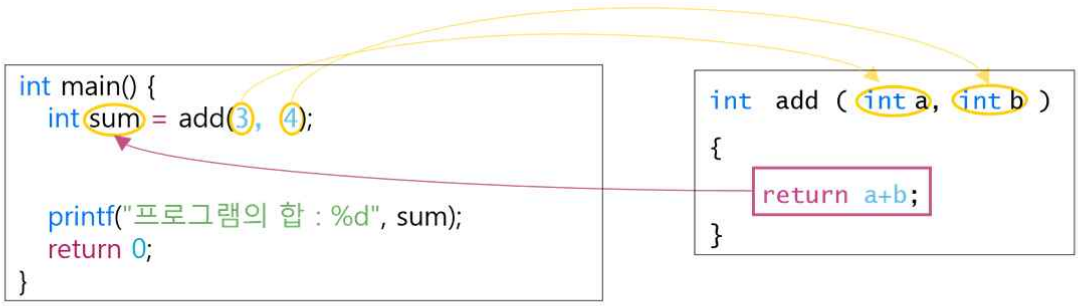
8.2 사용자 정의 함수

- 소스 8-1 실행 흐름



8.2 사용자 정의 함수

▪ 소스 8-1 인자값과 반환값 전달



8.2 사용자 정의 함수

소스 8-2

```
1  #include <stdio.h>
2  int max(int a, int b) {
3      return (a > b) ? a : b;
4  }
5  int main(void) {
6      int n1, n2;
7      printf("Enter two integers : ");
8      scanf("%d %d", &n1, &n2);
9      printf("Bigger value : %d\n", max(n1, n2));
10     return 0;
11 }
```

입력 예 1	1 5
출력 예 1	Enter two integers : 1 5 Bigger value : 5
입력 예 2	5 2
출력 예 2	Enter two integers : 5 2 Bigger value : 5

8.2 사용자 정의 함수

- 반환값의 데이터 타입
  - ✓ 반환형 또는 리턴 타입(return type)
    - ➢ 함수 선언 제일 앞에 오는 함수 반환값의 데이터 타입
  - ✓ 기본 데이터 타입에 따른 함수 선언 예시

반환 값		선언	예시
없음		void	void add(int a, int b) { … }
정수	정수	short, …	short add(short a, short b) { … }
		int, …	int add(int a, int b) { … }
		long, …	long add(long a, long b) { … }
	문자	char	char add(char a, char b) { … }
실수		float	float add(float a, float b) { … }
		double	double add(double a, double b) { … }

8.2 사용자 정의 함수

- 반환값의 데이터 타입
  - ✓ 사용 가능한 데이터 타입
    - ➢ 표에 명시되어 있는 데이터 타입
    - ➢ 각 데이터 타입의 포인터
    - ➢ 고급 데이터 타입 (추후 학습)
  - ✓ 반환값이 필요 없는 경우
    - ➢ void를 사용하여 명시
      - 명시 하지 않을 시, int형으로 간주

8.2 사용자 정의 함수

- 함수 이름
  - ✓ 식별자의 한 종류
  - ✓ 식별자 생성 규칙에 따라 작성 (2장에서 학습)
  - ✓ 생성 규칙
    - ➢ 영문자의 대소문자
    - ➢ 숫자
    - ➢ 밑줄 기호(_)
    - ➢ 함수의 첫 글자 : 영문자 또는 밑줄 기호(_)
    - ➢ 키워드 불가능

8.2 사용자 정의 함수

- 함수 이름
  - ✓ 함수 이름 예시

```
int func1(int);
int absolute(int);
float get_circle_area(float);
float squareArea(float, float);
int getTriangleArea(int, int);
```

