CADT
បណ្ឌិត្យសភាបច្ចេកវិទ្យាឌីជីថលកម្ពុជា
Cambodia Academy of Digital Technology

IDT
វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

Week 03

Variable, Data Types and Operators

Prepared by: **Leangsiv HAN**

01st March 2024

Last Week

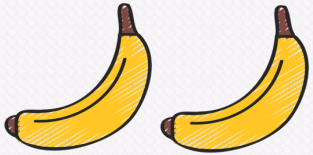
You have learned about:

- The number system used in computers.
- The types of number systems, digit symbols, and bases.
- The calculation of binary arithmetic.
- The method of number system conversions.

Let’s get started with this scenario!

We have:

Fruit A = 

Fruit B = 

Fruit Total = A + B = 5

What if?

We have:

Fruit A = 3

Fruit B = 2

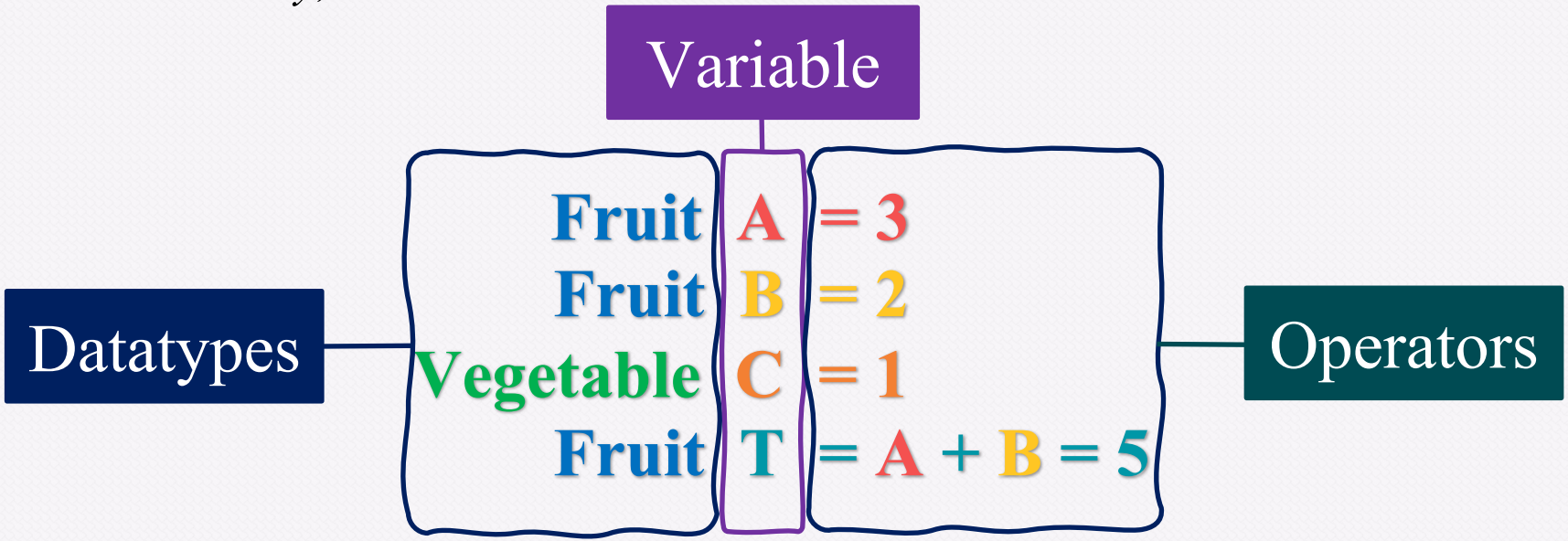
Vegetable C = 

Fruit Total = A + B + ~~C~~ = 5

Question?

Do you think it is related to Programming Language?

- *Similarly,*



Learning Objectives

By the end of this lesson, you will be to:

- ❑ Comprehend the overview of programming in C.
- ❑ Identify different data types used in programming.
- ❑ Recognize the concept of tokens in programming.
- ❑ Utilize variables and understand their usage.
- ❑ Apply various operators used in programming and comprehend their functions.

