상수와 기본 자료형

- 숫자 자료형, 문자 자료형, 상수, 그리고 형 변환 -

성공회대학교 IT융합자율학부
소프트웨어공학전공
홍 성 준

C언어가 제공하는 기본 자료형의 이해

◎ 자료형 (data type)

- 데이터를 표현하는 방법

  (참고) 변수 : 데이터의 저장을 위해 할당된 메모리 공간에 붙여진 이름

실수를 저장할 것인가? 정수를 저장할 것인가?

· 실수냐 정수냐에 따라 값을 저장하는 방식 달라지기 때문에 용도를 결정해야 한다.

얼마나 큰 수를 저장할 것인가?

· 큰 수를 표현하기 위해서는 많은 바이트(bytes)가 필요하다.

- "정수를 저장하려고 하는데, 크기가 4바이트면 좋겠어요. 변수 이름은 num 으로 할게요."

C언어가 제공하는 기본 자료형의 이해

◎ 기본 자료형의 종류와 데이터의 표현 범위

자료형		크기	값의 표현범위
정수형	char	1바이트	−128이상 +127이하
	short	2바이트	−32,768이상 +32,767이하
	int	4바이트	−2,147,483,648이상 +2,147,483,647이하
	long	4바이트	−2,147,483,648이상 +2,147,483,647이하
	long long	8바이트	−9,223,372,036,854,775,808이상
			+9,223,372,036,854,775,807이하
실수형	float	4바이트	$\pm 3.4\times10^{-37}$이상 $\pm 3.4\times10^{+38}$이하
	double	8바이트	$\pm 1.7\times10^{-307}$이상 $\pm 1.7\times10^{+308}$이하
	long double	8바이트 이상	double 이상의 표현범위

- 컴파일러에 따라 약간의 차이가 있음
  - C표준에서는 자료형 별 상대적 크기만 표준화 하고 구체적인 크기를 정하진 않음
    (예) short과 int는 최소 2바이트로 하되, int는 short보다 크기가 크거나 같아야 한다.

C언어가 제공하는 기본 자료형의 이해

◎ 연산자 sizeof 를 이용한 자료형의 크기 확인

- sizeof() 연산자
  - 변수, 상수 및 자료형 이름을 피연산자(인자)로 하여 데이터형의 크기를 반환

```
int main(void)
{
     int num = 10;
     int sz1 = sizeof(num);
     int sz2 = sizeof(int);
     . . . .
}
```

C언어가 제공하는 기본 자료형의 이해

◎ 연산자 sizeof 를 이용한 자료형의 크기 확인

- sizeof() 연산자
  - 변수, 상수 및 자료형 이름을 피연산자(인자)로 하여 데이터형의 크기를 반환
- SizeOfOperator.c

```
int main(void)
{
    char ch=9;
    int inum=1052;
    double dnum=3.1415;
    printf("변수 ch의 크기: %d \n", sizeof(ch));
    printf("변수 inum의 크기: %d \n", sizeof(inum));
    printf("변수 dnum의 크기: %d \n", sizeof(dnum));

    printf("char의 크기: %d \n", sizeof(char));
    printf("int의 크기: %d \n", sizeof(int));
    printf("long의 크기: %d \n", sizeof(long));
    printf("long long의 크기: %d \n", sizeof(long long));
    printf("float의 크기: %d \n", sizeof(float));
    printf("double의 크기: %d \n", sizeof(double));
    return 0;
}
```

```
변수 ch의 크기: 1
변수 inum의 크기: 4
변수 dnum의 크기: 8
char의 크기: 1
int의 크기: 4
long의 크기: 4
long long의 크기: 8
float의 크기: 4
double의 크기: 8
```