함수 기능 추측 불가능

함수 기능 추측 가능

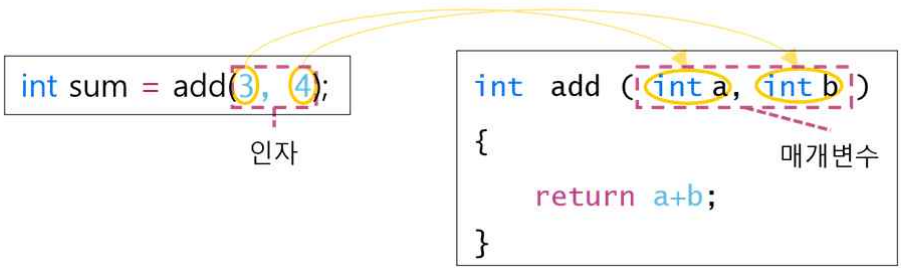
  - ✓ 함수의 역할을 추측할 수 있도록 작성
  - ✓ 동일한 이름의 함수는 중복으로 선언 불가능
  - ✓ 잘 작성한 함수 이름은 코드 생산성을 높이는데 효과적

8.2 사용자 정의 함수

- 매개변수(parameter)
  - ✓ 함수를 호출한 곳에서 함수로 전달하는 값을 저장하는 변수
  - ✓ 매개변수의 인자값(argument value)을 저장하여 함수 내부에서 그 값을 사용할 수 있도록 해주는 역할
    - ➢ 함수를 호출한 측과 함수 내부를 연결
  - ✓ 인자(argument)
    - ➢ 함수를 호출할 때 함수에 전달하는 것
    - ➢ 변수, 상수, 함수, 연산식 등이 사용됨

8.2 사용자 정의 함수

- 매개변수(parameter)
  - ✓ 함수의 호출과 인자의 전달



8.2 사용자 정의 함수

- 매개변수(parameter)
  - ✓ 매개변수는 개수 제한 없음
    - ➢ 매개변수가 없는 함수도 선언 가능
  - ✓ 반드시 데이터 타입이 정의되어야 함
  - ✓ 각 매개변수는 콤마 연산자(,)로 구분하여 선언
  - ✓ 매개변수가 없을 경우
    - ➢ void로 기재 (권장)
    - ➢ 괄호()를 비워두기

8.2 사용자 정의 함수

- 매개변수(parameter)
  - ✓ 사용자 정의 함수의 매개변수 작성 예시

```
double inputHeight(void);
void setAge(int value);
double getCircleArea(double);
float max(float a, float b, float c, float d);
```

  - ✓ 매개변수의 데이터 타입이 명시되지 않은 경우
    - ➢ int형으로 간주
    - ➢ 컴파일 단계에서 경고 메시지 출력

8.2 사용자 정의 함수

- 매개변수(parameter)
  - ✓ [ ]
    - ➢ 매개변수 이름 생략 가능
  - ✓ [ ]
    - ➢ 매개변수의 이름 필요
  - ✓ [ ]
  - ✓ [ ]
    - ➢ 함수 종료 시, 매개변수 사용 불가
    - ➢ 오로지 함수 내부에서만 사용할 수 있는 변수

8.2 사용자 정의 함수

소스 8-3

```
1  #include <stdio.h>
2  int test(int a) {
3      a +=10;
4      printf("In test : a = %d\n", a);
5      return a;
6  }
7  int main(void) {
8      int a =5;
9      printf("Before test : a = %d\n", a);
10     printf("Return value of test = %d\n", test(a));
11     printf("After test : a = %d\n", a);
12     return 0;
13 }
```

출력 예

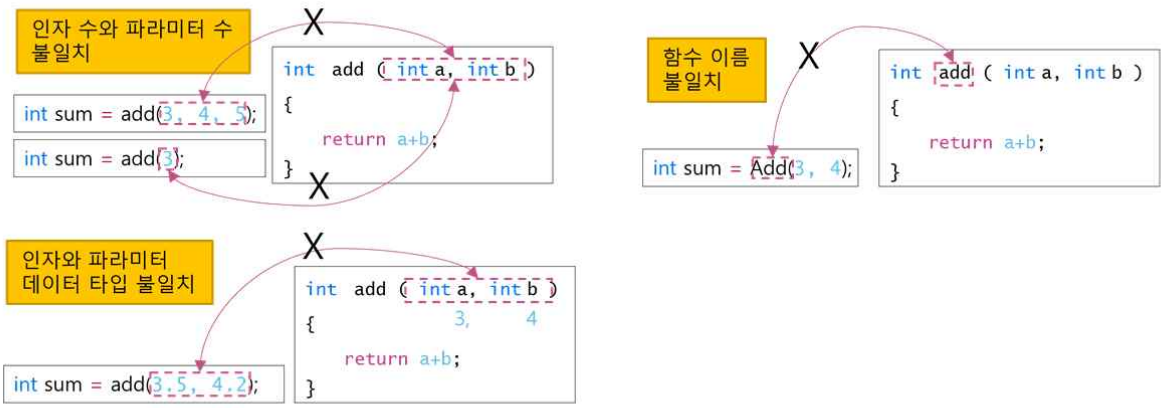
```
Before test : a = 5
In test : a = 15
Return value of test = 15
After test : a = 5
```

