

Slides

Development > Programming Languages > C++

## The C++ 20 Masterclass : From Fundamentals to Advanced

Learn and Master Modern C++ From Beginning to Advanced in Plain English : C++11, C++14, C++17, C++20 and More!

4.7 ★★★★★

Created by [Daniel Gakwaya](#)

# Section : Operations on data

Slide intentionally left empty

# Operations on Data

add

subtract

divide

modulus

decrement

multiply

Built in  
utilities

Formatting

increment

Slide intentionally left empty

# Basic Operations

add

subtract

divide

modulus

multiply

## Addition

```
//Addition
int sum { number1 + number2 };
int other_sum = number1 + number2 + number1;

std::cout << "The sum is : " << sum << std::endl;
std::cout << "Other sum is : " << other_sum << std::endl;
```



## Subtraction

```
//Subtraction  
int diff { number1 - number2 };  
int other_diff = number1 - number2 - number2;  
  
std::cout << "The difference is : " << diff << std::endl;  
std::cout << "Other diff is : " << other_diff << std::endl;
```

## Multiplication

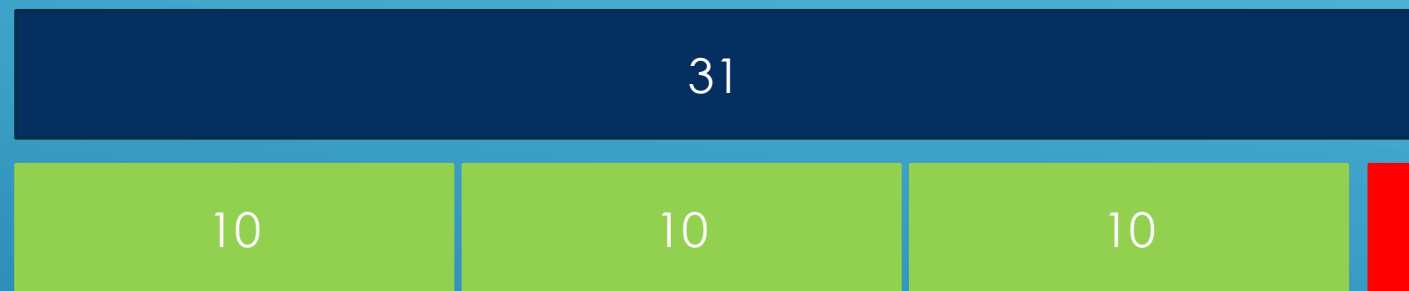
```
//Multiplication
int product {number1 * number2 };
int other_product = number1 * 2 * number2;

std::cout << "The product is : " << product << std::endl;
std::cout << "Other product is : " << other_product << std::endl;
```

## Division

```
//Integer Division  
int quotient { number1 / number2 };  
int other_quotient = number1 / 17;  
  
std::cout << "The quotient is : " << quotient << std::endl;  
std::cout << "The other quotient is : " << other_quotient << std::endl;
```

31/10

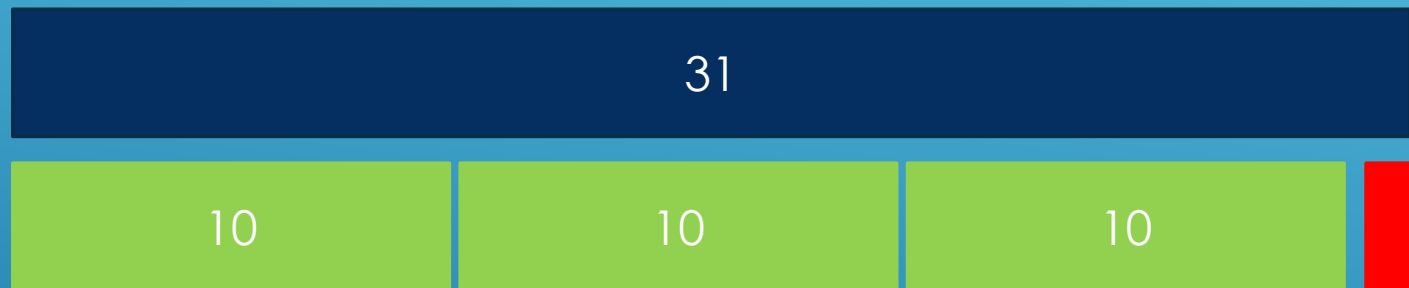


12

## Modulus

```
//Integer Division  
int quotient { number1 / number2 };  
int other_quotient = number1 / 17;  
  
std::cout << "The quotient is : " << quotient << std::endl;  
std::cout << "The other quotient is : " << other_quotient << std::endl;
```

$31 \% 10$



14

Slide intentionally left empty

# Precedence and Associativity





$a + b * c - d / e - f + g$

Precedence : which operation to do first  
Associativity : which direction or which order



