



C 언어 프로그래밍

Part 01. C 언어 준비 학습

Chapter 03. 변수와 타입

목차

1. 메모리와 데이터 타입
2. 변수란?
3. 정수형
4. 실수형
5. 문자형과 문자열

[실전예제] int와 float 사이의 오차 구하기

학습목표

- 데이터 타입의 필요성과 종류를 알아본다.
- 변수의 개념을 알고 정의해본다.
- 정수형의 종류를 살펴보고 오버플로우와 언더플로우의 개념을 이해한다.
- 실수형을 사용하는 경우를 살펴본다.
- 문자형과 문자열의 차이점을 이해할 수 있다.

01

메모리와 데이터 타입

01. 메모리와 데이터 타입

I. 메모리

- 비트(Bit, Binary Digit)

- 0 혹은 1을 나타낼 수 있는 모눈 한 개는 컴퓨터의 최소 저장 단위
- 1비트 : {0, 1} 두 가지 상태를 나타낼 수 있음
- 2비트 : {00, 01, 10, 11} 네 가지 상태를 나타낼 수 있음

1비트 {0, 1} → 2¹개

2비트 {00, 01, 10, 11} → 2²개

3비트 {000, 001, 010, 011, 100, 101, 110, 111} → 2³개

⋮

8비트 {00000000, ~, 11111111} → 2⁸개

↑
(= 1바이트)

그림 3-1 비트의 표현 가능한 상태 개수

01. 메모리와 데이터 타입

I. 메모리

• 바이트

- 2^{10} 인 1,024배마다 단위 표기를 다르게 함
- 컴퓨터는 바이트마다 주소를 부여

표 3-1 바이트 단위 표기

1KB(Kilo Byte)	1,024Byte
1MB(Mega Byte)	1,024KB
1GB(Giga Byte)	1,024MB
1TB(Tera Byte)	1,024GB
1PB(Peta Byte)	1,024TB

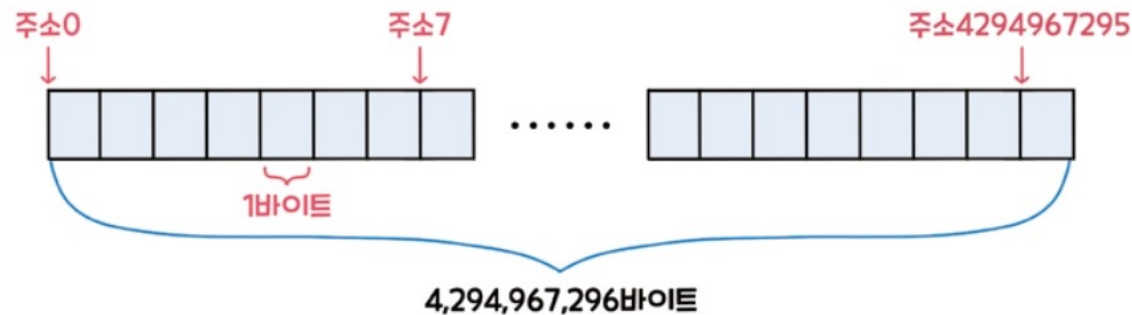


그림 3-2 메모리의 구성 및 주소

01. 메모리와 데이터 타입

II. 데이터 타입

- 타입의 개념

- 메모리 영역을 크기와 데이터의 종류에 따라서 분류한 것
- 메모리 영역의 상태를 타입에 따라 해석할 수 있음



그림 3-3 겉포장(라벨)만으로도 내용물을 쉽게 파악할 수 있다.

01. 메모리와 데이터 타입

III. 데이터의 분류

• 타입의 분류

- 수 : 정수형, 실수형으로 나눌 수 있음
- 문자, 문자열 : 아스키코드, 유니코드로 나눌 수 있음
- 분류 된 데이터는 해당 분류를 나타내는 타입의 메모리 영역에만 저장됨

하나 더 알기

수의 타입을 세분화하는 이유

- 한정된 크기의 메모리 영역으로 표현할 수 있는 데이터의 범위가 달라짐
- 따라서 수의 경우 실수형, 부호 있는 정수형, 부호 없는 정수형으로 세분화됨

01. 메모리와 데이터 타입

IV. 타입 개념의 확장

- 기본 타입
 - 숫자나 문자 하나를 저장할 수 있는 메모리 영역의 분류 체계
- 집합 타입
 - 기존의 타입들을 조합하여 하나의 타입으로 만든 것
 - C 언어의 경우 배열(Array) 타입, 구조체(Struct)를 집합 타입으로 제공

01. 메모리와 데이터 타입

확인문제1

1. 4바이트의 메모리는 몇 가지 상태를 나타낼 수 있는지 고르시오.