8.2 사용자 정의 함수

- 함수 선언 시 주의사항
  - ✓ main() 함수 내부의 변수와 사용자 정의 함수의 매개변수는 각각 독립적인 변수
  - ✓ 매개변수의 수와 타입에 맞게 인자를 정확하게 전달
  - ✓ 함수 이름에서 대소문자를 정확하게 확인 후 호출
    - ➢ C 언어는 영문자 대소문자를 구분함

8.2 사용자 정의 함수

- 함수 선언 시 주의사항
  - ✓ 함수 호출 시, 컴파일 에러가 발생하는 상황들



8.2 사용자 정의 함수

소스 8-4

```c
#include <stdio.h>
/*--- 3개의 정수 중 최솟값 반환 ---*/
double minAmong3(double a, double b, double c) {
    double min = a;
    if (b < min) min = b;
    if (c < min) min = c;
    return min;
}
int main(void) {
    double a, b, c;
    printf("Enter 3 numbers : ");
    scanf("%lf %lf %lf", &a, &b, &c);
    printf("The minimum value is %f\n", minAmong3(a, b, c));
    return 0;
}
```

입력 예	12 3.14 2021.11
출력 예	Enter 3 numbers : 12 3.14 2021.11 The minimum value is 3.140000

8.2 사용자 정의 함수

소스 8-5

```c
#include <stdio.h>
/*--- 문자 ch를 n개 연속해서 출력 ---*/
void printChars(char ch, int n) {
    while (n-->0)
        printf("%c", ch);
}
/*--- 문자 ch를 n개 연속해서 출력하고 줄 바꿈 ---*/
void printCharsLine(int ch, int n) {
    printChars(ch, n);
    printf("\n");
}
/*--- 경고음을 n번 연속해서 발생 ---*/
void alert(int n) {
    printChars('\a', n);
}
```

```c
int main(void) {
    int n;
    printCharsLine('=', 5); /* '='를 5개 연속 출력하고 줄 바꿈 */
    printf("Input a Number : ");
    scanf("%d", &n);
    printCharsLine('-', 5); /* '-'를 5개 연속 출력하고 줄 바꿈 */
    alert(n);     /* 경고음 n번 출력 */
    printf("\n");
    printCharsLine('=', 5); /* '='를 5개 연속 출력하고 줄 바꿈 */
    return 0;
}
```

입력 예	5
출력 예	===== Input a number : 5 ----- =====

8.3 함수의 인자 전달 방법

- C 언어 함수에서 인자를 전달하는 방법

✓ [ ]

- 인자값을 매개변수에 복사하여 함수로 전달
- 인자로 변수, 상수, 함수, 연산식 등을 사용

✓ [ ]

- 인자값으로 주소를 전달하는 방식
- 호출하는 인자와 함수의 매개변수를 이용하여 값을 공유
- C 언어는 주소에 의한 호출과 참조에 의한 호출을 혼용하여 사용
- 타 객체지향 언어에서 참조에 의한 호출은 C 언어에서의 주소에 의한 호출과 다른 의미로 사용되는 경우가 있음

8.3 함수의 인자 전달 방법

소스 8-6

```
1  #include <stdio.h>
2  void swap(int x, int y) {
3      int temp = x;
4      x = y;
5      y = temp;
6  }
7  int main(void)
8  {
9      int a, b;
10     printf("Enter two int : ");
11     scanf("%d %d", &a, &b);
12     swap(a, b);
13     printf("Result of swap() : A=%d, B=%d\n", a, b);
14     return 0;
15 }
```