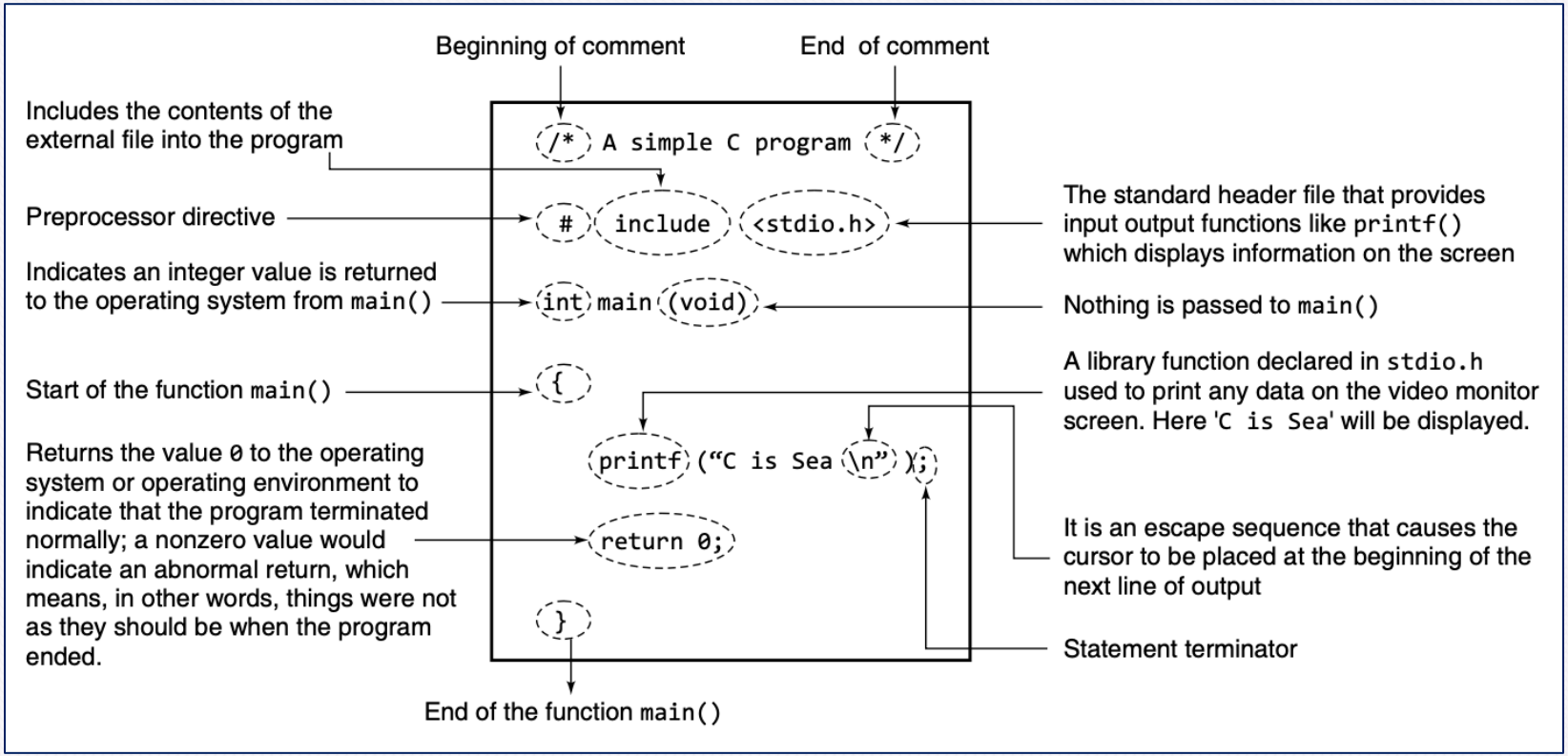
Introduction

Definition:

- **Syntax** is the set of rules that dictate how a program is written and how it should be structured.
- It includes the appropriate use of *keywords, operators, variables, data types, functions, and control structures*.

