상수, 변수, 자료형

Jo, Heeseung

이 장의 내용

자료형이란 무엇인가?

상수와 변수

정수형

부동소수형

문자형

자료형이란 무엇인가?

자료형(data type)

- 프로그램에서 표현 혹은 저장하는 데이터의 종류 혹은 유형

C 언어의 자료형

- C 언어는 1개의 문자형, 4개의 정수형, 3개의 부동소수형을 제공

자료형	크기에 따라			
문자형	char			
정수형	short	int	long	long long
부동소수형	float	double	long double	

표 3.1 C 언어의 자료형 분류

변수

변수(variable)

- 데이터를 저장하는 데 사용되는 기억 장소의 이름

사용 전 변수 선언

- 변수 이름과 저장할 데이터 값의 유형(자료형)을 지정

```
자료형 변수이름;
char c;
short sum;
```

메모리와 변수

변수를 위한 메모리 할당

- 문자형 변수 c를 위해서는 1 바이트를 할당
- short 변수 sum에 대해서는 2 바이트를 할당

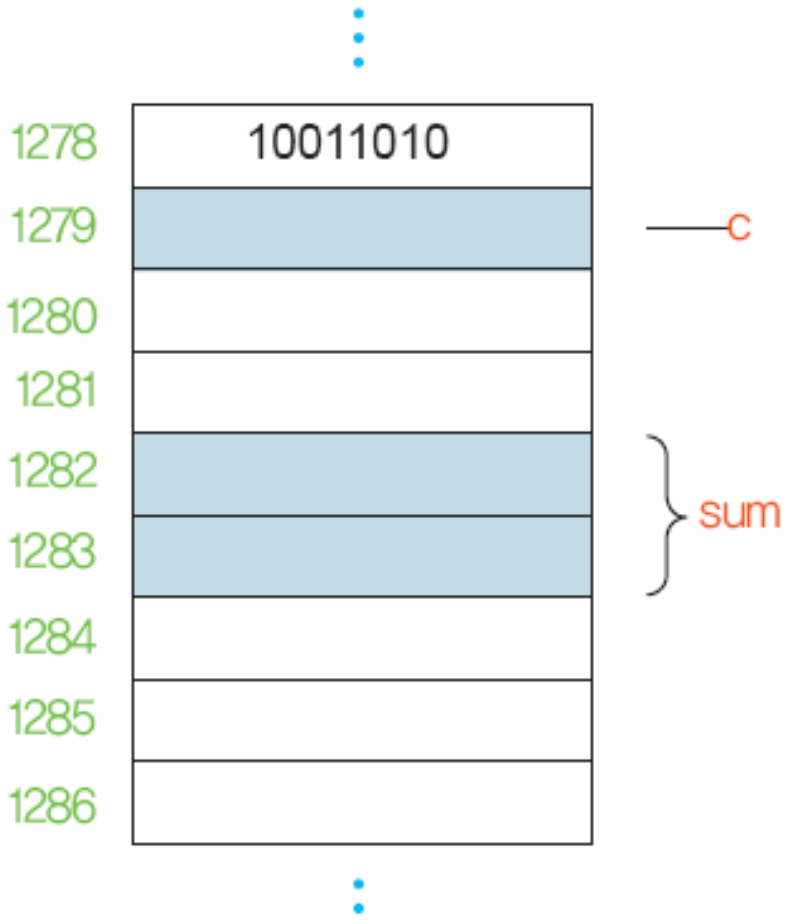

그림 3.1 메모리와 변수

상수

상수(constant)

- 프로그램이 실행되는 동안 값이 변하지 않는 데이터
- 정수형 상수, 문자형 상수, 부동소수형 상수로 구분
- 예
  - 'A', 'x'는 문자형 상수
  - -10, 0, 55는 정수형 상수
  - 3.14, 300.25는 부동소수형 상수

식별자

- 변수, 상수, 함수 등에 대한 이름
- 문자와 숫자로 구성되며 첫 번째 글자는 반드시 문자
- 밑줄문자 '_' 도 하나의 문자로 사용될 수 있음
- 대소문자를 구별

바른 식별자	잘못된 식별자
x25	25x
X	X#
age01	01age
sum	sum*
_jumsu	-jumsu
address_01	address-01

표 3.2 바른 식별자, 잘못된 식별자

변수의 초기화

변수를 선언하면서 그 초기값을 정할 수도 있음

- 변수 초기화는 반드시 해야 하는 것은 아니고 선택사항

```
int sum = 0;
```

그림 3.2 변수의 초기화

변수의 초기화

프로그램 3.1 sum.c

```
//*************************************************************************
//   sum.c
//
//   정수 변수의 선언과 초기화 그리고 사용을 보여준다.
//*************************************************************************

int main()
{
     int sum = 0;

     printf("합이 %d 입니다. \n", sum);
     sum = 10;
     printf("합이 %d 입니다. \n", sum);

     return 0;
}
```

