고려대학교 ALPS 2021

C언어 스터디

Week 0x01

알고리즘, 변수, 자료형,
연산자, 표준 입출력

0x00 Introduction

소개

- 컴퓨터학과 20학번 노정훈
- 2021 ALPS 부회장
- 잘 부탁드립니다! ☺

0x00 Introduction

스터디 소개

본 스터디는 C언어의 기초와 그에 필요한 간단한 컴퓨터 구조,
이를 활용한 간단한 알고리즘에 대해 **9주** 동안 다룰 예정입니다.

컴퓨터 학과 커리큘럼의 첫걸음과 같은 C언어는,
고전적이고, 복잡하고, 어렵다는 오해를 사기 십상입니다.

저 또한 특히 악명 높은, 배열이나 포인터 부분에서
어려움을 느끼고, 많이 검색하고, 선배들에게 질문을 많이 했던 기억이 납니다.

하지만 C언어는 그렇게 어렵지 않고, 스터디를 잘 따라와 주시면
컴프 강의의 높은 성적과, 다음 스터디인 초급 알고리즘 스터디에 참여할 정도의
실력을 얻으실 수 있도록 노력하겠습니다.

저와 여러분의 선배들이 겪었던 어려움들을 조금이나마 덜어드리고,
여러분의 시간을 아끼고, 더 재미있게 C언어를 배울 수 있도록
본 스터디를 실시해보고자 합니다.

0x00 Introduction

어떻게 배울 것인가?

```c
#include <stdio.h>

int main() {
  printf("Hello World\n");
  return 0;
}
```

최대한 많은 설명

공부, 특히 그 중에서도 프로그래밍 공부의 난이도는 모두에게 동등할 수 없습니다. 프로그래밍 자체가 여러 개념이 얽혀 있어 순차적으로 공부하기가 어렵기 때문입니다.

당장 프로그래밍 을 배울때 가장 처음 배우는 위의 Hello World! 만 하더라도, printf 등은 바로 배울 수 있지만, int main, \n 등이 무엇인지는 알지 못합니다. 하지만, 스터디를 통해 여러분 모두가 개념을 확실히 이해할 수 있도록, 최대한의 설명을 곁들여 드리겠습니다.

0x00 Introduction

어떻게 배울 것인가?

레퍼런스와 그 이상

인터넷으로 C를 공부하다 보면 정확하지 않거나, 명칭의 혼동이 발생하기가 쉽습니다. 이를 해결하고자 다양한 자료를 통해 강의에 정확하고 필요한 내용만 수록하겠습니다.

또한, C를 배우면서 더 알아 볼만한 것들과 알아두면 좋은 것들을 적어두겠습니다. 사소하지만 컴퓨터를 이해하는 데 도움을 줄 수 있다고 생각합니다. 심심하실 때 한 번 검색해보세요 😄

실습과 과제

시간이 남는다면 스터디 시간에 간단히 실습을 진행해볼 예정이며, 과제를 내는 것을 그리 좋아하진 않지만, 스터디 방침상 과제가 나갈 예정입니다! 9주차 중 7주차 이상 과제를 해결하면 회비를 환급받을 수 있으니 화이팅입니다! 과제는 백준 그룹 내에서 풀 수 있으며 모든 강의 자료와 강의 영상은 구글 드라이브를 통해 제공됩니다!

0x00 Introduction

진짜로 시작하기 전에.

프로그래밍 공부는 암기가 아닙니다.

프로그래밍 공부는 전혀 암기가 아닙니다.
수많은 언어의 수많은 문법, 그리고 그 문법들을 활용해 사람들이 만든 오픈 소스 라이브러리의 함수들까지, 그렇게 많은 내용을 외울 수 있는 사람은 존재하지 않을 것이라고 확신합니다.

저 또한 프로그래밍을 할 때 구글링 하는 것들이 절반입니다.