$a/b * c + d - e + f$

It is better to make the intent in your code as clear as possible by clearly using `()` to indicate which operations you want done first

Slide intentionally left empty

# Prefix and Postfix + and -

## Regular Increment/Decrement

```
→ int value { 5 };  
  
//Increment by one  
value = value + 1;  
std::cout << "The value is : " << value << std::endl;  
  
value = 5; // Reset value to 5  
  
//Decrement by one  
value = value - 1;  
std::cout << "The value is : " << value << std::endl;
```

## Postfix Increment/Decrement

```
value = 5;

std::cout << "The value is (incrementing) : " << value++ << std::endl;
std::cout << "The value is : " << value << std::endl;

std::cout << std::endl;

//Decrement with postfix

//Reset value to 5
value = 5;

std::cout << "The value is (decrementing) : " << value-- << std::endl;
std::cout << "The value is : " << value << std::endl;
```



## Postfix Increment/Decrement

```
value = 5;

std::cout << "The value is (incrementing) : " << value++ << std::endl; //5
std::cout << "The value is : " << value << std::endl; //6

std::cout << std::endl;

//Decrement with postfix

//Reset value to 5
value = 5;

std::cout << "The value is (decrementing) : " << value-- << std::endl;
std::cout << "The value is : " << value << std::endl;
```

## Postfix Increment/Decrement

```
value = 5;

std::cout << "The value is (incrementing) : " << value++ << std::endl; //5
std::cout << "The value is : " << value << std::endl; //6

std::cout << std::endl;

//Decrement with postfix

//Reset value to 5
value = 5;

std::cout << "The value is (decrementing) : " << value-- << std::endl; //5
std::cout << "The value is : " << value << std::endl; //4
```

## Prefix Increment/Decrement

```
value = 5;

++value;
std::cout << "The value is (prefix++) : " << value << std::endl;

//Reset value to 5
value = 5;
std::cout << "The value is (prefix++ in place) : " << ++value << std::endl;

std::cout << std::endl;

//Prefix : Decrementing

//Reset value to 5;
value = 5;
--value;
std::cout << "The value is (prefix--) : " << value << std::endl;

//Reset value to 5;
value = 5;
std::cout << "The value is (prefix-- in place) : " << --value << std::endl;
```

## Prefix Increment/Decrement

```
value = 5;

++value;
std::cout << "The value is (prefix++) : " << value << std::endl; //6

//Reset value to 5
value = 5;
std::cout << "The value is (prefix++ in place) : " << ++value << std::endl;

std::cout << std::endl;

//Prefix : Decrementing

//Reset value to 5;
value = 5;
--value;
std::cout << "The value is (prefix--) : " << value << std::endl;

//Reset value to 5;
value = 5;
std::cout << "The value is (prefix-- in place) : " << --value << std::endl;
```

## Prefix Increment/Decrement

```
value = 5;

++value;
std::cout << "The value is (prefix++) : " << value << std::endl; //6

//Reset value to 5
value = 5;
std::cout << "The value is (prefix++ in place) : " << ++value << std::endl; //6

std::cout << std::endl;

//Prefix : Decrementing

//Reset value to 5;
value = 5;
--value;
std::cout << "The value is (prefix--) : " << value << std::endl;

//Reset value to 5;
value = 5;
std::cout << "The value is (prefix-- in place) : " << --value << std::endl;
```

## Prefix Increment/Decrement

```
value = 5;

++value;
std::cout << "The value is (prefix++) : " << value << std::endl; //6

//Reset value to 5
value = 5;
std::cout << "The value is (prefix++ in place) : " << ++value << std::endl; //6

std::cout << std::endl;

//Prefix : Decrementing

//Reset value to 5;
value = 5;
--value;
std::cout << "The value is (prefix--) : " << value << std::endl; //4

//Reset value to 5;
value = 5;
std::cout << "The value is (prefix-- in place) : " << --value << std::endl;
```

## Prefix Increment/Decrement

```
value = 5;

++value;
std::cout << "The value is (prefix++) : " << value << std::endl; //6

//Reset value to 5
value = 5;
std::cout << "The value is (prefix++ in place) : " << ++value << std::endl; //6

std::cout << std::endl;

//Prefix : Decrementing

//Reset value to 5;
value = 5;
--value;
std::cout << "The value is (prefix--) : " << value << std::endl; //4

//Reset value to 5;
value = 5;
std::cout << "The value is (prefix-- in place) : " << --value << std::endl; //4
```

- Prefix and postfix increment/decrement operators are cool.
- But they only increment by one, what if we need to increment by a value other than 1, say 5 or 7. We'll see a way to do that in the next lecture
- There is no `value**` or `**value`, or `//value` or `value//`. They don't make sense, these prefix/postfix operators are only available for `+` and `-`, again because that's where they make sense



