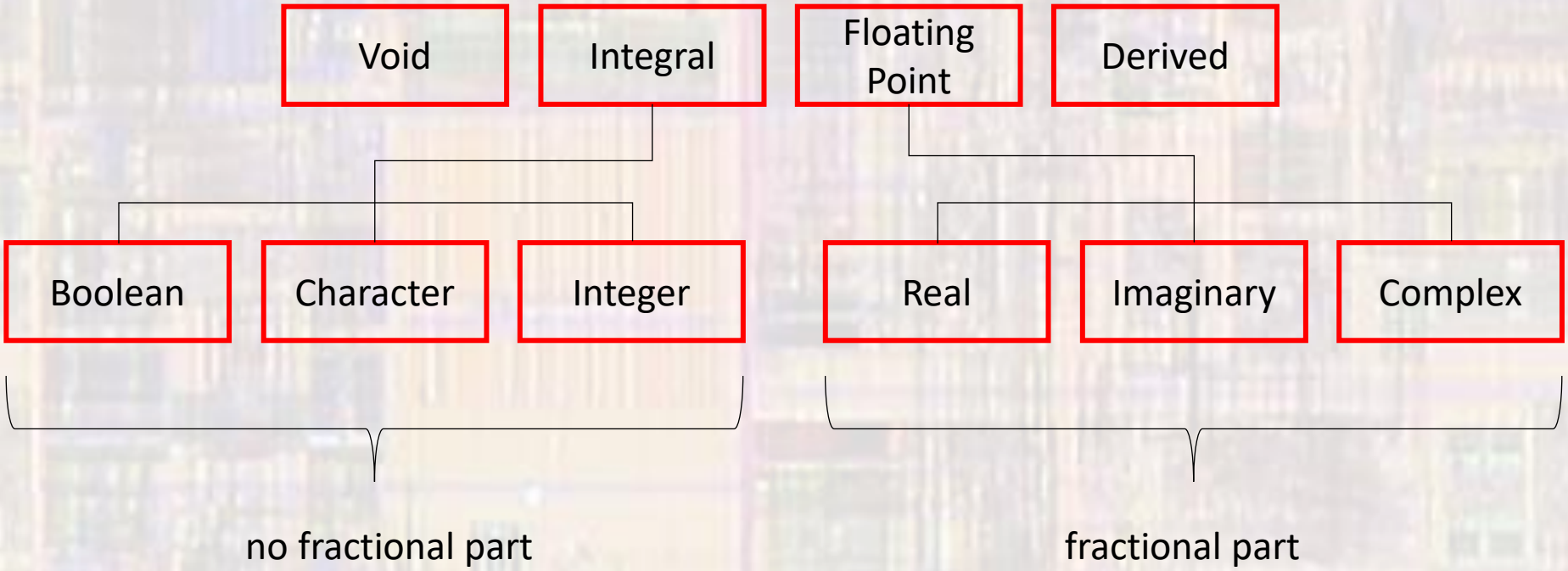
Type Conversion

Last updated 9/4/21

Type Conversion

- Type conversion – changing a value from one type to another



Type Conversion

- Suppose we had the following expression:

voltage * current

where: voltage was a variable of type int (5)
current was a variable of type float (2.5)

what would the expression evaluate to?

Type Conversion

- Implicit Type Conversion
  - Type conversions done automatically by the compiler
  - Each type has a RANK

bool < char < short < int < long < long long < float < double < long double

complex types match the floating types

Type Conversion

- Implicit Type Conversion

  int * float → float

  1) int expression promoted to float
  2) multiplication
  3) result is of type float

  char + long int → long int

  1) char expression promoted to long int
  2) addition
  3) result is of type long int

Type Conversion

- Implicit Type Conversion
  - No Side Effect

int days;
float rate;

days * rate → float

days remains an int

No variable types are changed in this process

days temporarily promoted to a float
Multiplication is performed – resulting in a float

Type Conversion

- Explicit Type Conversion
  - Cast or casting
  - Force a type conversion on an expression
  - Use the unary operator “type cast”

(desired_type) var

Precedence	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	()	Function call	
	[]	Array subscripting	
	.	Structure and union member access	
	->	Structure and union member access through pointer	
	(*type*){*list*}	Compound literal(C99)	
2	++ --	Prefix increment and decrement	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	Type cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of	
	_Alignof	Alignment requirement(C11)	
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	

Type Conversion

- Explicit Type Conversion

```
int a;
int b;
a = 5;
b = 2;

a/b              ?

(float) a / b    ?

a / (float) b    ?

(float) (a/b)    ?
```

Type Conversion

- Explicit Type Conversion

```
int a;
int b;
a = 5;
b = 2;
```

a/b → 2

(float) a / b → 5.0 / 2 → 5.0 / 2.0 → 2.5

a / (float) b → 5 / 2.0 → 5.0 / 2.0 → 2.5

implicit type conversion

(float) (a/b) → (float) (5/2) → (float) 2 → 2.0

Type Conversion

- Explicit Type Conversion
  - No Side effect

```
int a;
int b;
a = 5;
b = 2;
```

No variable types are changed in this process

a/b → 2

a temporarily promoted to a float
Division is performed – resulting in a float

(float) a / b → 2.5

a / (float) b → 2.5

(float) (a/b) → 2.0

a = 5

b = 2

a = 5, b = 2

all still type int

Type Conversion

- Assignment Type Conversion
  - Assignment operator =
    - value – evaluate right side expression
    - side effect – left side is assigned the value

```
int a;
int b;
int c;
a = 5;
b = 6;
c = 7;
```

a = b + c;

evaluate right side (b + c) → value is 13

side effect – a is assigned the value 13

Type Conversion

- Assignment Type Conversion
  - Regardless of any implicit or explicit type conversions the assignment operator side effect cannot change the type of a variable

```
int a;
float b;
int c;
b = 12.3;
c = 5;
a = b / c;
```

c is promoted to type float

right side is evaluated 12.3 / 5.0 → 2.46

value is demoted to match the receiving variable (side effect): a = 2

Type Conversion

- Assignment Type Conversion
  - Regardless of any implicit or explicit type conversions the assignment operator side effect cannot change the type of a variable

```
int a;
int b;
float c;
a = 5;
b = 7;
c = a + b;
```

right side is evaluated → 12 of type int

value is promoted to match the receiving variable (side effect): c = 12.0