입력 예	12 8
출력 예	Enter two int : 12 8 Result of swap() : A=12, B=8

8.3 함수의 인자 전달 방법

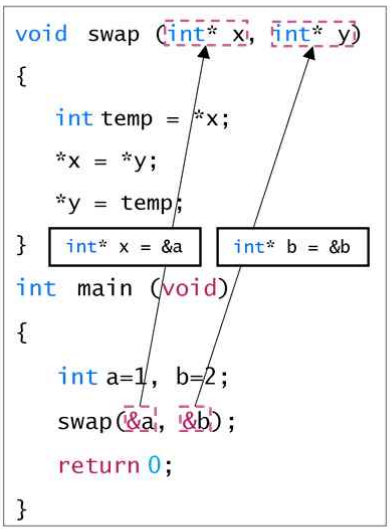
소스 8-6-1

```c
#include <stdio.h>
void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}
int main(void)
{
    int a, b;
    printf("Enter two int : ");
    scanf("%d %d", &a, &b);
    swap(&a, &b);
    printf("Result of swap() : A=%d, B=%d\n", a, b);
    return 0;
}
```

입력 예	12 8
출력 예	Enter two int : 12 8 Result of swap() : A=8, B12

8.3 함수의 인자 전달 방법

- 주소에 의한 호출 방식
  - ✓ 변수의 주소 전달하기 : 주소 연산자 & 사용
  - ✓ 주소값이 인자로 전달되는 과정



8.3 함수의 인자 전달 방법

소스 8-7

```c
#include <stdio.h>
void printArray(int array[], int len) {
   int i;
   for(i=0; i<len; i++) {
      printf("%d ", array[i]);
   }
   printf("\n");
}
int main(void) {
   int score[] = { 99, 84, 91, 78, 89 };
   int array_length =sizeof(score) /sizeof(score[0]);
   printArray(score, array_length);
   return 0;
}
```

출력 예	99 84 91 78 89

8.3 함수의 인자 전달 방법

소스 8-7-1

```c
#include <stdio.h>
void printArray(int* ptr, int len) {
   int i;
   for(i=0; i<len; i++) {
      printf("%d ", *(ptr++));
   }
   printf("\n");
}
int main(void) {
   int score[] = { 99, 84, 91, 78, 89 };
   int array_length =sizeof(score) /sizeof(score[0]);
   printArray(score, array_length);
   return 0;
}
```

출력 예	99 84 91 78 89

8.3 함수의 인자 전달 방법

- 배열을 인자로 받는 사용자 정의 함수
  - ✓ 배열의 이름은 배열의 0번 인덱스의 주소를 가지는 포인터 상수
  - ✓ 사용자 정의 함수에서 배열을 처리하는 방식
    - ➢ 배열의 시작 주소를 배열로 받아서 처리
    - ➢ 배열의 시작 주소를 포인터 매개변수로 받아서 처리
  - ✓ 사용자 정의 함수에서 배열의 값 변경 시
    - ➢ 주소에 의한 호출(call by address)
    - ➢ `main()` 함수의 배열의 값도 변경

8.3 함수의 인자 전달 방법

- 매개변수를 상수로 선언 - const 키워드
  - ✓ 주소에 의한 호출
    - ➢ 함수 내부에서 매개변수 값 변경을 원하지 않는 경우
  - ✓ 값에 의한 호출
    - ➢ 파라미터에 저장된 값이 변경되어 실행에 좋지 않은 영향을 미칠 경우

8.3 함수의 인자 전달 방법

- 가변인자
  - ✓ 인자의 개수와 데이터 타입을 미리 정의해 두지 않고 사용하는 함수의 인자
  - ✓ 가변인자 함수
    - ➢ 매개변수의 데이터 타입과 개수가 동적으로 변하는 함수
    - ➢ 대표적인 가변인자 함수
      - `printf()`, `scanf()`