Slide intentionally left empty

# Compound Operators

```

>int value {45};

std::cout << "The value is : " << value << std::endl;

std::cout << std::endl;
value +=5; // equivalent to value = value + 5
std::cout << "The value is (after +=5) : " << value << std::endl; // 50

std::cout << std::endl;
value -=5; // equivalent to value = value - 5
std::cout << "The value is (after -=5) : " << value << std::endl; // 45

std::cout << std::endl;
value *=2;
std::cout << "The value is (after *=2) : " << value << std::endl; // 90

std::cout << std::endl;
value /= 3;
std::cout << "The value is (after /=3) : " << value << std::endl; // 30

std::cout << std::endl;
value %= 11;
std::cout << "The value is (after %=11) : " << value << std::endl; // 8

```

Slide intentionally left empty

# Relational Operators : Comparing Stuff

```

int number1 {45};
int number2 {60};

std::cout << "number1 : " << number1 << std::endl;
std::cout << "number2 : " << number2 << std::endl;

std::cout << std::endl;
std::cout << "Comparing variables" << std::endl;

std::cout << std::boolalpha ; // Make bool show up as true/false instead of 1/0
//Stress the need for parentheses here
std::cout << "number1 < number2 : " << (number1 < number2) << std::endl;
std::cout << "number1 <= number2 : " << (number1 <= number2) << std::endl;
std::cout << "number1 > number2 : " << (number1 > number2) << std::endl;
std::cout << "number1 >= number2 : " << (number1 >= number2) << std::endl;
std::cout << "number1 == number2 : " << (number1 == number2) << std::endl;
//std::cout << std::noboolalpha;
std::cout << "number1 != number2 : " << (number1 != number2) << std::endl;

```

Slide intentionally left empty

# Logical Operators



AND

&&

OR

||

NOT

!

## AND

<b>a</b>	<b>b</b>	<b>a &amp;&amp; b</b>
false	false	false
false	true	false
true	false	false
true	true	true

OR

<b>a</b>	<b>b</b>	<b>a    b</b>
false	false	false
false	true	true
true	false	true
true	true	true

NOT

<b>a</b>	<b>!a</b>
true	false
false	true

```
bool a {true};  
bool b {false};  
bool c {true};
```

```
std::cout << std::boolalpha;  
std::cout << "a : " << a << std::endl;  
std::cout << "b : " << b << std::endl;  
std::cout << "c : " << c << std::endl;
```

```
//AND : Evaluates to true when all operands are true.  
//      A single false operand will drag  
//      the entire expression to evaluating false.
```

```
std::cout << std::endl;  
std::cout << "Basic AND operations" << std::endl;
```

```
std::cout << " a && b : " << (a && b) << std::endl;  
std::cout << " a && c : " << (a && c) << std::endl;  
std::cout << " a && b && c : " << (a && b && c) << std::endl;
```

```
//AND : Evaluates to true when all operands are true.  
//      A single false operand will drag  
//      the entire expression to evaluating false.  
  
std::cout << std::endl;  
std::cout << "Basic AND operations" << std::endl;  
  
std::cout << " a && b : " << (a && b) << std::endl;  
std::cout << " a && c : " << (a && c) << std::endl;  
std::cout << " a && b && c : " << (a && b && c) << std::endl;
```

```
//OR : Evaluates to true when at least one operand true.  
//      A single true operand will push  
//      the entire expression to evaluating true.  
std::cout << std::endl;  
std::cout << "Basic OR operations" << std::endl;  
std::cout << " a || b : " << (a || b) << std::endl;  
std::cout << " a || c : " << (a || c) << std::endl;  
std::cout << " a || b || c : " << (a || b || c) << std::endl;
```

```
//NOT : Negates whatever operand you put it with
std::cout << std::endl;
std::cout << "Basic NOT operations" << std::endl;

std::cout << "!a : " << !a << std::endl;
std::cout << "!b : " << !b << std::endl;
std::cout << "!c : " << !c << std::endl;
```



