5. ↗ ~에게 or 으로

배열 : aa[0], aa[1], aa[2]… 와 같이 첨자를 붙임.

↳ 데이터형식 배열이름[개수] :

ex) int aa[3];

배열 크기 : sizeof(배열이름) / sizeof(배열의 형식);

include <string.h> → 문자열 처리 헤더파일

strlen( ) → 문자열의 개수를 알려줌.

strcpy(a, b) → 문자열 a에다가 b를 복사함.

strcat(a, b) → 문자열 a 뒤에 b를 집합한다.

strcmp(a, b) → a-b의 결과값을 돌려줌.
즉, ↳ =0 이라면 문자열이 같다.

gets(ss) : ss라는 문자열을 받음. = Scanf

Puts(ss) : ss라는 문자열을 출력. = Printf
자동 줄바꿈.

2차원 배열

int aa[행][열]

int aa[0][0] : 포인터값

3차원 배열

int aa[층][행][열];

a[0] a[1] ↓ 층은 배열개수임.

6. Pointer.

Int = 4 bite

char = 1 bite

& aa[0] → 주소 = 1031번지라면 aa[1]의 주소는 1035이다.

aa+1 == & aa[1]

포인터 선언 : * p 변수명;

ex) int * pnum; ~으로부터

※ char ch;
char *p;
ch='A';
p=&ch;
*p='A';
↓ 1bite

같은 데이터 형식에 같은 포인터 사용가능

char* pch = str

str[3] == *(p+3)

Printf("%d", P) → 주소값에 표시 ex) 1016

Printf("%d", *P) → 주소에 위치한 실질적인 값 출력 ex) 123

주소값을 표시하려면 &를 넣는다.

포인터 변수는 무조건 4bite, 반드시 주소만 대입

7.

Function (함수) : 입력에 대한 출력을 제공

↳반환 자료형 ↗ 매개변수목록

int sum (x, y)
↳함수이름

{ ↳ 함수몸체 }

→

매개변수 데이터 형식
↗ 과 이름

int sum (int a, int b){

int result; → 지역변수

result = a+b;

return result; → 반환.
↳ int로 정의 했기에 반환값도 정도형이나 정수로 되어야함.

}

반환값을 담을 변수선언 ← hap = sum(100, 200);
함수호출

↗반환값이 없음.

void func1( ){

printf(~);

}

func1( ); → 함수 실행

for, while, if 도 지역변수 존재

전역변수 global

int, float, char, int*, double*, char*도 함수가능
↳주소값 함수.

getchar(); 남아있는 버퍼를 비운다.

복사하는 값은 값으로도 전달가능하고 주소로도 전달이 가능하다.

void func1 (int a){
a = a+1;
}
10

int main( ){
int a = 10;
func1(a);
}

10을 넣고 1로 변화된다.

기존값 변환x

void func1 (int* a){

*a = *a+1
↳해당주소의 실질적인 값

Void main( ){
int a=10;
func1(&a);
}

주소값을 보냄

기존값이 변함 (주소참조 이기때문)

8.

표준 입력 : scanf, gets, getchar
표준 출력 : printf, puts, putchar

#include<conio.h>
puts( ), gets( ) : 문자열만 취급. → 처리속도가 빠름
getch( ) : 모니터에 보여지지 않는다.]→ 문자 하나만 입력
getche( ) : 모니터에 보여준다.]
getchar( ) : 모든 지정된 문자를 하나만 입력받음.

Putchar( ) : 문자 하나만 출력
Putch( )
getch를 확인하기 위해서 Putch를 씀.

파일 입출력 : fscanf, fgets, fgetc
fprintf, fputs, fputc

* 열기모드
r : 읽기
w : 쓰기

1. 파일 포인터 선언
FILE *변수이름;
2. 변수이름 = fopen("파일이름", "열기 모드") ← 파일 경로도 포함(//)
3. 내용작성
4. fclose(파일포인터);

fgets(문자열을 담을 배열의 변수명, 읽을 최대 문자수, 파일 포인터);
fscanf(파일 포인터, "서식", 입력할 매개변수들 …);
fputs(출력할 데이터, 파일 포인터);
fprintf(파일 포인터, "서식", 출력할 매개변수들 …);

Dos 명령어

type 파일이름 : 지정한 파일의 내용을 출력
Copy 소스파일 타겟파일 : 타겟 파일에 소스 파일 내용 복사.

9.

상수 : 바뀌지 않는 수
포인터 상수 : 바뀌지 않는 메모리 내에서 위치함
포인터 변수선언 : *을 이용 → 주소값만 저장해야 한다.
&은 일반 변수의 주소를 알아내기 위해 사용하는 연산자.

동적 결정 : 발생하는 데이터 크기에 따라 실행
정적 배열 : 메모리 낭비가 심함

포인터 변수 = (포인터 변수의 데이터형 *) malloc(sizeof(포인터 변수형) × 필요한 크기)

```
int *p;
    p=(int*)malloc(sizeof(int)x 필요한크기);
    free();
      ↳ 남은 메모리를 운영체제에 반납
    p=(int*)calloc(필요한크기, sizeof(int)); → 처음부터 0으로 할당됨.
    p=(int*)realloc(p, sizeof(int)x 필요한 크기);
          ↳ 이미 할당한 메모리크기 재할당.
```

11.

구조체 : 서로 다른 변수를 하나의 블록으로 묶는것!

```
struct bibim {
    int a;
    char b;
    float c;
    char d[5];
};
```

struct 구조체형이름 구조체변수;

선언부

ex) struct bibim b1;

b1. int a; char b; float c; char d[5]; → b1.a = 10;

초기화 가능 ← struct bibim b1[3];

b1[0].a ~~~

```
struct bibim *p;
p = &b1;
p->a = 10;
printf("%d", a); -> 10
```

union bibim { int a; char b; }; ← 내용물중 1개만 사용가능.

int (≡ ← char

union bibim b1;

b1.a = 10;

enum week { sun, mon, tue ···· } ← 연거형으로 앞터 1개씩 증가

0 1 2 ···

enum week ww;

ww = sun; → 0으로 출력됨