C언어 스터디 2차시

컴퓨터학부 23 유상원

목차

- 변수와 상수
- 입력
- 연산

변수와 상수

변수

- 특정 값을 저장하는 메모리 공간
- 저장하는 값의 종류를 자료형이라고 함 (e.g. 정수, 실수 등)
- 선언하는 위치에 따라 지역변수, 전역변수 등 종류가 다양함

변수

- 변수 선언
  - 형태: [자료형] [식별자];
  - e.g. `int a;`
- 변수 선언 및 초기화
  - 형태: [자료형] [식별자] = [리터럴];
  - e.g. `int b = 123;`
- 여러 개의 변수를 동시에 선언 및 초기화
  - 형태: [자료형] [식별자] (= [리터럴]), [식별자] (= [리터럴]), ...;
  - e.g. `double pi = 3.14, tau = 6.28, e;`

변수

- 사실 이 세 가지 형태가 전부가 아님
  - Storage Class (e.g. auto, register, static, extern, typedef)
  - Type Qualifier (e.g. const, volatile, restrict, _Atomic)
  - e.g. `register int a = 123; volatile int b = 456;`

- 자세한 설명은 생략
  - C언어를 배워봤는데 처음 보는 키워드라면 한번 찾아보는 것도 나쁘지 않음

자료형

자료형	종류	크기*	범위*	Suffix
char	정수	8bits (1byte)	-128 ~ 127	
unsigned char	부호 없는 정수	8bits (1byte)	0 ~ 255	
short	정수	16bits (2bytes)	-32,768 32,767	
unsigned short	부호 없는 정수	16bits (2bytes)	0 ~ 65,535	
int	정수	32bits (4 bytse)	-2,147,483,648 ~ 2,147,483,647	
unsigned int	부호 없는 정수	32bits (4 bytes)	0 ~ 2,147,483,648	u 또는 U

*자료형의 크기는 CPU 아키텍처마다 다르며, 위의 표는 x86-64 Linux를 기준으로 함

자료형

자료형	종류	크기*	범위*	Suffix
long	정수	32bits (4bytes)	-2,147,483,648 ~ 2,147,483,647	l 또는 L
unsigned long	부호 없는 정수	32bits (4bytes)	0 ~ 2,147,483,648	ul 또는 UL
long long	정수	64bits (8bytes)	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	ll 또는 LL
unsigned long long	부호 없는 정수	64bits (8bytes)	0 ~ 18,446,744,073,709,551,615	ull 또는 ULL

*자료형의 크기는 CPU 아키텍처마다 다르며, 위의 표는 x86-64 Linux를 기준으로 함

자료형

자료형	종류	크기*	Suffix
float	실수	32bits (1byte)	f 또는 F
double	실수	64bits (8bytes)	
long double	실수	80bits (10bytes)	l 또는 L

*자료형의 크기는 CPU 아키텍처마다 다르며, 위의 표는 x86-64 Linux를 기준으로 함

자료형

- 일반적으로...
  - 문자는 char 형에 저장
  - 정수는 int형에 저장
  - 큰 정수는 long 형에 저장
  - 실수는 double 형에 저장

자료형

- 정수형 변수에서 나타낼 수 있는 범위를 벗어난 경우 오버플로우 발생
  - e.g. int형 변수에 3,000,000,000을 넣으면 -1,294,967,296이 됨

- 실수형 변수는 근사값을 저장*
  - 두 값을 직접 비교하는 것을 지양하는 게 좋음

- 실수형 변수는 크기가 클 수록 정확도가 높아짐
  - double 이상의 자료형을 사용하는 것을 권장

*IEEE 754 참조: https://en.wikipedia.org/wiki/IEEE_754

상수

- const 키워드로 시작, 나머지는 변수 선언과 동일
  - e.g. `const int a = 123;`

- 초기 값을 지정하지 않아도 컴파일은 됨*
  - `const int a;` → 컴파일은 되지만 의미 없음

- 읽기 전용, 나중에 값을 바꿀 수 없음
  - `const int a = 123;`
  - `a = 456;` → 컴파일 오류

*C++에서는 컴파일 오류 발생

#define 매크로

- 형식: #define [치환할 대상] [치환할 값]
- 전처리기가 치환할 대상을 치환할 값으로 모두 바꿈
- 코드 편집기에서 '모두 바꾸기' 기능을 사용하는 느낌
- 메모리 공간을 차지하지 않음
- e.g. `#define PI 3.14`

참고할 점

- 변수가 선언되는 곳에 따라 초기화 여부가 다름
  - 초기화 하지 않고 전역변수로 선언한 경우 0으로 초기화 됨
    - 이 경우 변수가 .bss 섹션에 저장되는데, .bss 섹션은 프로그램 시작시 0으로 초기화 됨
  - 지역변수로 선언한 경우 초기화 하지 않으면 쓰레기 값이 들어가있음
    - 이 경우 변수가 스택 영역에 저장되는데, 스택 영역은 이전에 쓰인 값이 지워지지 않음
  - 웬만해서는 변수를 초기화 하고 사용하는 것이 좋음
    - 변수를 초기화 하지 않아서 오작동 하는 경우가 생각보다 자주 일어남

