2장. 변수와 자료형

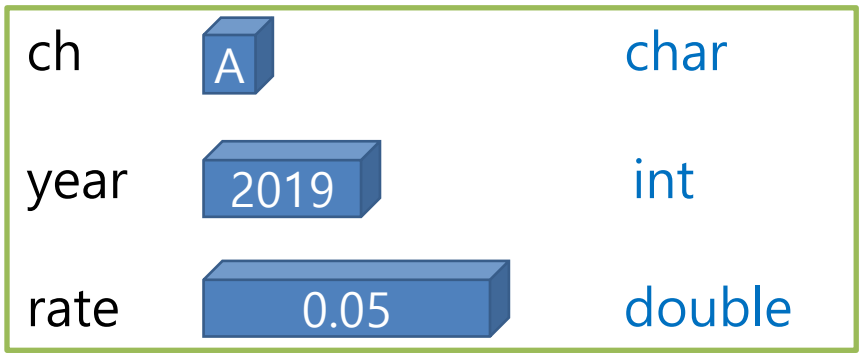
Visualstudio 2019

변수(Variable)

- **변수란?**
  - 프로그램 내부에서 사용하는 데이터를 저장해두는 메모리 공간
  - 한 순간에 한 개의 값만을 저장한다.(배열-여러 개 저장)
- **변수의 선언 및 사용**
  - 자료형 **변수이름**;
  - 자료형 **변수이름 = 초기값;**

변수 선언문

```
char ch;

int year = 2019;

double rate = 0.05;
```



변수의 선언과 사용

- **변수이름**
  - 영문자, 숫자, 밑줄 사용{ 예) int num_1 }
  - 변수명의 첫 글자는 밑줄이나 영문자여야 한다.

    숫자로 시작할 수 없고, 공백도 허용되지 않음{ 오류. int 8number }
  - 예약어는 사용할 수 없다 { char for3, int void20 }

```c
int num = 10;
int age = 35;
char grade_eng = 'A';

printf("num : %d\n", num);
printf("Age : %d\n", age);
printf("grade of english : %d\n", grade_eng);
```

```c
int a;
int b;

a = 7;
b = 5;

printf("시은이 나이는 %d살이고, 우진이 나이는 %d살입니다.\n", a, b);
```

자료형(data type)

- **자료형이란?**
  - 데이터를 저장하는 공간의 유형
  - 사용할 데이터의 종류에 따라 메모리 공간을 적절하게 설정해 주는 것

자료형		용량 (bytes)	주요 용도	범위
정수형	**char**	1	문자 또는 작은 정수 표현	-128~127
	short	2	정수 표현	-32768~32767
	int	4	큰 범위의 정수 표현	-2147483648~2147483647
	long	8	큰 범위의 정수 표현	-2^{63}~$(2^{63}-1)$
실수형	**float**	4	실수 표현	10^{-38} ~ 10^{38}
	double	8	정밀한 실수 표현	10^{-380} ~ 10^{380}

- 정수형 양수 표현범위를 2배로 늘릴 때는 자료형 앞에 unsigned를 붙일수 있는데. 이 경우 동일한 공간으로 0을 포함한 양수만을 표현하게 된다.
- 예 : char -128~127 -----→ unsigned char는 0~255 범위 표현

정수 자료형

- **정수 자료형(char, short, int, long)**

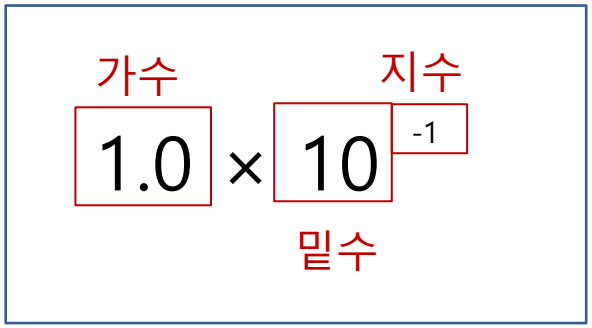
```
//자료형의 크기
printf("char 형 = %dByte\n", sizeof(char));
printf("short 형 = %dByte\n", sizeof(short));
printf("int 형 = %dByte\n", sizeof(int));
printf("long 형 = %dByte\n", sizeof(long));
printf("float 형 = %dByte\n", sizeof(float));
printf("double 형 = %dByte\n", sizeof(double));

//long형은 8byte이나 윈도 운영체제에서는 4byte임.
```