① 2^4 개

② 2^8 개

③ 2^{16} 개

④ 2^{32} 개

2. 다음 빈칸에 들어갈 단어를 채우시오.

타입은 메모리 영역의 와 해당 영역에 저장되는 의 종류에 따른 분류이다.

3. 집합 타입 관점에서 다음 제품 중 나머지 넷과 성격이 다른 것을 고르시오.

① 연필 한 다스

② 담배 한 갑

③ 학용품 세트

④ 마스크 50매

4. 연필 두 자루와 연필 세 자루는 서로 다른 타입이라고 가정할 때, 연필 3자루와 지우개 4개로 만들 수 있는 타입은 몇 종류인가?

02

변수란?

02. 변수란?

I. 변수의 개념

- 변수

- 이름이 붙어있어서 쉽게 값을 읽고 쓸 수 있는 메모리 영역

- 변수명

- 메모리 영역에 붙인 이름이자 식별자
- 어떤 데이터를 저장하는지 알 수 있도록 쉽게 작성하는 것이 좋음
- 여러 단어를 조합하여 적당한 길이로 만드는 것이 좋음



그림 3-4 장소를 나타내는 방법

02. 변수란?

II. 변수의 정의

- 이름이 붙은 메모리 영역을 마련하는 것

[코드 3-1] 변수의 정의

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int a;  ← 정수를 저장할 수 있는 변수 a
06      printf("a의 주소: %p", &a);
07  }
```

a의 주소: 00000039F52FF950

컴퓨터에 따라서 혹은 실행할 때마다 마련된 메모리 영역이 달라지면서 출력되는 주소는 달라집니다.

02. 변수란?

II. 변수의 정의

- 변수 초기화
 - 변수가 정의되고 생성되는 순간에 특정한 값을 넣음
- 변수 대입
 - 변수가 생성된 이후에 값을 넣음
- 변수의 초기화와 대입 형식

타입 변수명 = 초기값 ; (단, '= 초기값'은 생략 가능하다.)

02. 변수란?

II. 변수의 정의

[코드 3-2] 변수의 초기화와 대입

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int a = 1;
06      a = 2;
07  }
```

초기값
초기화
대입

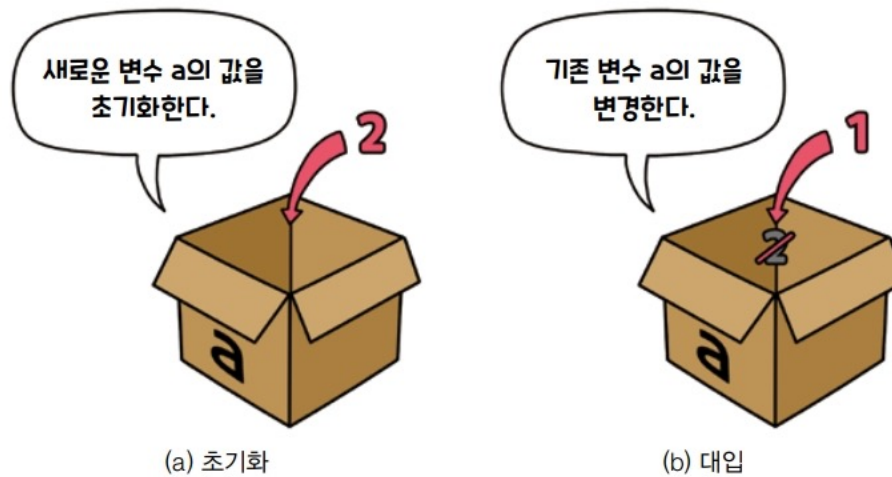


그림 3-5 초기화와 대입의 차이점

02. 변수란?

