Welcome to Course

Programming Fundamentals

Week #4– Lecture 1 - 2

Print an equation

- Asks the user for 3 numbers.
- Asks the user for 2 operator symbols (like +, -, *, /).
- Shows the equation exactly as entered, without solving it.

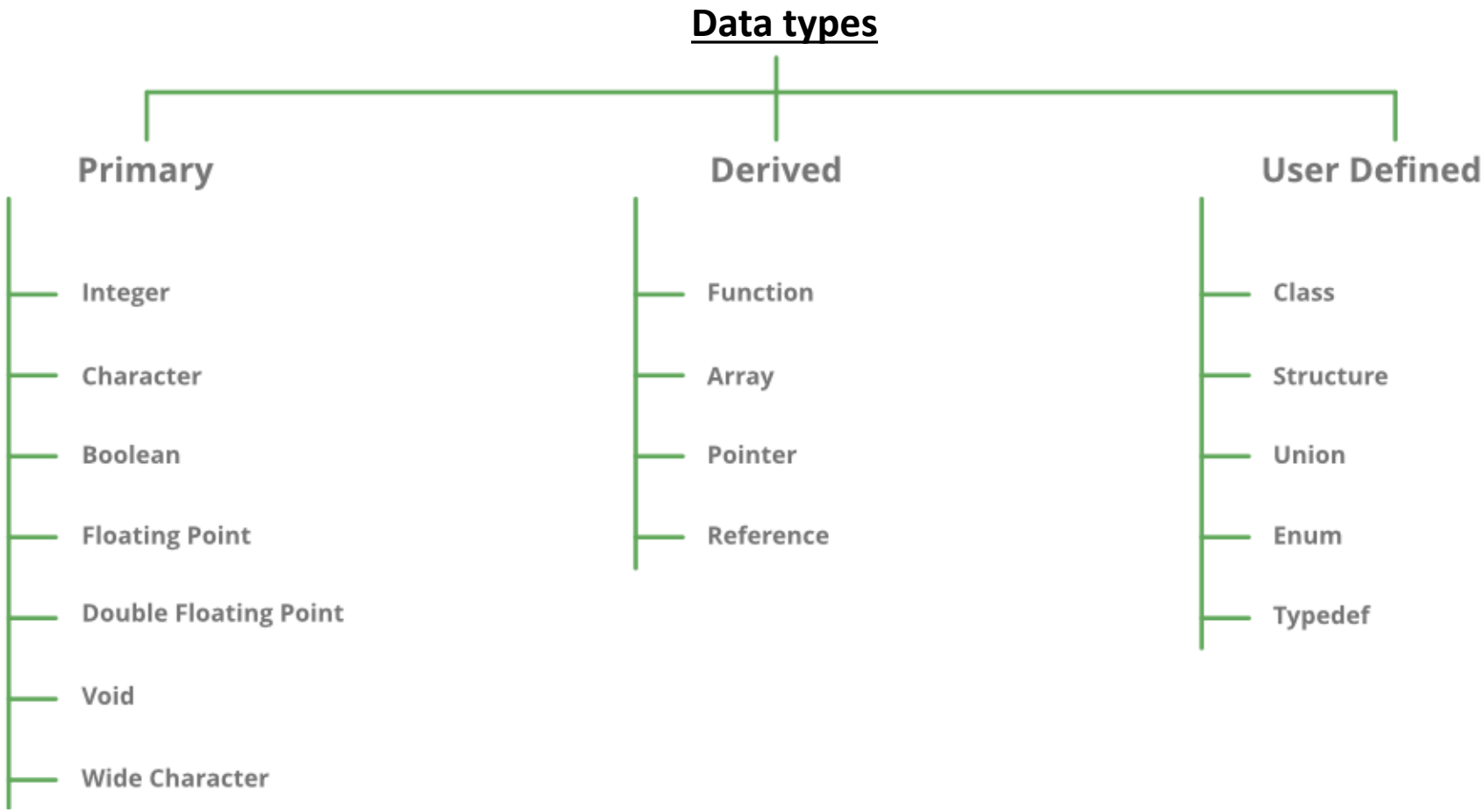
- Example

- Enter first number: 5
- Enter first operator: +
- Enter second number: 8
- Enter second operator: *
- Enter third number: 2