C언어가 제공하는 기본 자료형의 이해

◎ 정수형 데이터를 처리하기 위한 일반적인 자료형의 선택

- '저장하고자 하는 값의 범위'를 고려
  - short 형은 -32,768 ~ +32,767 까지 표현하기 때문에 32,768을 표현하기 위해선 int 형을 사용해야 함
- CharShortBaseAdd.c

```
int main(void)
{
    char num1=1, num2=2, result1=0;
    short num3=300, num4=400, result2=0;

    printf("size of num1 & num2: %d, %d \n", sizeof(num1), sizeof(num2));
    printf("size of num3 & num4: %d, %d \n", sizeof(num3), sizeof(num4));
    printf("size of char add: %d \n", sizeof(num1+num2));
    printf("size of short add: %d \n", sizeof(num3+num4));

    result1=num1+num2;
    result2=num3+num4;
    printf("size of result1 & result2: %d, %d \n", sizeof(result1), sizeof(result2));
    return 0;
}
```

```
size of num1 & num2: 1, 1
size of num3 & num4: 2, 2
size of char add: 4
size of short add: 4
size of result1 & result2: 1, 2
```

C언어가 제공하는 기본 자료형의 이해

◎ 실수형 데이터를 처리하기 위한 일반적인 자료형의 선택

- float 형, double 형 둘 다 표현할 수 있는 범위는 넓음
- 실수 자료형의 선택에 중요한 건 '정밀도'
- 정밀도 : 오차가 발생하지 않는 소수점 이하의 자릿수

실수 자료형	소수점 이하 정밀도	바이트 수
float	6자리	4
double	15자리	8
long double	18자리	12

C언어가 제공하는 기본 자료형의 이해

◎ 실수형 데이터를 처리하기 위한 일반적인 자료형의 선택

- CircleArea.c

```
int main(void)
{
    double rad;
    double area;
    printf("원의 반지름 입력: ");
    scanf("%lf", &rad);
    area = rad*rad*3.1415;
    printf("원의 넓이: %f \n", area);
    return 0;
}
```

```
원의 반지름 입력: 2.4
원의 넓이: 18.095040
```

C언어가 제공하는 기본 자료형의 이해

◎ unsigned을 붙여 0과 양의 정수(음수가 아닌 정수)만 표현하기

정수 자료형	크기	값의 표현범위
char	1바이트	−128이상 +127이하
unsigned char		0이상 (128 + 127)이하
short	2바이트	−32,768이상 +32,767이하
unsigned short		0이상 (32,768 + 32,767)이하
int	4바이트	−2,147,483,648이상 +2,147,483,647이하
unsigned int		0이상 (2,147,483,648 + 2,147,483,647)이하
long	4바이트	−2,147,483,648이상 +2,147,483,647이하
unsigned long		0이상 (2,147,483,648 + 2,147,483,647)이하
long long	8바이트	−9,223,372,036,854,775,808이상 +9,223,372,036,854,775,807이하
unsigned long long		0이상 (9,223,372,036,854,775,808 + 9,223,372,036,854,775,807)이하

문자의 표현방식과 문자를 위한 자료형

◎ 아스키(ASCII) 코드

- 문자를 표현하기 위해 숫자를 문자에 맵핑(mapping)한 코드
- 미국 표준 협회(ANSI: American National Standards Institute)에서 제정
- 알파벳과 일부 특수문자를 포함하여 총 128개의 문자로 구성

아스키 코드	아스키 코드 값
A	65
B	66
C	67
`	96
~	126

```
int main(void)
{
    char ch1 = 'A';
    char ch2 = 'C';
    . . . .
}
```

```
int main(void)
{
    char ch1 = 65;
    char ch2 = 67;
    . . . .
}
```

- C언어에서는 작은 따옴표 ‘ ’ 로 감싸서 문자를 표현

문자의 표현방식과 문자를 위한 자료형

◎ 아스키(ASCII) 코드 (cont.)

- 모든 아스키 코드는 1바이트로 충분히 표현 가능하기 때문에, 문자를 저장하기 위해 char 형을 주로 사용
- HowChar.c

```
int main(void)
{
    char ch1='A', ch2=65;
    int ch3='Z', ch4=90;

    printf("%c %d \n", ch1, ch1);
    printf("%c %d \n", ch2, ch2);
    printf("%c %d \n", ch3, ch3);
    printf("%c %d \n", ch4, ch4);
    return 0;
}
```

```
A 65
A 65
Z 90
Z 90
```

◎ char 형은 문자형인가 정수형인가?