II. 변수의 정의

- 각종 타입 변수 정의

[코드 3-3] 각종 타입 변수의 정의

```
01  #include <stdio.h>
02
03  int main()
04  {
05      char c;      ← 1바이트 정수
06      short s;     ← 2바이트 정수
07      int i;       ← 4바이트 정수
08      long long ll; ← 8바이트 정수
09      float f;     ← 4바이트 실수
10      double d;    ← 8바이트 실수
11  }
```


02. 변수란?

II. 변수의 정의

- 같은 타입 변수 여러 개 정의 형식

타입 변수명1 = 초기값1, 변수명2 = 초기값2, ... ; (단, '=' 초기값'은 생략 가능하다.)

[코드 3-4] 같은 타입 변수 여러 개 정의

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int a1, a2;
06      int a3 = 1, a4;
07      int a5 = 1, a6 = 2;
08  }
```

} 변수 초기화

02. 변수란?

III. 타입 한정자 const

- 타입 한정자
 - 타입에 추가적인 속성을 지정
- **const**
 - 해당 타입의 변수는 정의된 이후 값을 대입할 수 없음



그림 3-6 초기화 이후 값을 변경할 수 없는 const 타입

02. 변수란?

III. 타입 한정자 const

[코드 3-5] const

```
01  #include <stdio.h>
02
03  int main()
04  {
05      const int ci;
06      ci = 1;
07  }
```

오류

02. 변수란?

III. 타입 한정자 const

[코드 3-6] const 초기화

```
01  #include <stdio.h>
02
03  int main()
04  {
05      const float PI = 3.141592;
06      const int Twelve = 12;
07
08      print f("PI: %f, Twelve: %d", PI, Twelve);
09  }
```

각 타입에 맞는 초기값으로 초기화

PI: 3.141592, Twelve: 12

02. 변수란?

III. 타입 한정자 const

- 같은 타입의 변수 여러 개 정의할 경우 변수들 모두 const 속성을 가짐

[코드 3-7] const 변수 여러 개 정의

```
01  #include <stdio.h>
02
03  int main()
04  {
05      const int c1, c2;
06      c1 = 1;
07      c2 = 2;
08  }
```

오류

02. 변수란?

IV. 타입 별칭 정의 typedef

- 타입 별칭 정의
 - 변수를 정의할 때 사용
- 타입 별칭 형식

```
typedef 타입 별칭;
```

- int의 별칭으로 MyInt 정의해보기

① int 타입 변수 MyInt를 초기값 없이 정의

```
int MyInt;
```

② 맨 앞에 typedef 붙여줌

```
typedef int MyInt;
```

02. 변수란?

IV. 타입 별칭 정의 typedef

- int의 별칭으로 MyInt 정의해보기

[코드 3-8] typedef

```
01  #include <stdio.h>
02
03  int main()
04  {
05      typedef int MyInt;
06
07      int a = 1;
08      MyInt b = 21
09      b = a;
10  }
```

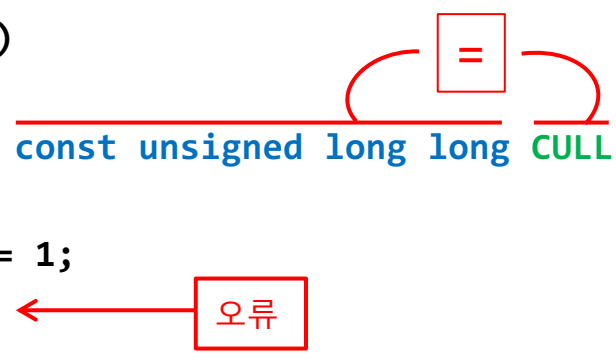
02. 변수란?

IV. 타입 별칭 정의 typedef

- typedef의 효용성

[코드 3-9] typedef

```
01  #include <stdio.h>
02
03  int main()
04  {
05      typedef const unsigned long long CULL;
06
07      CULL c = 1;
08      c = 2;
09  }
```



02. 변수란?

확인문제2

1. 다음 변수명 중에서 잘못된 것을 고르시오.

① Coke

② pepsi_

③ 7up

④ Wator

2. 다음 변수들의 정의가 잘못된 이유를 설명하고, 올바르게 고치시오.