- Equation: 5 + 8 * 2

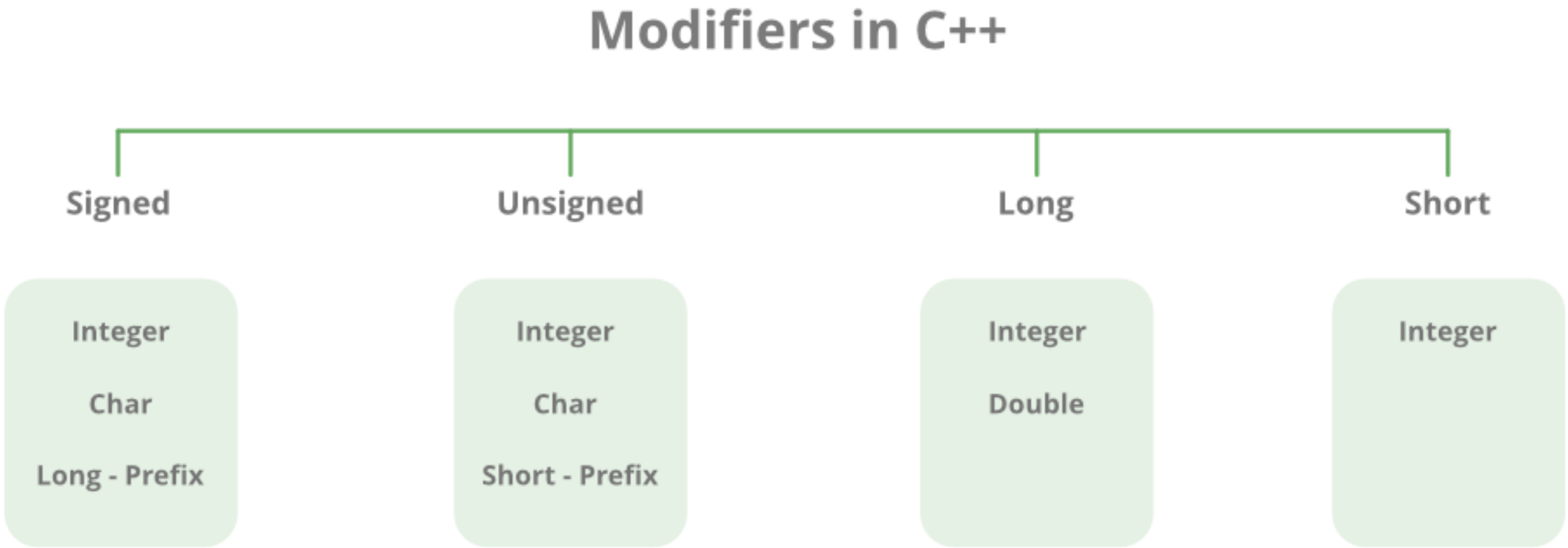
Data Types

Data types in C/C++



https://www.geeksforgeeks.org/c-data-types/

Datatype Modifiers



https://www.geeksforgeeks.org/c-data-types/

Data types and size (in bytes) ranges

Data Type	Size (in bytes)	Range
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	8	-2,147,483,648 to 2,147,483,647
unsigned long int	8	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 or 4	1 wide character

Note: Above values may vary from compiler to compiler. In the above example, we have considered GCC 32 bit

2.6

Integer Data Types

Integer Data Types

- Integer variables can hold whole numbers such as 12, 7, and -99.

Table 2-6 Integer Data Types, Sizes, and Ranges

Data Type	Size	Range
`short`	2 bytes	–32,768 to +32,767
`unsigned short`	2 bytes	0 to +65,535
`int`	4 bytes	–2,147,483,648 to +2,147,483,647
`unsigned int`	4 bytes	0 to 4,294,967,295
`long`	4 bytes	–2,147,483,648 to +2,147,483,647
`unsigned long`	4 bytes	0 to 4,294,967,295

Defining Variables

- Variables of the same type can be defined
  - On separate lines:

```
int length;
int width;
unsigned int area;
```