정수 자료형

```c
char ch = 'A';
printf("%c\n", ch);
printf("%d\n", ch);

char val = -128;
printf("%d\n", val);  //-128

char val2 = 128;
printf("%d\n", val2);  //-128, 오버플로우(overflow) 발생

unsigned char val3 = 256;  //0~255, 오버플로우(overflow) 발생
printf("%d\n", val3);

int iNum = 2147483647;
printf("%d\n", iNum);

int iNum2 = 2147483648;  //-2147483648, 오버플로우(overflow) 발생
printf("%d\n", iNum2);

float fNum = 1.1234567;           //소수이하 6자리 표현
double dNum = 1.1234567890123456; //소수점 이하 15자리 표현

printf("float형 : %f\n", fNum);
printf("double형 : %.16lf\n", dNum);
```

실수 자료형

- **실수 자료형(float, double)**
  - 실수값 3.14를 표현하면 3이라는 정수부분과 .14라는 소수 부분을 따로 표현할수 있고, 0과 1사이에는 무한개의 실수가 있다.
  - 부동 소수점 방식 : 실수를 지수부와 가수부로나눔.
  - 정밀도의 차이 : float형(소수이하 6자리), double형(소수이하 15자리)




```
float fNum = 1.1234567;              //소수이하 6자리 표현
double dNum = 1.1234567890123456; //소수점 이하 15자리 표현

printf("float형 : %f\n", fNum);
printf("double형 : %.16lf\n", dNum);
```

▶0.1을 표현하는 방식

문자 자료형

- **문자 자료형(char, 배열)**
  - 문자 1개를 표현할때 홑따옴표(' ')로 감싸준다.
  - 아스키 코드(ASCII코드) – 각 문자에 따른 특정한 숫자 값(코드 값)을 부여

    영문자, 숫자 만 표현 가능
  - 유니 코드 – 한글, 중국어등 아스키 코드로 표현할 수 없는 문자를 2바이트 이상의 크기로 표현할수 있는 표준 코드

```
char ch = 'A';
printf("%c\n", ch);
printf("%d\n", ch);

ch = 66;
printf("%c\n", ch+1);
printf("%d\n", ch+1);
```

http://www.unicode.org/charts/PDF/UAC00.pdf

문자 자료형

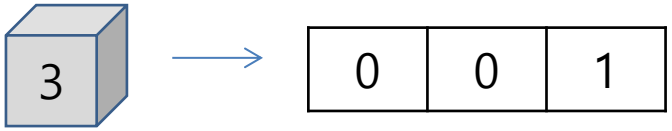
- **문자 자료형(char, 배열)**

```c
//영문자
char ch = 'a';
printf("%c\n", ch);
printf("%d\n", ch);

ch = 98;
printf("%c\n", ch);
printf("%c\n", ch + 1);
printf("%d\n", ch + 1);

//영문 문자열과 한글은 배열로 표현
char f[6] = "apple";  //맨뒤 '\0'(null)까지 6개의 메모리 공간 필요
char h[3] = "가";
char h2[3] = "\uD55C";  //유니코드
char h3[3] = "\uAE00";

printf("%s\n", f);
printf("%s\n", h);
printf("%s\n", h2);  //한
printf("%s\n", h3);  //글
```