(Ref. StudySmarter)

Parts of C Program REVISITED



Structure of C Program

Preprocessor directives

Global declarations

```
int main (void) {
    Local Definitions
    Statements
    return 0;
}
```

Backslash Codes

Code	Meaning
\a	Ring terminal bell (a is for alert) [ANSI extension]
\?	Question mark [ANSI extension]
\b	Backspace
\r	Carriage return
\f	Form feed
\t	Horizontal tab
\v	Vertical tab
\0	ASCII null character
\\	Backslash
\"	Double quote
\'	Single quote
\n	New line
\o	Octal constant
\x	Hexadecimal constant

COMMON USAGE

Comments (1/4)

- The **Comments** in C are human-readable *explanations or notes* in the source code of a C program.
- A comment *makes the program easier to read and understand*. These are the statements that are not executed by the compiler or an interpreter. It is considered to be a *good practice to document* our code using comments.
- In C there are **two types of comments** in C language:
  1. *Single-line comment*
  2. *Multi-line comment*

Comments (2/4)

1. Single-line Comment in C

A single-line comment in C starts with (//) double forward slash. It extends till the end of the line and we don't need to specify its end.

Syntax: // This is a single-line comment

Comments (3/4)

1. Multi-line Comment in C

The Multi-line comment in C starts with a forward slash and asterisk (/*) and ends with an asterisk and forward slash (*/). Any text between /* and */ is treated as a comment and is ignored by the compiler.

Syntax: **/*Comment starts**

```
   continues
   continues
   .
   .
   .
Comment ends*/
```

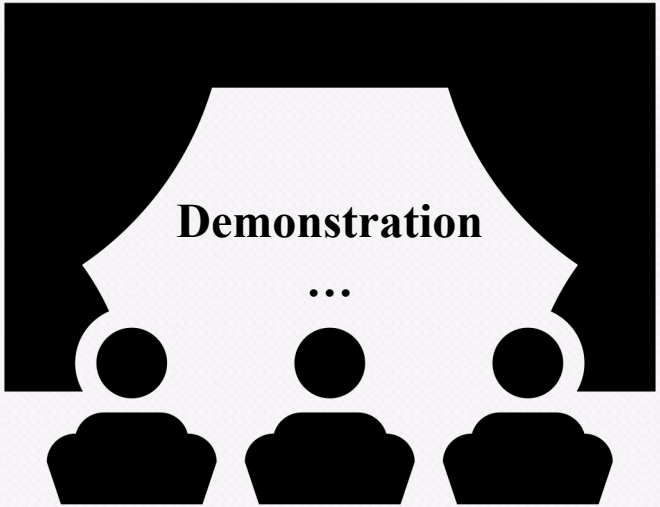
Comments (4/4)

When and Why to use Comments in C programming?

- A person reading a large code will be bemused if comments are not provided about details of the program.
- C Comments are a way to make a code more readable by providing more descriptions.
- C Comments can include a description of an algorithm to make code understandable.
- C Comments can be used to prevent the execution of some parts of the code.

(Ref. C Comments)

Let’s look at the code in IDE together!



Token

Tokens are the basic *lexical building blocks* of *source code*. In other words, tokens are *one or more symbols* understood by the compiler that help it interpret the program code.

There are **five classes** of tokens: *identifiers, reserved words (keywords), operators, separators, and constants*.

Identifier

Identifier or name is a *sequence of characters* created by the programmer to *identify or name a specific object*.

Some rules must be kept in mind when naming identifiers. These are stated as follows:

1. The first character *must be an alphabetic character* (lower-case or capital letters) or an underscore '_'.
2. *All characters must be alphabetic characters, digits, or underscores.*
3. The first 31 characters of the identifier are significant. Identifiers that share the same first 31 characters may be indistinguishable from each other.
4. *A keyword cannot be duplicated by an identifier*. A keyword is word which has special meaning in C.

Naming Conversion

There are four popular ways that developers use to name things in code, including:

Case	Example
snack_case	theory_class, lab_class
camelCase	theoryClass, labClass
PascalCase	TheoryClass, LabClass
kebab-case	theory-class, lab-class

Keywords

- **Keywords** are the vocabulary of C. Because they are special to C, one cannot use them for variable names.
- There are **32 words** defined as keywords in C.
- These have *predefined uses and cannot be used for any other purpose in a C program*. They are used by the compiler to compile the program.
- They are *always written in lower-case letters*.