- char 형은 원래 문자의 표현을 목적으로 정의된 자료형으로 '문자형'으로 분류되지만, 실제 '정수형 ' 에 속함

상수에 대한 이해

◎ 상수 (constant)

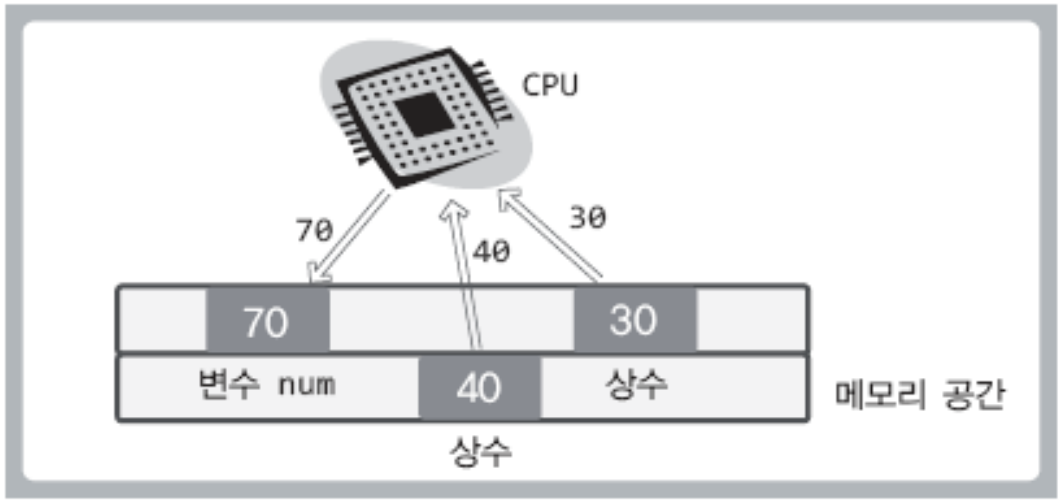
- 변경이 불가능한 데이터 ↔ 변수

◎ 리터럴 상수(literal constant)

- 연산을 위해 메모리 상에 저장되는 이름 없는 상수

  (예) 30, 40, 2.12, 7.49, 'a' 등

```
int main(void)
{
    int num = 30 + 40;
    . . . .
}
```



- 단계 1. 정수 30과 40이 메모리 공간에 상수의 형태로 저장된다.
- 단계 2. 두 상수를 기반으로 덧셈이 진행된다.
- 단계 3. 덧셈의 결과로 얻어진 정수 70이 변수 num에 저장된다.

상수에 대한 이해

◎ 리터럴 상수의 자료형

- 정수는 기본적으로 int 형으로 표현
- 실수는 기본적으로 double 형으로 표현
- 문자는 기본적으로 int 형으로 표현
- LiteralSize.c

```
int main(void)
{
    printf("literal int size: %d \n", sizeof(7));
    printf("literal double size: %d \n", sizeof(7.14));
    printf("literal char size: %d \n", sizeof('A'));
    return 0;
}
```

```
literal int size: 4
literal double size: 8
literal char size: 4
```

상수에 대한 이해

◎ 접미사를 이용한 다양한 리터럴 상수의 표현

```
int main(void)
{
    float num1 = 5.789;    // 경고 메시지 발생
    float num2 = 3.24 + 5.12;    //경고 메시지 발생
    return 0;
}
```

```
float num1 = 5.789f;        // 경고 메시지 발생 안 함
float num2 = 3.24F + 5.12F;    // 소문자 f 대신 대문자 F를 써도 된다!
```

상수에 대한 이해

◎ 접미사를 이용한 다양한 리터럴 상수의 표현 (cont.)

접미사	자료형	사용의 예
U	unsigned int	unsigned int n = 1025U
L	long	long n = 2467L
UL	unsigned long	unsigned long n = 3456UL
LL	long long	long long n = 5768LL
ULL	unsigned long long	unsigned long long n = 8979ULL

[표 05-4: 정수형 상수의 표현을 위한 접미사]

접미사	자료형	사용의 예
F	float	float f = 3.15F
L	long double	long double f = 5.789L

[표 05-5: 실수형 상수의 표현을 위한 접미사]

상수에 대한 이해

◎ 심볼릭 상수 (symbolic constant)

