Variable

Jinwoo Choi

Variable(변수)란?

데이터를 담는 상자

얼만큼의 공간이 필요할까?

Declaring Variables

type variableName

```
int myVariable1;

float myVariable2;

char myVariable3;
```

Initializing Variables

```
type variableName = value;

int myVariable1 = 12345;

int mightGiveWarning;
mightGiveWarning = 12345;

float myVariable2 = 12.345;
```

Variable Types

1 Byte = 8 Bit

1	0	1	1	1	0	0	1
+-	2^6	2^5	2^4	2^3	2^2	2^1	2^0

User-defined Data Types

Arrays Data Type

Pointers Data Type

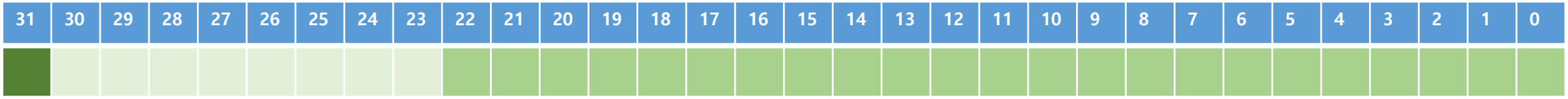
Integer Data Types

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	8 bytes	-9223372036854775808 to 9223372036854775807
unsigned long	8 bytes	0 to 18446744073709551615

Floating-Point Data Types

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

IEEE 754 Floating-Point Standard



$0.25_{10} \rightarrow 0.01_2 \rightarrow (-1)^0 * 1.00_2 * 2^{-2}$

Apply Bias: $E = -2 + 127 = 125_{10} \rightarrow 01111101_2$

0 | 01111101 | 00000000000000000000000

$-1.25_{10} \rightarrow -1.01_2 \rightarrow (-1)^1 * 1.01_2 * 2^0$

Apply Bias: $E = -0 + 127 = 127_{10} \rightarrow 01111111_2$

1 | 01111111 | 01000000000000000000000

Bias: 127(8bit), 1023(11bit)

Infinity: All **exponents** are 1, All **Mantissa** are 0, if **–inf** : **sign** = 1 if **inf** : **sign** = 0

NaN(Not a Number): All **exponents** are 1, if **Mantissa** has at least one 1

Zero: All **exponents** and **Mantissa** are 0, if **–0** : **sign** = 1 if **0** : **sign** = 0

sizeof

sizeof operator

Queries size of the object or type.

Used when actual size of the object must be known.

Syntax

`sizeof(` *type* `)`	(1)
`sizeof` *expression*	(2)

Both versions return a value of type `size_t`.

```
int i;
sizeof(i)
-> 4

sizeof(double)
-> 8
```

Explanation

1) Returns the size, in bytes, of the object representation of *type*
2) Returns the size, in bytes, of the object representation of the type of *expression*. No implicit conversions are applied to *expression*.

Variable Naming Rules

- Names can contain letters, digits and underscores
- Names must begin with a letter or an underscore (_)
- Names are case-sensitive('myVar' and 'myvar' are different variables)
- Names cannot contain whitespaces or special characters like !, #, %, etc.
- Reserve words (such as int) cannot be used as names

Variable Naming Rules

C keywords

This is a list of reserved keywords in C. Since they are used by the language, these keywords are not available for re-definition. As an exception, they are not considered reserved in *attribute-tokens*(since C23)

alignas (C23)	extern	sizeof	_Alignas (C11)(deprecated in C23)
alignof (C23)	false (C23)	static	_Alignof (C11)(deprecated in C23)
auto	float	static_assert (C23)	_Atomic (C11)
bool (C23)	for	struct	_BitInt (C23)
break	goto	switch	_Bool (C99)(deprecated in C23)
case	if	thread_local (C23)	_Complex (C99)
char	inline (C99)	true (C23)	_Decimal128 (C23)
const	int	typedef	_Decimal32 (C23)
constexpr (C23)	long	typeof (C23)	_Decimal64 (C23)
continue	nullptr (C23)	typeof_unqual (C23)	_Generic (C11)
default	register	union	_Imaginary (C99)
do	restrict (C99)	unsigned	_Noreturn (C11)(deprecated in C23)
double	return	void	_Static_assert (C11)(deprecated in C23)
else	short	volatile	_Thread_local (C11)(deprecated in C23)
enum	signed	while	

Explicit Type Conversion

```
int i_1 = 9;

int i_2 = 6;

float f_1 = i_1 / i_2;

float f_2 = (float)i_1 / i_2;

printf("%f %f", f_1, f_2);
```

LAB – Variable Declaration

- Create a file named 'VariableDeclaration_YourName.c'.
- Declare and initialize at least 10 types of variables