프로그래밍 공부는 언어의 원리와, 프로그래밍적인 사고, 그리고 이해를 기반으로 한다고 저는 생각합니다. 이것은 외우는 것이라기 보다, 이해와 경험을 통해 익숙해지는 것이며, 설령 기억이 안 난다고 해도,
인터넷엔 양질의 자료들이 많으니, “이런 게 있었지” 하며 다시 검색하고, 리마인드할 수 있으니, 외우려고 하지 말고
이해를 중점적으로 공부합시다 :)

0x00 Introduction

진짜로 시작하기 전에.

질문은 언제나 환영입니다!

채팅으로나, 스터디 단톡에서나, 혹은 사람 많은 곳에서 질문하기가 좀 그렇다 싶으시다면! 개인 톡으로도 언제든 질문은 환영합니다! 여러분 모두가 잘 이해하실 수 있도록 열심히 노력하겠습니다.

PPT가 더럽더라도..

PPT가 좀 더러워지더라도, PPT가 강의 자료로 배포될 것이기에 설명할 내용들을 PPT에 거의 다 수록했습니다. 스터디 당시에는 좀 난잡하게 보이더라도 이해해주세요!

컴퓨터 학과의 첫 걸음이자, ALPS 커리큘럼의 제일 앞에서 여러분과 함께하게 되어 영광입니다

이제 C언어와 프로그래밍을 향해 떠나봅시다~

Contents

Week 0x01

알고리즘, 변수, 자료형,
연산자, 표준 입출력

Week 0x02

조건문, 분기문, 반복문,
비트 연산, 아스키 코드

Week 0x03

배열, 포인터

Week 0x04

함수, 포인터, 변수 범위

Week 0x05

메모리, 동적 할당

Week 0x06

이중 포인터, 구조체

Week 0x07

구조체 연산, 별칭, 파일 입출력

Week 0x08

시간 복잡도, 공간 복잡도, 알고리즘,
브루트 포스

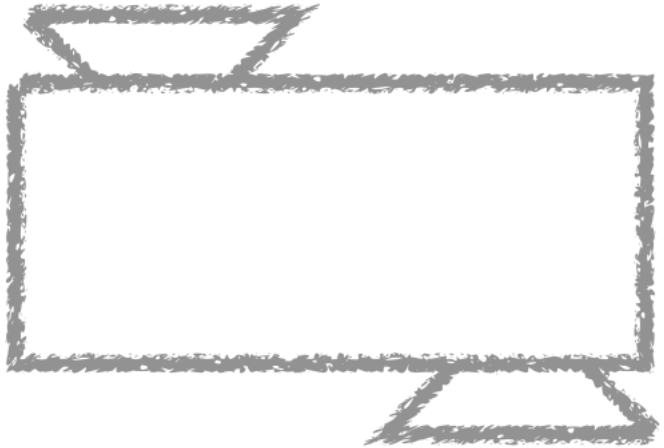
Week 0x09

재귀를 이용한 알고리즘

0x01 Algorithm

알고리즘이란?

수학과 컴퓨터 과학, 언어학 또는 관련 분야에서 어떠한 문제를 해결하기 위해 정해진 일련의 절차나 방법을 공식화한 형태로 표현한 것, 계산을 실행하기 위한 단계적 절차



상자의 연산 과정과 비슷

간단히 말하자면, 입력이 주어졌을 때 맞는 출력을 주는 일련의 과정

프로그래밍을 배우는 이유

자세한 건 8주차부터!

0x02 Variables

변수란?

정의

프로그래밍 언어에서 값을 사용하기 위한 방법.

수학에서 상수만이 아닌 변수 x, y 등을 다루듯이,
문제 풀이나 어떠한 과정의 일반화를 위해 필요함

예를 들어 계산기 프로그램을 만든다고 하면, 피연산자들을 담을 값이 필요. (연산의 대상)

이런 과정에서 **값을 저장**하기 위해 사용하는 것이 바로 변수

0x02 Variables

변수란?

사용법, 선언

변수를 사용하려면, 내가 이 변수를 사용하겠다는 "**선언**"이 필요!