Keywords in C

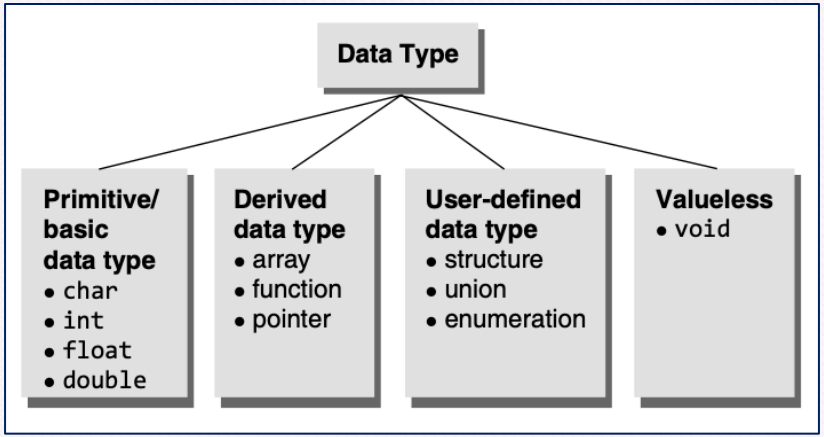
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Self-study

- Identify the three remaining classes of tokens:
  1. *Operators*
  2. *Separators*
  3. *Constant.*

Data Types

The **type or data type** of a variable determines the *set of values* that a variable might take and the *set of operations* that can be applied to those values.



Classification of data types

Basic Data Types

Data Type	Meaning	Size (in bit) [for a 32-bit computer]	Size (in bit) [for a 64-bit computer]
char	character	8	8
int	integer	32	32
float	floating-point	32	32
double	double precision floating point	64	64
void	valueless	8	8

More data types

Below is a list of ranges along with the memory requirement and format specifiers on the ***32-bit GCC compiler***.

Data Type	Size (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	1.2E-38 to 3.4E+38	%f
double	8	1.7E-308 to 1.7E+308	%lf
long double	16	3.4E-4932 to 1.1E+4932	%Lf

Ref. Data Types in C

Format specifiers for *printf*

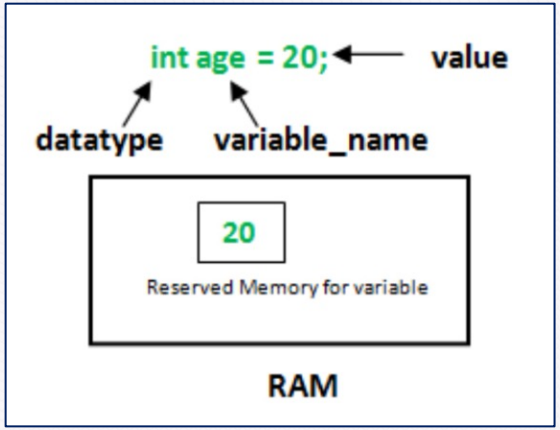
Conversion code	Usual variable type	Display
%c	`char`	single character
%d (%i)	`int`	signed integer
%e (%E)	`float` or `double`	exponential format
%f	`float` or `double`	signed decimal
%g (%G)	`float` or `double`	use `%f` or `%e`, whichever is shorter
%o	`int`	unsigned octal value
%p	pointer	address stored in pointer
%s	array of `char`	sequence of characters (string)
%u	`int`	unsigned decimal integer
%x (%X)	`int`	unsigned hex value
%%	none	no corresponding argument is converted, prints only a `%`.
%n	pointer to `int`	the corresponding argument is a pointer to an integer into which the number of characters displayed is placed.

Commonly used format specifiers

Coversion Code	Ususal variable type	Display
%c	char	single character
%d (%i)	int	signed integer
%f	float or double	signed decimal
%p	pointer	address stored in pointer
%s	array of char	sequence of characters (string)

Variable

A variable in C language is the *name associated* with some *memory location* to *store data* of different types.



We can store different types of data in the variable and reuse the same variable for storing some other data any number of times.