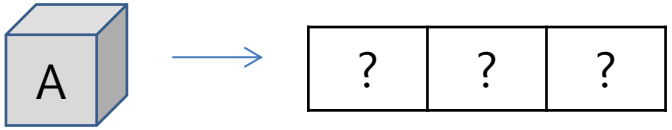
http://www.unicode.org/charts/PDF/UAC00.pdf

아스키 코드

10진수를 2진수로 변환



문자를 2진수로 변환



저장하는 문자에 해당하는 숫자를 지정하고 메모리에 저장할때는 그 숫자를 비트 단위로 바꾸어 저장

- 아스키 코드(ASCII Code)

아스키 코드는 미국 ANSI에서 표준화한 정보교환용 7비트 부호체계이다.

000(0x00)부터 127(0x7F)까지 총 128개의 부호가 사용된다.

영문 키보드로 입력할 수 있는 모든 기호들이 할당되어 있는 부호 체계이다.

아스키 코드

- 아스키 코드(ASCII Code)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u

아스키 아트(ASCII Art)

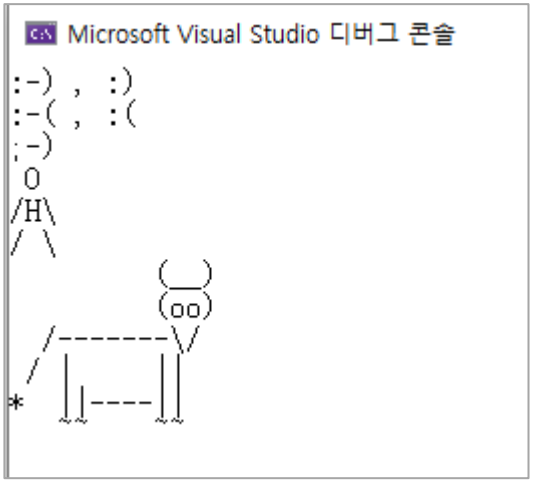
- **아스키 아트(ASCII Art)**

  아스키 코드에 포함되는 문자나 기호를 이용한 그림

  웃는 표정 :-) , :)

  찡그린 얼굴 :-(

```
printf(":-) , :) \n");
printf(":-( , :( \n");
printf(";-) \n");

printf(" O\n");
printf("/H\\\n");
printf("/ \\\n");

printf("          (__)\n");
printf("          (oo)\n");
printf("  /-------\\/ \n");
printf(" / |      ||  \n");
printf("*  ||----||  \n");
printf("   ~~    ~~  \n");
```



컴퓨터에서 데이터 표현하기

- **비트(binary digit)**

  컴퓨터가 표현하는 데이터의 최소 단위로 2진수 하나의 값을 저장할 수 있는 메모리의 크기

  컴퓨터는 0과 1로만 데이터를 저장함(0-> 신호꺼짐, 1-> 신호켜짐)

- 비트로 표현할 수 있는 수의 범위

비트수	표현할 수 있는 범위(십진수)	
1bit	0, 1(0~1)	2^1
2bit	00, 01, 10, 11(0~3)	2^2
3bit	000, 001, 010, 011, 100, 101, 110, 111(0~7)	2^3

10진수를 2진수로 바꾸기

- **진수 표현**

10진수	2진수	16진수	10진수	2진수	16진수
1	00000001	1	9	00001001	9
2	00000010	2	10	00001010	A
3	00000011	3	11	00001011	B
4	00000100	4	12	00001100	C
5	00000101	5	13	00001101	D
6	00000110	6	14	00001110	E
7	00000111	7	15	00001111	F
8	00010000	8	16	000010000	10

자리 올림 발생

컴퓨터에서 데이터 표현하기

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

printf("a의 값 : %d \n", a);
printf("b의 값 : %d \n", b);

//메모리 주소 - 16진수로 표기
printf("변수 a의 시작주소 : %x \n", &a);
printf("변수 b의 시작주소 : %x \n", &b);