이것을 **변수 선언** *Variable declaration* 이라고 함

다음과 같이 할 수 있음

More
컴퓨터가 각 코드를 구분하도록
세미콜론, ; 을 코드 끝에 붙여줘야 함

```
자료형  이름;
int num;
int a, b;
```

파이썬과 달리 C에는 변수의 형태인 자료형이 존재
변수를 쓰기 위해서는 변수의 자료형과 이름이 필요

자료형은 어떤 형태의 변수를 저장할 것인지를 말함(정수, 문자, 소수 등)
옆의 예시는 정수를 담는 int라는 자료형을 가지는 num이라는 변수를 선언

위의 a,b 처럼 여러 변수를 , 를 통해 한 번에 선언하는 것도 가능!

자료형은 조금 있다가 더 자세히 다루겠습니다!

0x02 Variables

변수란?

More
변수 이름을 정하는 관행들
camelCase, snake_case, PascalCase 등
코드를 더 알아보기 쉽게!

변수 이름 규칙

변수 또한 컴퓨터(컴파일러)가 인식할 수 있어야 하므로, 변수 이름에는 규칙이 존재

-영문 문자와 숫자 사용 가능
- 대소문자 구분
- 숫자부터 시작하면 안됨
- _로 시작할 수 있음
- 예약어 keyword (int, if 등) 사용 불가

외워야 할 내용 같아 보이지만, just 어떻게 변수 이름을 작성할 수 있는지를 설명하는 내용이므로 외울 필요는 없음 (나중엔 자연스럽게 느껴집니다!)

중요한 건 **숫자로 시작할 수 없다**는 점, C 언어의 문법에 쓰이는 단어들을 사용할 수 없다는 점입니다.

예를 들어 int 자료형의 변수 명을 int라고 설정한다면, “int int;” 라고 변수 선언을 하는 것인데, 나중에 이 변수를 사용할 때 이게 자료형인지 이름인지 컴퓨터가 이해할 수가 없겠죠.

즉, C언어의 문법에서 쓰이는 단어들로 변수 이름을 작성한다면,
컴퓨터가 그게 문법인지, 변수인지 알 수 없으므로 쓸 수 없다는 것입니다.

당연한 내용이므로 외울 필요는 전혀 없습니다!

0x02 Variables

변수란?

변수에 값 저장하기, 할당

변수를 선언했으면, 사용을 해야겠죠. 그렇다면 값도 저장해야 할 것입니다.
이 행위를 **할당** ***Assignment*** 이라고 합니다.

값 할당은 선언과 동시에 할 수도 있고, 이것을 **초기화** ***Initialization*** 라고 함
변수 초기화를 하지 않으면, 즉 값이 할당되어 있지 않은 변수는 **쓰레기 값**을 가짐

<u>즉, 값이 할당되지 않은 변수를 사용하면 안됨!(프로그램이 생각과 다르게 작동)</u>

```
자료형 이름 = 값;   이름 = 값;
int num = 10;       num = 245;
int a = 3, b = 2;    a =1, b=3;
```

값 할당 방법은 대입 연산자를 사용하여 왼쪽과 같으며,
이것은 같다는 의미의 등호가 아님!

여러 변수에 한꺼번에 값을 초기화하거나 할당해도 됨
전자는 초기화, 후자는 나중에 할당하는 방식

지금까지는 정수를 나타내는 int 자료형에 변수를 저장해 왔습니다.
그렇다면 문자, 소수 등은 어떻게 나타낼까요?
자료형을 통해 배워보도록 하겠습니다!

