C Identifiers

"Identifiers" or "symbols" are the names you supply for variables, types, functions, and label

Types of identifiers

- Internal identifier
- External identifier

Internal Identifier

- If the identifier is not used in the external linkage, then it is known as an internal identifier.

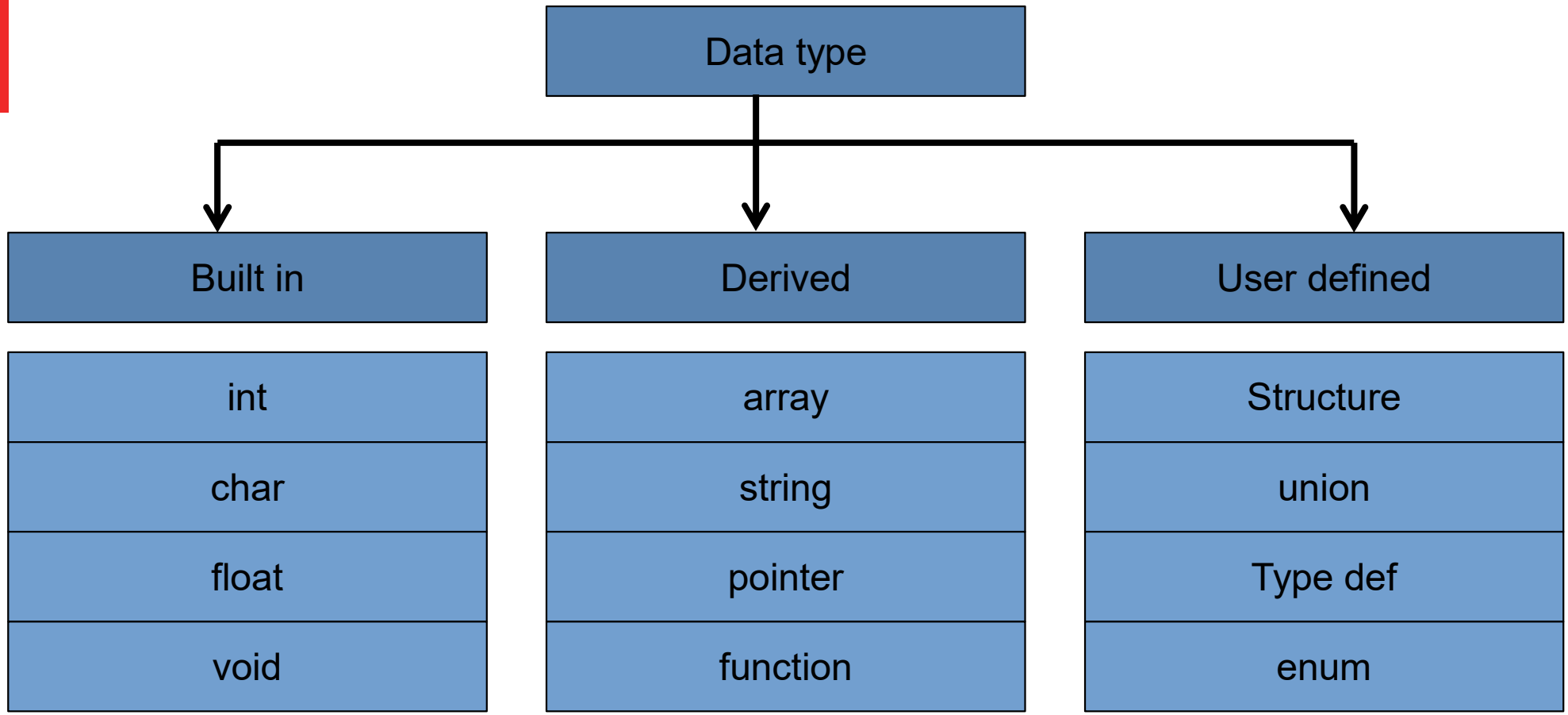
External Identifier

- If the identifier is used in the external linkage, then it is known as an external identifier. The