## Combine logical operators in expression

```
std::cout << "!(a &&b) || c : " << (!(a &&b) || c) << std::endl;
```

## Logical and Relational combined in expressions

```
int d{45};
int e{20};
int f{11};

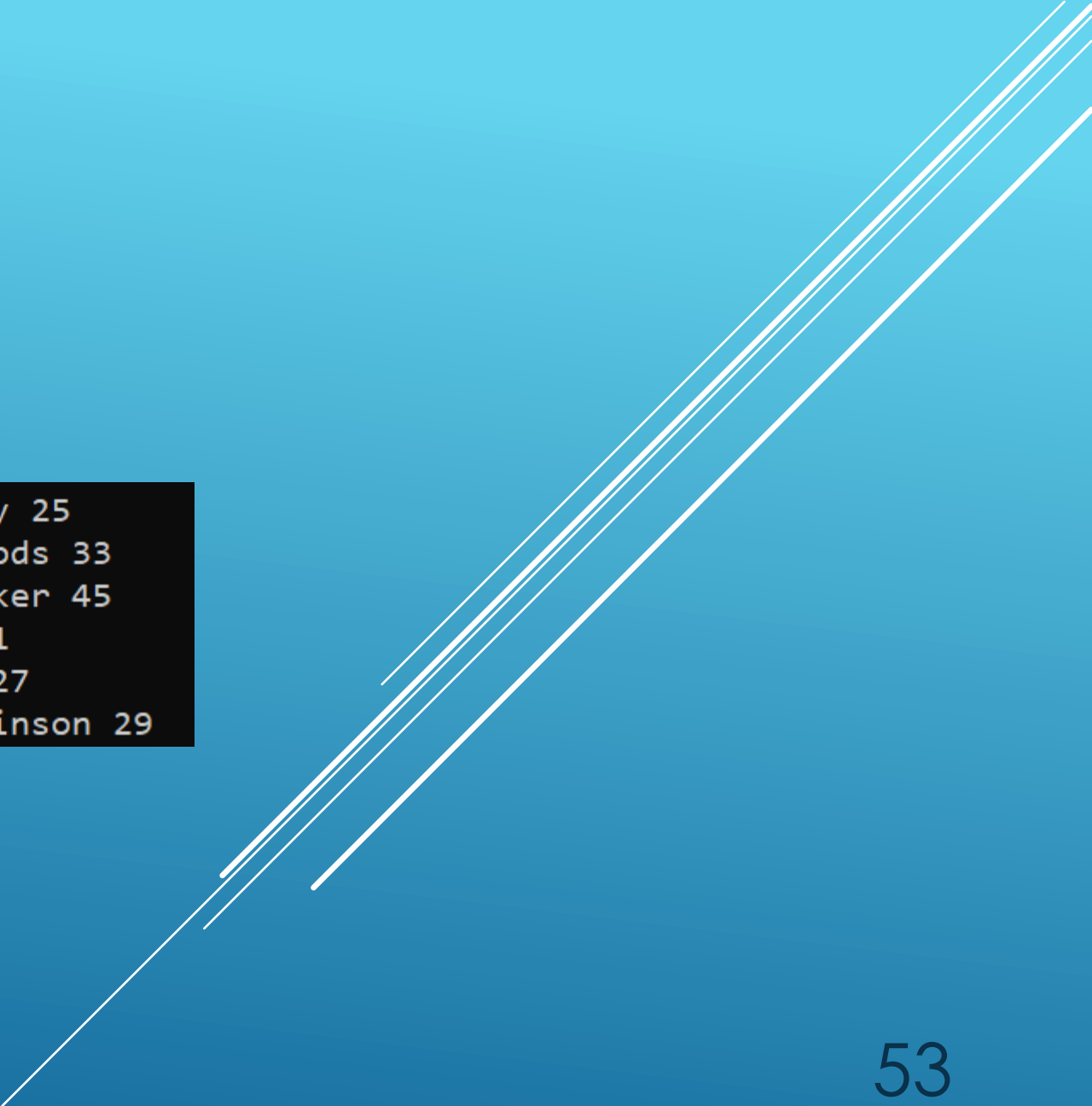
std::cout << std::endl;
std::cout << "Relational and logic operations on integers" << std::endl;
std::cout << "d : " << d << std::endl;
std::cout << "e : " << e << std::endl;
std::cout << "f : " << f << std::endl;

std::cout << std::endl;

std::cout << "(d > e) && (d > f) : " << ((d > e) && (d > f)) << std::endl;
std::cout << "(d==e) || (e <= f) : " << ((d==e) || (e <= f)) << std::endl;
std::cout << "(d < e) || (d > f) : " << ((d < e) || (d > f)) << std::endl;
std::cout << "(f > e) || (d < f) : " << ((f > e) || (d < f)) << std::endl;
std::cout << "(d > f) && (f <= d) : " << ((d > f) && (f <= d)) << std::endl;
std::cout << "(d > e) && (d <= f) : " << ((d > e) && (d <= f)) << std::endl;
std::cout << "(! a) && (d == e) : " << ((! a) && (d == e)) << std::endl;
std::cout << "(! a) && (d == e) : " << ((! a) && (d == e)) << std::endl;
```

Slide intentionally left empty

# Output Formatting

Several thin, white, parallel lines of varying lengths and angles are scattered across the right side of the slide, creating a dynamic, abstract pattern.

```
Daniel Gray 25  
Stanley Woods 33  
Jordan Parker 45  
Joe Ball 21  
Josh Carr 27  
Izaiah Robinson 29
```

LastName	Firstname	Age
Daniel	Gray	25
Stanley	Woods	33
Jordan	Parker	45
Joe	Ball	21
Josh	Carr	27
Izaiah	Robinson	29

```
#include <ios>
```

```
#include <iomanip>
```