0x03 Data Type

여러가지 자료형들

컴퓨터의 저장 방법

자료형 ***Data Type*** 은 변수가 어떤 형태의 자료를 저장할 것인지 지정하는 것
알다시피 컴퓨터는 데이터를 저장할 때, 0과 1을 사용한 이진법을 사용

여기서 0과 1을 담는 기본 단위를 binary digit, 즉 이진수를 나타내는 **비트** ***Bit*** 라고 하며,
저장되는 공간, 컴퓨터의 기억장치를 일단은 **메모리** ***Memory*** 라고 함

32bit, 64bit 운영 체제 라는 말도 여기서 온 것

KB, MB, GB 등 컴퓨터에서 “바이트”라는 말을 사용 **1 byte = 8 bit**
변수는 컴퓨터 메모리의 공간을 차지하는데, 변수의 크기 단위로 바이트를 사용함

예를 들어 1바이트 크기의 자료형이 있다고 해봅시다.
이 자료형은 1 바이트, 즉 8 비트의 메모리를 사용합니다.
그리고 각각의 비트에는 0 또는 1 2가지 상태가 가능합니다.
그렇다면 8비트 안에서 표현 가능한 경우의 수는 2^8 이죠.

0x03 Data Type

2진수 체계

그렇다면 8비트로 2^8가지 수를 어떻게 표현할까요?

예를 들어 135라는 수가 있다고 해봅시다.

135는 10진수로 나타낸 것이고, (우리가 사용하는 수의 체계, 0~9로 수를 나타냄)
이것을 풀어서 써보자면,

$135 = 10^2 * 1 + 10^1 * 3 + 10^0 * 5$ 가 됩니다.

2진수로는 어떻게 나타낼까요?
위의 예시를 보면, 10진수로 나타낼 때 10의 거듭제곱에 자릿수를 곱한 것을 볼 수 있죠.

2진수도 이와 같습니다. 0, 1로 수를 나타내며, 2의 거듭제곱으로 자릿수를 표현합니다.

$$135 = 2^7(=128) * 1 + 2^6 * 0 + 2^5 * 0 + 2^4 * 0 + 2^3 * 0 + 2^2(=4) * 1 + 2^1 * 1 + 2^0 * 1$$

이므로, 10000111이 됩니다.
2진수로 변환하는 방법이나, 8진수와 16진수(원리는 같습니다)는
다음에 짚어보는 걸로 하고 지금 중요하지 않고, 지루하니까 넘어가도록 하겠습니다.

0x03 Data Type

정수를 저장하는 자료형

다음은 정수를 저장하는 자료형들

자료형	크기	범위
char	1바이트 (8비트)	-128 ~ 127 (2^8 = 256)
unsigned char	1바이트 (8비트)	0 ~ 255
short	2바이트 (16비트)	-32,768 ~ 32,767 (2^16 = 65,536)
unsigned short	2바이트 (16비트)	0 ~ 65,535
int (= long)	4바이트 (32비트)	-2,147,483,648 ~ 2,147,483,647 (2^32 = 4,294,967,296)
unsigned int (= unsigned long)	4바이트 (32비트)	0 ~ 4,294,967,295
long long	8바이트 (64비트)	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
unsigned long long	8바이트 (64비트)	0 ~ 18,446,774,073,709,551,615

unsigned 란 부호가 없다는 의미로, 부호 비트 사용 X
⇒ 원래보다 더 큰 수를 저장 가능

TMI1) int는 정수라는 의미의 Integer의 약자
TMI2) short와 long, long long 뒤에는 int가 생략되어 있다
TMI3) 부호가 있는 자료형의 앞에는 signed가 생략되어 있다

메모리 공간 사용의 효율성도 중요하므로 **변수의 용도에 맞게** 다른 자료형 사용!

데이터를 저장할 때,

부호를 나타내기 위해서 맨 첫 비트를 사용
보통 양수가 0, 음수가 1
이전의 135를 예로 들면
010000111

그러므로 char 자료형이
$2^8 = 256$ 이 아닌
$2^7 - 1 = 127$ 까지 나타냄

1을 빼는 이유는
0000 0000 = 0이기 때문에
0을 나타낼 비트가 필요

0x03 Data Type

소수, 즉 실수의 저장 방법

실수의 저장 방법은 정수보다 더 복잡합니다.
어떻게 0과 1만으로 소수를 나타낼 수 있을까요?

첫 비트로 부호를 나타내듯,
일부 비트는 정수 부분, 나머지는 소수 부분을 나타낸다고 가정합시다.
그렇다면 아래와 같이 나타낼 수 있겠죠.