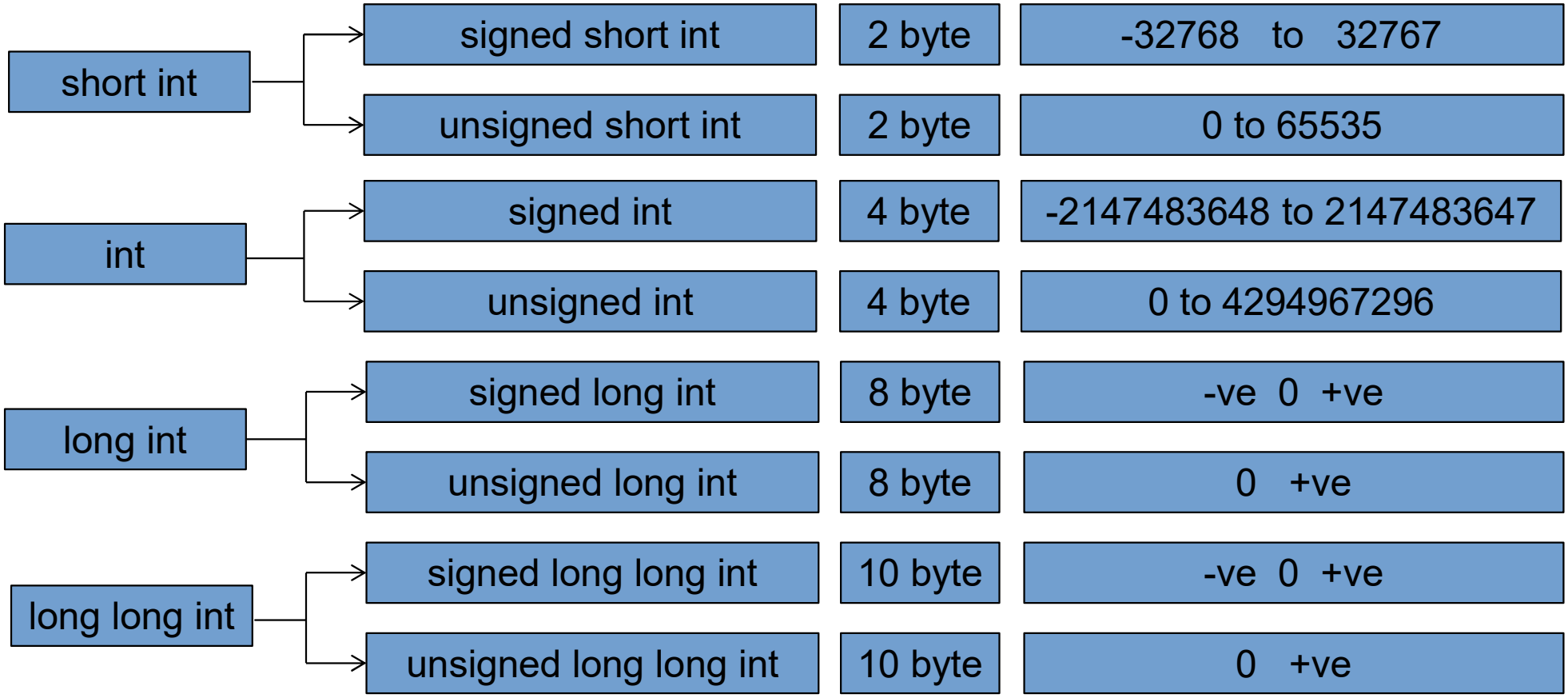
Rules for constructing C identifiers

- The first character of an identifier should be either an alphabet or an underscore, and ther
- It should not begin with any numerical digit.
- In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say tha
- Commas or blank spaces cannot be specified within an identifier.
- Keywords cannot be represented as an identifier.
- The length of the identifiers should not be more than 31 characters.
- Identifiers should be written in such a way that it is meaningful, short, and easy to read