```
int a1 = 1, a2 =2, float f1 = 1.1, f2 = 2.2;
```

02. 변수란?

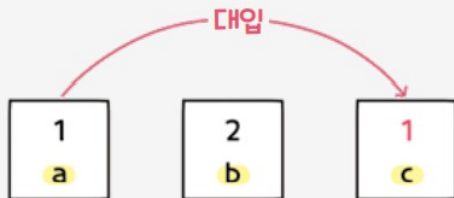
LAB 3-1

a와 b 값을 서로 바꾸기

각각 1과 2로 초기화된 두 정수 변수 a, b의 값을 서로 바꾸는 코드를 작성해봅시다.

a와 b를 교환하는 방법은 다음과 같다.

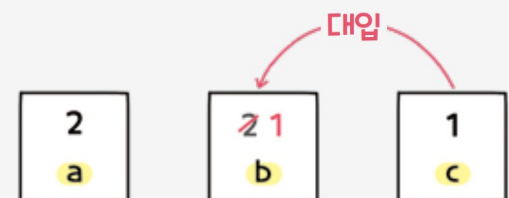
❶ a의 값을 c에 넣는다($c = a$);



❷ b의 값을 a에 넣는다($a = b$);



❸ c의 값을 b에 넣는다($b = c$);



02. 변수란?

LAB 3-1

정답

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int a = 1, b = 2;
06      int c = a;
07      a = b;
08      b = c;
09  }
```

← b = a는 안 됨

03

정수형

03. 정수형

I. 부호 있는 정수 타입

- 부호 있는 정수
 - 모든 정수(음의 정수, 0, 자연수)

표 3-2 부호 있는 정수 타입

타입	크기	표현 범위
char	1바이트	-128 ~ 127
short	2바이트	-32,768 ~ 32,767
int	4바이트	-2,147,483,648 ~ 2,147,483,647
long long	8바이트	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

03. 정수형

II. 부호 없는 정수 타입

- 부호 없는 정수
 - 음이 아닌 정수(0, 자연수)

표 3-3 부호 없는 정수 타입

타입	크기	표현 범위
unsigned char	1바이트	0 ~ 255
unsigned short	2바이트	0 ~ 65,535
unsigned int	4바이트	0 ~ 4,294,967,295
unsigned long long	8바이트	0 ~ 18,446,744,073,709,551,615

하나 더 알기 (unsigned) long 타입

- 어떤 시스템에서 long을 사용한 코드가 다른 시스템에서는 타입 크기가 달라서 오동작을 일으킬 수 있음
- 따라서 long은 없는 타입으로 여기고 절대 쓰지 말기를 권장

03. 정수형

III. 정수 타입 변수의 초기화와 대입

[코드 3-10] int 변수 정의

```
01 #pragma warning (disable:4700)
02 #include <stdio.h>
03
04 int main()
05 {
06     int i1 = 3;
07     int i2;
08
09     print("i1: %d\r\ni2: %d", i1, i2);
10 }
```

오류를 무시하고 강제로 실행

```
i1: 3
i2: -1781858304
```

03. 정수형

III. 정수 타입 변수의 초기화와 대입

[코드 3-11] 변수 대입

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i;
06      unsigned int ui;
07
08      i = -999;
09      ui = 777;
10
11      print("i: %d\r\nui: %d", i, ui);
12  }
```

i: -999
ui: 777

03. 정수형

III. 정수 타입 변수의 초기화와 대입

[코드 3-12] 정수가 아닌 실수의 반환

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int i1 = 3.141592;
06      int i2 = -3.141592;
07
08      print("i1: %d\r\ni2: %d", i1, i2);
09  }
```

