2일차

1일차 복습
- GitHub관련 오늘 중으로 마무리 지어라 - 내가 도와서라도 끝낸다.

터미널(쉘): 우분투 서버(OS)에서 하나씩 열 수 있는 ???
터미널에서 쉘 명령어를 통해 원하는 작업을 시행한다
기본 bash 쉘을 사용하고 있다
초기 명령어

zsh(지쉘) : 쉘 명령어와 명령어를 공유하기에 차이가 없다
 - 사용할 것이라면 zsh uhmyood를 추천한다

*** 쉘 명령어를 찾아봐라 - 숙제

C언어

- CC(컴파일) 과정
텍스트(소스) --전처리ㅡ어셈블리 ㅡ 오브젝트 ㅡ > 바이너리 파일(실행파일)
limit.c ----------> limit : CC(컴파일)과정

위 내용을 여러 파일을 한번에 다루기 위해 makefile을 사용하였음

- 타입 Type(형) (2진수)
운영체제에 따라 데이터 크기의 변동이 있을 수 있기 때문에 type의 형태를 외울게 아니라 이해가 필요함

-- 숫자형 - 정수형 : int, char ㄱ
  | long, long long | -> 2의 보수를 통한 음수로 변경가능
  ㄴ 부동소수형 : double, float ㄴ

  - 음수의 표현 : unsigned int

2일차 수업

- include
#include A
#include B
에 대해 A와 B를 불러오는 것이다.
만약 A파일 안에 #include B가 있는 상태로 위와 같이 사용한다면
A와 B를 불러오는데, A안에 B를 한번 더 불러오기에 A, B, B를 불러오는 것 이기에
이러한 부분에서 include의 사용에 주의해야한다

- main 함수의 형태
int /main/()/{ return 0;}
리턴값의 타입 / 식별자(함수명) / (나중에 설명해줌) / { ~~~return (맨앞의 선언에 해당하는 데이터);}
선언의 예)
int -> 숫자
void -> 공란
char -> 문자

- Type
-- -정수형
  ㄴ 문자형 char()
  ㄴ void - 함수 출력을 하지 않을 때
        ㄴ 함수 입력을 안 받을 때
        ㄴ 임의의 인자를 전달*
임의의 인자: type 미정이라 뭘 줄진 몰라도 뭔갈 줄 것이기에 미리 선언이라도 한다는 느낌?

%f(실수형)이나 %lf에 대해 중간에 .(숫자)의 형태로 자리수를 지정해줄 수 있다
ex) %.2f -> 소수점 2번째 자리까지 출력
ex) %.3f -> 소수점 3번째 자리까지 출력

4. 연산자(계산)
폰 노이만 머신 -- 폰 노이만 구조 (바이트머신 : 1바이트 단위로 처리)
ㄴ 하버드 구조

┌code ┐ ┌ cpu ┐
└information ┘ -> └ ┘ 세트로 cpu에 들어가서 처리되면 폰 노이만 구조

[code] ┌ cpu ┐
[information] -> └ ┘ 각 개별로 cpu에 들어가서 처리되면 하버드 구조

컴퓨터의 구조 CS(computer science) - 대학 과정일 정도로 복잡하니 알아서 찾아봐라
cpu / ram / harddisk

변수(variable): 변화가능한 수 ┌ a
└ b
└ var_a
- 저장 가능한 메모리 공간에 확보 (식별화)
ㄴ 램 ┬ 메모리
| 힙 : 아무 자리에나 malloc, calloc freo의 방식으로 저장
(자유롭다보니 메모리 리킹이 일어날 수 있음)
ㄴ 스택 : 주로 우리가 한 코딩: 바닥에서부터 쌓아올리는 방법
(함수 형태로 쌓아 올리다 보니 변수의 태가 변하면 치명적임)
동적할당 변수 -> 힙 방식으로 램에서 처리하는 변수

상수(constant): 코드가 변하지 않음
0 1 2 3 4 5 6 7 8 9 -> 이런 숫자에 대해 이진법으로 설명하고 일일이 만들지 않도록함
램의 메모리에 얹어버림

```
#include <stdio.h>
int main(void){
    int a, b;
    printf("Input a, b : ");
    scanf("%d %d", &a, &b);

    printf("PLUS %d + %d = %d\n", a, b, a+b);
    printf("MINUS %d - %d = %d\n", a, b, a-b);
    printf("MULTIPLY %d * %d = %d\n", a, b, a*b);
    printf("DIVISION %d / %d = %d\n", a, b, a/b);
    printf("나머지 %d 나머지 %d = %d\n", a, b, a%b);
    return 0;
}
```