`std::endl`

`std::flush`

`std::right`

`std::boolalpha`

`std::showpoint`

`std::setfill()`

`std::left`

`std::setw()`

`std::hex`

`std::internal`

`std::fixed`

`std::setprecision()`

`std::dec`

`std::scientific`

`std::showbase`

`std::uppercase`

`std::showpos`

`std::oct`

56



std::endl

```
std::cout << "Hello";  
std::cout << "World";  
  
std::cout << std::endl;  
  
std::cout << "-----" << std::endl;  
  
std::cout << "Hello" << std::endl;  
std::cout << "World" << std::endl;  
  
std::cout << std::endl;  
  
std::cout << "Hello\n";  
std::cout << "World\n";
```

```
HelloWorld  
-----  
Hello  
World  
  
Hello  
World
```

`std::flush`

causes immediate sending of data to  
the device connected to the stream

`std::cout`

`std::flush`

`std::cout`

`std::flush`

Hello

`std::cout`

`std::flush`

Hello  
World

`std::cout`

`std::flush`

Hello  
World  
How  
Are  
You

`std::cout`

`std::flush`

Hello  
World  
How  
Are  
You

## std::setw()

```
std::cout << "Unformatted table : " << std::endl;
std::cout << "Daniel" << " " << "Gray" << " 25" << std::endl;
std::cout << "Stanley" << " " << "Woods" << " 33" << std::endl;
std::cout << "Jordan" << " " << "Parker" << " 45" << std::endl;
std::cout << "Joe" << " " << "Ball" << " 21" << std::endl;
std::cout << "Josh" << " " << "Carr" << " 27" << std::endl;
std::cout << "Izaiah" << " " << "Robinson" << " 29" << std::endl;
```

```
std::cout << std::endl;
std::cout << "Formatted table : " << std::endl;
```

```
std::cout << std::setw(10) << "Lastname" << std::setw(10) << "Firstname" << std::setw(5) << "Age" << std::endl;
std::cout << std::setw(10) << "Daniel" << std::setw(10) << "Gray" << std::setw(5) << "25" << std::endl;
std::cout << std::setw(10) << "Stanley" << std::setw(10) << "Woods" << std::setw(5) << "33" << std::endl;
std::cout << std::setw(10) << "Jordan" << std::setw(10) << "Parker" << std::setw(5) << "45" << std::endl;
std::cout << std::setw(10) << "Joe" << std::setw(10) << "Ball" << std::setw(5) << "21" << std::endl;
std::cout << std::setw(10) << "Josh" << std::setw(10) << "Carr" << std::setw(5) << "27" << std::endl;
std::cout << std::setw(10) << "Izaiah" << std::setw(10) << "Robinson" << std::setw(5) << "29" << std::endl;
```

```
Unformatted table :
Daniel Gray 25
Stanley Woods 33
Jordan Parker 45
Joe Ball 21
Josh Carr 27
Izaiah Robinson 29
```

```
Formatted table :
Lastname Firstname Age
Daniel      Gray      25
Stanley     Woods     33
Jordan      Parker    45
Joe         Ball      21
Josh        Carr      27
Izaiah      Robinson  29
```



## Right justified

```
col_width = 20;
```

```
std::cout << std::right;
std::cout << std::setw(col_width) << "Lastname" << std::setw(col_width)
std::cout << std::setw(col_width) << "Daniel" << std::setw(col_width) <<
std::cout << std::setw(col_width) << "Stanley" << std::setw(col_width) <<
std::cout << std::setw(col_width) << "Jordan" << std::setw(col_width) <<
std::cout << std::setw(col_width) << "Joe" << std::setw(col_width) << "Ba
std::cout << std::setw(col_width) << "Josh" << std::setw(col_width) << "Ca
std::cout << std::setw(col_width) << "Izaiah" << std::setw(col_width) << "
```

Head	Head
1	2

## Left justified

```
col_width = 20;

std::cout << std::left;
std::cout << std::setw(col_width) << "Lastname" << std::setw(col_width) <<
std::cout << std::setw(col_width) << "Daniel" << std::setw(col_width) <<
std::cout << std::setw(col_width) << "Stanley" << std::setw(col_width) <<
std::cout << std::setw(col_width) << "Jordan" << std::setw(col_width) <<
std::cout << std::setw(col_width) << "Joe" << std::setw(col_width) << "B"
std::cout << std::setw(col_width) << "Josh" << std::setw(col_width) << "C"
std::cout << std::setw(col_width) << "Izaiah" << std::setw(col_width) <<
```

Head	Head
1	2

## Internal justified

```
//Internal justified : sign is left justified , data is right justified
std::cout << std::endl;
std::cout << "Internal justified : " << std::endl;
std::cout << std::right;
std::cout << std::setw(10) << -123.45 << std::endl;
std::cout << std::internal;
std::cout << std::setw(10) << -123.45 << std::endl;
```