이 방식을 **고정 소수점** ***Fixed-point*** 라고 부릅니다.

2^3	2^2	2^1	2^0	2^-1	2^-2	2^-3	합
1	0	0	1	0	1	0	8 + 1 + 0.25 = 9.25

0x03 Data Type

소수, 즉 실수를 나타내는 자료형

고정 소수점 방식은 나타낼 수 있는 수의 범위도 좁고, 부정확하기까지 합니다.
그러나 실수를 나타내는 다음 자료형을 보면 몹시 범위가 넓습니다.

자료형	크기	범위
float	4바이트 (32비트)	1.175494e-38 ~ 3.402823e+38
double (= long double)	8바이트 (64비트)	2.225074e-308 ~ 1.797693e+308

이것은 **부동 소수점** ***Floating-point*** 방식을 사용한 것입니다.

부동 소수점은 아닐 부, 움직일 동 이란 뜻이 아니고,
Float라는 말을 생각해보면, 소수점이 둥둥 떠다닌다고 생각해 볼 수 있습니다.

이전 슬라이드의 예시로 간단하게 설명하자면,
1001.010으로 나타낸 것을,
1.001010 * 8(2의 3승)로 표현합니다.

복잡해보이지만, 정확히 알 필요는 없고 다음만 알아두면 됩니다.
소수점이 움직이고, 소수 부분을 지수 형태로 저장합니다.
그러므로 매우 정확한 소수 표현은 불가능합니다.

C에서 문자는 “아스키 코드”를 이용하는데, 이것은 다음 주차에 알아보겠습니다!

More
부동 소수점에 관심이 있다면
IEEE 754 (IEEE Standard for Floating-Point Arithmetic)
을 찾아보세요!

0x04 Operators

연산자

연산자와 우선 순위

변수에 데이터를 저장했으니, 컴퓨터 즉 계산기라는 이름에 맞게
이제 데이터들을 사용해 계산, 즉 연산을 해보겠습니다.

연산을 하기 위해서는, +-*/ 와 같은 연산 기호, 즉 연산자가 필요하겠죠.

이런 연산자들을 공부할 때 생각할 것은 **우선순위와 결합성** ***Associativity*** 입니다.

우선순위란 어떤 것을 먼저 계산하는가에 관한 것입니다.
곱셈을 덧셈보다 먼저 한다. 간단하죠?
C의 연산자는 많고, 그에 따라 우선순위 또한 많은데,
자연스럽게 익숙해질 내용들입니다.

```
More
 괄호
   =>나머지
     =>논리
       =>할당
```

괄호 안의 연산을 제일 먼저 하고, 나머지 연산(사칙 연산 포함 익숙한 연산들)을 하고,
논리 연산(AND, OR, NOT 등)을 하고, 할당(대입)을 한다고 간단하게만 알아두시면,
더 배우시면서 익숙해질 수 있습니다!

0x04 Operators

연산자

연산자의 결합성

결합성은 어떤 방향으로 연산이 이루어지느냐 입니다.
예를 들어, 대입 연산자 “=”를 사용할 때, a=5;
라고 한다면, 5를 a에 대입하는 것이므로, 오른쪽에서 왼쪽 방향이라고 할 수 있겠죠.

반면, a/b를 계산한다면, a를 b로 나누는 것이므로, /는 왼쪽에서 오른쪽 방향입니다.

우선순위와 결합성은 외우는 것이 아닙니다. 새로운 문법을 배울 때 그 문법과 그에 필요한 새로운 연산자의 작용을 이해한다면, 더 쉽게 프로그래밍을 배우실 수 있을 것입니다.

지금 다 이해하실 필요는 없고, 조건문에서 논리 연산자를, 비트 연산에서 비트 연산자를, … 이처럼 순차적으로 배우시면 되니까, 이해가 안 간다고 포기하지 말고 따라오세요!

0x04 Operators

연산자

산술 연산자와 할당 연산자

오늘은 산술과 할당(대입)연산자들만 알아보고 가도록 하겠습니다.

