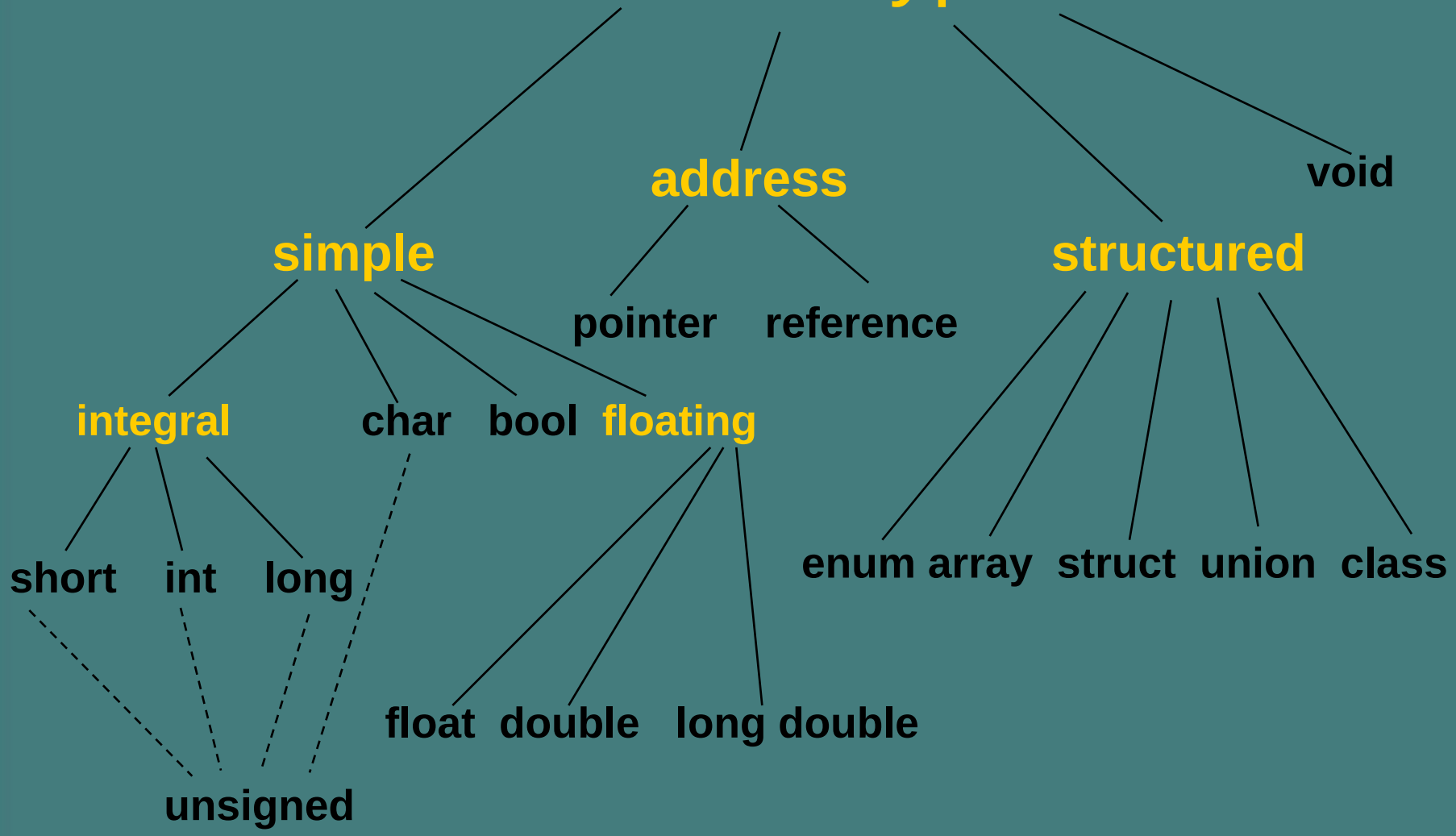
Chapter 2

Numeric Types and Expressions

Chapter 2 Topics

- Overview of C++ Data Types
- C++ Identifiers
- Declarations for Numeric Types
- Arithmetic Expressions
- Assignment Operator

C++ Data Types


address
void
simple
structured
pointer
reference
integral
char
bool
floating
short
int
long
enum
array
struct
union
class
float
double
long double
unsigned

Standard Data Types in C++

❖ Integral Types

- represent whole numbers and their negatives
- declared as short, int or long

❖ Floating Types

- represent real numbers with a decimal point
- declared as float, double or long double

❖ Character Types

- represent single characters
- declared as char

Samples of C++ Data Values

int sample values

4578 -4578 0

Either the integer part or the fractional part, but not both, may be missing.

float sample values

95.274 95. .265

9521E-3 -95E-1 95.213E2

in scientific notation

char sample values

'B' 'd' '4' '?' '*'

C++ Data Type: String

- a string is a sequence of characters enclosed in double quotes
- string sample values

  `"Hello"` `"Year 2000"` `"1234"`
- the empty string (null string) contains no characters and is written as `""`

