BASIC PROGRAMMING

VARIABLES OPERATORS EXPRESSIONS

Le Thi Ngoc Tho, PhD
Faculty of Information Technology, HUTECH
ltn.tho@hutech.edu.vn

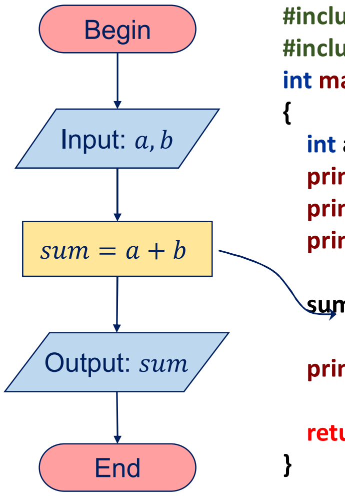
1

CONTENTS

- Program Structure
- Variable Types:
  - Names
  - Data Types & Sizes
  - Constants
- Operators
- Expressions

2

PROGRAM STRUCTURE



```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a, b, sum;
   printf("Input two integers:");
   printf("\n a = "); scanf("%d", &a);
   printf("\n b = "); scanf("%d", &b);

   sum = a + b;

   printf("\nSum is %d\n", sum);

   return 0;
}
```

C LANGUAGE

- C characters:
    - Letters: A…Z, a…z
    - Digits: 0…9
    - Arithmetic symbols: + - * / = > < ( )
    - Special symbols: . , " ' _ @ # $! ^ [ ] { } …
    - Space
- C keywords: words predefined by Compiler

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

VARIABLES

- Variables: Container to store the data,
- Variables are named for convenience.
- Characters Allowed :
  - Underscore(_)
  - Capital Letters (A – Z)
  - Small Letters (a – z)
  - Digits (0 – 9)
- Restrictions:
  - NO Blanks & Commas
  - NO Special Symbols
  - First character must be letter or underscore
  - Must NOT be keywords.
- Note: C is case sensitive

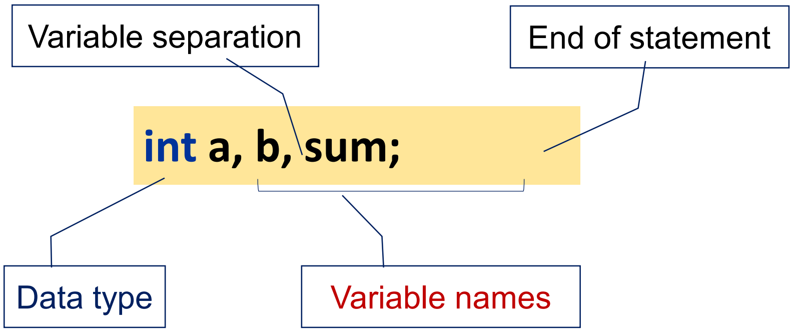
VARIABLES

- E.g.,
  - Correct: **a, A, x, x1, sum, _sum**
  - Incorrect: **1x, n!, al pha**
- Which of the following are correct variable names in C:

_a1	✓	**_1a**	
5linear	✗	**1_a**	
Delta		**a_1**	
x		**f(x)**	
sum_even		**Del ta**	
while		**WHILE**	

VARIABLES

- Declaration of variables: a statement



VARIABLES

- Data Types:
  - char: character
  - int: integer
  - float: single-precision floating point.
  - double: double-precision floating point.
- Qualifiers can be applied to basic types. unsigned and signed apply to integers:
  - signed int sh;
  - unsigned int counter;

VARIABLES

- Data Types: **char**
  - Use 1 byte to represent characters in ASCII table*.

Type	Domain
char	In ASCII table, character has value from - -128 to 127
unsigned char	Character has value from 0 to 255
signed char	Character has value from -128 to 127

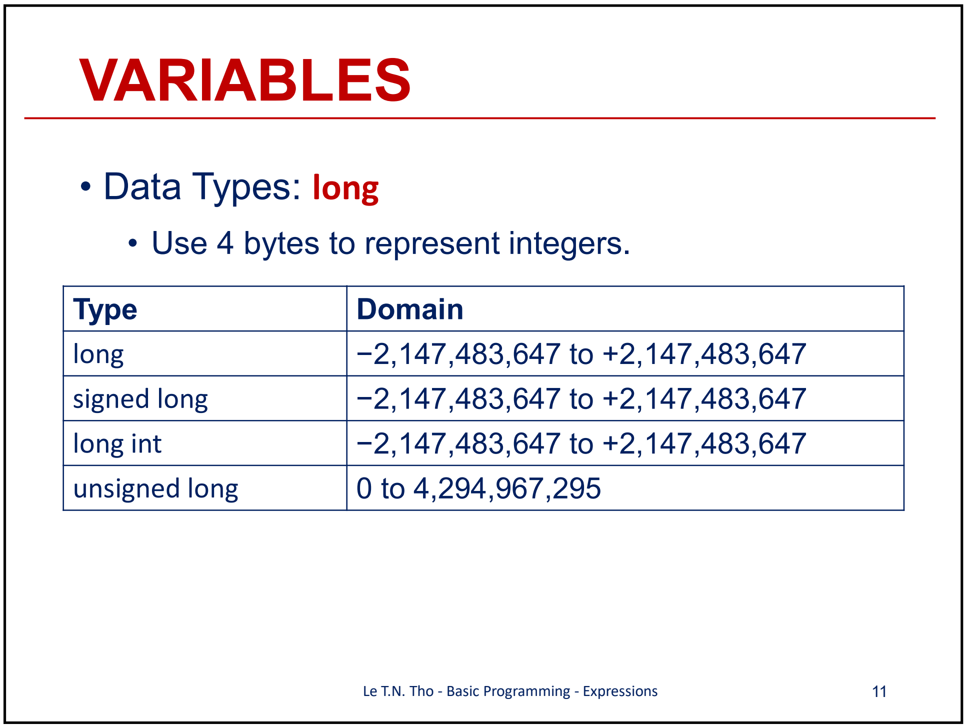
  - E.g., In ASCII table, character ‘A’ as value 65 and character ‘a’ has value 97.

(*): http://www.asciitable.com/

VARIABLES

- Data Types: **int**
  - Use 2 bytes to represent integers.

Type	Domain
int	-32,768 to +32,767
signed int	-32,768 to +32,767
unsigned int	0 to 65,535
short int	-32,768 to +32,767