산술 연산자에는 `+,-,*,/,%,++,--` 가 있습니다.
+,-,*,/ 는 각각 사칙 연산이며, %는 나머지 연산, 예를 들어 8%7= 1입니다.
++,-- 는 한 변수에 결합하여 각각 1을 증가, 또는 감소시켜 주는 증감연산자입니다.

한 변수에 결합하므로 앞, 뒤 중 어디에 붙냐에 따라 약간의 차이가 있습니다.
또, int형에만 쓸 수 있습니다.

```
--b; // 전위 연산자, b를 1 감소시키고 문장 실행
a++; // 후위 연산자, 문장을 실행하고 a를 1 증가시킴
```

할당 연산자에는 `=, +=, -=, *=, /=, %= …` 이 있습니다.
=는 아까 봤던 대입 연산자가 맞고, 그 외에 다른 연산자가 붙어있는 연산자는,
해당 변수에 연산을 한 뒤 대입을 한다는 의미입니다.
예를 들어, a+=5; 라는 말은, a = a+5; 를 줄인 것입니다.
이해가 잘 안 가실텐데, '=' 연산자는 오른쪽에서 왼쪽으로 결합하므로
a+5연산이 먼저 적용되어 a에 5를 더한 값이 a에 새롭게 대입된다는 의미입니다.

0x04 Operators

연산자

연산자	유형	결합성
[ ] ( ) . -> ++ -- (후위)	식	왼쪽에서 오른쪽
sizeof & * + - ~ ! ++ -- (전위)	단항	오른쪽에서 왼쪽
형식 캐스팅(Cast operator)	단항	오른쪽에서 왼쪽
* / %	곱하기	왼쪽에서 오른쪽
+ -	더하기	왼쪽에서 오른쪽
<< >>	비트 시프트	왼쪽에서 오른쪽
< > <= >=	관계	왼쪽에서 오른쪽
== !=	같음	왼쪽에서 오른쪽
&	비트 AND	왼쪽에서 오른쪽
^	비트 제외 OR (XOR)	왼쪽에서 오른쪽
|	비트 포함 OR	왼쪽에서 오른쪽
&&	논리 AND	왼쪽에서 오른쪽
||	논리 OR	왼쪽에서 오른쪽
? :	조건식	왼쪽에서 오른쪽
= *= /= %= += -= <<= >>= &= ^= |=	단순 및 복합 할당	오른쪽에서 왼쪽
,	순차적 계산	오른쪽에서 왼쪽

구분	연산자
할당 연산자*Assignment operator*	=, +=, -=, *=, /=, %=, … 등등
산술 연산자*Arithmetic operator*	+, -, *, /, %, ++, --
관계 연산자*Relational operator*	<, >, <=, >=, ==, !=
논리 연산자*Logical operator*	&&, ||, ==
비트 연산자*Bit operator*	&, |, ~, ^, <<, >>
삼항 연산자*Ternary operator*	?

이런 연산자들이 있다는 것만 일단 알아두고 넘어갑시다!

0x05 Stdio

표준 입출력 *STanDard Input/Output, stdio*

지금까지 변수니, 자료형이니, 연산자니 너무 뜬구름잡는 얘기처럼 들리셨을 수도 있습니다.
이제는 드디어 우리가 프로그래밍으로 연산을 시킨 결과를 알아볼 수 있습니다.
1주차 마지막 챕터인 표준 입출력을 배워봅시다!

처음 Visual Studio를 키면, 아까 봤던 Hello World! 코드가 있습니다.
#include <stdio.h> 는 뭐고, printf는 무엇일까요?
이것이 바로 표준 입출력 파트에서 배워볼 내용들입니다.

C에는 여러가지 일을 하는 함수들이 정의되어 있습니다.
하지만 모든 함수들이 포함되어 있지는 않죠.
C가 나올 당시에는 컴퓨터 사양이 그리 좋지 않았기 때문입니다.