Ref. C Variables

C Variable Syntax

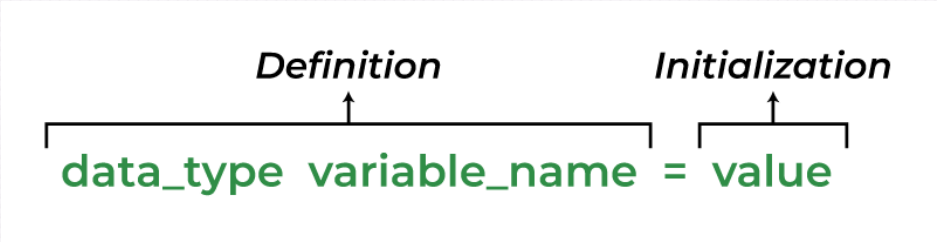
The syntax to declare a variable in C specifies the name and the type of the variable.

```
data_type variable_name = value;     // defining single variable
                or
data_type variable_name1, variable_name2;     // defining multiple variable
```

Note:

- data_type: Type of data that a variable can store.
- variable_name: Name of the variable given by the user.
- value: value assigned to the variable by the user.

3 aspects of defining a variable



1. C Variable Declaration

Variable declaration in C tells the compiler about the existence of the variable with the given name and data type.When the variable is declared compiler automatically allocates the memory for it.

3 aspects of defining a variable

2. C Variable Definition

In the definition of a C variable, the compiler allocates some memory and some value to it. A defined variable will contain some random garbage value till it is not initialized.

Example:

```
int apple;
char gender;
```

3 aspects of defining a variable

3. C Variable Initialization

Initialization of a variable is the process where the user assigns some meaningful value to the variable.

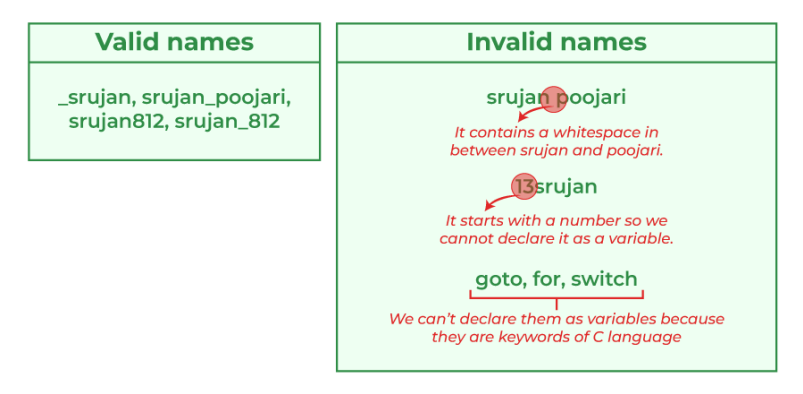
Example:

```
int num; // variable definition
num = 10; // initialization
```

or

```
int num = 10; // variable declaration and definition
```

Rules for Naming Variables in C



You can assign any name to the variable as long as it follows the following rules:

- A variable name must only contain alphabets, digits, and underscore.
- A variable name must start with an alphabet or an underscore only. It cannot start with a digit.
- No whitespace is allowed within the variable name.
- A variable name must not be any reserved word or keyword.

Constant Variable in C

A **constant variable** in C is a read-only variable whose value cannot be modified once it is defined. We can declare a constant variable using the const keyword.

```
const data_type variable_name = value; // Syntax of Const Variable
```

Self-study

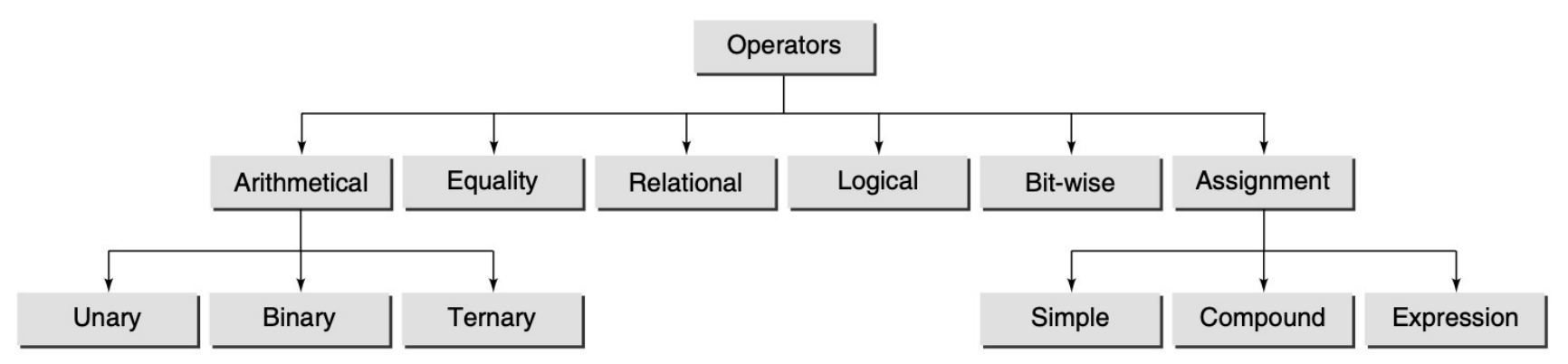
❖ Research about C Variable Types.