i1: 3

i2: -3

03. 정수형

IV. 오버플로우와 언더플로우

- 오버플로우
 - 표현 범위의 최댓값을 넘어선 경우
- 언더플로우
 - 표현 범위의 최솟값보다 작아진 경우

03. 정수형

IV. 오버플로우와 언더플로우

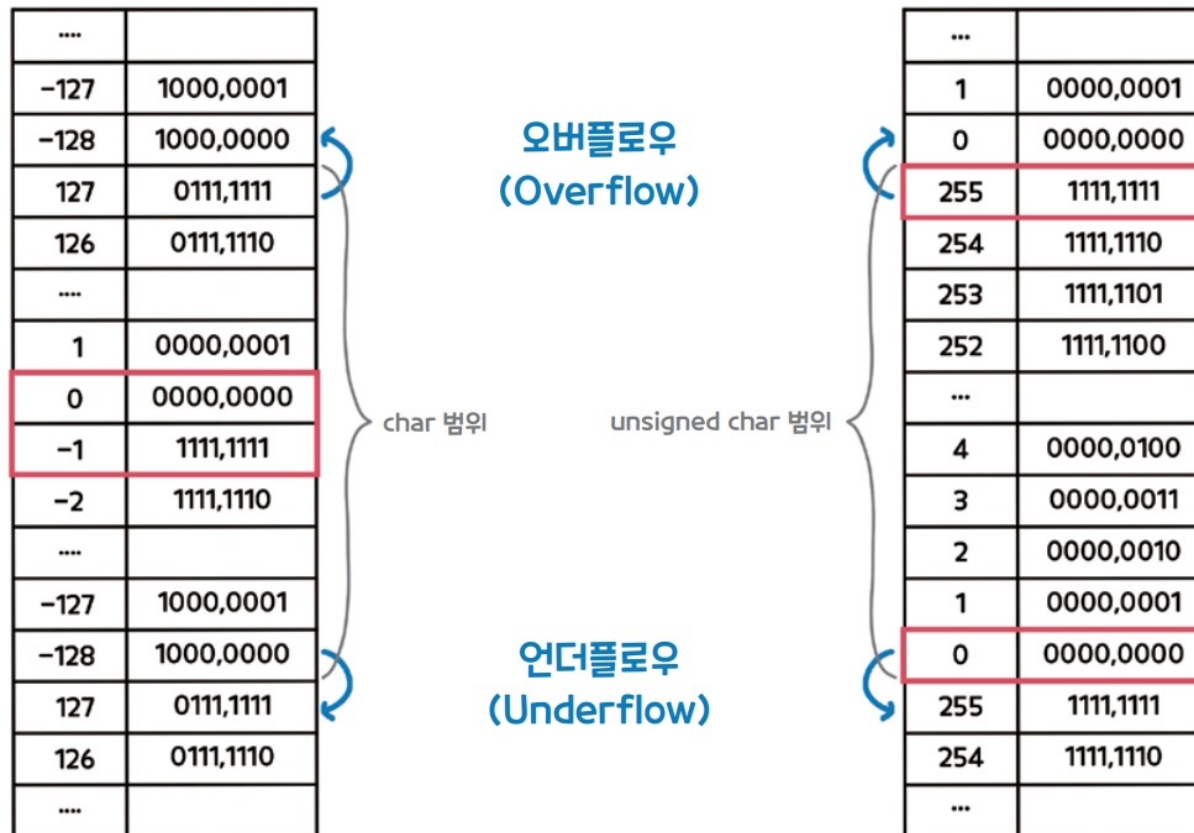


그림 3-7 char와 unsigned char의 오버플로우와 언더플로우

03. 정수형

IV. 오버플로우와 언더플로우

[코드 3-13] 범위 밖의 정수의 변환

```
01  #include <stdio.h>
02
03  int main()
04  {
05      char c1 = 128;
06      char c2 = -129;
07      printf("c1: %d\r\nc2: %d\r\n", c1, c2);
08
09      unsigned char uc1 = 256;
10      unsigned char uc2 = -1;
11      print("uc1: %u\r\unc2: %d", uc1, uc2);
12  }
```

최댓값 +1

최솟값 -1

최댓값 +1

c1: -128
c2: 127
uc1: 0
unc2: 255

03. 정수형

IV. 오버플로우와 언더플로우

[코드 3-14] 범위 밖의 정수의 변환

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int hits1 = 2147483648;
06      unsigned int hits2 = 4294967296;
07
08      print("hits1=%d, hits2=%d", hits1, hits2);
09  }
```

최댓값 +1

최댓값 +1

hits1=-2147483648, hits2=0

03. 정수형

확인문제3

1. 메모리 크기의 관점에서 사람의 태어난 연도를 저장하는 변수로 가장 적당한 타입을 고르시오.

① char

② short

③ int

④ long long

2. unsigned int는 사용하기에 타입명이 길다. unsigned int의 별칭을 UINT로 정의하시오.

3. 다음 코드에서 변수 a에 들어갈 값으로 옳은 것을 고르시오.