8.3 함수의 인자 전달 방법

소스 8-8

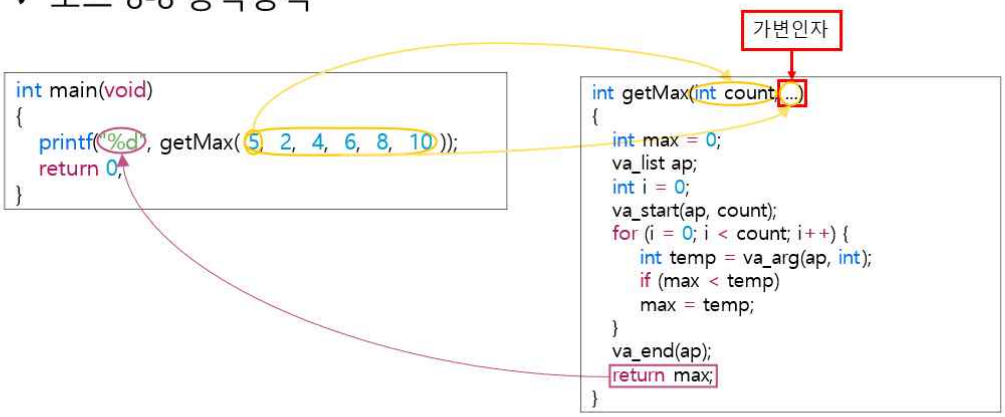
```c
#include <stdio.h>
#include <stdarg.h>
int getMax(int count, ...) {
    int max =0;
    va_list ap;
    int i =0;
    va_start(ap, count);
    for (i =0; i < count; i++) {
        int temp = va_arg(ap, int);
        if (max < temp)
            max = temp;
    }
    va_end(ap);
    return max;
}
int main(void) {
    printf("%d", getMax(5, 2, 4, 6, 8, 10));
    return 0;
}
```

출력 예	10

8.3 함수의 인자 전달 방법

- 가변인자

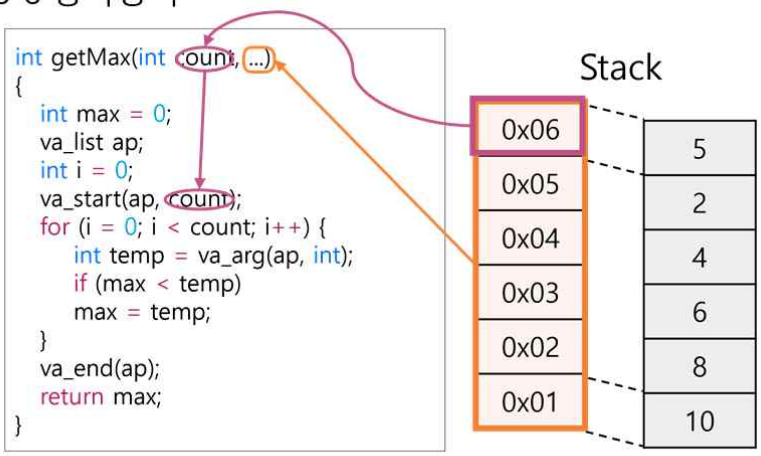
✓ 소스 8-8 동작방식



8.3 함수의 인자 전달 방법

- 가변인자

✓ 소스 8-8 동작방식



8.3 함수의 인자 전달 방법

- 가변인자
  - ✓ stdarg.h 헤더파일
    - ➢ 가변인자를 제어하기 위한 헤더파일
    - ➢ stdarg.h 헤더파일 매크로의 종류

반환 값	매크로	함수 원형
함수	va_start	void va_start(va_list ap, parmN);
	va_arg	type va_arg(va_list ap, type);
	va_copy	void va_copy(va_list dest, va_list src);
	va_end	void va_end(va_list ap);
자료형	va_list	-

8.3 함수의 인자 전달 방법

- 가변인자
  - ✓ stdarg.h 헤더파일
    - ➢ va_list
      - 가변인자가 저장된 스택 메모리 변수를 담기 위한 공간
    - ➢ va_start()
      - 가변인자 중 첫 번째 인자를 지정
      - parmN : 가변인자의 첫 번째 인자