//10을 10진수, 2진수, 16진수로 표
int num = 10;
int bNum = 0b1010;
int hNum = 0xA;
printf("%d\n", num);
printf("%d\n", bNum);
printf("%d\n", hNum);
```

```c
int num1 = 0b00000000000000000000000000000101;  //5
int num2 = 0b11111111111111111111111111111011;  //-5
int sum = 0;

printf("%d\n", num1);
printf("%d\n", num2);

sum = num1 + num2;

printf("%d\n", sum);
```

부호 있는 수를 표현하는 방법

- **음의 정수는 어떻게 표현할까?**
  - 정수의 가장 왼쪽에 존재하는 비트는 부호비트입니다.
    (양의 정수는 0, 음의 정수는 1을 붙인다.)
  - 음수를 만드는 방법은 2의 보수를 취한다.(1의 보수는 0과 1을 반대로 바꿈)

형 변환(Type Conversion)

- **묵시적 형 변환(자동)**

  1) 작은 자료형에서 큰 자료형으로 변환
  덜 정밀한 수에서 더 정밀한 수로 대입되는 경우

  예) int iNum = 20;
  float fNum = iNum;

  2) 연산 중에 자동 변환되는 경우

  예) int num1=100; // 정수
  double num2=3.14; // 실수
  printf("%lf ₩n", num1+num2); // 정수 + 실수

- **명시적 형 변환(강제)**

  1) 큰 자료형에서 작은 자료형으로 변환
  변환 자료형을 명시해야 하고(괄호사용), 자료의 손실이 발생

  예) double dNum = 12.34;
  int iNum = (int)dNum;

형 변환(Type Conversion)

```c
//묵시적 형변환(자동 형변환)
int iNum = 20;
float fNum = iNum;  //큰자료형 = 작은 자료형

printf("%d\n", iNum);
printf("%f\n", fNum);  //20.0
printf("%f\n", iNum + fNum);  //40.0

//명시적 형변환(강제 형변환) - cast(캐스트)라 함.
double dNum = 2.54;
int iNum2 = (int)dNum;  //작은 자료형 = 큰 자료형

printf("%d\n", iNum2);  //2

//연산
dNum = 1.2;
fNum = 0.9;

iNum = (int)dNum + (int)fNum;
printf("%d\n", iNum); //1

iNum = (int)(dNum + fNum);
printf("%d\n", iNum);  //2
```

형 변환 실습 문제

변수 두 개를 선언해서 10과 2.0을 대입하고, 두 변수의 사칙연산의 결과를 정수로 출력해 보세요.

```c
//사칙연산
int num1 = 10;
double num2 = 2.0;

printf("%d\n", (int)(num1 + num2));
printf("%d\n", (int)(num1 - num2));
printf("%d\n", (int)(num1 * num2));
printf("%d\n", (int)(num1 / num2));
```

상수(constant)

- **상수(constant)**
  - 한번 설정해 두면 그 프로그램이 종료 될 때까지 결코 변경될 수 없는 값
  - 상수를 숫자로 직접 나타내는 것보다 이름을 붙여 사용하는 것이 좋다.
  - 상수 이름은 대문자로 관례적으로 사용.

(1) **#define 상수이름 상수값** – 매크로 상수로 정의(전처리, 컴파일되지 않음)

(2) **const** 자료형 **상수이름** = **상수값**;

```
#define PI 3.141592

const double PI = 3.141592;

//PI = 3.14 (변경할 수 없다.)
```

상수(constant)

- 상수(constant)

```c
//const 키워드로 상수 선언
const int MIN_NUM = 1;
const int MAX_NUM = 100;

//MAX_NUM = 200; 컴파일 에러, 상수는 변경 불가

printf("%d\n", MIN_NUM);
printf("%d\n", MAX_NUM);

//원의 넓이 계산하기
const double PI = 3.14;
int radius = 5;
double area;

area = PI * radius * radius;
printf("원의 넓이 = %.1lf\n", area);
```