C 언어

01 프로그램 만들기

'프로그래밍 언어'

230208

C언어
↓파생
C++, C#

이후 ┌ 객체지향언어 : 자바, 파이썬, C++
└ 자료구조, 알고리즘 공부

OS 개발을 위한 언어 → 임베디드 프로그램에 많이 쓰임

'프로그래밍 절차 이해·데이터 처리 과정을 메모리 상황과 함께 고민'

pros
- 시스템 프로그래밍 가능 (하드웨어 제어)
- 이식성 갖춘 프로그램 만들수 있음
- 함수 사용하여 개별 프로그래밍 가능

소스파일 (C언어) → 컴파일 → 실행파일 (기계어)

에러수정 : 디버깅 debugging

컴파일 : Ctrl + Shift + B

실행 : Ctrl + F5

소스파일 → 전처리 → 전처리된 소스파일 → 컴파일 → 개체파일 → 링크 → 실행파일
(Startup code → 링크)

stdio : standard input output (표준입출력)

2-1 상수와 데이터 출력

main 함수

함수원형 function prototype: `int main(void)`

몸통:
```
{
    // 실행코드
    return 0; → 프로그램 종료
}
```

//, /**/ → 주석문, 컴퓨터 처리 X. for 인간

; → 문장 끝 표시

printf

print formatted

행 바꿀 때 **제어문자** 사용 (백슬래시 \)

\n	줄 바꿈 new line	("Goot\bd\tchance\n")
\a	벨소리 alert	Goot \bd→ Good
\r	맨 앞으로 이동 carriage return	\t→ Good V chance
\t	탭 위치로 이동 tab	("Cow\rWla\n");
\b	한 칸 왼쪽으로 이동 backspace	Cow \r→ Wow

정수와 실수 출력

%d demical / %lf long float

```
print ("변환 문자", 숫자)
printf ("%d\n", 10);
printf ("%lf\n", 3.4);
```

%lf로 출력시 소수점 이하 6자리까지 출력

↳ 자릿수 변경 시 %와 lf 사이 소수점 찍고 자리수 지정 `printf("%.1lf\n", 3.45);` 첫째 자리까지 출력 둘째 자리에서 반올림

변환문자 여러개 사용하기 `printf("%d과 %d의 합은 %d입니다. \n", 10, 20, 10+20);`

↳ 변환 문자의 개수와 출력할 값의 개수가 일치해야 함. 차례로 출력

Keypoint

C 프로그램은 main 함수로 시작

//. /**/ 주석문

printf 함수는 데이터를 화면에 출력할 때 사용

제어문자를 문자열 안에 포함시키면 그 기능에 따라 출력형태가 바뀐다.

확인문제

1. ② 2. %d, %d %lf 3. Happy Baby

2-2 상수와 데이터 표현 방법

진법별 수 표현 방법

		12 표현식
10진수	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	12
숫자 앞 0 8진수	0 1 2 3 4 5 6 7 10 11 12 13 14 15 16 17	014
숫자 앞 0x 16진수	0 1 2 3 4 5 6 7 8 9 A B C D E F	0xC

+) 8진수로 출력하려면 %o, 16진수로 출력하려면 %x

실수 상수 표현법

정규화 표기법 (normalization) : 소수점 앞에 0이 아닌 유효숫자 한 자리를 사용하여 지수형태로 바꾼것

%.le 변환문자 : 소수형 형태의 실수 → 지수형태로 출력

→ 소수점 이하 자리수 설정시 %와 le 사이에 점을 찍고 원하는 숫자 함께 사용

```
printf("%.le\n", 0.0000314); → 3.140000e-05
printf("%.2le\n", 0.0000314); → 3.14e-05
```

문자와 문자열 상수 표현법

'문자' (작은 따옴표) "문자열" (큰 따옴표)

%c %s (변환문자 없어도 출력 가능)

상수가 컴파일 된 후의 비트형태

	크기(byte)	크기(bit)	바뀌는 형태	
정수	4	32	2진수	코드가 컴파일러를 거쳐 컴파일 되어야
실수	8	64	IEEE 754표준 double형	비교소 연산자는 명령어가 되고
문자	4	32	아스키코드값과 같은 2진수	상수는 연산 가능한 형태로 바뀜.

bit (1과 0) —8개→ byte (2^8 → 256가지) (1바이트 = 256가지)

1바이트 = 8비트

정수는 4바이트로 처리하나, 8바이트로 만들고 싶을때 정수에 접미사 LL or ll (Long Long)을 붙여 사용.