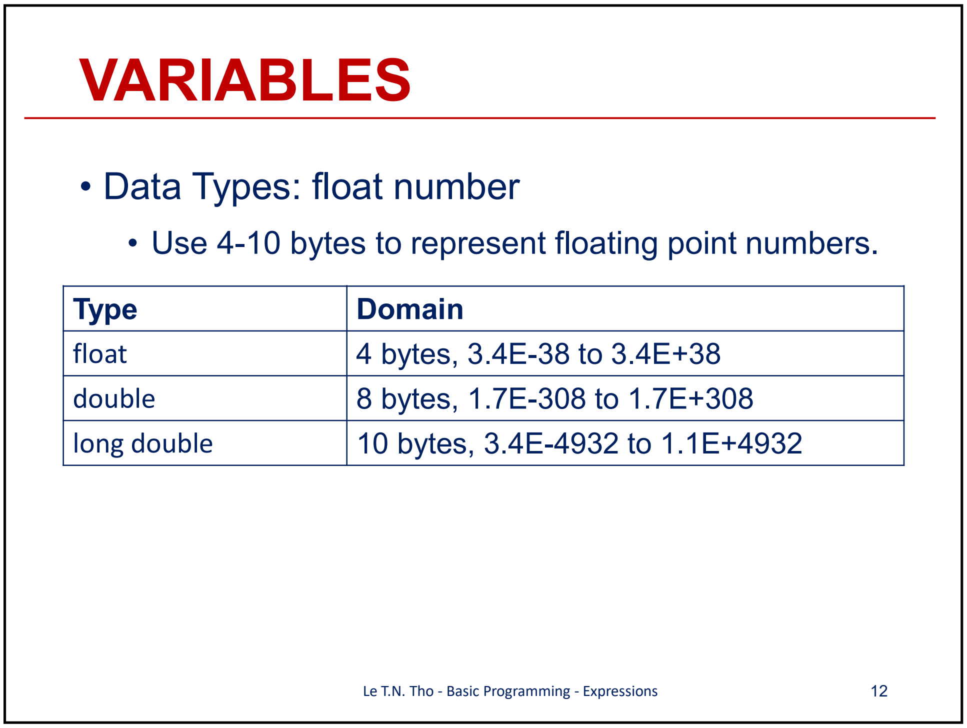


VARIABLES

- Data Types: **long**
  - Use 4 bytes to represent integers.

Type	Domain
long	−2,147,483,647 to +2,147,483,647
signed long	−2,147,483,647 to +2,147,483,647
long int	−2,147,483,647 to +2,147,483,647
unsigned long	0 to 4,294,967,295

11

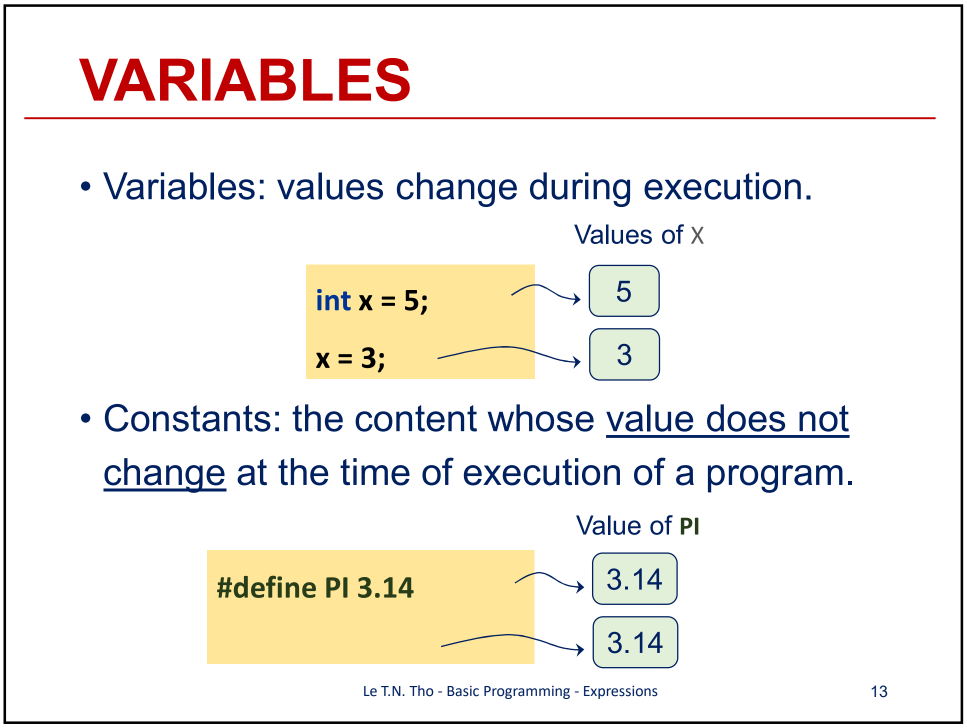


VARIABLES

- Data Types: float number
  - Use 4-10 bytes to represent floating point numbers.

Type	Domain
float	4 bytes, 3.4E-38 to 3.4E+38
double	8 bytes, 1.7E-308 to 1.7E+308
long double	10 bytes, 3.4E-4932 to 1.1E+4932

12



VARIABLES

- Variables: values change during execution.

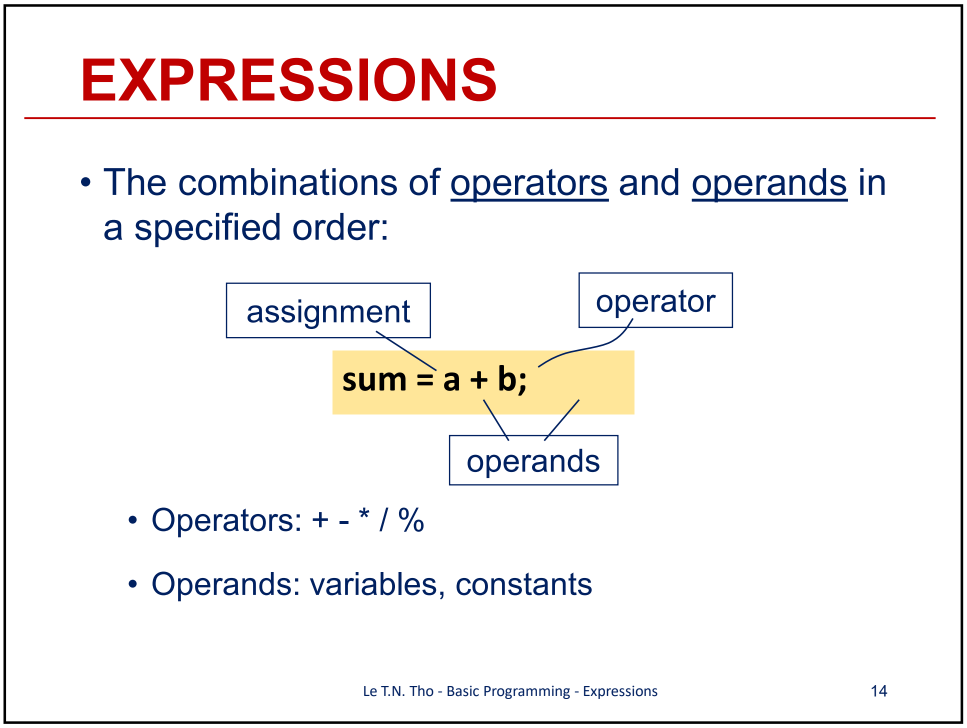
```
int x = 5;
x = 3;
```

Values of x
5
3

- Constants: the content whose value does not change at the time of execution of a program.

```
#define PI 3.14
```