```
int a = 1.7 + 0.4;
```

① 1

② 2

③ 2.1

④ 3

03. 정수형

LAB 3-2

부호 없는 정수형 타입의 최댓값 출력하기

부호 없는 정수형 타입인 `unsigned char`, `unsigned short`, `unsigned int`의 최댓값을 출력하는 프로그램을 작성해봅시다.

```
uc: 255
us: 65535
ui: 4294967295
```

- 1 모든 비트가 1일 때 부호 없는 정수형은 최댓값을 나타내지만 부호 있는 정수형의 경우 `-1`을 나타낸다. 따라서 각각의 부호 없는 정수형 변수를 `-1`로 초기화할 경우 모든 비트가 1로 설정되면서 각각의 최댓값을 가지게 된다.
- 2 부호 없는 정수형을 출력할 경우 형식지정자 `%u`를 사용한다.

03. 정수형

LAB 3-2

정답

```
01  #include <stdio.h>
02
03  int main()
04  {
05      unsigned char uc = -1;
06      unsigned short uc = -1;
07      unsigned int ui = -1;
08
09      printf("uc: %u\r\nus: %u\r\nui: %u", uc, us, ui);
10  }
```


04

실수형

04. 실수형

I. 실수형의 분류

- 실수형
 - 모든 수를 대표함
 - 부호 있는 실수형 타입만 제공함

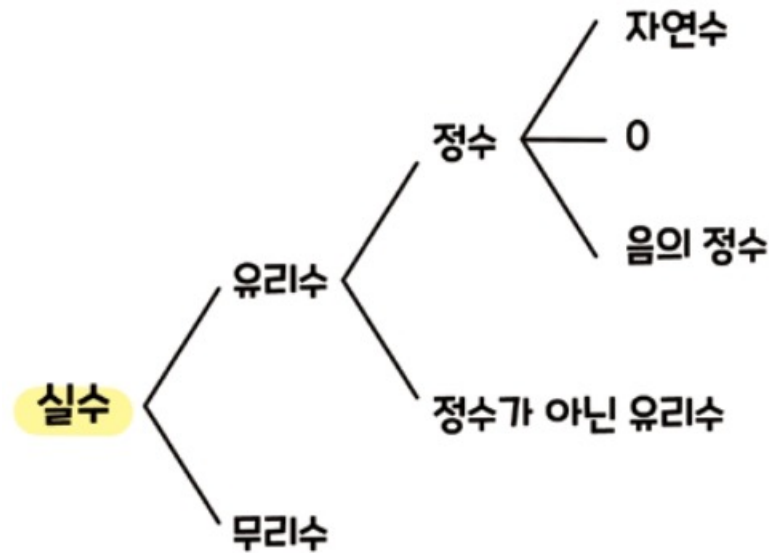


그림 3-8 실수의 범위

04. 실수형

II. 실수형 타입의 오차

- 실수형 타입
 - 부호 있는 실수형 타입만 제공
- float과 double
 - float은 무한개의 수를 대응해서 나타낼 수는 없음
 - 따라서 표현할 수 있는 가장 근접한 수로 바꾸는 과정을 거쳐 리터럴 값과 실제 저장된 값 사이에 오차가 발생함
 - 이러한 오차를 줄이기 위해 double을 사용하기도 함

표 3-4 실수형 타입

타입	크기	표현 범위
float	4바이트	$-3.4 \times 10^{38} \sim 3.4 \times 10^{38}$
double	8바이트	$-1.8 \times 10^{308} \sim 1.8 \times 10^{308}$

04. 실수형

II. 실수형 타입의 오차

[코드 3-15] float의 오차