음수의 변환

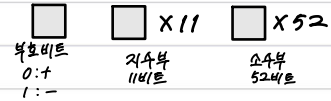
이유 : 특별한 변환 과정없이 양수와 음수 바로 더할 수 있음.

: 2의 보수로 처리 (2진수의 0과 1을 바꾼 상태(1의보수)에 1을 더한 값)

-10 절댓값→ 10 2진수→ 00000000 x 3 00001010 1의보수→ 1111 1111 x 3 11110101 1을더함→ 11110110

실수 상수 (IEEE 754 표준 double)

64비트 (부호 1 + 지수부 11 + 소수부 52)



+6.5 : 2진수 110.1 → 정규화 1.101 x 2^{+2} ← 2진수식까지

① 부호 +

② 지수 2 + bias(1023) → 1025 : 10000000001 (지수부)

③ 유효숫자 1101 → 101 0x49

(소제외하고 집어넣기)

3-1 변수

정수 int　　문자 char

실수 double　　문자열 char 배열

230303 Fri

int형 변수 선언 방법

```
int a;
```
자료형 변수명

변수 선언과 대입 규칙

대입연산자 사용하여 변수값 초기화·저장

(초기화하지 않은 변수에는 쓰레기값 들어있음!)

1. 중괄호의 블록 { } 안에 변수를 선언하며 선언한 위치부터 블록끝까지 사용할 수 있음.
2. 컴파일러에 따라 변수의 선언 위치가 제한될 수 있음.
3. 변수의 자료형이 같으면 동시에 둘 이상의 변수를 선언할 수 있음. (ex) int a,b,c
4. 대입 연산자 (=)는 연산자 왼쪽의 변수에 오른쪽의 값을 저장함.

```
a = 10;   ← 상수 대입
b = a;    ← 변수 대입
c = a+20; ← 수식대입
```
(변수)

5. 변수는 대입 연산자 왼쪽에서는 저장공간, 오른쪽에서는 값

```
int a,b;
a = 10;   → 저장공간으로 사용하는 변수 l-value
b = a;    → 값으로 사용하는 변수 r-value
```
(저장공간, 값)

정수 자료형

변수 자료형, 데이터형

- 정수형
- 실수형

1 bite

각 자료형 → %d 사용

long형 → l을 붙여 %ld로 출력

long long형 → %lld로 출력

char	1바이트 (8비트)	
short	2바이트	
int	4 바이트	
long	4 바이트	
long long	8 바이트	

저장가능한 값의 범위 $-2^{비트수-1} \sim 2^{비트수-1}-1$

ex) char형 $-2^7 \sim 2^7-1 = -128 \sim 127$

컴파일러는 0~127 사이의 정수(아스키코드 값)로 바꾸어 처리 ⇒ Char형 변수로 저장시 가장 효율적임.

언제 어떤 자료형을 사용할지 고민이라면?

→ 특별한 경우가 아니라면 정수형은 int 사용!

→ long형은 큰 값을 저장할 때 사용

※ 자료형의 크기가 궁금하다면 sizeof 연산자로 확인 가능 (바이트)

```
printf("long long형의 크기 : %d바이트\n", sizeof(long long));
```

```
char kor = 70;
char eng = 80;
Char
```

unsigned 정수 자료형

↳ 음수가 없는 데이터 저장시!

%u

%lu

%llu

%d는 부호까지 생각해서 10진수로 출력하는 변환문자

%u는 부호 없는 10진수로 출력하는 변환문자

a = 4294967295 입력 후 %d로 출력 → -1

a = -1 입력 후 %u로 출력 → 4294967295

→ 메모리 내에 저장된 형태(32비트)가 1111 1111 ×4 로 같음.

∴ undesigned 자료형은 항상 양수만 저장하고 %u로 출력하기!

실수 자료형

%f %lf %Lf

유효숫자

float 7

double 15

long double 15 이상

※ 실수형은 저장하는 값에 따라 숫자가 정확하게 표현 될수도, 아닐수도 있음.