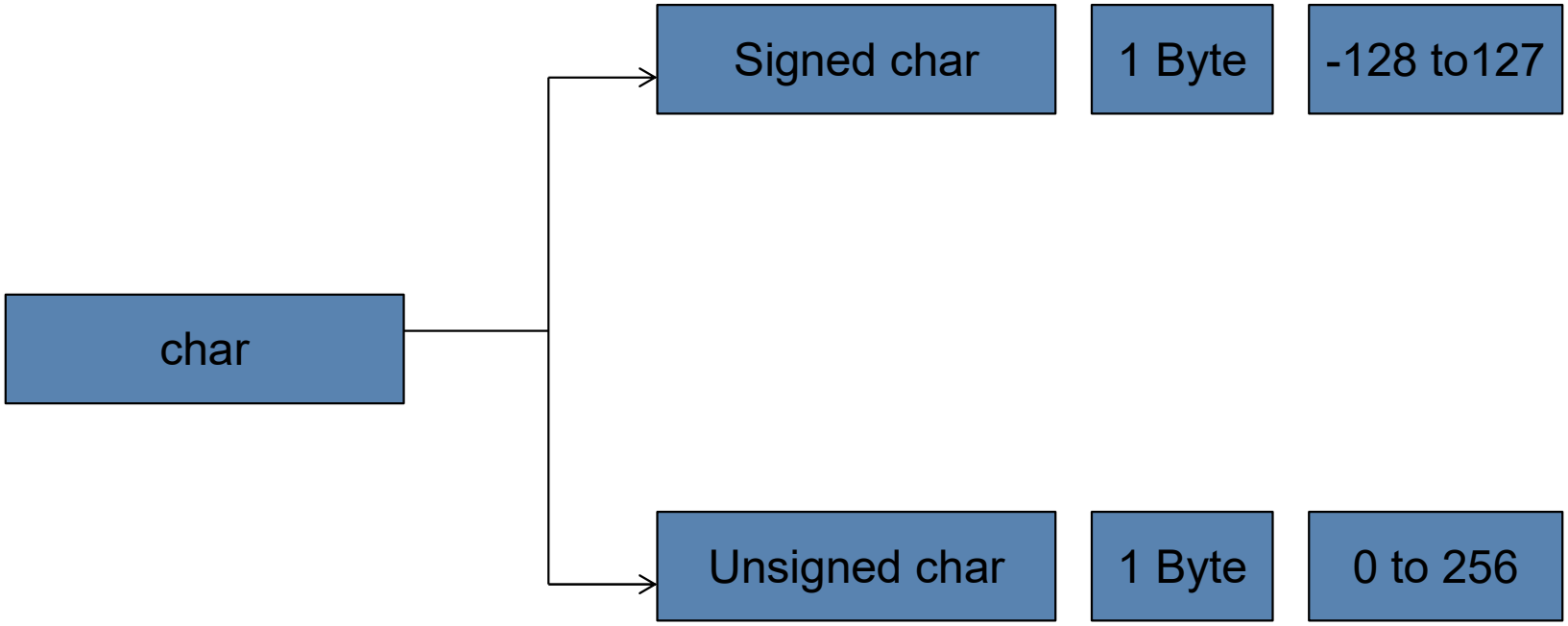
Data types


Data type
Built in
Derived
User defined
int
char
float
void
array
string
pointer
function
Structure
union
Type def
enum