```
01  #include <stdio.h>
02
03  int main()
04  {
05      int    i = 33554434;
06      float  f = 33554434;
07
08      print("i:%d\r\nf:%f", i, f);
09  }
```

정수 초기화 및
대입도 가능

} 같은 값으로 초기화

i: 33554434
f: 33554432.000000

04. 실수형

LAB 3-3

원의 둘레와 넓이 구하기

다음 실행 결과를 참고하여 반지름을 입력받아서 원의 둘레와 넓이를 출력하는 프로그램을 작성해 봅시다.

반지름을 입력하세요.

7

둘레: 43.982288, 넓이: 153.938008

- 1 정밀도를 최대한 높이고 오차를 줄이기 위해서 모든 변수의 타입은 `double`을 사용한다.
- 2 원주율은 `const double PI` 변수를 마련하여 3.141592로 초기화한다.
- 3 `scanf`를 통해서 `double` 변수 `r`에 입력을 받는다. 이때 `double` 변수에 값을 입력받기 위하여 `scanf`의 서식 문자열은 형식지정자 `%lf`를 사용해야 한다.
- 4 원 둘레와 넓이를 구한 뒤 출력한다. `printf`에서는 형식지정자 `%f`와 `%lf`가 동일하지만 `scanf`에서는 변수의 타입 `float`과 `double`을 나타낸다.

04. 실수형

LAB 3-3

정답

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main()
05  {
06      const double PI = 3.141592;
07      double r;
08      printf("반지름을 입력하세요.\r\n");
09
10      scanf("%lf", &r);
11      double len = 2 * PI * r;
12      double area = PI * r * r;
13
14      printf("둘레: %f, 넓이: %f", len, area);
15  }
```

05

문자형과 문자열

05. 문자형과 문자열

I. 기본 문자 타입 char

- char
 - 1바이트 크기의 부호 있는 정수 타입
 - 문자에 대응되는 정수형 코드가 저장됨
- 아스키코드(ASCII Code)
 - 문자와 정수형 코드의 대응 관계

05. 문자형과 문자열

I. 기본 문자 타입 char

표 3-5 아스키코드 표

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
032	20	SP	056	38	8	080	50	P	104	68	h
033	21	!	057	39	9	081	51	Q	105	69	i
034	22	"	058	3A	:	082	52	R	106	6A	j
035	23	#	059	3B	;	083	53	S	107	6B	k
036	24	\$	060	3C	<	084	54	T	108	6C	l
037	25	%	061	3D	=	085	55	U	109	6D	m
038	26	&	062	3E	>	086	56	V	110	6E	n
039	27	'	063	3F	?	087	57	W	111	6F	o
040	28	(064	40	@	088	58	X	112	70	p
041	29)	065	41	A	089	59	Y	113	71	q
042	2A	*	066	42	B	090	5A	Z	114	72	r
043	2B	+	067	43	C	091	5B	[115	73	s

05. 문자형과 문자열

I. 기본 문자 타입 char

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
044	2C	,	068	44	D	092	5C	\	116	74	t
045	2D	-	069	45	E	093	5D]	117	75	u
046	2E	.	070	46	F	094	5E	^	118	76	v
047	2F	/	071	47	G	095	5F	_	119	77	w
048	30	0	072	48	H	096	60	`	120	78	x
049	31	1	073	49	I	097	61	a	121	79	y
050	32	2	074	4A	I	098	62	b	122	7A	z
051	33	3	075	4B	K	099	63	c	123	7B	{
052	34	4	076	4C	L	100	64	d	124	7C	}
053	35	5	075	4D	M	101	65	e	125	7D	}
054	36	6	078	4E	N	102	66	f	126	7E	~
055	37	7	079	4F	O	103	67	g	127	7F	DEL

05. 문자형과 문자열

I. 기본 문자 타입 char

[코드 3-16] 문자 타입

```
01  #include <stdio.h>
02
03  int main()
04  {
05      char a = 65;
06      printf("%c = %d\r\n", a, a);
07
08      char b = 'B';
09      printf("%c = %d", b, b);
10  }
```

A = 65

B = 66

05. 문자형과 문자열

I. 기본 문자 타입 char

[코드 3-17] 문자열과의 차이

```
01  #include <stdio.h>
02
03  int main()
04  {
05      char a = "A";
06      char b = '한';
07      printf("%c %c", a, b);
08  }
```

문자가 아닌 문자열

< ?