문자열 저장

char %s

선언에 따라 크기가 달라지므로

대입 연산자 사용 불가.

char 배열명[문자열길이+1] = 문자열;

ex) char fruit[6] = "apple";

∵ 컴파일러가 문자열의 끝에 \0 (널 문자 null character)을 자동으로 추가하기 때문.

char 배열에 새로운 문자열을 저장하려면 strcpy 함수 사용

String copy, 문자열복사

const를 사용한 변수

변수선언시 const를 붙이면 초기화 된 값을 바꿀수 없다! → 변수를 상수처럼 사용 가능!

선언과 동시에 초기화 const 자료형 변수명 = 값;

예약어 와 식별자

예약어 reserved word key word

컴파일러와 사용방법이 약속된 단어

식별자 identifier

필요에 따라 만들어 사용하는 단어

식별자 - 알파벳 A~Z, a~z, 숫자 0~9, 밑줄 _ 로 만듦

숫자로 시작 불가. 대·소문자는 서로 다른 식별자로 인식

예약어는 식별자로 사용 불가

이름이 필요할 때 언제든지 만들어 사용하면 됨

4-2 그 외 유용한 연산자

형변환 연산자

피 연산자의 형태를 원하는 형태로 바꿈

int a = 10; 이라고 선언하면 a는 정수형이고 그 안의 값은 정수이다.

이 값을 일시적으로 실수로 사용하고 싶으면 a 앞에 괄호로 (double)처럼 원하는 자료형을 넣으면 됨

정수를 실수로 바꾸는 경우

(double) 10 → 10.0

(double): 연산자 / 10: 피연산자 (정수) / 10.0: 연산결과 (실수)

실수를 정수로 바꾸는 경우

(int) 10.7 → 10

10.7: 실수 / 10: 정수

자동 형 변환

: 암시적인 형 변환. 묵시적인 형 변환

예상치 못한 값의 변형이 생길수있음

컴파일 과정에서 피연산자의 형태가 다르면 형태를 일치시키는 작업을 수행

기본 규칙 : 크기가 작은 값이 큰 값으로 바뀌는 것

sizeof 연산자

: 피연산자 하나만 사용가능

피연산자의 크기를 바이트 단위로 계산해서 알려줌

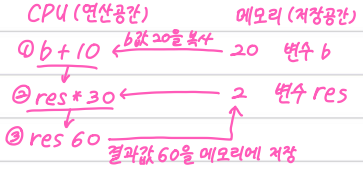
sizeof(int) sizeof(10) → 꼭 괄호 사용!

복합 대입 연산자

+=, -=, *=, /=, %=

왼쪽 피연산자는 반드시 변수가 와야한다.

오른쪽 항의 계산이 모두 끝난 다음에 복합대입연산자는 가장 마지막에 계산한다.



콤마 연산자

한 번에 여러 개의 수식을 차례로 나열해야할 때 사용.

대입연산자보다 우선순위가 낮음.

230314

조건 연산자

유일한 삼항 연산자. ?와 : 기호를 함께 사용

첫 번째 피연산자가 참이면 두 번째 피연산자가 결괏값이 되고

참이면 a선택 / 거짓이면 세 번째 피연산자가 결괏값이 됨.

(a > b) ? a : b

거짓이면 b선택

res = (a>b) ? a : b; ④

① a=10, b=20 대입시 10>20은 거짓.

② 조건연산자에서 거짓일 때 세번째 값(b) 선정

③ 선정한 b값을 res에 대입한다.

비트 연산자

비트논리연산자 비트이동연산자

데이터를 비트단위로 연산. & | ^ >> <<

논리곱 연산 & 배타적논리합연산 ^ 논리합 연산 | 부정연산자 ~ 비트이동연산자

a 10:1010
b 12:1100 → 1000:8

결괏값 서로 다를 때 참
0110:6

둘중하나라도 참이면 1
1110:14

~a:-11

01010 왼쪽한칸씩
→ 10100:20

오른쪽두칸씩
0010:2

연산자 우선순위와 연산 방향

단항 연산자 > 이항 연산자 > 삼항 연산자 순서로 연산

산술 연산자 > (비트 이동 연산자) > 관계 연산자 > 논리 연산자