실행 결과

```
합이 0 입니다.
합이 10 입니다.
```

대입문을 이용한 변수 값 변경

대입문(assignment statement)

- 변수에 새로운 값을 저장함으로써 기존 값을 변경
- 대입 연산자(=)의 오른쪽 식이 계산되고, 그 결과 값이 왼쪽 변수에 저장

```
sum = 10;
```

그림 3.3 변수 값 변경

프로그램 3.2 score.c

```c
//*********************************************************************
//  score.c
//
//  대입문을 사용한 변수 값 변경을 보여준다.
//*********************************************************************

#include <stdio.h>
//---------------------------------------------------------------------
// 몇 번의 점수를 출력한다.
//---------------------------------------------------------------------
int main()
{
    int score = 7; // 초기화

    printf("첫 번째 점수는 %d \n", score);

    score = 10;     // 대입문
    printf("두 번째 점수는 %d \n", score);

    score = 8;      // 대입문
    printf("세 번째 점수는 %d \n", score);
    score = 9;      // 대입문
    printf("네 번째 점수는 %d \n", score);

    return 0;
}
```

실행 결과

```
첫 번째 점수는 7
두 번째 점수는 10
세 번째 점수는 8
네 번째 점수는 9
```

변수 값 변경 과정

그림 3.4 변수 값 변경 과정

변수 값 증가

대입문 오른쪽과 왼쪽에 나타난 변수는 다른 의미

- 대입문 오른쪽 변수는 그 변수로부터 읽어온 값을 의미
- 대입문 왼쪽 변수는 그 변수에 저장(쓰기) 의미

```
sum = sum + 1;
```

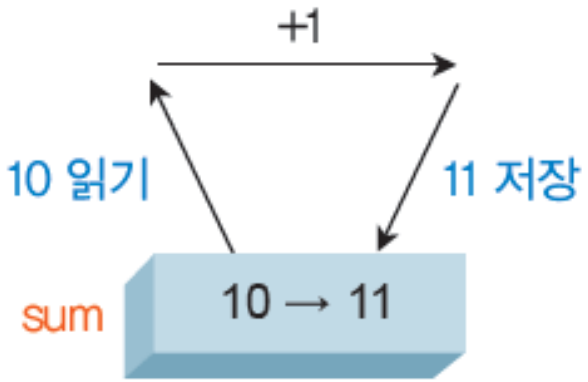

그림 3.5 변수에 1 더하기

- 변수에 접근(읽기)은 그 값을 변경시키지 않음
- 변수에 대입(쓰기)은 이전 데이터를 새로운 데이터로 대체

프로그램 3.3 scoreSum.c

```c
//*********************************************
//  scoreSum.c
//
//  대입문을 사용하여 값의 합을 저장한다.
//*********************************************

#include <stdio.h>
//---------------------------------------------
// 몇 번의 점수를 합하여 출력한다.
//---------------------------------------------------------------
int main()
{
    int sum = 0, score;
    score = 7;
    sum = sum + score; // score 값 더하기
    printf("첫 번째 점수 : %d 현재 합 : %d \n", score, sum);

    score = 10;
    sum = sum + score; // score 값 더하기
    printf("두 번째 점수 : %d 현재 합 : %d \n", score, sum);

    score = 8;
    sum = sum + score; // score 값 더하기
    printf("세 번째 점수 : %d 현재 합 : %d \n", score, sum);

    score = 9;
    sum = sum + score; // score 값 더하기
    printf("네 번째 점수 : %d 현재 합 : %d \n", score, sum);

    return 0;
}
```

실행 결과

```
첫 번째 점수: 7  현재 합: 7
두 번째 점수: 10  현재 합: 17
세 번째 점수: 8  현재 합: 25
네 번째 점수: 9  현재 합: 34
```

이름 상수

이름 상수(named constant)

- 상수에 이름이 부여되어 있는 것
- 이름상수에는 식별자가 부여되어 있으므로 이름상수는 변수와 유사하지만 상수이므로 값을 변경할 수 없음
- 예약어 const

```
const int MAX_STUDENT = 20000;
```