8.3 함수의 인자 전달 방법

- 가변인자
  - ✓ stdarg.h 헤더파일
    - ➢ va_arg()
      - va_list의 포인터를 다음 가변인자의 위치로 이동하는 함수
      - type : 가변인자의 데이터 타입을 선택
    - ➢ va_copy
      - 복사 대상(source) va_list를 목적지(destination) va_list로 복사
    - ➢ va_end
      - 모든 가변인자를 검색한 후 포인터를 NULL로 설정

8.4 main() 함수

- main() 함수
  - ✓ 진입점 함수
    - ➢ C 언어로 작성한 프로그램을 실행할 때 가장 먼저 main() 함수가 실행
    - ➢ 컴파일 후 프로그램 실행 시 가장 먼저 운영체제로부터 호출이 이루어짐
    - ➢ 일반적인 함수와 다른 특징

8.4 main() 함수

- 표준 C 언어에서 main() 함수의 형태
  - ✓ `int main(void)`
  - ✓ `int main(int argc, char* argv[])`
    - ➢ 실행 파일의 이름과 인자값을 함께 입력하여 실행
    - ➢ `int argc`
      - 프로그램 시작 시 입력받은 인자 값의 개수
    - ➢ `char* argv[]`
      - 입력받은 모든 인자값이 차례대로 저장
    - ➢ MinGW 환경을 이용하는 파워쉘이나 리눅스 환경의 터미널에서도 실행 방법 유사

8.4 main() 함수

소스 8-9

```
1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Hello world");
5     return 0;
6 }
```

출력 예	Hello world

소스 8-10

```
1 #include <stdio.h>
2 int main(int argc, char* argv[]) {
3 printf("%d\n", argc);
4 printf("%s\n", argv[0]);
5 printf("%s", argv[1]);
6 return 0;
7 }
```

실행 예

8.4 main() 함수

- return 0 이란?
  - ✓ C 표준에서의 프로그램이 종료될 때 프로그램 종료 상태를 명시
    - ➢ 정상 종료
      - EXIT_SUCCESS (전통적으로 0)
    - ➢ 비정상 종료
      - EXIT_FAILURE (0이외의 값)
  - ✓ return 0 이란 프로그램이 정상 종료한다는 의미
  - ✓ main() 함수의 리턴 값을 정의 하지 않았을 경우
    - ➢ 컴파일러에 의해 자동으로 return 0의 코드가 추가

8.5 재귀함수

- 재귀함수(recursive function, recursion)
  - ✓ 자기 자신을 반복적으로 호출하여 원하는 결과를 도출하는 함수
  - ✓ 재귀함수에 대한 유머

어느 한 컴퓨터공학과 학생이 유명한 교수님을 찾아가 물었다.
"재귀함수가 뭔가요?"
"잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.
마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.
그의 답은 대부분 옳았다고 하네.
그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어.
"재귀함수가 뭔가요?"
"잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 ...

8.5 재귀함수

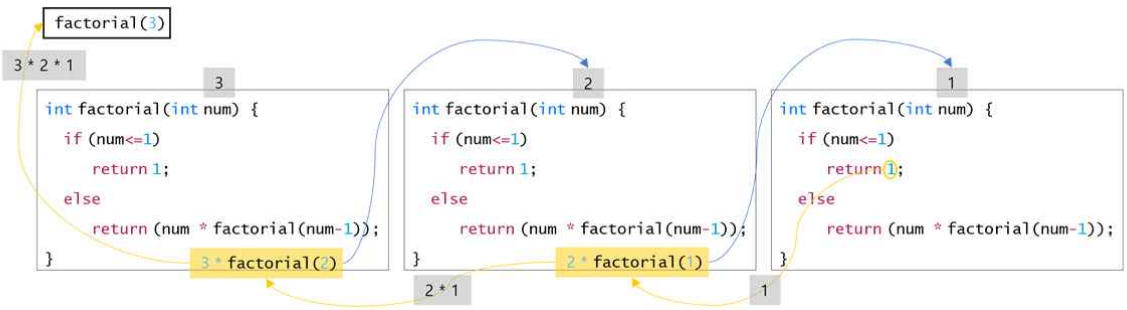
소스 8-11

```c
#include <stdio.h>
int factorial(int num) {
    if (num <= 1)
        return 1;
    else
        return (num * factorial(num-1));
}
int main(void) {
    int n;
    scanf("%d", &n);
    printf("%d! = %d\n", n, factorial(n));
    return 0;
}
```