More About Type String

- string is not a built-in (standard) type
  - it is a programmer-defined data type
  - it is provided in the C++ standard library
- string operations include
  - comparing 2 string values
  - searching a string for a particular character
  - joining one string to another

Chapter 2 Topics

- Overview of C++ Data Types
- C++ Identifiers
- Declarations for Numeric Types
- Arithmetic Expressions
- Assignment Operator

Identifiers

- an identifier must start with a letter or underscore, and be followed by zero or more letters (A-Z, a-z), digits (0-9), or underscores

- **VALID**

age_of_dog **taxRateY2K**

PrintHeading **ageOfHorse**

- **NOT VALID**

age# **2000TaxRate** **Age-Of-Cat**

More About Identifiers

- some C++ compilers recognize only the first 32 characters of an identifier as significant
- then these identifiers are considered the same:

  age_Of_This_Old_Rhinoceros_At_My_Zoo

  age_Of_This_Old_Rhinoceros_At_My_Safari

- consider these:

  Age_Of_This_Old_Rhinoceros_At_My_Zoo

  age_Of_This_Old_Rhinoceros_At_My_Zoo

Chapter 2 Topics

- Overview of C++ Data Types
- C++ Identifiers
- Declarations for Numeric Types
- Arithmetic Expressions
- Assignment Operator

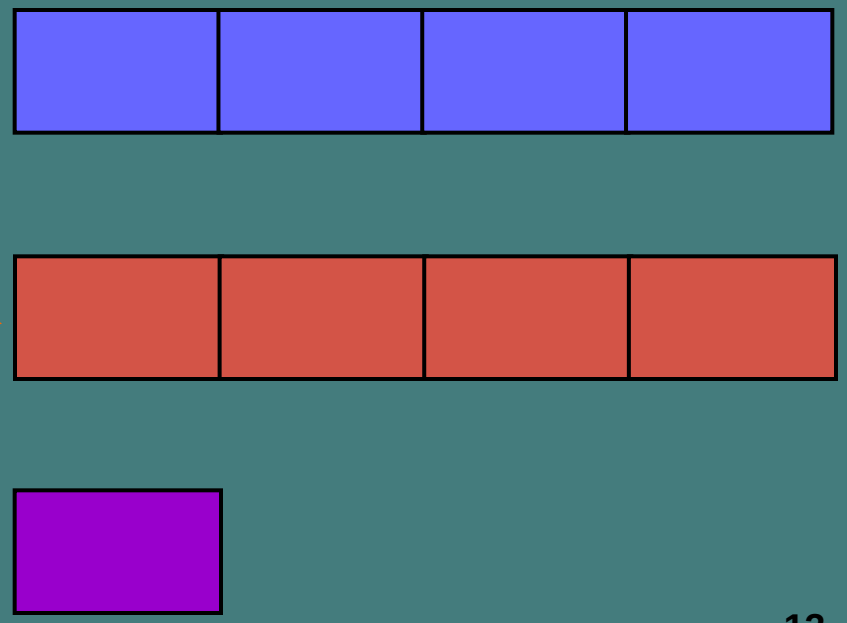
What is a Variable?

- A variable is a location in memory which we can refer to by an identifier, and in which a data value that can be changed is stored.
- declaring a variable means specifying both its name and its data type

What does a Variable Declaration Do?

A declaration tells the compiler to allocate enough memory to hold a value of this data type, and to associate the identifier with this location.

```
int     ageOfDog;
float   taxRateY2K;
char    middleInitial;
```



What is a Named Constant?

- A named constant is a location in memory that we can refer to by an identifier, and in which a data value that cannot be changed is stored.

VALID CONSTANT DECLARATIONS

```
const   string   STARS  =  "****" ;
const   float    NORMAL_TEMP  =  98.6 ;
const   char     BLANK = '  ' ;
const   int      VOTING_AGE  =  18 ;
const   float    MAX_HOURS  =  40.0 ;
```

Giving a Value to a Variable

You can assign (give) a value to a variable by using the **assignment operator =**

VARIABLE DECLARATIONS

```
string   firstName ;
char     middleInitial ;
char     letter ;
int      ageOfDog;
```

VALID ASSIGNMENT STATEMENTS

```
firstName  =  “Fido” ;
middleInitial  =  ‘X’ ;
letter  =  middleInitial ;
ageOfDog  =  12 ;
```

Chapter 2 Topics

- Overview of C++ Data Types
- C++ Identifiers
- Declarations for Numeric Types
- Arithmetic Expressions
- Assignment Operator

What is an Expression in C++?