정수형

크게 4개의 정수형

- char, short, int, long

자료형	기억 장소	최소값	최대값
char	8비트	$-2^7 = -128$	$2^7 - 1 = 127$
short	16비트	$-2^{15} = -32,76$	$2^{15} - 1 = 32,767$
int	32비트	$-2^{31} = -2,147,483,648$	$2^{31} - 1 = 2,147,483,647$
long	32비트	$-2^{31} = -2,147,483,648$	$2^{31} - 1 = 2,147,483,647$
long long	64비트	$-2^{63} = -9,223,372,036,854,775,808$	$2^{63} - 1 = 9,223,372,036,854,775,807$

표 3.3 C의 기본 정수형의 범위 예(Microsoft Visual C/C++ 2010)

정수 표현

8 비트를 이용한 양수와 음수 표현 예

- 첫 번째 비트는 부호 비트로서, 0은 양수 1은 음수를 나타냄
- 나머지 7개의 비트들은 수의 크기를 나타냄

음수 표현

- 연산의 효율성을 위해 수의 크기를 2의 보수 형태로 나타냄

수의 표현	이진수
+5	00000101
5의 1의 보수	11111010
5의 2의 보수	11111011
–5	11111011

표 3.4 +5, –5의 표현

양수, 음수 표현

양수	이진수	음수	이진수
+127	01111111	−128	10000000
+126	01111110	−127	10000001
...	...	...	...
+5	00000101	−6	11111010
+4	00000100	−5	11111011
+3	00000011	−4	11111100
+2	00000010	−3	11111101
+1	00000001	−2	11111110
0	00000000	−1	11111111

표 3.5 양수, 음수 표현

변수에 기억공간 할당

기억공간 할당

- 정수형 종류에 따라 해당 변수에 다른 크기로 할당

sizeof() 연산자

- 자료형의 크기를 나타내는 연산자

프로그램 3.4 int.c

```
//*************************************************************
//  int.c
//
//  정수형들의 크기를 프린트한다.
//*************************************************************

#include <stdio.h>
int main()
{
    printf("char 크기:        %d\n", sizeof(char));
    printf("short 크기:       %d\n", sizeof(short));
    printf("int 크기:         %d\n", sizeof(int));
    printf("long 크기:        %d\n", sizeof(long));
    printf("long long 크기:   %d\n", sizeof(long long));

    return 0;
}
```

실행 결과

```
char 크기:        1
short 크기:       2
int 크기:         4
long 크기:        4
long long 크기:   8
```

프로그램 3.5 korea.c

```c
//*****************************************************************
//  korea.c
//
//  대한민국에 대한 데이터를 프린트한다.
//*****************************************************************

#include <stdio.h>
int main()
{
    short no_univ = 276;                // 대학 수
    int population = 48295000;          // 인구
    long budget = 2370000000000000L;    // 년 예산

    printf("대한민국에 대한 데이터입니다.\n");
    printf("대학 수: %d \n", no_univ);
    printf("인구:    %d 명\n", population);
    printf("예산:    %d 원\n", budget);

    return 0;
}
```

실행 결과

```
대한민국에 대한 데이터입니다
대학수: 276
인구:    48295000
예산:    -590360576 원
```

정수형 오버플로우

정수형 오버플로우

- 정수형에서 저장할 수 있는 수보다 더 큰 수나 더 작은 수를 저장

예) 32비트 int

- 가장 큰 수($2^{31}-1$ = 2,147,483,647)에 1을 더하면 어떻게 될까?

```
  01111111 11111111 11111111 11111111
+ 00000000 00000000 00000000 00000001
-----------------------------------
  10000000 00000000 00000000 00000000
```

질문

```
long budget = 23700000000000000L;
```

unsigned 정수형

unsigned 정수형

- 0과 양의 정수만을 나타낼 수 있는 정수형
- 음수를 표현할 수 없는 대신에 나타낼 수 있는 양의 정수가 두 배

자료형	기억 장소	최소값	최대값
unsigned char	8비트	0	$2^8-1=$ 255
unsigned short	16비트	0	$2^{16}-1=$ 65,535
unsigned int	32비트	0	$2^{32}-1=$ 4,294,967,295
unsigned long	32비트	0	$2^{32}-1=$ 4,294,967,295

표 3.6 C의 unsigned 정수형의 표현 범위 예

부동소수형

C 언어의 3개의 부동소수형

- float, double, long double

부동소수점(floating-point)이란?