Value of PI
3.14
3.14



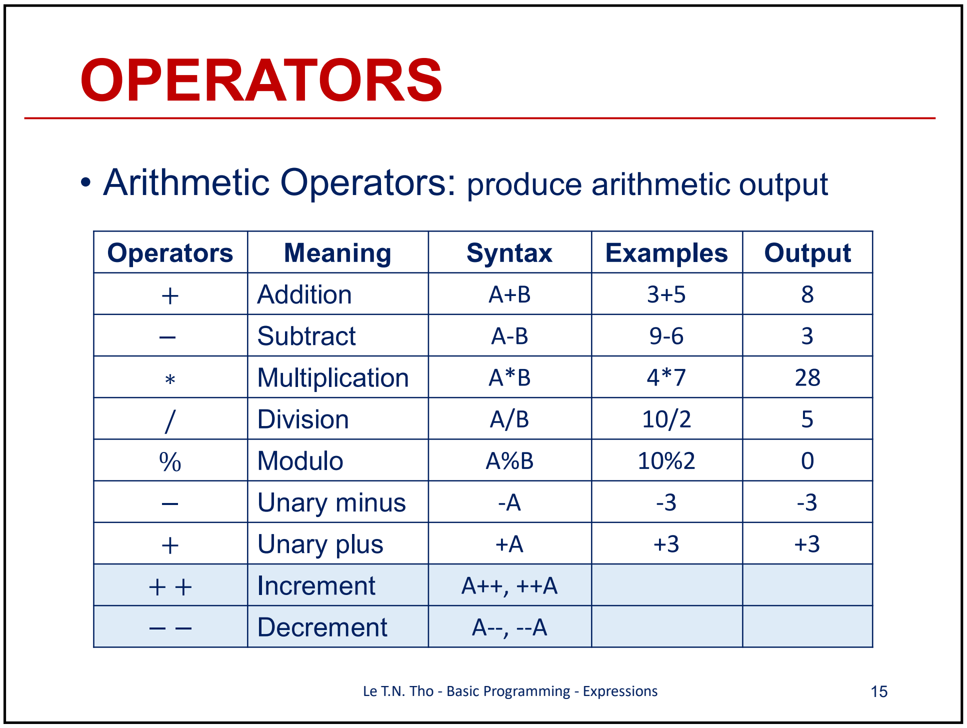
EXPRESSIONS

- The combinations of operators and operands in a specified order:

```
sum = a + b;
```

assignment
operator
operands

  - Operators: + - * / %
  - Operands: variables, constants

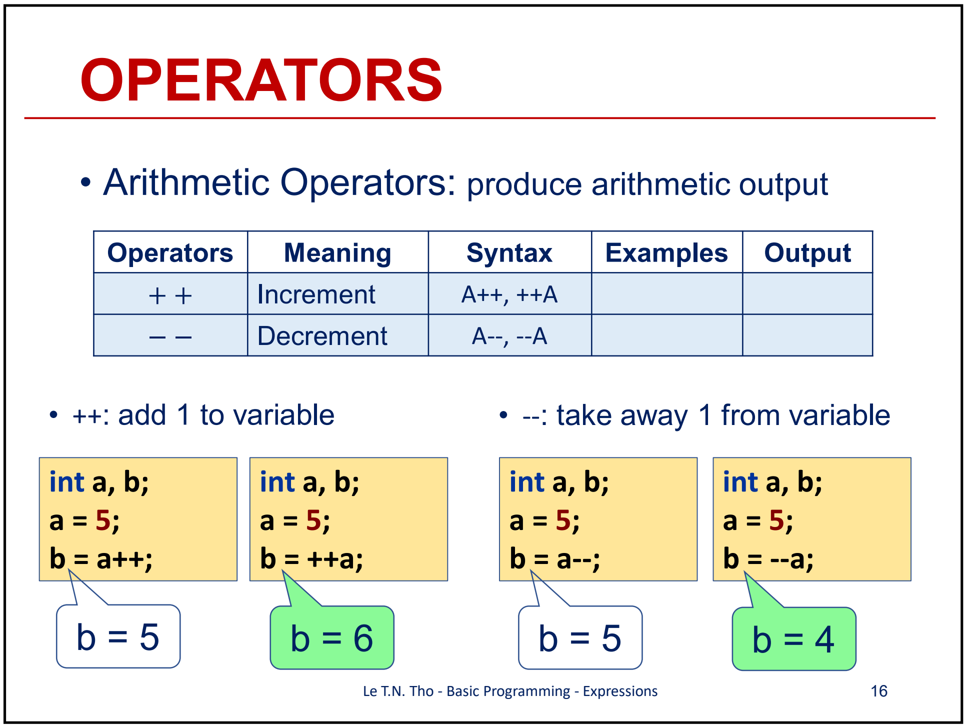


OPERATORS

- Arithmetic Operators: produce arithmetic output

Operators	Meaning	Syntax	Examples	Output
+	Addition	A+B	3+5	8
–	Subtract	A-B	9-6	3
*	Multiplication	A*B	4*7	28
/	Division	A/B	10/2	5
%	Modulo	A%B	10%2	0
–	Unary minus	-A	-3	-3
+	Unary plus	+A	+3	+3
+ +	Increment	A++, ++A		
– –	Decrement	A--, --A		

15



OPERATORS

- Arithmetic Operators: produce arithmetic output

Operators	Meaning	Syntax	Examples	Output
+ +	Increment	A++, ++A		
– –	Decrement	A--, --A		

- ++: add 1 to variable
- --: take away 1 from variable

```
int a, b;
a = 5;
b = a++;
```