- An **expression** is a valid arrangement of variables, constants, and operators.
- in C++ each expression can be evaluated to compute a value of a given type
- the value of the expression

  9 + 5 is 14

Operators can be

binary	**involving 2 operands**	**+**
unary	**involving 1 operand**	**++**
ternary	**involving 3 operands**	***? :***

Some C++ Operators

Precedence	Operator	Description
Higher	( )	Function call
	+	Positive
	-	Negative
	*	Multiplication
	/	Division
	%	Modulus (remainder)
	+	Addition
	-	Subtraction
Lower	=	Assignment

Precedence

- higher Precedence determines which operator is applied first in an expression having several operators
- parentheses can be used to change the usual order

Associativity

- left to right Associativity means that in an expression having 2 operators with the same priority, the left operator is applied first

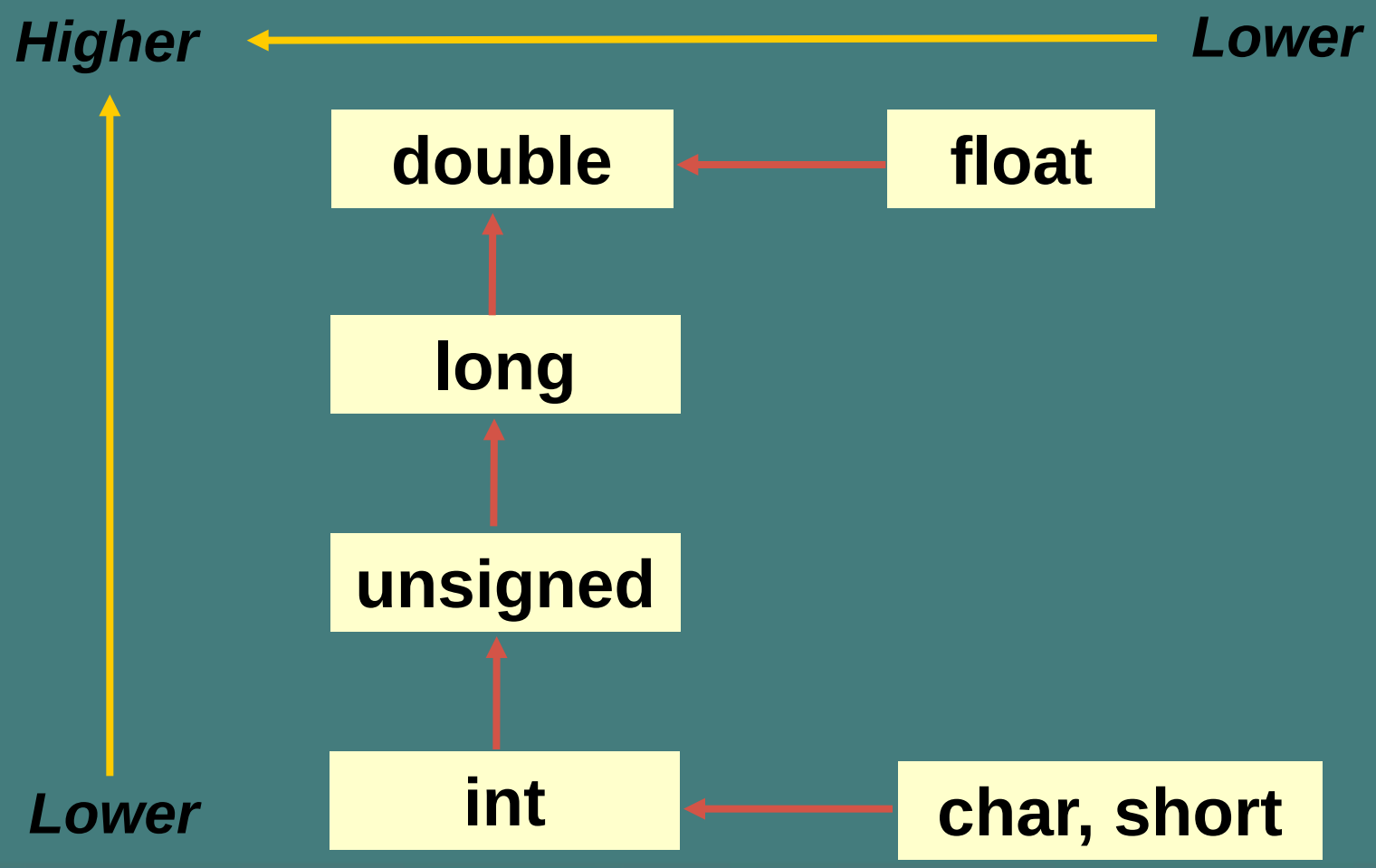
- in C++ the binary operators

  * , / , % , + , - are all left associative

Type Casting is Explicit Conversion of Type

int(4.8)	has value	4
float(5)	has value	5.0
float(7/4)	has value	1.0
float(7) / float(4)	has value	1.75

Type Coercion is Implicit Conversion of Type



Chapter 2 Topics

- Overview of C++ Data Types
- C++ Identifiers
- Declarations for Numeric Types
- Arithmetic Expressions
- Assignment Operator

Assignment Operator Syntax

Variable = Expression

First, Expression on right is evaluated.

Then the resulting value is stored in the memory location of Variable on left.

NOTE: An automatic type coercion occurs after evaluation but before the value is stored if the types differ for Expression and Variable

What value is stored?

```
float  a;
float  b;
a = 8.5;
b = 9.37;



a = b;
```

a	8.5
b	9.37

a	9.37
b	9.37

What is stored?

```
float  someFloat;

someFloat = 12;
```

?

someFloat

// causes implicit type conversion

12.0

someFloat

What is stored?

```
int  someInt;

someInt = 4.8;
```

someInt

// causes implicit type conversion

someInt