  - On the same line:

```
int length, width;
unsigned int area;
```

- Variables of different types must be in different definitions

Integer Types in Program 2-10

Program 2-10

```
1  // This program has variables of several of the integer types.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int checking;
8     unsigned int miles;
9     long days;
10
11    checking = -20;
12    miles = 4276;
13    days = 189000;
14    cout << "We have made a long journey of " << miles;
15    cout << " miles.\n";
16    cout << "Our checking account balance is " << checking;
17    cout << "\nAbout " << days << " days ago Columbus ";
18    cout << "stood on this spot.\n";
19    return 0;
20 }
```

This program has three variables: checking, miles, and days

Integer Literals

- An integer literal is an integer value that is typed into a program's code. For example:

```
itemsOrdered = 15;
```

  In this code, 15 is an integer literal.

Integer Literals in Program 2-10

Program 2-10

```
1  // This program has variables of several of the integer types.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int checking;
8     unsigned int miles;
9     long days;
10
11    checking = -20;
12    miles = 4276;
13    days = 189000;
14    cout << "We have made a long journey of " << miles;
15    cout << " miles.\n";
16    cout << "Our checking account balance is " << checking;
17    cout << "\nAbout " << days << " days ago Columbus ";
18    cout << "stood on this spot.\n";
19    return 0;
20 }
```

Integer Literals

- Integer literals are stored in memory as `int`s by default
- To store an integer constant in a long memory location, put '`L`' at the end of the number: `1234L`
- Constants that begin with '`0`' (zero) are base 8: `075`
- Constants that begin with '`0x`' are base 16: `0x75A`

2.7

The `char` Data Type

The `char` Data Type

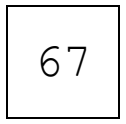
- Used to hold characters or very small integer values
- Usually 1 byte of memory
- Numeric value of character from the character set is stored in memory:

CODE:

```
char letter;
letter = 'C';
```

MEMORY:

letter



Character Literals

- Character literals must be enclosed in single quote marks. Example:

```
'A'
```

Character Literals in Program 2-13

Program 2-13

```
1  // This program uses character literals.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     char letter;
8
9     letter = 'A';
10    cout << letter << endl;
11    letter = 'B';
12    cout << letter << endl;
13    return 0;
14 }
```

Program Output

```
A
B
```

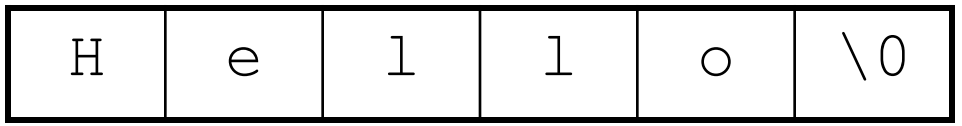
Character Strings

- A series of characters in consecutive memory locations:

  `"Hello"`

- Stored with the null terminator, `\0`, at the end:

- Comprised of the characters between the `"  "`



2.8

The C++ `string` Class

The C++ `string` Class

- Special data type supports working with strings
- `#include <string>`
- Can define `string` variables in programs:

```
string firstName, lastName;
```

- Can receive values with assignment operator:

```
firstName = "George";
lastName = "Washington";
```

- Can be displayed via `cout`

```
cout << firstName << " " << lastName;
```

The `string` class in Program 2-15

Program 2-15

```
1  // This program demonstrates the string class.
2  #include <iostream>
3  #include <string> // Required for the string class.
4  using namespace std;
5
6  int main()
7  {
8     string movieTitle;
9
10    movieTitle = "Wheels of Fury";
11    cout << "My favorite movie is " << movieTitle << endl;
12    return 0;
13 }
```

Program Output

```
My favorite movie is Wheels of Fury
```

2.9

Floating-Point Data Types

Floating-Point Data Types

- The floating-point data types are:
  `float`
  `double`
  `long double`

- They can hold real numbers such as:
  `12.45` `-3.8`

- Stored in a form similar to scientific notation

- All floating-point numbers are signed

Floating-Point Data Types

Table 2-8 Floating Point Data Types on PCs

Data Type	Key Word	Description
Single precision	`float`	4 bytes. Numbers between ±3.4E-38 and ±3.4E38
Double precision	`double`	8 bytes. Numbers between ±1.7E-308 and ±1.7E308
Long double precision	`long double*`	8 bytes. Numbers between ±1.7E-308 and ±1.7E308