입력 예	10
출력 예	10! = 3628800

8.5 재귀함수

- 재귀함수 동작방식



8.5 재귀함수

▪ 유클리드 호제법

✓ 두 정수 간 최대 공약수를 찾는 공식

➢ 1071과 1029의 최대 공약수 x를 찾는 순서

1071	%	1029	=	42
1029	%	42	=	21
42	%	21	=	0

$\therefore x = 21$

1. 두 정수 중 큰 정수를 기준으로 나누고, 그 나머지를 큰 정수의 값으로 대체
2. 1. 을 반복하여 연산
3. 연산 시 나머지가 0일 경우 마지막으로 나눈 수가 최대 공약수

8.5 재귀함수

소스 8-12

```c
/* 유클리드 호제법을 이용한 최대 공약수 구하기 */
#include <stdio.h>
/*--- 정숫값 x, y의 최대 공약수 반환---*/
int gcd(int x, int y) {
    if (y ==0)
        return x;
    else
        return gcd(y, x % y);
}
int main(void) {
    int x, y;
    printf("Enter two integer numbers : ");
    scanf("%d %d", &x, &y);
    printf("The greatest common denominator is %d.\n", gcd(x, y));
    return 0;
}
```

입력 예	12 8
출력 예	Enter two integer numbers : 12 8 The greatest common denominator is 4.

8.5 재귀함수

- 재귀함수 특징
  - ✓ 수학적인 알고리즘을 적용하기에 편리
  - ✓ 간단한 프로그램 코드로 복잡한 연산 수행 가능
  - ✓ 문제 해결 원리를 그대로 적용하여 알고리즘 개발 및 프로그램 구현 가능
  - ✓ 지속적으로 자기 자신을 호출하는 과정에서 많은 메모리를 소모
  - ✓ 반복 호출 횟수가 늘어날수록 시간이 오래 걸림

8.6 함수 포인터

- 함수 포인터
  - ✓ 함수가 저장된 메모리의 주소를 가리키는 포인터
  - ✓ 데이터 값을 가리키는 대신, 메모리 내에 실행 가능한 코드를 가리킴
  - ✓ 함수 포인터를 이용하여 함수 호출 및 인자 전달 가능
  - ✓ 간접 함수 호출
    - ➢ 함수 포인터 변수 또는 함수 포인터로 함수를 호출

8.6 함수 포인터

- 함수 포인터 선언 예시

```
int (*fp1)();
int (*fp2)(int);
int (*fp3)(int, int);
double (*fp4)();
void (*fp5)();
```

```
int   (*fp)   (int, int);
```

자료형 변수이름 매개변수

8.6 함수 포인터

- 함수 포인터 선언 구성
  - ✓ 리턴타입
    - ➢ 가리키고자 하는 함수의 리턴 타입
  - ✓ 함수 포인터 변수 이름
    - ➢ 함수를 가리키기 위해 사용할 포인터 변수
  - ✓ 매개변수 리스트
    - ➢ 함수 포인터가 가리킬 함수의 매개변수 개수와 타입

8.6 함수 포인터

소스 8-13

```c
#include <stdio.h>
int add(int first, int second) {
    return first + second;
}
int subtract(int first, int second) {
    return first - second;
}
int main(void) {
    int (*plus)(int, int) = add;
    int (*minus)(int, int) = subtract;
    printf("7 + 5 = %d\n", plus(7, 5));
    printf("20 - (7 + 5) = %d\n", minus(20, plus(7, 5)));
    return 0;
}
```