Head		Head	
-	3.33	2	
-	34.79		

## std::setfill

```
col_width = 20;

std::cout << std::left;
std::cout << std::setfill('-'); // The fill character
std::cout << std::setw(col_width) << "Lastname" << std::setw(col_width) << "Firstname" << std::setw(col_width) << "Age" << std::endl;
std::cout << std::setw(col_width) << "Daniel" << std::setw(col_width) << "Gray" << std::setw(col_width) << "25" << std::endl;
std::cout << std::setw(col_width) << "Stanley" << std::setw(col_width) << "Woods" << std::setw(col_width) << "33" << std::endl;
std::cout << std::setw(col_width) << "Jordan" << std::setw(col_width) << "Parker" << std::setw(col_width) << "45" << std::endl;
std::cout << std::setw(col_width) << "Joe" << std::setw(col_width) << "Ball" << std::setw(col_width) << "21" << std::endl;
std::cout << std::setw(col_width) << "Josh" << std::setw(col_width) << "Carr" << std::setw(col_width) << "27" << std::endl;
std::cout << std::setw(col_width) << "Izaiah" << std::setw(col_width) << "Robinson" << std::setw(col_width) << "29" << std::endl;
```

```
Lastname-----Firstname-----Age-----
Daniel-----Gray-----25-----
Stanley-----Woods-----33-----
Jordan-----Parker-----45-----
Joe-----Ball-----21-----
Josh-----Carr-----27-----
Izaiah-----Robinson-----29-----
```

## std::boolalpha

//boolalpha and noboolalpha : control bool output format : 1/0 or true/false

```
bool condition {true};
```

```
bool other_condition {false};
```

```
std::cout << "condition : " << condition << std::endl;
```

```
std::cout << "other_condition : " << other_condition << std::endl;
```

```
std::cout << std::endl;
```

```
std::cout << std::boolalpha;
```

```
std::cout << "condition : " << condition << std::endl;
```

```
std::cout << "other_condition : " << other_condition << std::endl;
```

```
std::cout << std::endl;
```

```
std::cout << std::noboolalpha;
```

```
std::cout << "condition : " << condition << std::endl;
```

```
std::cout << "other_condition : " << other_condition << std::endl;
```

## std::showpos

```
//showpos and noshowpos : show or hide the + sign for positive numbers
```

```
int pos_num {34};  
int neg_num {-45};
```

```
std::cout << "pos_num : " << pos_num << std::endl;  
std::cout << "neg_num : " << neg_num << std::endl;
```

```
std::cout << std::endl;  
std::cout << std::showpos;  
std::cout << "pos_num : " << pos_num << std::endl;  
std::cout << "neg_num : " << neg_num << std::endl;
```

```
std::cout << std::endl;  
std::cout << std::noshowpos;  
std::cout << "pos_num : " << pos_num << std::endl;  
std::cout << "neg_num : " << neg_num << std::endl;
```

```
pos_num : 34  
neg_num : -45  
  
pos_num : +34  
neg_num : -45  
  
pos_num : 34  
neg_num : -45
```

## std::dec std::oct and std::hex

```
//different number systems : std::dec, std::hex, std::oct

int pos_int {717171};
int neg_int {-47347};
double double_var {498.32};

std::cout << std::endl;
std::cout << "pos_int in different bases : " << std::endl;
std::cout << "pos_int (dec) : " << std::dec << pos_int << std::endl;
std::cout << "pos_int (hex) : " << std::hex << pos_int << std::endl;
std::cout << "pos_int (oct) : " << std::oct << pos_int << std::endl;

std::cout << std::endl;
std::cout << "neg_int in different bases : " << std::endl;
std::cout << "neg_int (dec) : " << std::dec << neg_int << std::endl;
std::cout << "neg_int (hex) : " << std::hex << neg_int << std::endl;
std::cout << "neg_int (oct) : " << std::oct << neg_int << std::endl;

std::cout << std::endl;
std::cout << "double_var in different bases : " << std::endl;
std::cout << "double_var (dec) : " << std::dec << double_var << std::endl;
std::cout << "double_var (hex) : " << std::hex << double_var << std::endl;
std::cout << "double_var (oct) : " << std::oct << double_var << std::endl;
```

```
pos_int in different bases :
pos_int (dec) : 717171
pos_int (hex) : af173
pos_int (oct) : 2570563

neg_int in different bases :
neg_int (dec) : -47347
neg_int (hex) : ffff470d
neg_int (oct) : 37777643415

double_var in different bases :
double_var (dec) : 498.32
double_var (hex) : 498.32
double_var (oct) : 498.32
```