연습

- 정수형 변수 a와 b를 선언하고, a를 123으로 초기화
- 실수형 변수 c를 선언하고, c를 4.56으로 초기화

연산

대입

```
int a, b = 1, c = 2;

a = b + c;
b = 123;
c = 456;
a = b - c;
```

연산자 종류

- 산술 연산자 (e.g. 덧셈, 뺄셈, 곱셈, 나눗셈, 나머지)
- 비교 연산자 (e.g. 같다, 다르다, 크다, 크거나 같다, 작다, 작거나 같다)
- 논리 연산자 (e.g. AND, OR, NOT)
- 비트 연산자 (e.g. AND, OR, NOT, XOR, Shift)

- 대입 연산자: 산술·비교·논리·비트 연산과 함께 값을 대입

산술 연산자

- a + b: 덧셈
- a – b: 뺄셈
- a * b: 곱셈
- a / b: 나눗셈
- a % b: 나머지

산술 연산자

- a++: 값을 반환하고 1만큼 증가 (후위)
- a--: 값을 반환하고 1만큼 감소 (후위)
- ++a: 1만큼 증가하고 값을 반환 (전위)
- --a: 1만큼 감소하고 값을 반환 (전위)

비교 연산자

- a < b: a가 b보다 작으면 true, 그렇지 않으면 false 반환
- a > b: a가 b보다 크면 true, 그렇지 않으면 false 반환
- a <= b: a가 b보다 작거나 같으면 true, 그렇지 않으면 false 반환
- a >= b: a가 b보다 크거나 같으면 true, 그렇지 않으면 false 반환
- a == b: a와 b가 동일하면 true, 그렇지 않으면 false 반환
- a != b: a와 b가 다르면 true, 그렇지 않으면 false 반환

논리 연산자

- a && b: a와 b가 모두 true이면 true 반환, 그렇지 않으면 false 반환
- a || b: a와 b 중 하나라도 true이면 true 반환, 그렇지 않으면 false 반환
- !a: a가 true면 false를, false면 true를 반환

비트 연산자

- a & b: a와 b에 bitwise AND 연산 수행
- a ¦ b: a와 b에 bitwise OR 연산 수행
- a ^ b: a와 b에 bitwise XOR 연산 수행
- ~a: a에 bitwise NOT 연산 수행
- a << b: a를 b만큼 왼쪽으로 쉬프트 수행
- a >> b: a를 b만큼 오른쪽으로 쉬프트 수행

대입 연산자

- =, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=

- a += b는 a = a + b와 같은 형태로 볼 수 있음
- 나머지도 대입 연산자도 마찬가지...

연산자 우선순위

우선순위	연산자	방향	우선순위	연산자	방향
1	++ (후위), -- (후위), (), [], ., ->	→	9	^	→
2	++ (전위), -- (전위), +(부호), -(부호), !, ~	←	10	|	→
3	*, /, %	→	11	&&	→
4	+, -	→	12	||	→
5	<<, >>	→	13	?:	←
6	<, <=, >, >=	→	14	대입 연산자	←
7	==, !=	→	15	,	→
8	&	→			

입력

일단 따라하기

```c
#include <stdio.h>

int main(void) {
    int a = 0;
    scanf("%d", &a);
    printf("a = %d\n", a);

    return 0;
}
```

일단 따라하기

```c
#include <stdio.h>

int main(void) {
    int a = 0; 변수 a를 선언하면서 0으로 초기화
    scanf("%d", &a); a의 위치에 입력받은 10진수 저장
    printf("a = %d\n", a); a의 값을 출력

    return 0;
}
```

scanf

- printf와 사용법이 상당히 유사함
- 형식 지정자도 printf와 거의 비슷함*
- 하지만 인자에 변수를 넘길 때 변수 이름에 &를 붙임
  - &를 붙이면 그 변수의 주소를 반환함
  - &를 붙이지 않으면 그 변수의 값을 반환함
  - scanf에 변수의 위치를 알려 준 다음, 그곳에 입력 받은 값을 넣도록 하는 것
  - 포인터에 관한 자세한 내용은 다음에 다룰 예정

*https://cplusplus.com/reference/cstdio/scanf/ 참조

연습

- scanf로 정수 입력받기
- scanf로 실수 입력받기
- scanf로 문자 입력받기

*https://cplusplus.com/reference/cstdio/scanf/ 참조

과제

새싹 – 입력과 계산

문제 번호	제목
1000	A+B
1001	A-B
10998	A×B
10869	사칙연산
1008	A/B
11382	꼬마 정민

‘https://noj.am/(문제 번호)’ 로 문제 페이지를 바로 열 수 있음

감사합니다