- 부동소수점에서 점(point)은 소수점을 말하며, 이 소수점이 수의 어느 위치에도 올 수 있으므로 부동소수점이라고 함

자료형	기억 장소	최소값	최대값
float	4바이트 (32비트)	7개의 유효숫자를 가지며, −1.0E+38의 근사값	7개의 유효숫자를 가지며, 1.0E+38의 근사값
double	8바이트 (64비트)	15개의 유효숫자를 가지며, −1.0E+308의 근사값	15개의 유효숫자를 가지며, 1.0E+308의 근사값
long double	8바이트 (64비트)	15개의 유효숫자를 가지며, −1.0E+308의 근사값	15개의 유효숫자를 가지며, 1.0E+308의 근사값

표 3.7 부동소수형의 예(Microsoft Visual C++ 2010)

부동소수형 표현

float 형 자료의 저장과 표현 범위

- 아래 double 형과 유사하게 부호, 지수부, 가수부로 나뉘어짐
- 전체 비트 수가 double보다 작을 수 있음(예: 16비트)

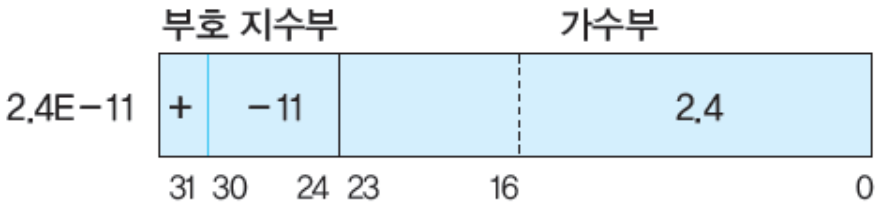

그림 3.6 float 형 자료의 저장 형태

double 형 자료의 저장과 표현 범위

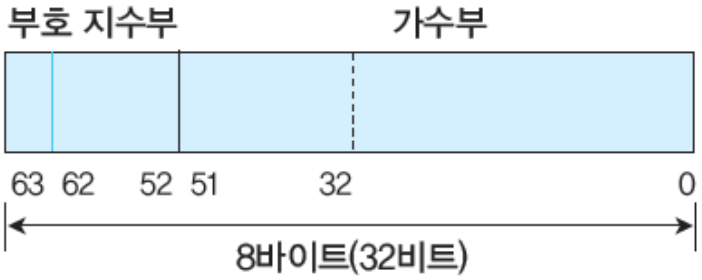

그림 3.7 double 형 자료의 저장과 표현 범위

프로그램 3.6 float.c

```c
//*****************************************************************
//  float.c
//
//  부동소수형들의 유효숫자와 크기를 프린트한다.
//*****************************************************************

#include <stdio.h>

int main()
{
    float x = 0.12345678901234567890F;
    double y = 0.12345678901234567890;
    long double z = 0.12345678901234567890L;
    printf("%20.19f\n", x);
    printf("%20.19f\n", y);
    printf("%20.19f\n", z);

    printf("float 크기:       %d\n", sizeof(float));
    printf("double 크기:      %d\n", sizeof(double));
    printf("long double 크기: %d\n", sizeof(long double));

    return 0;
}
```

실행 결과

```
0.1234567910432815600
0.1234567890123456800
0.1234567890123456800
float 크기:       4
double 크기:      8
long double 크기: 8
```

문자형

각 문자에 고유번호를 부여한 코드를 사용하여 표현

C에서는 ASCII 코드를 사용

- ASCII(American Standard Code for Information Interchange)

128개의 문자를 7 비트를 사용하여 표현(8비트로 확장됨)

- 대문자(A, B, C 등등)
- 소문자(a, b, c 등등)
- 구두점(punctuation)(마침표, 세미콜론, 쉼표 등등)
- 숫자(digit)(0에서 9까지)
- 공백 문자(' ')
- 특수 문자(&, ¦, \ 등)
- 제어 문자
  - 열복귀(carriage return), 널(null), 문서-끝-표시자(end-of-text)
- 액센트(accent)가 있는 문자

ASCII 코드표

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	、	p
1	SOH	DC1XON	!	1	A	Q	a	q
2	STX	DC2	“	2	B	R	b	r
3	ETX	DC3XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	‘	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	〈	L	₩	l	|
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	〉	N	^	n	~
F	SI	US	/	?	O	_	o	del

표 3.8 ASCII 코드표

프로그램 3.7 code.c

```
//*************************************************************************
//  code.c
//
//  문자의 코드 값을 프린트한다.
//*************************************************************************

int main()
{
    char c;
    int i;

    c = 'a';
    printf("%c %d \n", c, c);
    c = 'A';
    printf("%c %d \n", c, c);
    c = '1';
    printf("%c %d \n", c, c);
    c = '$';
    printf("%c %d \n", c, c);
    c = '+';
    printf("%c %d \n", c, c);
    i = 'a';
    printf("%c %d \n", i, i);
    i = 'A';
```

```
    printf("%c %d \n", i, i);
    i = '1';
    printf("%c %d \n", i, i);
    i = '$';
    printf("%c %d \n", i, i);
    i = '+';
    printf("%c %d \n", i, i);

    return 0;
}
```