## std::showbase

//showbase and noshowbase : show the base for integral types

```
pos_int = 717171;
```

```
std::cout << "pos_int (noshowbase : default) : " << std::endl;
std::cout << "pos_int (dec) : " << std::dec << pos_int << std::endl;
std::cout << "pos_int (hex) : " << std::hex << pos_int << std::endl;
std::cout << "pos_int (oct) : " << std::oct << pos_int << std::endl;
```

```
std::cout << "pos_int (showbase) : " << std::endl;
std::cout << std::showbase;
std::cout << "pos_int (dec) : " << std::dec << pos_int << std::endl;
std::cout << "pos_int (hex) : " << std::hex << pos_int << std::endl;
std::cout << "pos_int (oct) : " << std::oct << pos_int << std::endl;
```

```
pos_int (noshowbase : default) :
pos_int (dec) : 717171
pos_int (hex) : af173
pos_int (oct) : 2570563
pos_int (showbase) :
pos_int (dec) : 717171
pos_int (hex) : 0xaf173
pos_int (oct) : 02570563
```



## std::uppercase

```
//uppercase and nouppercase
```

```
pos_int = 717171;
```

```
std::cout << "pos_int (nouppercase : default) : " << std::endl;
std::cout << "pos_int (dec) : " << std::dec << pos_int << std::endl;
std::cout << "pos_int (hex) : " << std::hex << pos_int << std::endl;
std::cout << "pos_int (oct) : " << std::oct << pos_int << std::endl;

std::cout << std::endl;
std::cout << "pos_int (uppercase) : " << std::endl;
std::cout << std::uppercase;
std::cout << "pos_int (dec) : " << std::dec << pos_int << std::endl;
std::cout << "pos_int (hex) : " << std::hex << pos_int << std::endl;
std::cout << "pos_int (oct) : " << std::oct << pos_int << std::endl;
```

```
pos_int (nouppercase : default) :
pos_int (dec) : 717171
pos_int (hex) : 0xaf173
pos_int (oct) : 02570563

pos_int (uppercase) :
pos_int (dec) : 717171
pos_int (hex) : 0XAF173
pos_int (oct) : 02570563
```

```
double a{ 3.1415926535897932384626433832795 };
double b{ 2006.0 };
double c{ 1.34e-10 };

std::cout << std::endl;
std::cout << "double values (default ) : " << std::endl;
std::cout << "a : " << a << std::endl;
std::cout << "b : " << b << std::endl;
std::cout << "c : " << c << std::endl;

std::cout << std::endl;
std::cout << "double values (fixed) : " << std::endl;
std::cout << std::fixed;
std::cout << "a : " << a << std::endl;
std::cout << "b : " << b << std::endl;
std::cout << "c : " << c << std::endl;
```

```
double values (default : use scientific where necessary) :
a : 3.14159
b : 2006
c : 1.34E-10

double values (fixed) :
a : 3.141593
b : 2006.000000
c : 0.000000
```

## std::scientific – std::fixed (contd)

```
std::cout << std::endl;
std::cout << "double values (scientific) : " << std::endl;
std::cout << std::scientific;
std::cout << "a : " << a << std::endl;
std::cout << "b : " << b << std::endl;
std::cout << "c : " << c << std::endl;

std::cout << std::endl;
std::cout << "double values (back to defaults) : " << std::endl;
std::cout.unsetf(std::ios::scientific | std::ios::fixed);
std::cout << "a : " << a << std::endl;
std::cout << "b : " << b << std::endl;
std::cout << "c : " << c << std::endl;
```

```
double values (scientific) :
a : 3.141593E+00
b : 2.006000E+03
c : 1.340000E-10

double values (back to defaults) :
a : 3.14159
b : 2006
c : 1.34E-10
```

## std::setprecision

```
//setprecision() : the number of digits printed out for a floating point.  
// Default is 6  
a = 3.1415926535897932384626433832795;  
  
std::cout << std::endl;  
std::cout << "a (default precision(6)) : " << a << std::endl;  
std::cout << std::setprecision(10);  
std::cout << "a (precision(10)) : " << a << std::endl;  
std::cout << std::setprecision(20);  
std::cout << "a (precision(20)) : " << a << std::endl;  
std::cout << std::setprecision(50);  
std::cout << "a (precision(50)) : " << a << std::endl;  
  
//Reset precision to 6  
std::cout << std::setprecision(6) << std::endl;
```

Select Microsoft Visual Studio Debug Console

```
a (default precision(6)) : 3.14159  
a (precision(10)) : 3.141592654  
a (precision(20)) : 3.141592653589793116  
a (precision(50)) : 3.141592653589793115997963468544185161590576171875
```

## std::showpoint

```
//showpoint and noshowpoint : show trailing zeros if necessary
```

```
double d {34.1};  
double e {101.99};  
double f {12.0};  
int     g {45};
```

```
std::cout << std::endl;  
std::cout << "noshowpoint (default) : " << std::endl;  
std::cout << "d : " << d << std::endl;  
std::cout << "e : " << e << std::endl;  
std::cout << "f : " << f << std::endl;  
std::cout << "g : " << g << std::endl;
```

```
std::cout << std::endl;  
std::cout << "showpoint: " << std::endl;  
std::cout << std::showpoint;  
std::cout << "d : " << d << std::endl;  
std::cout << "e : " << e << std::endl;  
std::cout << "f : " << f << std::endl;  
std::cout << "g : " << g << std::endl;
```

```
noshowpoint (default) :
```

```
d : 34.1  
e : 101.99  
f : 12  
g : 45
```

```
showpoint:
```

```
d : 34.1000  
e : 101.990  
f : 12.0000  
g : 45
```