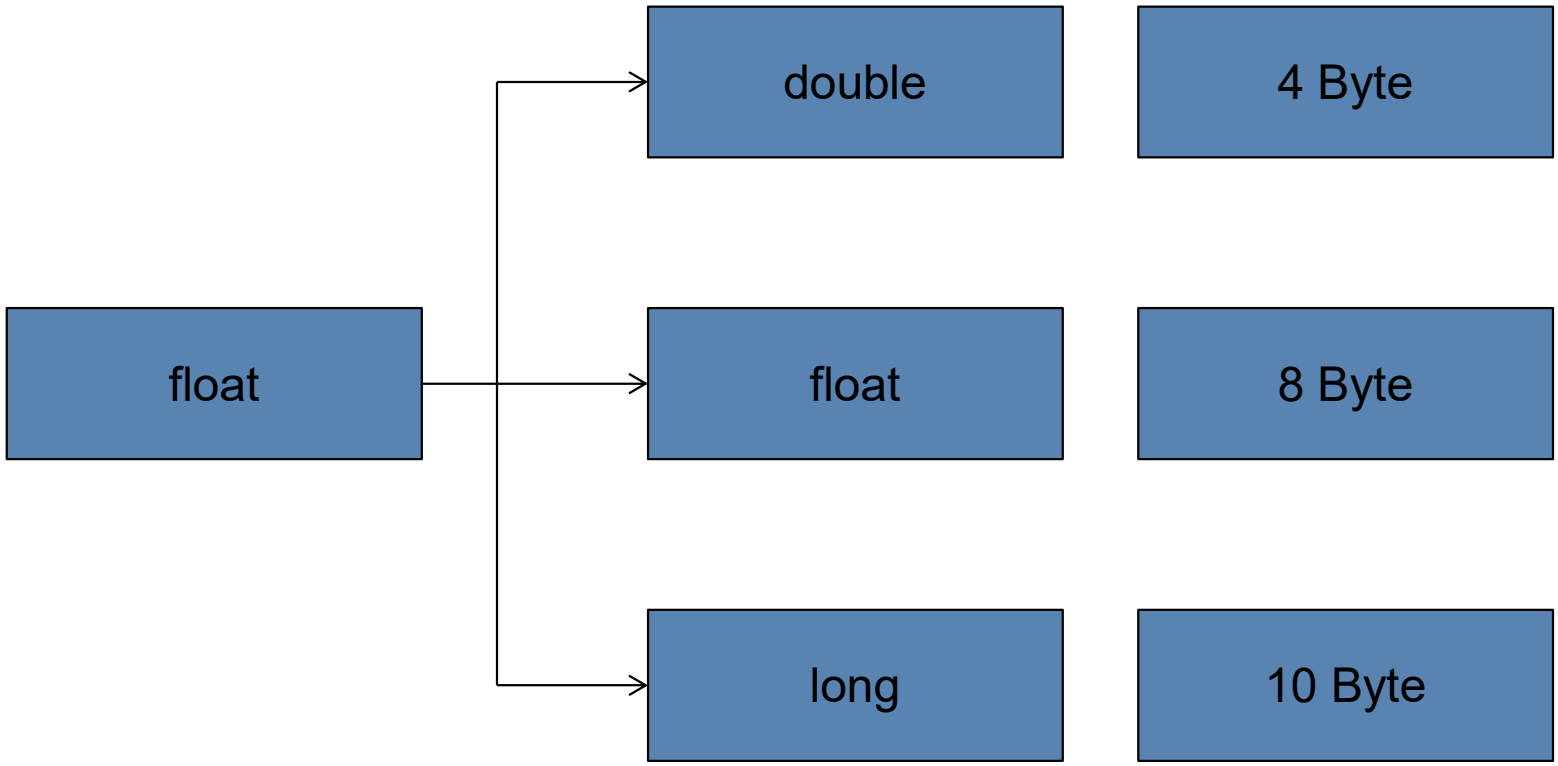
Integer


short int
signed short int
2 byte
-32768 to 32767
unsigned short int
2 byte
0 to 65535
int
signed int
4 byte
-2147483648 to 2147483647
unsigned int
4 byte
0 to 4294967296
long int
signed long int
8 byte
-ve 0 +ve
unsigned long int
8 byte
0 +ve
long long int
signed long long int
10 byte
-ve 0 +ve
unsigned long long int
10 byte
0 +ve

char


char
Signed char
1 Byte
-128 to127
Unsigned char
1 Byte
0 to 256

Float



Format specifiers

The format specifier is used during input and output. It is a way to tell the compiler what type