실행 결과

```
a 97
A 65
1 49
$ 36
+ 43
a 97
A 65
1 49
$ 36
+ 43
```

문자형 관련 입출력 함수

문자형 관련 입출력 함수

- getchar()
- putchar()

int getchar()

- 매개변수가 없으며 한 문자를 읽어 그 문자의 ASCII 값을 반환
- 파일 끝에 도달하면 EOF를 반환
- <stdio.h> 파일을 #include 해야 사용 가능

int putchar(int c)

- 한 글자를 받아 화면에 출력
- 출력한 문자의 ASCII 값을 반환
- 오류 시에는 EOF 를 반환
- <stdio.h> 파일을 #include 해야 사용 가능

다음 예제 프로그램 char.c

- ① 키보드로부터 한 개 문자 읽기
- ② 문자로 출력
- ③ ASCII 값을 보기 위해 정수로 출력

프로그램 3.8 char.c

```c
//*************************************************************************
//  char.c
//
//  문자를 읽어서 문자와 ASCII 값을 프린트한다.
//*************************************************************************

#include <stdio.h>
int main()
{
    int c;

    printf("한 문자를 입력하시오 : ");
    c = getchar();                              // 단계 ①

    printf("읽은 문자 :           %c \n", c);  // 단계 ②
    printf("읽은 문자의 ASCII 값 : %d \n", c);  // 단계 ③

    return 0;
}
```

실행 결과

```
한 문자를 입력하시오 : A
읽은 문자 :           A
읽은 문자의 ASCII 값 : 65
```

프로그램 3.9 putchar.c

```
//*************************************************************************
//  putchar.c
//
//  문자들을 프린트한다.
//*************************************************************************

#include <stdio.h>
int main()
{
    putchar('H');
    putchar('e');
    putchar('l');
    putchar('l');
    putchar('o');
    putchar(33);          // ASCII 값 33에 해당하는 문자를 출력
    putchar('\n');        // 화면에 새 줄을 삽입하여 줄을 바꾼다.

    return 0;
}
```

실행 결과

```
Hello!
```

이스케이프 시퀀스

이스케이프 시퀀스(escape sequence)

- 백슬래쉬 문자(\)로 시작하고, 다음 문자는 특별한 방식으로 해석
- 예를 들어 이중 인용부호 문자(") 출력: \"

이스케이프 시퀀스	의미
\b	백스페이스
\t	탭
\n	줄바꿈
\r	열복귀
\"	이중 인용부호
\'	단일 인용부호
\\	백슬래시

표 3.9 이스케이프 시퀀스

프로그램 3.10 escape.c

```
//*************************************************************************
//  escape.c
//
//  이스케이프 시퀀스들을 프린트한다.
//*************************************************************************

#include <stdio.h>
int main()
{
    putchar('a');
    putchar('\b');
    putchar('\t');
    putchar('b');
    putchar('\n');
    putchar('c');
    putchar('\r');
    putchar('d');
    putchar('\"');
    putchar('\'');
    putchar('\\');

    return 0;
}
```

실행 결과

```
a       b
d"'\
```

프로그래밍 실습 1

1. ASCII 코드표의 일부를 프린트하는 프로그램을 작성
   - 다음 선언을 가정하여 프로그램을 작성

```
int main( )
{
    char c;

}
```

(1) 각 알파벳 대문자와 그 코드 값을 프린트
   - 변수 c에 문자 'A'를 대입한 후에 1씩 증가시키면서 26개를 차례로 프린트

(2) 각 알파벳 소문자와 그 코드 값을 프린트
   - 변수 c에 문자 'a'를 대입한 후에 1씩 증가시키면서 26개를 차례로 프린트

(3) 숫자와 그 코드 값을 프린트
   - 변수 c에 문자 '0'을 대입한 후에 1씩 증가시키면서 10개를 차례로 프린트

프로그래밍 실습 2

2. 5의 배수를 순서대로 계산하여 프린트하고 그 합을 계산하여 프린트

(1) int 변수 i를 선언하고 0으로 초기화, 이 변수를 5씩 증가하면서 50까지의 5의 배수를 프린트하는 프로그램을 작성

```
int main( )
{
    int i;

}
```

(2) 실습 1의 프로그램을 확장하여 0부터 50 사이의 5의 배수들의 합을 계산하여 프린트하는 프로그램을 작성