  (can have same type but requires a different name).
- Variables must follow the naming rules.
- You must define at least one variable that contains the value of

  sizeof("any variable or variable type");

```
#include<stdio.h>

int main(void)
{
    //todo: declare more than 10 types of variables;
    return 0;
}
```

Input/Output

C Standard Output

Most common input/output function in standard I/O library

Defined in header <stdio.h>

```
int printf( const char*          format, ... );          (1)   (until C99)
int printf( const char* restrict format, ... );                (since C99)
```

https://en.cppreference.com/w/c/io/fprintf

printf(“This is just a simple example for the output function”);

printf(“This prints out the number : %d”, 12345);

printf(“This goes in to the first row. \nAnd this goes in to the second row.”);

C Standard Output – Escape Sequence

Escape Sequence	Name	Description
\a	Alarm or Beep	It is used to generate a bell sound in the C program.
\b	Backspace	It is used to move the cursor one place backward.
\f	Form Feed	It is used to move the cursor to the start of the next logical page.
\n	New Line	It moves the cursor to the start of the next line.
\r	Carriage Return	It moves the cursor to the start of the current line.
\t	Horizontal Tab	It inserts some whitespace to the left of the cursor and moves the cursor accordingly.
\v	Vertical Tab	It is used to insert vertical space.
\\	Backlash	Use to insert backslash character.
\’	Single Quote	It is used to display a single quotation mark.
\”	Double Quote	It is used to display double quotation marks.
\?	Question Mark	It is used to display a question mark.
\ooo	Octal Number	It is used to represent an octal number.
\xhh	Hexadecimal Number	It represents the hexadecimal number.
\0	NULL	It represents the NULL character.

C Standard Output – Escape Sequence

```c
#include<stdio.h>

int main(void)
{
    printf("Hello World! \nThis is a new line\n");
    printf("This is a \"Double Quote\".\n");
    char c = '\'';
    printf("Use \' if it is a char type. You can use ' in string type\n");
    printf("This is a \t horizontal tab");

    return 0;
}
```

C Standard Output – Format specifier

Format Specifier	Description
%c	For character type.
%d	For signed integer type.
%e or %E	For scientific notation of floats.
%f	For float type.
%g or %G	For float type with the current precision.
%i	signed integer
%ld or %li	Long
%lf	Double
%Lf	Long double
%lu	Unsigned int or unsigned long

C Standard Output – Format specifier

%lli or %lld	Long long
%llu	Unsigned long long
%o	Octal representation
%p	Pointer
%s	String
%u	Unsigned int
%x or %X	Hexadecimal representation
%n	Prints nothing
%%	Prints % character

C Standard Output – Format specifier

```c
#include<stdio.h>

int main(void)
{
    int i_v = 96;
    float f_v = 11.3f;
    double d_v = 107.3;
    char c_v = '\\';
    int i_1 = 96, i_2 = 113;

    printf("This is an integer value: %d\n", i_v);
    printf("This is an float value: %f\n", f_v);
    printf("This is an double value: %lf\n", d_v);
    printf("Result of %d + %d is %d.\n", i_1, i_2, i_1 + i_2);
    printf("Guess how it will be printed: %c", c_v);

    return 0;
}
```

C Standard Input

Most common input/output function in standard I/O library

MSVC environment

scanf_s("%d", &i_v)

Defined in header <stdio.h>

```
int scanf( const char          *format, ... );
int scanf( const char *restrict format, ... );
```

(1) (until C99) (since C99)

https://en.cppreference.com/w/c/io/fscanf

We want to put input value to the variable 'i'.
To do that, we need to know where 'i' exists.
& gives the address where 'i' exists.

```
int i;

scanf("%d", &i);
```

```
scanf("%d", &i_v);

scanf("%f", &f_v);

scanf("%lf", &d_v);

scanf("%c", &c_v);
```

LAB – I/O

- Create a file named 'InputOutput_YourName.c'.
- Copy and paste variables you declared from 'VariableDeclaration.c'
- Write a program that gets input for the variables that you defined already and print out to the console. (except %p, %n, and %%)
- You can try gets(), puts(), getchar(), putchar() in lab if you want.

```
#include<stdio.h>

int main(void)
{
    int i;
    scanf("%d", &i);
    printf("i variable has value %d.\n", i);
    return 0;
}
```

C Standard Output

Defined in header <stdio.h>

```
char* gets( char* str );                    (1)   (removed in C11)
char* gets_s( char* str, rsize_t n );       (2)   (since C11)
```

Defined in header <stdio.h>

```
int puts( const char* str );
```

Defined in header <stdio.h>

```
int getchar( void );
```

Defined in header <stdio.h>

```
int putchar( int ch );
```