Floating-Point Literals

- Can be represented in
  - Fixed point (decimal) notation:

    `31.4159` `0.0000625`
  - E notation:

    `3.14159E1` `6.25e-5`
- Are `double` by default
- Can be forced to be float (`3.14159f`) or long double (`0.0000625L`)

Floating-Point Data Types in Program 2-16

Program 2-16

```
1  // This program uses floating point data types.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     float distance;
8     double mass;
9
10    distance = 1.495979E11;
11    mass = 1.989E30;
12    cout << "The Sun is " << distance << " meters away.\n";
13    cout << "The Sun\'s mass is " << mass << " kilograms.\n";
14    return 0;
15 }
```

Program Output

```
The Sun is 1.49598e+011 meters away.
The Sun's mass is 1.989e+030 kilograms.
```

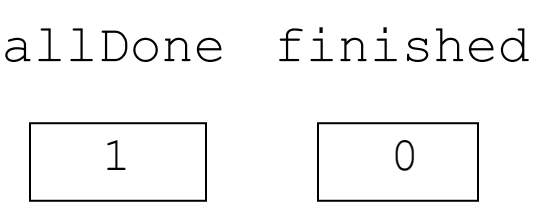
2.10

The `bool` Data Type

The `bool` Data Type

- Represents values that are `true` or `false`
- `bool` variables are stored as small integers
- `false` is represented by 0, `true` by 1:

```
bool allDone = true;
bool finished = false;
```



Boolean Variables in Program 2-17

Program 2-17

```
1  // This program demonstrates boolean variables.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     bool boolValue;
8
9     boolValue = true;
10    cout << boolValue << endl;
11    boolValue = false;
12    cout << boolValue << endl;
13    return 0;
14 }
```

Program Output

```
1
0
```

2.11

Determining the Size of a Data Type

Determining the Size of a Data Type

The `sizeof` operator gives the size of any data type or variable:

```
double amount;
cout << "A double is stored in "
     << sizeof(double) <<
"bytes\n";
cout << "Variable amount is
stored in "
     << sizeof(amount)
      << "bytes\n";
```

2.12

Variable Assignments and Initialization

Variable Assignments and Initialization

- An assignment statement uses the = operator to store a value in a variable.

```
item = 12;
```

- This statement assigns the value 12 to the `item` variable.

Assignment

- The variable receiving the value must appear on the left side of the = operator.
- This will NOT work:

```
// ERROR!
12 = item;
```

Variable Initialization

- To initialize a variable means to assign it a value when it is defined:

```
int length = 12;
```

- Can initialize some or all variables:

```
int length = 12, width = 5, area;
```

Variable Initialization in Program 2-19

Program 2-19

```
1  // This program shows variable initialization.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int month = 2, days = 28;
8
9     cout << "Month " << month << " has " << days << " days.\n";
10    return 0;
11 }
```

Program Output

```
Month 2 has 28 days.
```

2.13

Scope

Scope

- The scope of a variable: the part of the program in which the variable can be accessed
- A variable cannot be used before it is defined

Variable Out of Scope in Program 2-20

Program 2-20

```
1  // This program can't find its variable.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     cout << value; // ERROR! value not defined yet!
8
9     int value = 100;
10    return 0;
11 }
```

`#define` directive in Program 2-31

Program 2-31

```
1  // This program calculates the circumference of a circle.
2  #include <iostream>
3  using namespace std;
4
5  #define PI 3.14159
6  #define DIAMETER 10.0
7
8  int main()
9  {
10    // Variable to hold the circumference
11    double circumference;
12
13    // Calculate the circumference.
14    circumference = PI * DIAMETER;
15
16    // Display the circumference.
17    cout << "The circumference is: " << circumference << endl;
18    return 0;
19 }
```

Program Output

```
The circumference is: 31.4159
```

2.17

Programming Style

Programming Style

- The visual organization of the source code
- Includes the use of spaces, tabs, and blank lines
- Does not affect the syntax of the program
- Affects the readability of the source code

Programming Style

Common elements to improve readability:

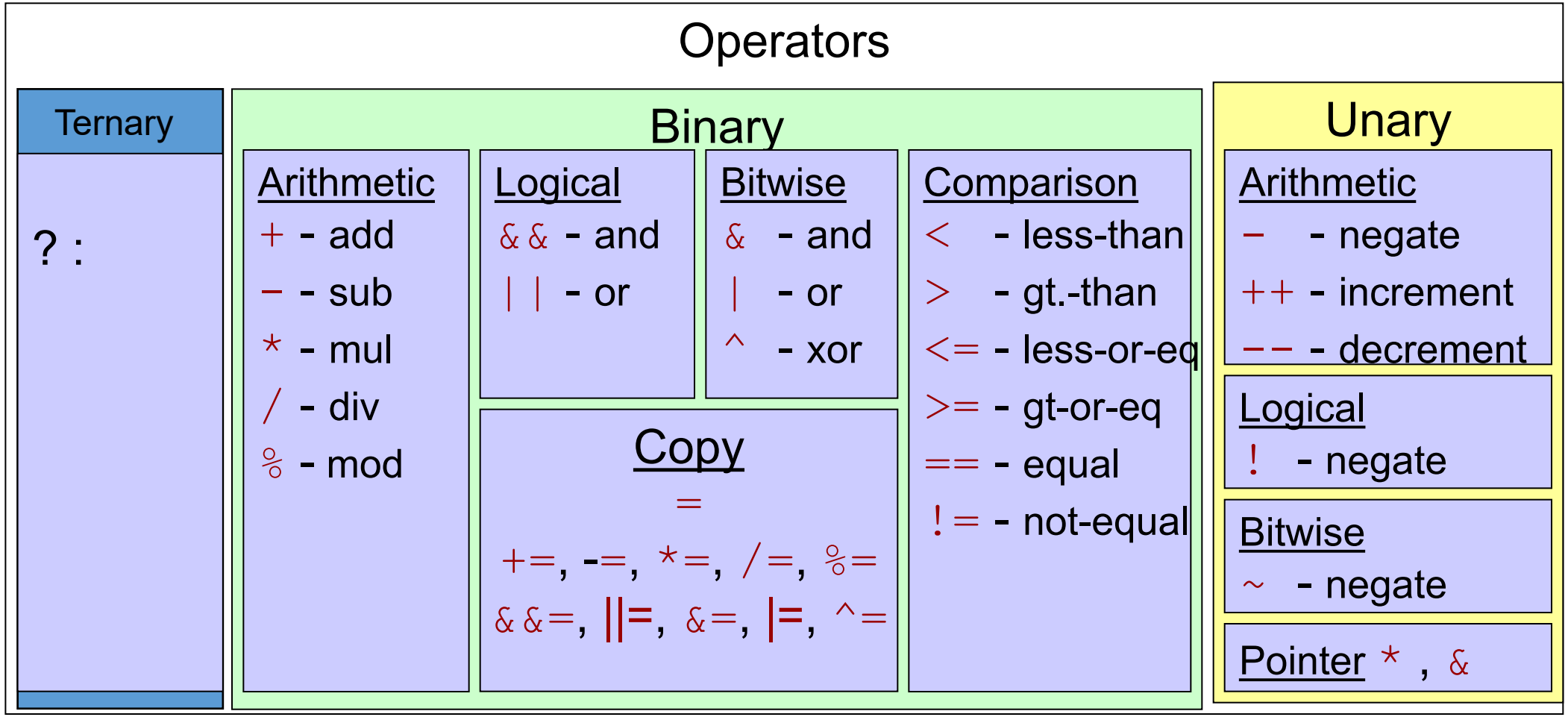
- Braces { } aligned vertically
- Indentation of statements within a set of braces
- Blank lines between declaration and other statements
- Long statements wrapped over multiple lines with aligned operators

Operators and Mathematical expressions in C++

Operators in C++

- Arithmetic operators, Relational or comparison operators
- Logical operators, Bitwise operators, Ternary operators
- Size of operator
  - is a compile-time operator that determines the size in bytes of a variable or data type e.g., sizeof (data type)
- Type casting operator
  - Example: c = (int) a;
- new operator and many more
  - Syntax: pointer-variable = new data-type;
  - Example: int *p = new int;
- Three things are important about operators
  - Precedence, Associativity and Arity

C++ operator map



47