The C variables can be classified into the following types:

1. **Local Variables** 🌟
2. **Global Variables** 🌟
3. **Static Variables** 🌟
4. Automatic Variables
5. Extern Variables
6. Register Variables

Operators

An Operator is a symbol that specifies the mathematical, logical, or relational operation to be performed.



Classification of operators in C language

Types of Operators

Type of operator	Operator symbols with meanings
Arithmetical	Unary + (Unary) – (Unary) ++ Increment –– Decrement
	Binary + Addition – Subtraction * Multiplication / Division % Modulo
	Ternary ?: Discussed later
Assignment	Simple Assignment =
	Compound Assignment +=, -=, *=, /=, %=, &=, ^=, |=
	Expression Assignment A= 5+(b=8 + (c=2)) -4
Relational	>, <, >=, <=
Equality	= = (Equal to) != (Not equal to)
Logical	&& (Logical AND) || (Logical OR) ! (Logical NOT)
Bitwise	& (Bitwise AND) | (Bitwise OR) ~ (Complement) ^ (Exclusive OR) >> (Right Shift) << (Left Shift)
Others	, (Comma) * (indirection), . (membership operator) -> (membership operator)

Arithmetic Operators (1/6)

There are three types of arithmetic operators in C:

- Binary operators
- Unary operators
- Ternary operators *(will be discussed in the lesson: Control Statement)*

Arithmetic Operators (2/6)

- A **binary Operator** is an operator that operates on two operands to produce a new value (result).
- Most common binary operators are +, -, *, /, and %.

Syntax: **Operand_1 Operator Operand_2**

Operator	Name	Example
+	Addition	`12 + 4.9 /* gives 16.9*/`
-	Subtraction	`3.98 - 4 /* gives -0.02*/`
*	Multiplication	`2 * 3.4 /* gives 6.8  */`
/	Division	`9 / 2.0 /* gives 4.5 */`
%	Remainder	`13 % 3 /* gives 1  */`

Arithmetic Operators (3/6)

- **Unary operators** are the operators that perform operations on a single operand to produce a new value.
- **Types of unary operators** are mentioned below:
  1. Increment (++)
  2. Decrement (--)
  3. Unary minus (-)
  4. Addressof operator (&)
  5. sizeof()

Arithmetic Operators (4/6)

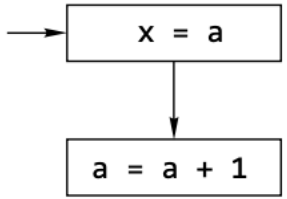
Basic rules for using increment (++) and decrement (– –)

- The operand *must be a variable* but *not a constant or an expression*.
- The operator ++ and -- may *precede* or *succeed* the operand.

Arithmetic Operators (5/6)

Postfix operation: Syntax: **Operand++ or Operand--**

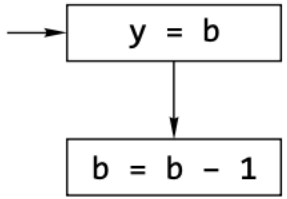
```
(a) x = a++;
```



First action: store value of `a` in memory location for variable `x`.