연산자
1) 산술연산자 : +, -, *, /, %
2) 대입연산자: =
3) 복합연산자: (1),2)의 결합) +=, -=, *=, =+
4) 증감연산자: ++, --
        - 전위: ++a -> 올려서 출력해라
        - 후위: a-- -> 출력하고 올려서 저장해라

전역변수: 프로그램 끝나도 살아남음
- 자동으로 초기화가 되기 때문에 많이 사용했었음
- 해킹의 단초라 사용을 안 하는 추세
지역변수: 프로그램 끝나면 사라짐
- 가비지값이 들어가 있을 수 있기에 초기화를 해주어야 함

대입연산자가 증감연산자보다 우위임

printf("%d %d %d\n", 100, 0144, 0x64);
진수 관련 출력 방법 :
0x 로 붙으면 16진법
0o로 붙으면 8진법 옥타로 묶어라
0b로 붙으면 2진법 바이너리로 묶어라

printf("%lldLL\n", 10000000000LL);
안과 밖에 LL을 써줌으로써 LongLong 형태라고 명시해준 것

printf("%f %f\n", 2.718, 3.141592F);
f 실수형으로 표현을 한다고 뒤에 F로 명시해준 것

printf("%d %d %d\n", 'A', 'a', '0');
뒤에 A a 0을 숫자형으로 바꾸면 아스키 코드값

printf("%c %c %c\n", 65, 97, 48);
뒤에 숫자를 글자형으로 바꾸면 아스키 코드 값에 의해 A a 0

```
//genderRatio.c
#include <stdio.h>

int main()
{
    int man, woman;
    double sum, manRate, womanRate;

    //입력받는 코드
    printf("남자의 수를 입력하시오 : ");
    scanf("%d", &man);
    printf("여자의 수를 입력하시오 : ");
    scanf("%d", &woman);
    //연산하는 코드
    sum = man + woman;
    manRate = man / sum * 100;
    womanRate = woman / sum * 100;

    printf("남자의 수는 %d명이고 여자의 수는 %d이다.\n", man, woman);
    printf("총 수는 %.f명\n남자의 비율은 %.2f%%\n여자의 비율은 %.2f%%\n", sum,
manRate, womanRate);

    return 0;
}
```

==>타입 캐스팅

```
sum = (double)man + (doulbe)woman;
```

```
printf("총 수는 %.f명\n남자의 비율은 %.2f%%\n여자의 비율은 %.2f%%\n", sum,
manRate, womanRate);
```

총수는 남자와 여자를 int로 선언해서 int에 해당하는 값이 나와야 하지 않나,
왜 float 형식을 쓰는가
sum = man + woman; 에서 sum이 이미 상단에서 double로 지정이 되어
%f의 타입으로 불러 줘야 합니다.

연산자2

1) 관계연산자: >,<,<=,>=, ==, !=

앞, 뒤의 관계에 대해 연산하기에 두 개의 데이터가 필요하다

이에 대한 결과는 yes나 no로 출력이 된다 ([a<b] -> yes(1) or no(0))

2) 논리연산자: &&(and), ||(or), !(not)

참(yes(1))과 거짓(no(0))에 대해 연산

yes &&(and) yes => yes

yes &&(and) no => no

no &&(and) yes => no

no &&(and) no => no

yes ||(or) yes => yes

yes ||(or) no => yes

no ||(or) yes => yes

no ||(or) no => no

!(not) yes => no

!(not) no => yes

3) 형(Type casting)변환연산자: (new type)변수

-> 변수에 대해 (new type)으로 변경함, 밑에서 추가로 형변환이 없다면 언급한 상태로 유지

4) sizeof연산자: 결과가 몇 바이트(byte)인가에 대해 long int(%ld)의 형태로 나옴

*8하면 비트(bite)수를 구할 수 있음

- 힙에서의 메모리 사용시 메모리 크기(변수의 크기)가 몇 바이트인지 지정할 수 있음

int a;

sizeof(a);

->a에 대한 int로 표현시 바이트수

5) 비트연산자: &(and), ~(not), |(or), <<, >>, ^(xor)

A: $01001110_{(2)}$

B: $01101010_{(2)}$

&: $01001010_{(2)}$ -> and

| : $01101110_{(2)}$ -> or

^: $00100100_{(2)}$ -> xor(서로 달라야 참)

~A : $10110001_{(2)}$ -> not

A<<1 : $10011100_{(2)}$ -> 왼쪽으로 1칸씩 밀기(밀때는 0으로, 숫자는 미는 칸 개수)

[챕터 4-완]