아무튼, 다른 유용한 함수들을 따로 만들어 보관해두었는데, 이것을 헤더 파일이라고 하며,
기본으로 탑재된 '표준 라이브러리'에서, 또는 사람들이 만든 '사용자 정의 라이브러리'에서
필요할 때만 코드에 포함시키도록 만들어 두었습니다.

그 중에서도, cmd와 같이 text기반의 'console'에서 키보드를 통해
입력과 출력을 받을 수 있는 표준 입출력이 정의된 헤더 파일이 바로 stdio.h 입니다.

0x05 Stdio

표준 입출력 *STanDard Input/Output, stdio*

#include <stdio.h> 를 통해 이 헤더를 코드에 포함시킬 수 있습니다.

C언어로 만등러진 코드는 컴파일 과정, 즉 프로그래밍 언어를 컴퓨터가 이해하도록 바꾸는 과정을 통해 실행 가능한 파일로 만들어집니다.

이 과정에서 제일 처음에 해당하는 전처리기에 명령을 주는 부분이므로
특이하게도 코드의 맨 앞에 #이 붙어있습니다.

stdio.h에는 printf, scanf, fprintf, fscanf 등등이 정의되어 있는데,
차차 나중에 알아보기로 하고, 오늘은 printf와 scanf에 대해 알아봅시다.

0x05 Stdio

표준 입출력 *STanDard Input/Output, stdio*

printf

먼저 printf는 문자열(숫자 포함)을 출력하는 함수입니다.

```
printf(“3”); // 3이 출력됩니다.
printf(“a”); // a가 출력됩니다.
printf(“Hello World!”); // Hello World가 출력됩니다.
```

만약 상수, 미리 정해진 문자열이 아닌, 변수의 데이터를 출력하고 싶다면 어떻게 할까요?

바로 **서식 지정자** ***Format specifier***을 이용합니다.
printf에서 f가 바로 그 formatted를 의미합니다.

즉, 변수의 데이터 값을 출력할 형태를 지정하여, 변수의 데이터를 출력할 수 있다는 것입니다.

사용 방법은 간단합니다. 문자열에 서식 지정자를 포함해서 넣어주고,
쉼표로 구분하여 해당 변수를 넣어주면 됩니다.

예를 들어, int 형의 정수를 출력하려면 %d 를 이용합니다.

```
Printf(“%d %d %d”, num1, num2, num3); // 띄어쓰기로 구분하여 순서대로 각각 수가 출력됨
```

0x05 Stdio

표준 입출력 *STanDard Input/Output, stdio*

서식 지정자 목록

서식 지정자	출력 형태
%c	문자 (char)
%s	문자열 (string)
%d, %i	부호 있는 10진 정수 (decimal, integer)
%f	float 형태의 고정 소수점으로 표현한 실수 (float)
%lf	double 형태의 고정 소수점으로 표현한 실수 (double)
%o	부호 없는 8진 정수 (octal)
%u	부호 없는 10진 정수 (unsigned int)
%x	부호 없는 16진 정수 (hexadecimal)
%X	부호 없는 16진 정수 (hexadecimal)
%lu	부호 없는 10진 정수 (unsigned long)
%llu	부호 없는 10진 정수 (unsigned long long)
%ld	long 형태의 10진 정수 출력
%lld	long long 형태의 10진 정수 출력
%%	% 기호 출력

%[플래그][폭][.정밀도][길이]서식 지정자

위와 같은 방법으로 더 정밀한 정보를 담을 수 있습니다.

- 플래그는 남은 빈칸을 뭐로 채울것이냐, 또는 부호를 의미하며,
- 폭은 말그대로 출력할 칸 수
- 정밀도는 소수점 몇째 자리까지 나타내는가
- 길이는 거의 사용 X
- 자세한 건 링크에!

printf("%6d", 40); // ____40 출력됨. 폭이 6칸이므로 앞에 4칸의 공백(_ 아님!)이 생긴다.
printf("%03d, 2); // 002 출력됨. 폭이 3칸 + 플래그 0이므로 남는 2칸에 0이 채워진다.
printf("%.3f", 1.27); // 1.270 출력됨. 정밀도가 3이므로 소수 셋째 자리까지 출력된다.