Second action: increment value of `a` by `1` and store result in memory location for variable `a`.

```
(b) y = b--;
```

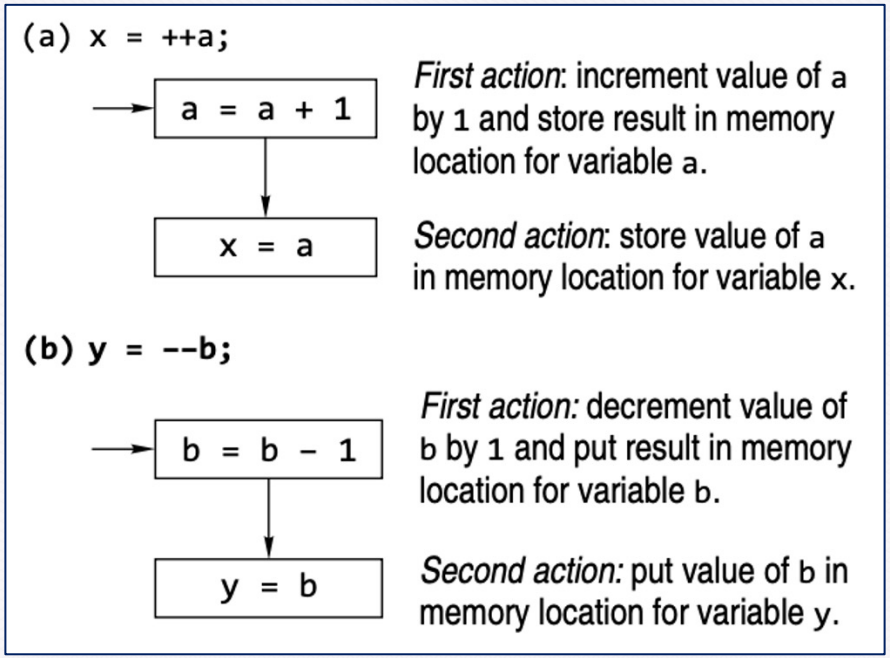


First action: put value of `b` in memory location for variable `y`.

Second action: decrement value of `b` by `1` and put result in memory location for variable `b`.

Arithmetic Operators (6/6)

Prefix operation: Syntax: **++Operand or --Operand**



Assignment Operators

Assignment	Symbol/Meaning
Simple Assignment	=
Compound Assignment	+=, -=, *=, /=, %=, &=, ^=, |=
Expression Assignment	A= 5+(b=8 + (c=2)) -4

Relational Operators

Operator	Action	Example
==	Equal	`5 == 5 /* gives 1 */`
!=	Not equal	`5 != 5 /* gives 0 */`
<	Less than	`5 < 5.5 /* gives 1 */`
<=	Less than or equal	`5 <= 5 /* gives 1 */`
>	Greater than	`5 > 5.5 /* gives 0 */`
>=	Greater than or equal	`6.3 >= 5 /* gives 1 */`

Relational & Equality Operators

Equality Operators

Relational Operators

Operator	Action	Example
==	Equal	`5 == 5 /* gives 1 */`
!=	Not equal	`5 != 5 /* gives 0 */`
<	Less than	`5 < 5.5 /* gives 1 */`
<=	Less than or equal	`5 <= 5 /* gives 1 */`
>	Greater than	`5 > 5.5 /* gives 0 */`
>=	Greater than or equal	`6.3 >= 5 /* gives 1 */`

Logical Operators

Operator	Action	Example	Result
!	Logical Negation	`!(5 == 5)`	`0`
&&	Logical AND	`5 < 6 && 6 < 6`	`0`
||	Logical OR	`5 < 6 || 6 < 5`	`1`

Self-study

- Explore more types of operators.

Bitwise	&	(Bitwise AND)
	|	(Bitwise OR)
	~	(Complement)
	^	(Exclusive OR)
	>>	(Right Shift)
	<<	(Left Shift)
Others	,	(Comma)
	*	(indirection),
	.	(membership operator)
	->	(membership operator)

Key Takeaways

You are now able to:

- ✓ Comprehend the overview of programming in C.
- ✓ Identify different data types used in programming.
- ✓ Recognize the concept of tokens in programming.
- ✓ Utilize variables and understand their usage.
- ✓ Apply various operators used in programming and comprehend their functions.

Reference

- Dey, P., & Ghosh, M. (2013). *Computer fundamentals and programming in C*.

Thank you !

Questions or Feedbacks?

Contact Me via:

leangsiv.han@cadt.edu.kh

@leangsiv

leangsivhan