출력 예

```
7 + 5 = 12
20 - (7 + 5) = 8
```


8.6 함수 포인터

- 함수 포인터 변수
  - ✓ 변수로 취급
  - ✓ 함수의 파라미터와 인자로도 사용 가능

8.6 함수 포인터

소스 8-14

```c
#include <stdio.h>
int add(int first, int second) {
    return first + second;
}
int subtract(int first, int second) {
    return first - second;
}
int operation(int first, int second, int (*functocall)(int, int)) {
    return (*functocall)(first, second);
}
int main(void) {
    int a, b;
    int (*plus)(int, int) = add;
    int (*minus)(int, int) = subtract;
    a = operation(7, 5, plus);
    b = operation(20, a, minus);
    printf("7 + 5 = %d\n", a);
    printf("20 - (7 + 5) = %d\n", b);
    return 0;
}
```

출력 예

```
7 + 5 = 12
20 - (7 + 5) = 8
```


8.7 Q&A

Q 27. 아래 코드에서 왜 f2()가 먼저 호출되나요? 제 생각으로는 콤마(,) 연산자는 왼쪽에서 오른쪽으로 코드를 실행하는 것으로 알고 있는데요.

```c
printf("%d %d", f1(), f2());
```

A. 콤마(,) 연산자는 왼쪽에서 오른쪽으로 코드를 실행합니다. 그러나 함수 호출에서 각 인자를 구별하기 위해 사용하는 콤마(,)는 콤마 연산자가 아닙니다. 따라서, 코드 동작 규칙에 따라 함수 f2()가 먼저 호출됩니다.

8.7 Q&A

Q 28. 함수에 몇 개의 인자가 전달되었는지를 정확히 알 방법이 있나요?

A . 호환성 있는 방법은 존재하지 않습니다. 어떤 오래된 시스템에서는 비표준 함수인 `nargs()`를 제공하기도 합니다. 그러나 이 함수는 인자의 개수를 반환하는 게 아니라 전달된 단어의 개수를 반환합니다. (구조체나 `long int`, 실수는 여러 개의 단어로 이루어져 있는 경우가 대부분입니다.) 가변 인자를 받아 처리하는 함수는 그 자체만으로 인자의 개수를 파악할 수 있어야 합니다. `printf()` 계열의 함수들은 포맷 문자열에서 (%d와 같은) 형식지정자를 보고 그 개수를 파악합니다.

A . 또 다른 방법으로, 가변 인자가 모두 같은 타입일 경우, 마지막 인자를 (0, -1, 또는 적절한 널 포인터와 같은) 어떤 특정한 값으로 설정해서 인자의 개수를 파악합니다. (질문 5.2, 15.4에서 `execl()`과 `vstrcat()` 함수의 사용법을 참고하시기 바랍니다). 마지막으로, 인자의 개수를 미리 파악할 수 있다면, 전체 인자의 개수를 인자로 전달하는 것도 좋은 방법입니다.

8.7 Q&A

Q 29. `main()`을 선언하는 정확한 방법이 궁금합니다. `void main()`으로 해도 좋은가요?

A. 많은 책이 `void main()`을 쓰고 있지만, 이는 잘못된 것입니다. `main()`의 선언은 다음 중에서 골라 써야 합니다: `int main(void);`, `int main(int argc, char *argv[]);` `argv`를, `char **argv`로 선언할 수도 있습니다. (물론 이때 `argv'라는 이름은 얼마든지 바꿀 수 있습니다.)

1.7 실습 및 과제

- 실습 및 과제 진행
  - DCU Code : http://code.cu.ac.kr
  - 8 주차 실습
    - 코딩 8-1, 코딩 8-2, 코딩 8-3, 코딩 8-4, 코딩 8-5, 코딩 8-6, 코딩 8-7, 코딩 8-8
  - 8 주차 과제
    - 8장 프로그래밍 연습 1-4번

1.8 참고 문헌

- 함수 (프로그래밍) : https://ko.wikipedia.org/wiki/함수_(프로그래밍)
- 엔트리 포인트 : https://ko.wikipedia.org/wiki/엔트리_포인트
- C 표준 라이브러리 : https://ko.wikipedia.org/wiki/C_표준_라이브러리
- Game case : https://en.wikipedia.org/wiki/Camel_case
- C Programming FAQs : http://cinsk.github.io/ko/cfaqs/index.html

E N D