b = 5

```
int a, b;
a = 5;
b = ++a;
```

b = 6

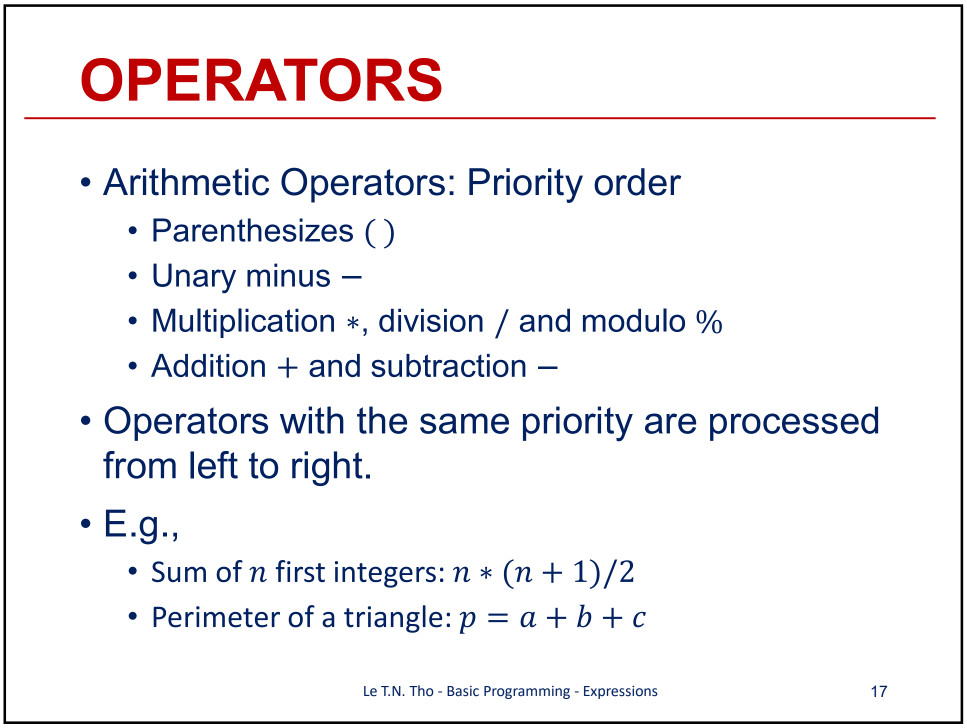
```
int a, b;
a = 5;
b = a--;
```

b = 5

```
int a, b;
a = 5;
b = --a;
```

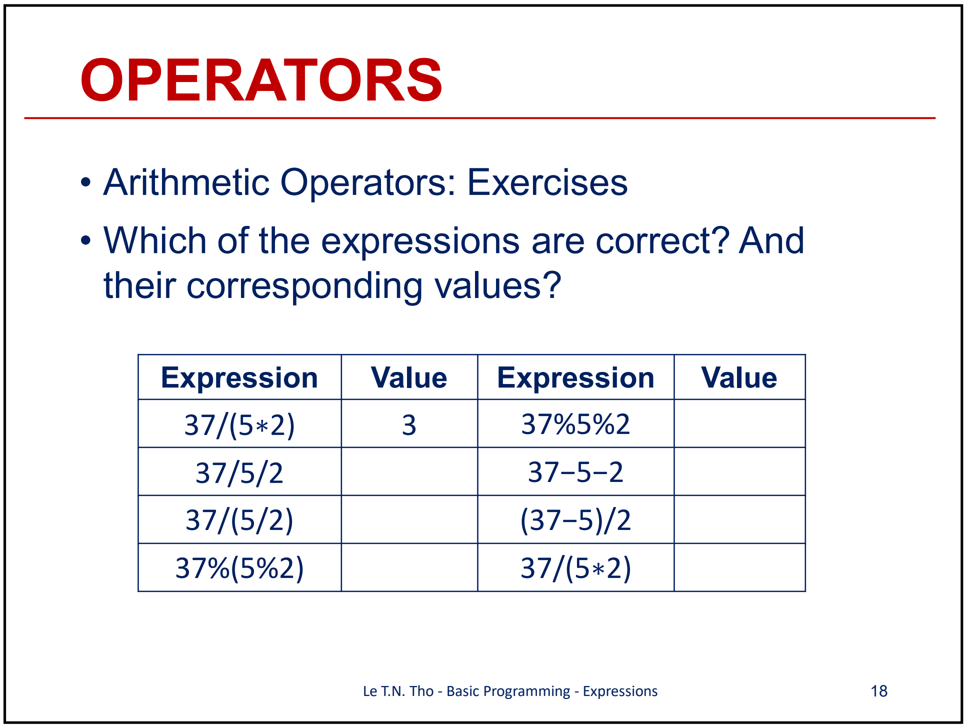
b = 4

16



OPERATORS

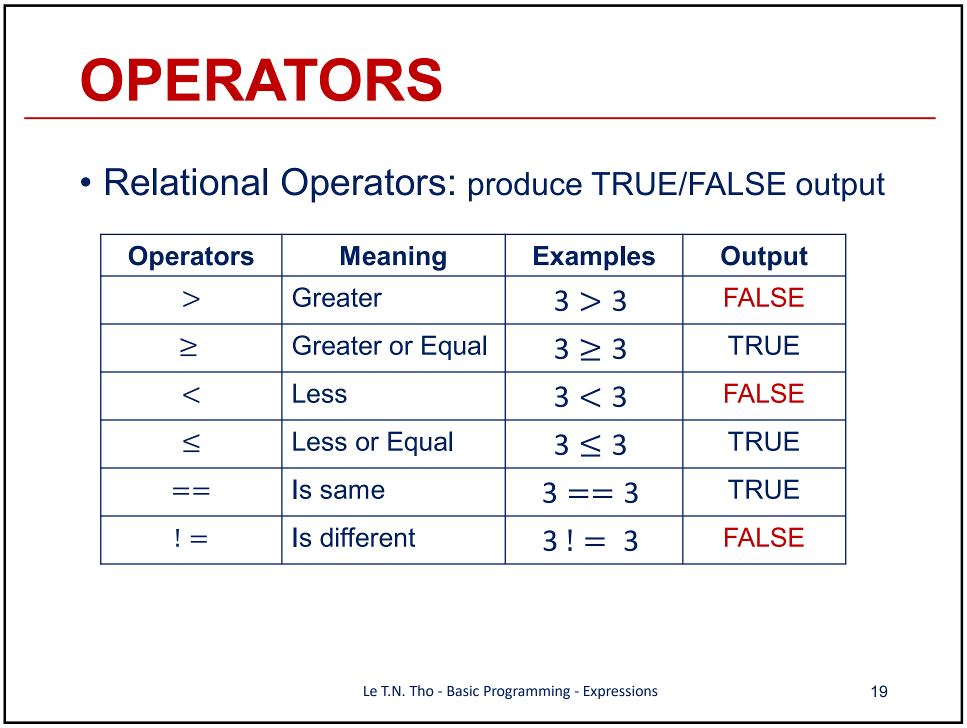
- Arithmetic Operators: Priority order
  - Parenthesizes ( )
  - Unary minus $-$
  - Multiplication $*$, division $/$ and modulo $\%$
  - Addition $+$ and subtraction $-$
- Operators with the same priority are processed from left to right.
- E.g.,
  - Sum of n first integers: $n * (n + 1)/2$
  - Perimeter of a triangle: $p = a + b + c$



OPERATORS

- Arithmetic Operators: Exercises
- Which of the expressions are correct? And their corresponding values?

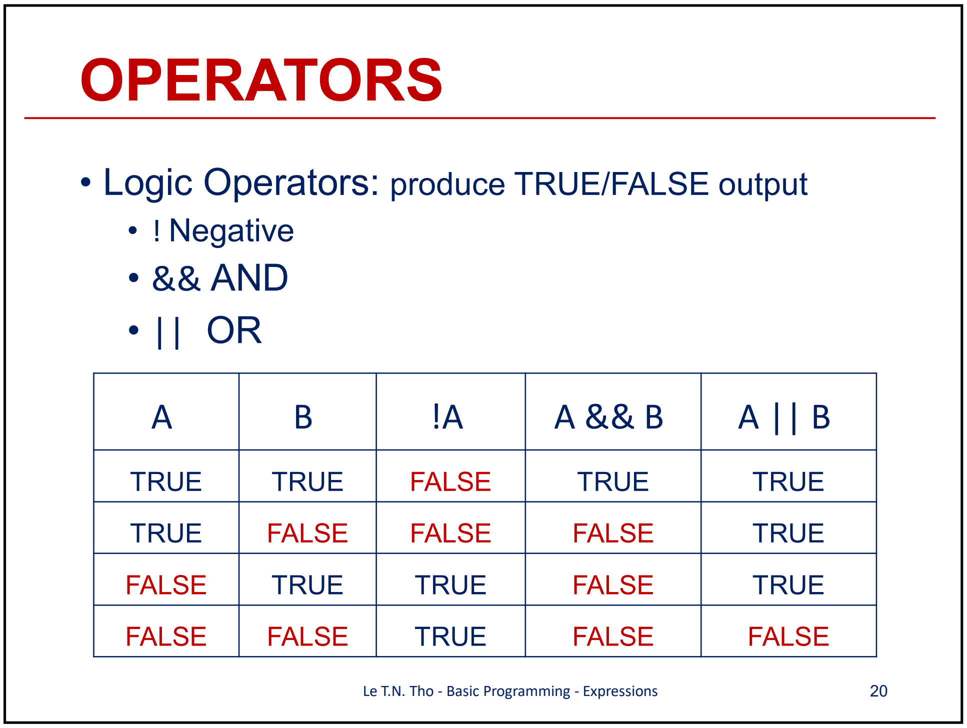
Expression	Value	Expression	Value
37/(5∗2)	3	37%5%2	
37/5/2		37−5−2	
37/(5/2)		(37−5)/2	
37%(5%2)		37/(5∗2)	