- const 키워드
  - 변수 선언시 const 키워드 추가
  - 선언과 동시에 반드시 초기화를 해야함
- 상수 이름 선언하는 규칙
  - 상수의 이름은 모두 대문자로 표시
    (예) MAX, MIN, PI
  - 둘 이상의 단어는 '_' 로 연결하여 명명
    (예) MAX_LENGTH, NUM_SAMPLES

```
int main(void)
{
    const int MAX=100;    // MAX는 상수! 따라서 값의 변경 불가!
    const double PI=3.1415;    // PI는 상수! 따라서 값의 변경 불가!
    . . . .
}
```

```
int main(void)
{
    const int MAX;    // 쓰레기 값으로 초기화 되어버림
    MAX=100;    // 값의 변경 불가! 따라서 컴파일 에러 발생!
    . . . .
}
```

자료 형의 변환

◎ 자료 형의 변환 (type casting)

- 자동 형 변환 (묵시적 형 변환; implicit type casting)
  - 자동으로 발생하는 형 변환
- 강제 형 변환 (명시적 형 변환; explicit type casting)
  - 프로그래머가 형 변환을 명시하여 강제로 일으키는 형 변환

자료 형의 변환

◎ 자동 형 변환

- 대입 연산의 전달 과정에서 발생하는 자동 형 변환 (cont.)

형 변환의 방식에 대한 유형별 정리

· 정수를 실수로 형 변환　　3은 3.0으로 5는 5.0으로(오차가 발생하게 된다).

· 실수를 정수로 형 변환　　소수점 이하의 값이 소멸된다.

· 큰 정수를 작은 정수로 형 변환　　작은 정수의 크기에 맞춰서 상위 바이트가 소멸된다.

자료 형의 변환

◎ 자동 형 변환

- 대입 연산의 전달 과정에서 발생하는 자동 형 변환 (cont.)
- AutoConvOne.c

```
int main(void)
{
    double num1=245;
    int num2=3.1415;
    int num3=129;
    char ch=num3;

    printf("정수 245를 실수로: %f \n", num1);
    printf("실수 3.1415를 정수로: %d \n", num2);
    printf("큰 정수 129를 작은 정수로: %d \n", ch);
    return 0;
}
```

```
정수 245를 실수로: 245.000000
실수 3.1415를 정수로: 3
큰 정수 129를 작은 정수로: -127
```

자료 형의 변환

◎ 자동 형 변환

- 정수의 승격 (integer promotion) 에 의한 자동 형 변환
  - CPU는 32비트 정수 연산에 최적화 되어, int 형보다 작은 정수형 데이터를 int 형으로 형 변환하여 연산

```
int main(void)
{
     short num1=15, num2=25;
     short num3=num1+num2;     // num1과 num2가 int형으로 형 변환
     . . . .
}
```

자료 형의 변환

◎ 자동 형 변환

- 피연산자의 자료형 불일치로 발생하는 자동 형 변환
  - 이항 연산자는 피연산자의 자료형이 일치해야 하는데, 일치하지 않는 경우 자동 형 변환이 발생

```
double num1 = 5.15 + 19;
```

  - 산술 연산에서의 자동 형 변환 규칙

(1) 크기가 클 수록 우선

(2) 정수형보다 실수형을 우선

자료 형의 변환

◎ 명시적 형 변환

- 연산 결과의 자료형은 피연산자의 자료형과 일치함
- ConvDiv.c

```
int main(void)
{
    int num1=3, num2=4;
    double divResult;
    divResult = num1 / num2;
    printf("나눗셈 결과: %f \n", divResult);
    return 0;
}
```

```
나눗셈 결과: 0.000000
```

자료 형의 변환

◎ 명시적 형 변환

- 형 변환 연산자 '( )'

```
divResult = (double)num1 / num2;
```

```
int main(void)
{
    int num1 = 3;
    double num2 = 2.5 * num1;
    . . . .
}
```

⟺

```
int main(void)
{
    int num1 = 3;
    double num2 = 2.5 * (double)num1;
    . . . .
}
```

자료 형의 변환

◎ 명시적 형 변환

- 형 변환 연산자 '( )'
- ConvDiv.c

```
int main(void)
{
    int num1=3, num2=4;
    double divResult;
    divResult = num1 / num2;
    printf("나눗셈 결과: %f \n", divResult);
    return 0;
}
```