폭, 정밀도 정도만 알아두시면 괜찮을 것 같습니다.
전혀 필요 없는데 컴프 시험에는 나올 수도 있습니다!

0x05 Stdio

표준 입출력 *STanDard Input/Output, stdio*

printf

표준 출력에서 개행, 즉 엔터를 하려면 어떻게 해야 할까요?
또, 문자열 안에 “를 포함하면, 컴퓨터가 문자열이 닫힌 것으로 간주해서
원하는 대로 출력이 되지 않는데 어떻게 해야 할까요?

이를 위해 **이스케이프 시퀀스** *Escape Sequence* 를 사용합니다!

이스케이프 시퀀스	ASCII 코드	표현
\a	0x07	알람***A****lert* (비프음, 벨)
\b	0x08	백스페이스***B****ackspace*
\f	0x0C	폼피드*form* ***F****eed* (커서를 다음 페이지로 넘김)
\n	0x0A	개행 문자***N****ew line* (커서를 다음 줄 첫번째로 넘김)
\r	0x0D	캐리지 리턴*carriage* ***R****eturn* (커서를 첫번째 위치로 넘김)
\t	0x09	키보드의 ***T****ab*
\v	0x0B	수직으로 탭***V****ertical tab*
\\	0x5C	백슬래시 문자
\’	0x27	작은 따옴표 (문자를 표현하는 ‘과 혼동하지 않기 위함)
\”	0x22	큰 따옴표 (문자열을 표현하는 ”와 혼동하지 않기 위함)
\?	0x3F	물음표 (삼항 연산자로 잘못 해석되는 경우를 막기 위함)

\n, \t, 따옴표 정도만 알아두시면 됩니다.

\, % 등의 특수 문자는 두 번 쓰면 나타낼 수 있다는 점도요!

어차피 쓰다 보면 익숙해지니 걱정 마세요!

0x05 Stdio

표준 입출력 *STanDard Input/Output, stdio*

scanf

scanf는 이름에서 알 수 있듯이, 서식 지정자에 맞춰 입력을 받는 함수입니다.

printf와 비슷하지만, 입력 값이 저장될 변수 앞에 &이 붙는다는 점이 다릅니다.
이는 나중에 포인터에 대해 다룰 때 공부하겠습니다.

예를 들면,
scanf("%d %lf", &a, &b); // 정수를 입력받아 a에 저장하고, 실수를 b에 저장함

입력을 받을 때에는, console이 사용자의 입력을 기다리게 되는데,

여기서 자유롭게 입력을 할 수 있고, Enter를 입력하면,
입력했던 모든 값이 buffer라는 것에 담겨 프로그램으로 전달됩니다.

위의 예시에서 두 서식 지정자 사이에 공백이 있는데,
이는 버퍼에 있는 모든 공백을 건너뛰겠다는 의미입니다.
몇 개의 공백이나 엔터가 있든, 뛰어넘어 실수 값을 찾아 b에 저장합니다.

단, A와 같이 입력 서식에 맞지 않는 값은 건너뛰지 못하기에,
입력 서식을 잘 저장해주는 것이 중요합니다.

0x06 Practice

실습

```
예시 -
https://www.acmicpc.net/problem/10869
```

0x07 Assignment

과제

https://www.acmicpc.net/problem/2557 - 2557: Hello World, 기본 출력
https://www.acmicpc.net/problem/10171 – 10171: 고양이, Escape Sequence
https://www.acmicpc.net/problem/1000 - 1000: A+B, 변수,사칙연산 + 출력
https://www.acmicpc.net/problem/1008 - 1008: A/B, 변수, 자료형, 사칙연산, 서식, 출력

3문제 이상

부족한 첫 수업
들어 주시느라 수고 많으셨습니다!

들어주셔서 감사하고, 질문과 밥약은 언제든 환영이에요

다음 주에 봅시다~