OPERATORS

- Relational Operators: produce TRUE/FALSE output

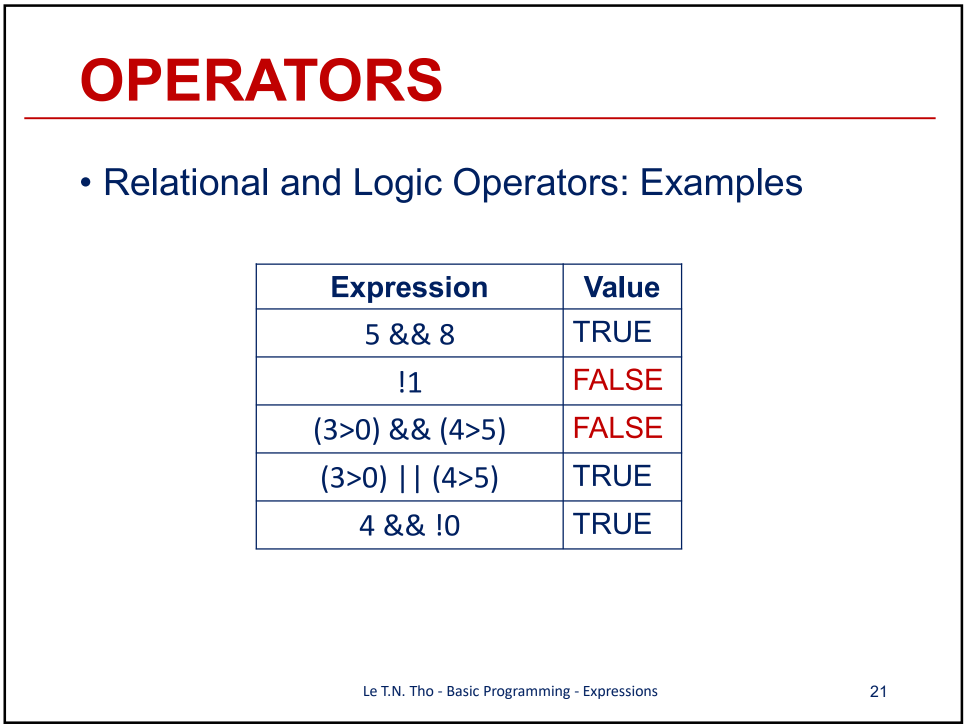
Operators	Meaning	Examples	Output
$>$	Greater	$3 > 3$	FALSE
$\geq$	Greater or Equal	$3 \geq 3$	TRUE
$<$	Less	$3 < 3$	FALSE
$\leq$	Less or Equal	$3 \leq 3$	TRUE
$==$	Is same	$3 == 3$	TRUE
$!=$	Is different	$3 != 3$	FALSE



OPERATORS

- Logic Operators: produce TRUE/FALSE output
  - ! Negative
  - && AND
  - | | OR

A	B	!A	A && B	A | | B
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE

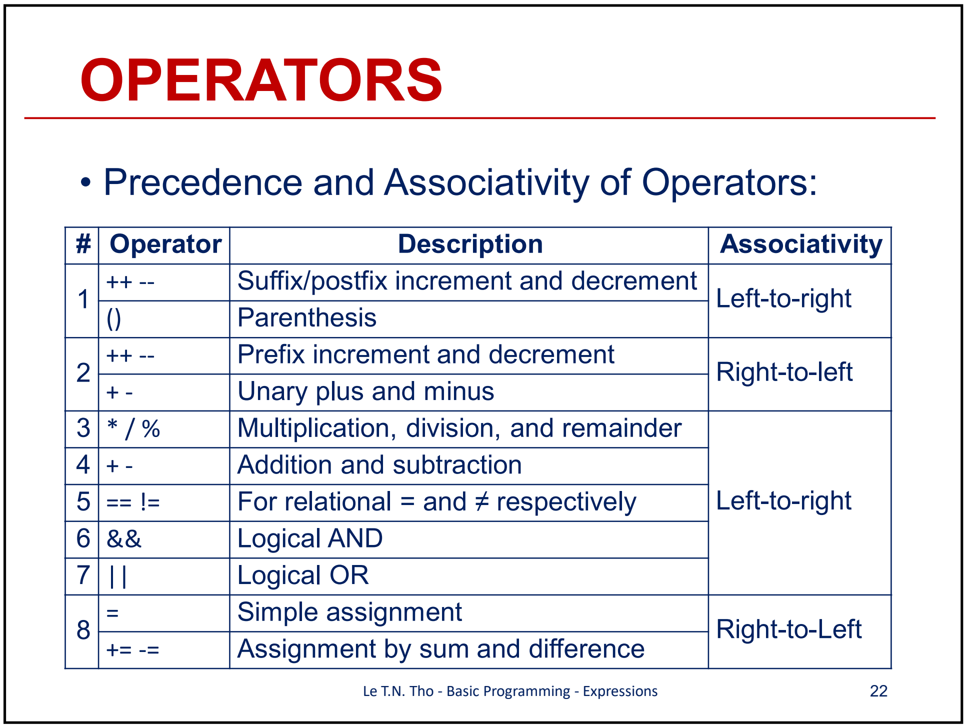


OPERATORS

- Relational and Logic Operators: Examples

Expression	Value
5 && 8	TRUE
!1	FALSE
(3>0) && (4>5)	FALSE
(3>0) || (4>5)	TRUE
4 && !0	TRUE

21

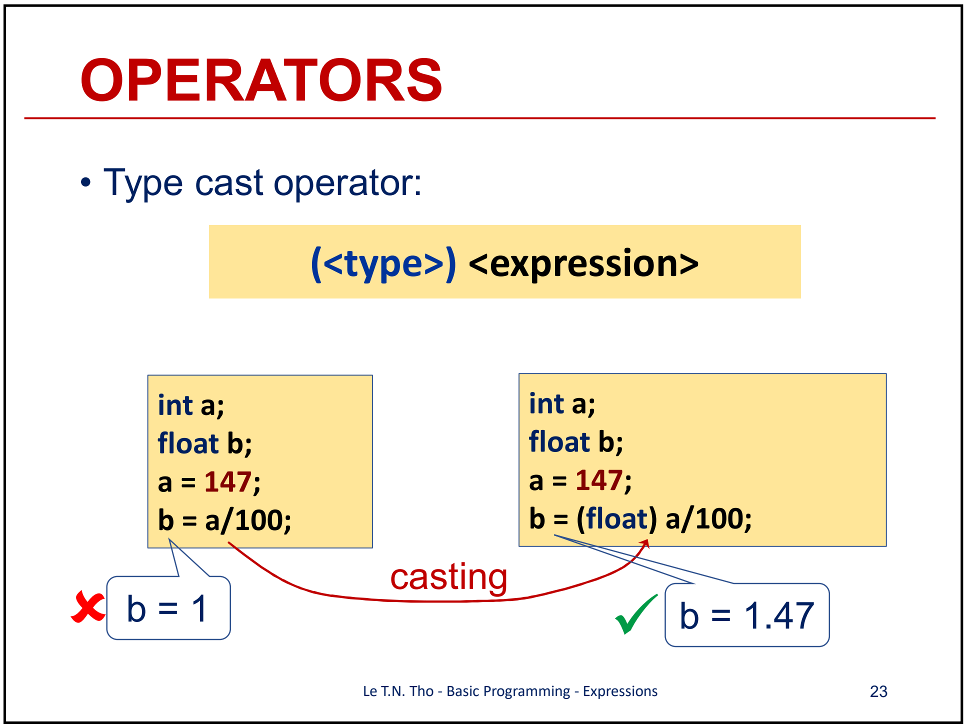


OPERATORS

- Precedence and Associativity of Operators:

#	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	()	Parenthesis	
2	++ --	Prefix increment and decrement	Right-to-left
	+ -	Unary plus and minus	
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	== !=	For relational = and ≠ respectively	
6	&&	Logical AND	
7	||	Logical OR	
8	=	Simple assignment	Right-to-Left
	+= -=	Assignment by sum and difference	

22



OPERATORS

- Type cast operator:

(<type>) <expression>

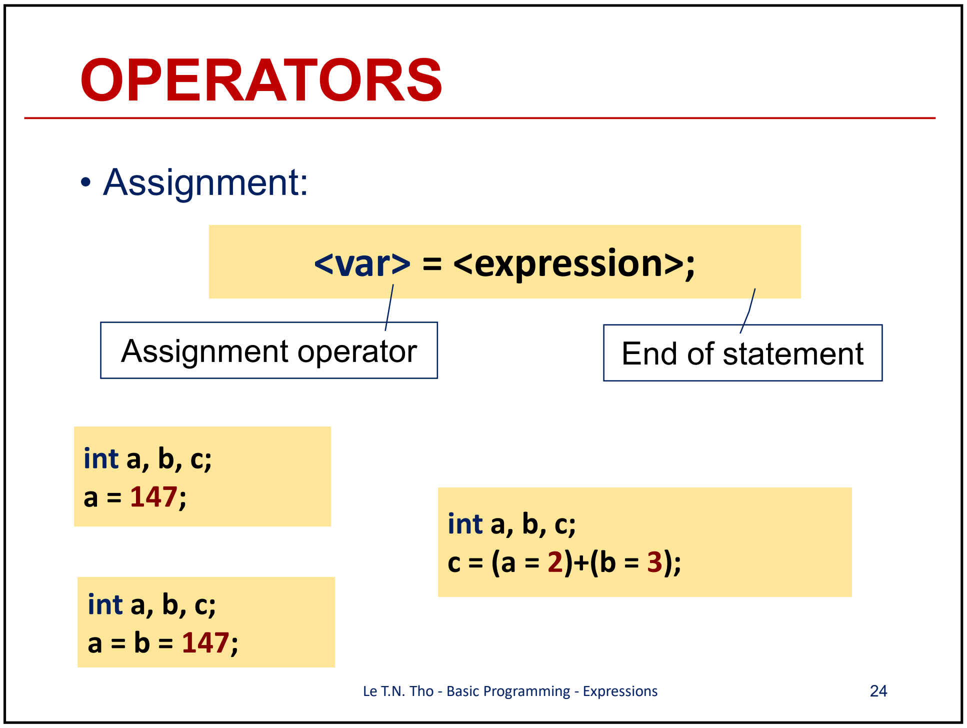
```
int a;
float b;
a = 147;
b = a/100;
```

```
int a;
float b;
a = 147;
b = (float) a/100;
```

✗ b = 1

casting

✓ b = 1.47



OPERATORS

- Assignment:

<var> = <expression>;

Assignment operator

End of statement

```
int a, b, c;
a = 147;
```

```
int a, b, c;
a = b = 147;
```

```
int a, b, c;
c = (a = 2)+(b = 3);
```

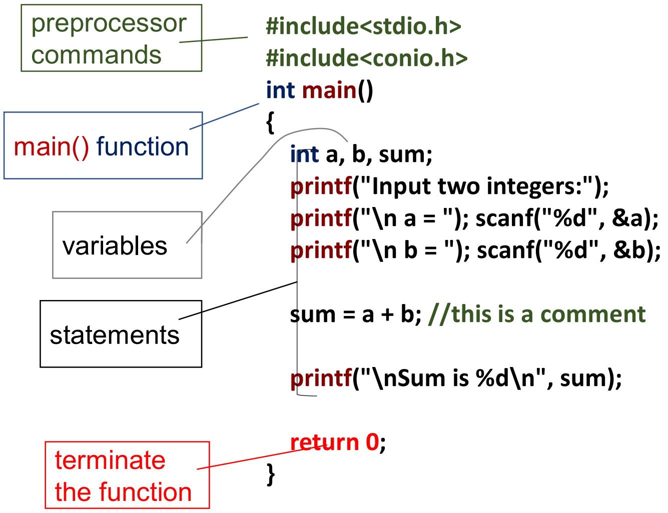
Conditional Expression

- Precedence and Associativity of Operators:

<exp1> ? <exp2> : <exp3>

- If exp1 is correct, get exp2 as the result, otherwise, get exp3 as the result.
- Meaning:
  - exp1: condition
  - exp2: true value
  - exp3: false value
- E.g.,
  - Get the greater number: (a>b) ? (a) : (b);
  - Get the absolute value: (a>0) ? (a) : (-a);

C PROGRAM STRUCTURE

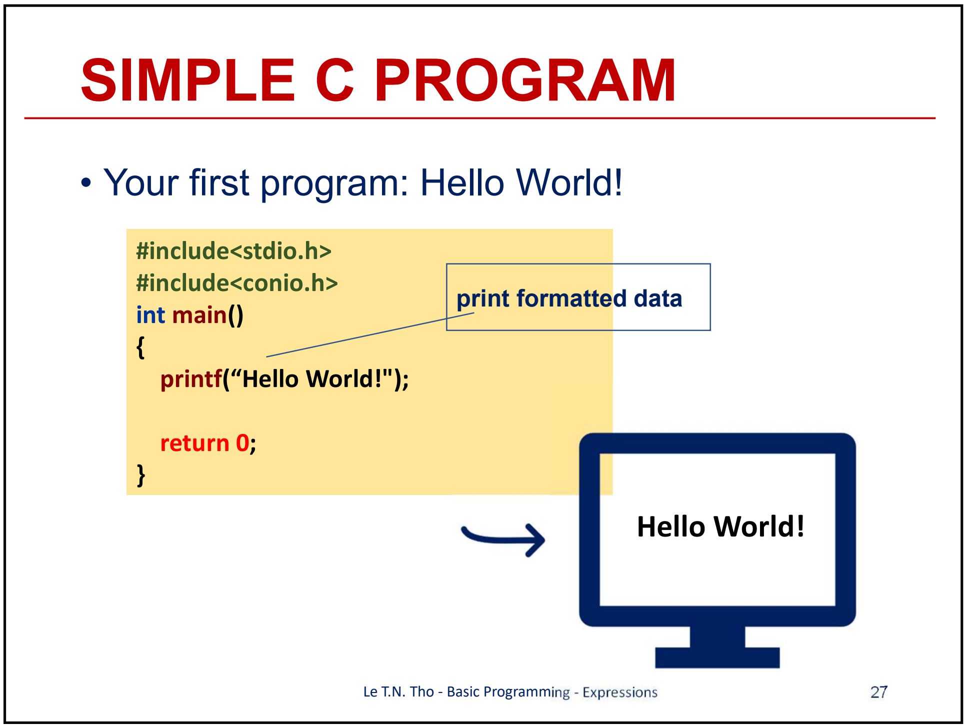


```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a, b, sum;
    printf("Input two integers:");
    printf("\n a = "); scanf("%d", &a);
    printf("\n b = "); scanf("%d", &b);

    sum = a + b; //this is a comment

    printf("\nSum is %d\n", sum);

    return 0;
}
```



SIMPLE C PROGRAM

- Your first program: Hello World!