The printf() is a library function to send formatted output to the screen. The function p

printf("format string",argument_list);

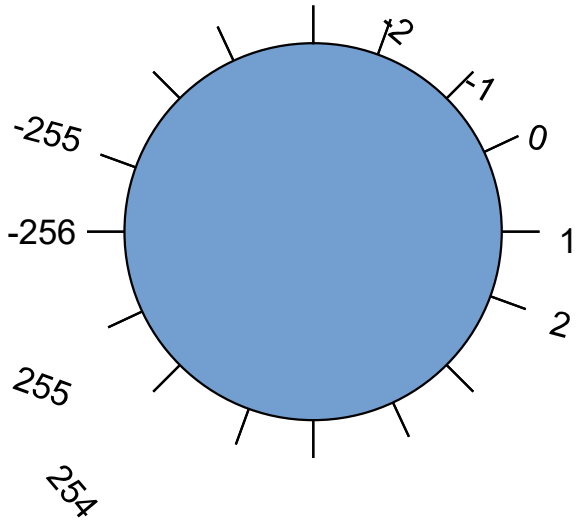
The scanf() function is used for input. It reads the input data from the console

scanf("format string",argument_list);

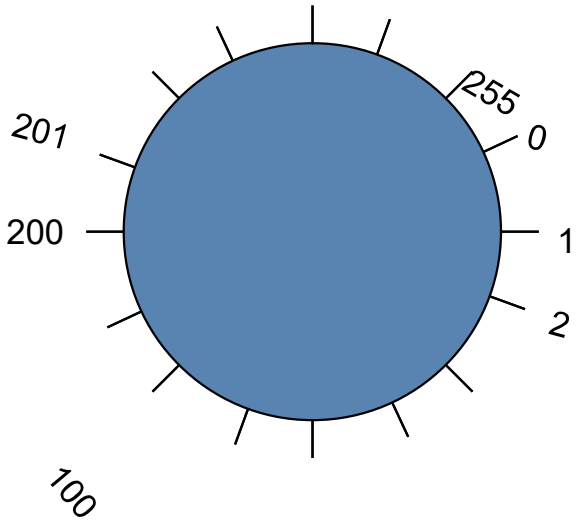
%d	int/signed int
%c	char
%f	float
%lf	double
%Lf	long double
%s	string
%u	unsigned int
%li	long int
%lli	long long int
%lu	unsigned long int

%llu	unsigned long long int
%hi	short int
%hu	Unsigned short
%x%p	Pointer/address
%e	scientific notation
%o	octal
%x	Hexadecimals

Relation between Format specifiers &Data type



Signed char



unsigned char

Operators in c

Arithmetic operators

Addition operator	(+)
Subtraction operator	(-)
Division operator	(/)
Multiplication operator	(*)
Modular operator	(%)