05. 문자형과 문자열

II. 문자열 타입

- 문자열 타입

- 문자열은 말 그대로 문자들이 나열된 것을 의미
- 몇 글자가 될지 정할 수 없기 때문에 엄밀히 말해서 문자열 타입이란 존재할 수 없음
- 문자 배열을 통해 문자열의 저장과 표현이 가능

확인문제4

알파벳 A를 첫 번째 문자라고 할 경우 아홉 번째 문자를 `char` 변수로 나타내시오.

05. 문자형과 문자열

LAB 3-4

대문자를 소문자로 변경하기

[표 3-5]의 아스키코드 표를 이용하여 다음 출력처럼 알파벳 대문자를 입력받아서 소문자로 출력하는 프로그램을 작성해봅시다.

알파벳 대문자를 입력하세요.

R

소문자: r

- 1 대문자 코드에 32를 더하면 소문자 코드이다.
- 2 대문자를 입력받을 변수 `upper`와 소문자를 나타낼 변수 `lower`를 정의한다.
- 3 형식지정자 `%c`를 이용하여 `upper`에 입력 대문자를 저장한다.
- 4 `upper - 'A'`는 A를 기준으로 몇 번째 대문자인지를 계산하므로 그 결과(순번)에 'a'를 더하면 a를 기준으로 해당 순번에 해당하는 소문자를 구할 수 있다.

05. 문자형과 문자열

LAB 3-4

정답

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main()
05  {
06      char upper, lower;
07
08      printf("반지름을 입력하세요.\r\n");
09
10      scanf("%c", &upper);
11      lower = 'a' + upper - 'A';
12      printf("소문자: %c", lower);
13  }
```

[실전예제]

int와 float 사이의 오차 구하기

[실전예제] int와 float 사이의 오차 구하기

[문제]

충분히 큰 정수를 입력받은 후, 해당 정수가 int형으로 저장될 때와 float형으로 저장될 때 그 사이에서 발생하는 오차를 구해봅시다.

실행 결과

정수를 입력하세요.

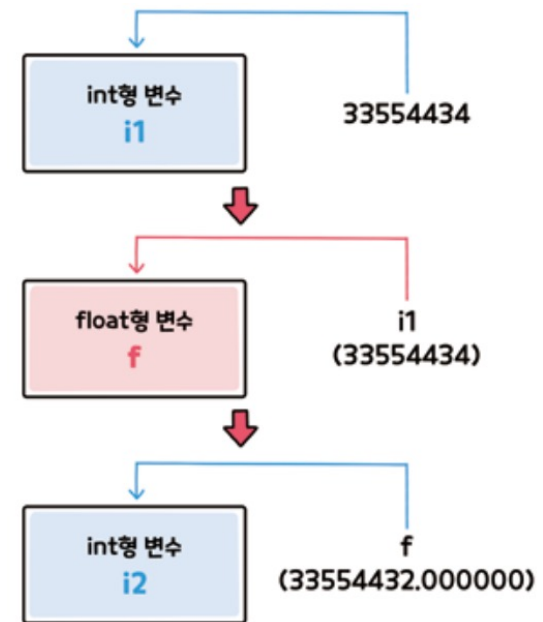
33554434

오차: 2

[실전예제] int와 float 사이의 오차 구하기

[해결]

1. 입력받은 정수를 int 변수 i1에 저장한다.
2. float 변수 f에 i1을 대입한다.
3. int 변수 i2에 f를 대입한다.
4. i1과 i2의 차를 출력한다.



[실전예제] int와 float 사이의 오차 구하기

[해결]

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main()
05  {
06      int i1, i2;
07      float f;
08      printf("정수를 입력하세요.\r\n");
09
10      scanf("%d", &i1);
11      f = i1;
12      i2 = f;
13
14      printf("오차: %d", i1 - i2);
15  }
```

Thank you!