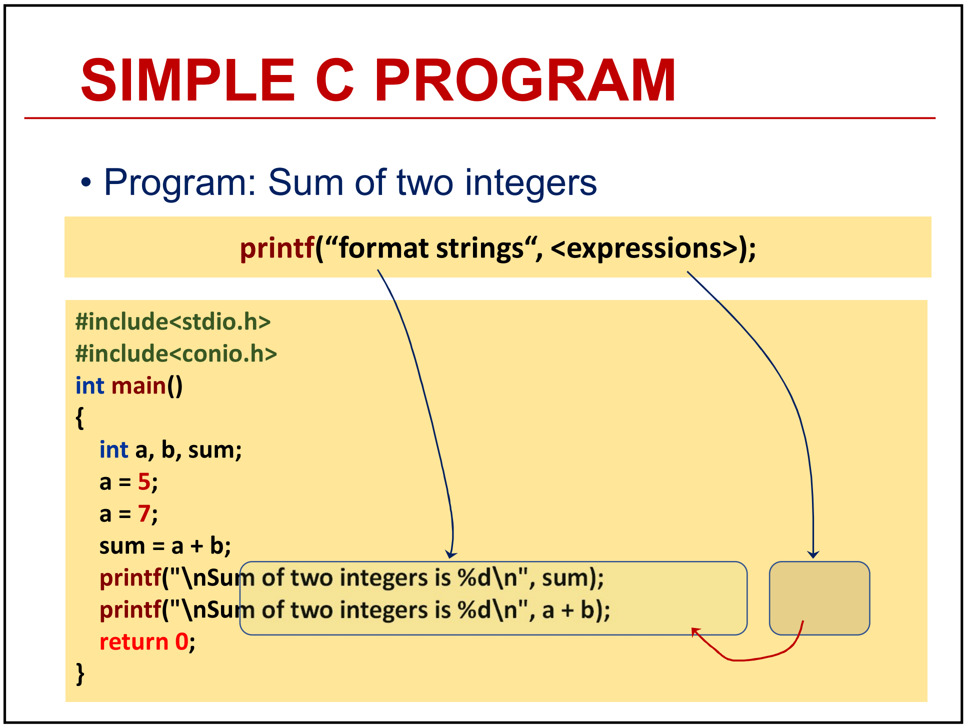
```
#include<stdio.h>
#include<conio.h>
int main()
{
   printf(“Hello World!");

   return 0;
}
```

print formatted data

Hello World!

27



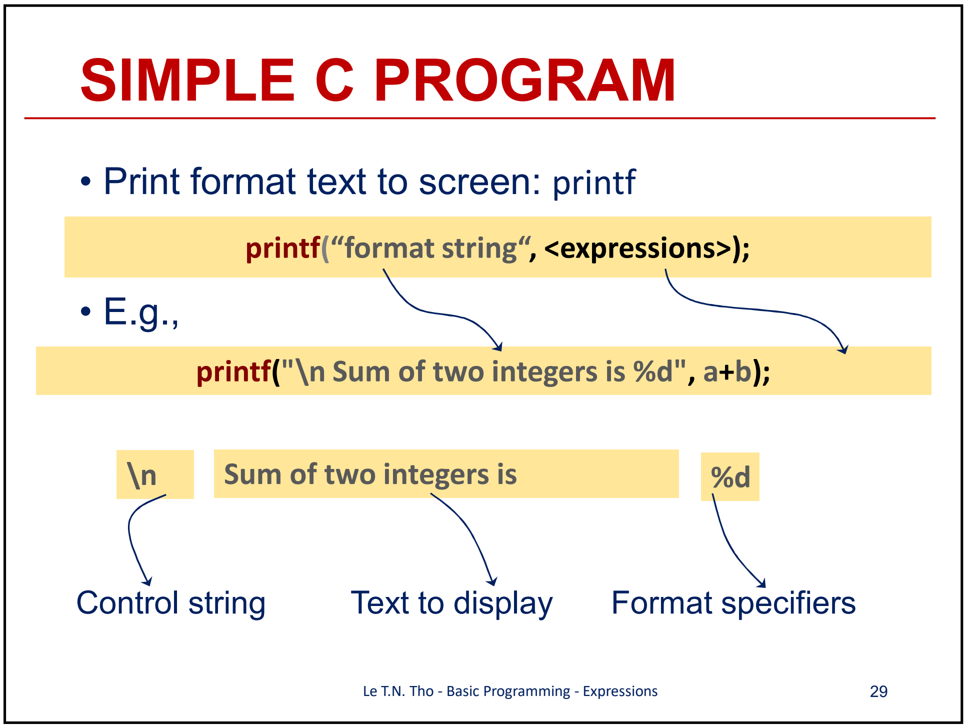
SIMPLE C PROGRAM

- Program: Sum of two integers

printf(“format strings“, <expressions>);

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a, b, sum;
   a = 5;
   a = 7;
   sum = a + b;
   printf("\nSum of two integers is %d\n", sum);
   printf("\nSum of two integers is %d\n", a + b);
   return 0;
}
```

28



SIMPLE C PROGRAM

- Print format text to screen: printf

```
printf(“format string“, <expressions>);
```

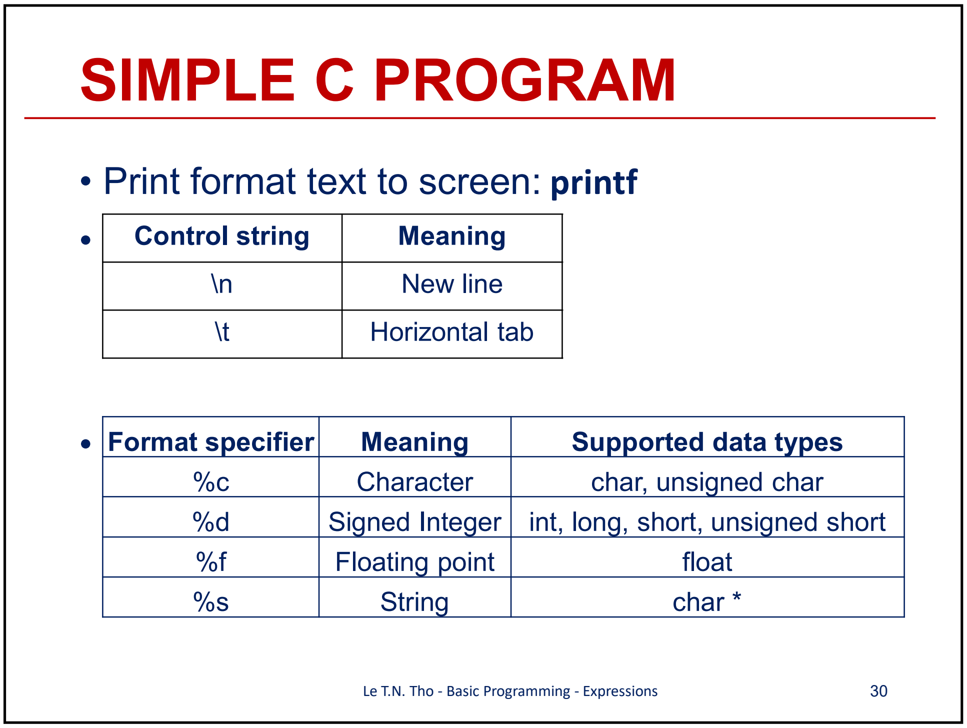
- E.g.,