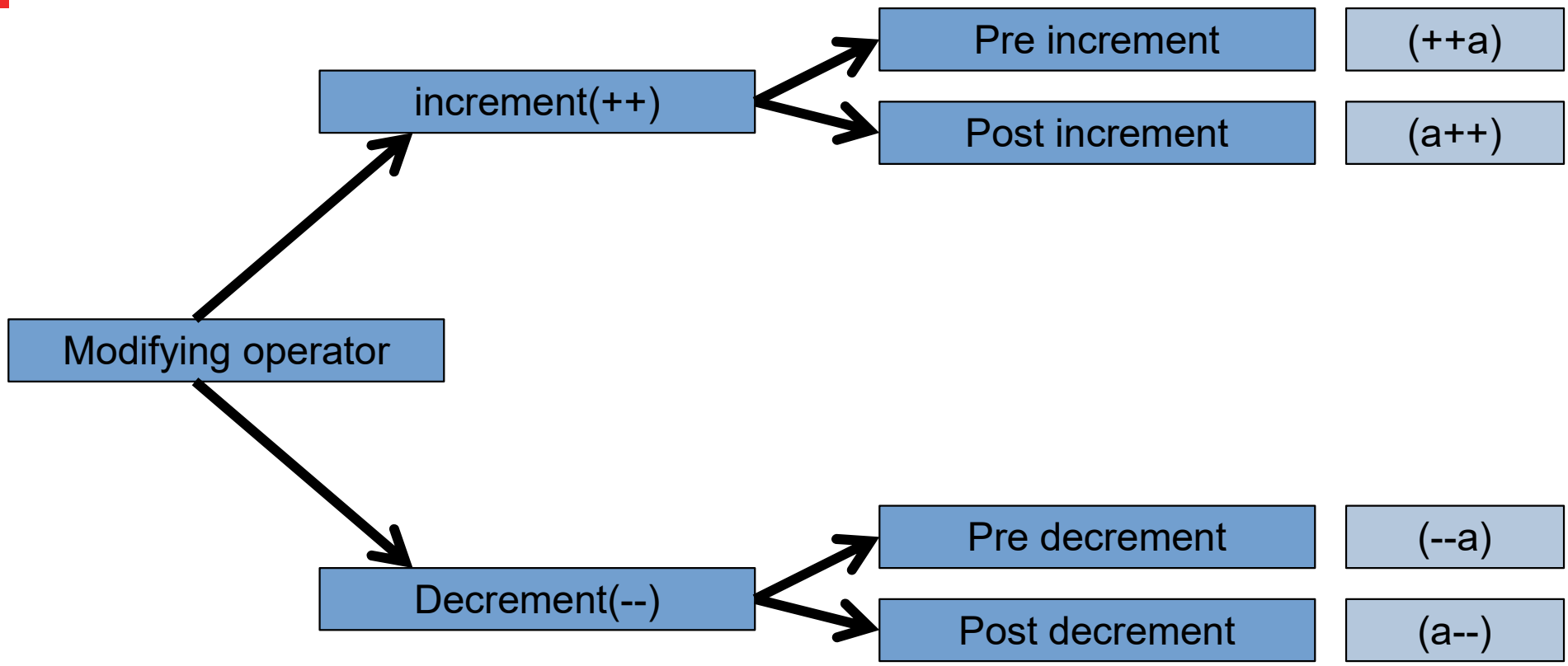
Relation operators

equal to	(==)
greater than or equal to	(>=)
greater than	(>)
less than or equal to	(<=)
less than	(<)
not equal to	(!=)

Assignment operators

Simple assignment	(=)
Multiplication assignment	(*=)
Division assignment	(/=)
Remainder assignment	(%=)
Addition assignment	(+=)
Subtraction assignment	(-=)
Left-shift assignment	(<<=)
Right-shift assignment	(>>=)
Bitwise-AND assignment	(&=)
Bitwise-exclusive-OR assignment	(^=)
Bitwise-inclusive-OR assignment	(|=)

Modifying operator



→Arithmetic operators are Unary operators

logical operators

AND	(&&)	True only if both operands are true
OR	(||)	True if either operand is true
NOT	(~)	Changes true to false and false to true

Bitwise operators

(&)	Bitwise AND operator
(|)	Bitwise OR operator
(^)	Bitwise exclusive OR operator
(~)	One's complement operator (unary operator)
(<<)	Left shift operator
(>>)	Right shift operator

Comma operator

The comma sign is used for mainly two different purposes in the C language – as an operato

For example

int x, y, z;

In the statement mentioned above, the comma acts as a separator and informs the compil

p = 40, 50, 60, 70, 80, 90;

q = (40, 50, 60, 70, 80, 90);

In the very first statement mentioned above, the value assigned to the variable p will be equ

In the case of the second statement, the value of q will be equivalent to 90. It is because the