```
printf("\n Sum of two integers is %d", a+b);
```

\n — Control string

Sum of two integers is — Text to display

%d — Format specifiers



SIMPLE C PROGRAM

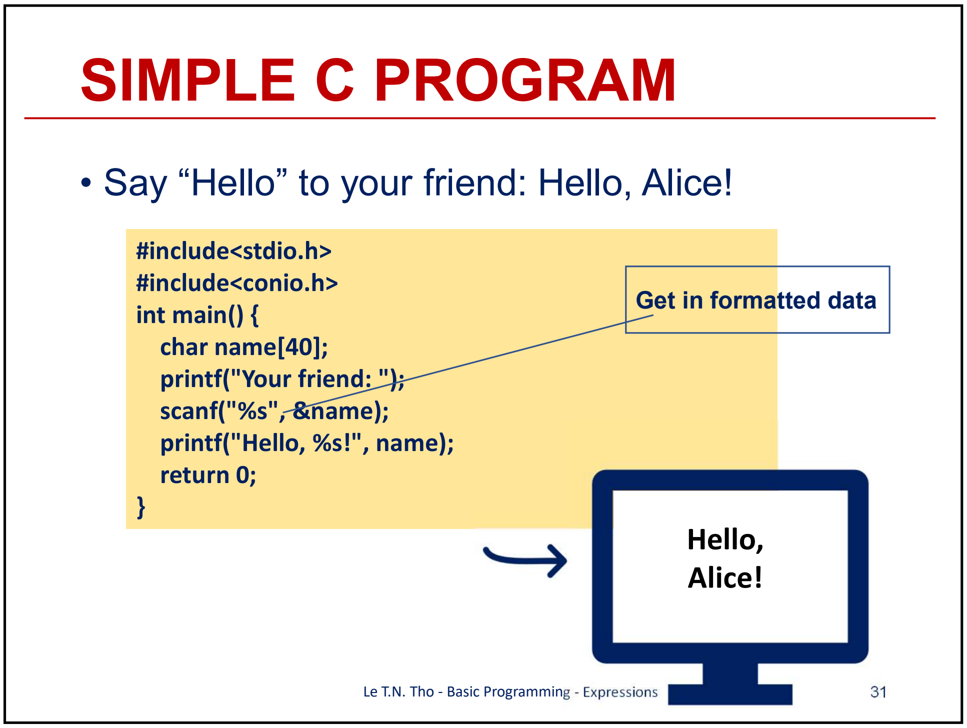
- Print format text to screen: **printf**

-

Control string	Meaning
\n	New line
\t	Horizontal tab

-

Format specifier	Meaning	Supported data types
%c	Character	char, unsigned char
%d	Signed Integer	int, long, short, unsigned short
%f	Floating point	float
%s	String	char *



SIMPLE C PROGRAM

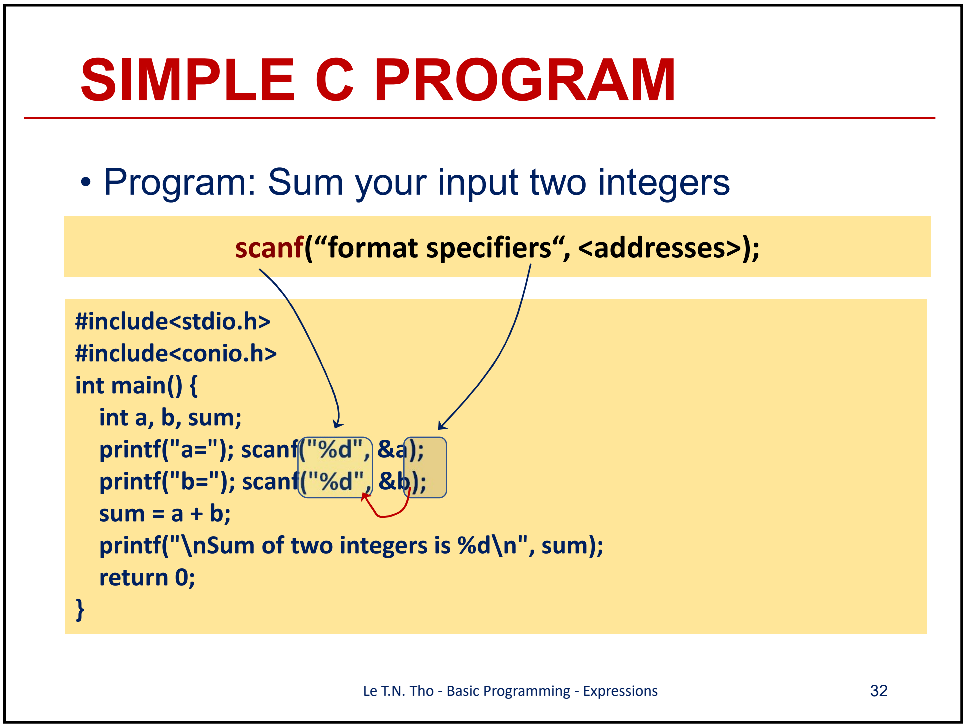
- Say “Hello” to your friend: Hello, Alice!

```
#include<stdio.h>
#include<conio.h>
int main() {
   char name[40];
   printf("Your friend: ");
   scanf("%s", &name);
   printf("Hello, %s!", name);
   return 0;
}
```

Get in formatted data

Hello,
Alice!

31



SIMPLE C PROGRAM

- Program: Sum your input two integers

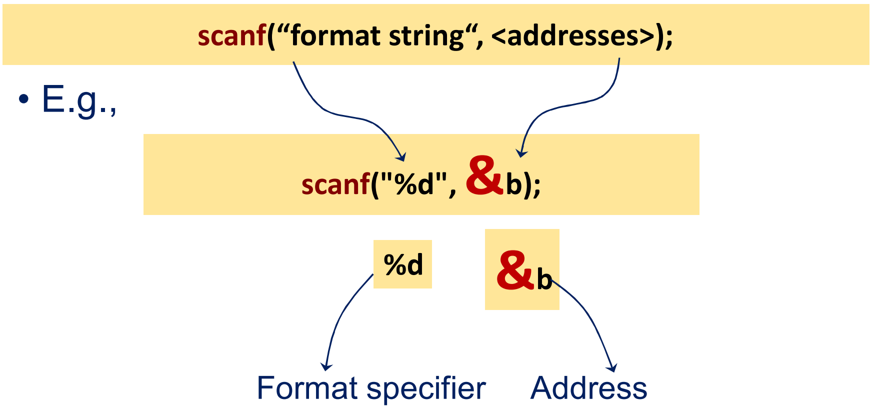
scanf(“format specifiers“, <addresses>);

```
#include<stdio.h>
#include<conio.h>
int main() {
   int a, b, sum;
   printf("a="); scanf("%d", &a);
   printf("b="); scanf("%d", &b);
   sum = a + b;
   printf("\nSum of two integers is %d\n", sum);
   return 0;
}
```

32

SIMPLE C PROGRAM

- Get format text from keyboard: scanf
- E.g.,



```
scanf("format string", <addresses>);
```

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

HOMEWORK

- Read the textbook: chapter 2
- Study how floating point numbers are represented.
  https://www.ntu.edu.sg/home/ehchua/programming/java/DataRepresentation.html

Any Questions?