Manipulator(s)	header	Purpose
std::endl	<ostream>	Insert new line character
std::flush	<ostream>	Flush the output stream
std::setw()	<iomanip>	changes the width of the next input/output field
std::left, std::right, std::internal	<ios>	Value Justification
std::boolalpha and std::noboolalpha	<ios>	Bool output format
std::showpos and std::noshowpos	<ios>	Show + sign for positive numbers
std::dec, std::hex, std::oct	<ios>	Controls the default number system
std::showbase and std::noshowbase	<ios>	Include prefix to show base
std::uppercase and std::nouppercase	<ios>	Show digits in uppercase

Manipulator(s)	header	Purpose
std::fixed and std::scientific	<ios>	Controls floating point output format
std::setprecision()	<iomanip>	Number of digits used to represent a floating point type
std::setfill()	<iomanip>	Changes the fill character
std::showpoint and std::noshowpoint	<ios>	Controls whether trailing zeros are shown

Reference Doc

<https://en.cppreference.com/w/cpp/io/manip>





# Numeric Limits

```
#include <limits>
```

```
std::numeric_limits<T>::min()
```

```
std::numeric_limits<T>::max()
```

```
std::numeric_limits<T>::lowest()
```

Floating point

$-3.40282e+38$

lowest

0

$1.17549e-38$

min

$3.40282e+38$

max

85

Signed Integer

lowest

0

-32768

min

32767

max

86

Unsigned Integer(Short)

lowest

0

0

min

65535

max

87

Reference Doc

[https://en.cppreference.com/w/cpp/types/numeric\\_limits](https://en.cppreference.com/w/cpp/types/numeric_limits)



Slide intentionally left empty

# Math Functions

```
#include <cmath>
```

`std::floor()`

`std::ceil()`

`std::abs()`

`std::cos()`

`std::exp()`

`std::log()`

`std::pow()`

`std::sqrt()`

`std::round`

`std::sin()`

`std::tan()`

92

Reference Doc

<https://en.cppreference.com/w/cpp/header/cmath>

```
→ double weight { 7.7 };  
  
//floor  
std::cout << "Weight rounded to floor is : " << std::floor(weight) << std::endl;  
  
//ceil  
std::cout << "Weight rounded to ceil is : " << std::ceil(weight) << std::endl;
```

```
//abs
double savings {-5000 };
weight = 7.7;
std::cout << "Abs of weight is : " << std::abs(weight) << std::endl;
std::cout << "Abs of savings is : " << std::abs(savings) << std::endl;
```


$$\exp(x) = e^x$$



```
//exp :  $f(x) = e^x$  , where  $e = 2.71828$  .  
double exponential = std::exp(10);  
std::cout << "The exponential of 10 is : " << exponential << std::endl;
```

```
//pow  
std::cout << "3 ^ 4 is : " << std::pow(3,4) << std::endl;  
std::cout << "9^3 is : " << std::pow(9,3) << std::endl;
```

```
//log : reverse function of pow. if  $2^3 = 8$  , log 8 in base 2 = 3. Log is like asking
// to which exponent should we elevate 2 to get eight ? Log, by default computes the log
// in base e. There also is another function which uses base 10 called log10

// Try the reverse operation of  $e^4 = 54.59$  , it will be log 54.59 in base e = ?
std::cout << "Log ; to get 54.59, you would elevate e to the power of : " << std::log(54.59) << std::endl;

//log10 ,  $10^4 = 10000$  , to get 10k , you'd need to elevate 10 to the power of ? , this is log in base 10
std::cout << "To get 1000, you'd need to elevate 10 to the power of : " << std::log10(10000) << std::endl;
```

```
//sqrt
std::cout << "The square root of 81 is : " << std::sqrt(81) << std::endl;

//round. Halfway points are rounded away from 0. 2,5 is rounded to 5 for example
std::cout << "3.654 rounded to : " << std::round(3.654) << std::endl;
std::cout << "2.5 is rounded to : " << std::round(2.5) << std::endl;
std::cout << "2.4 is rounded to : " << std::round(2.4) << std::endl;
```

## Trigonometric functions

<b>sin</b> <b>sinf</b> (C++11) <b>sinl</b> (C++11)	computes sine ( $\sin x$ ) (function)
<b>cos</b> <b>cosf</b> (C++11) <b>cosl</b> (C++11)	computes cosine ( $\cos x$ ) (function)
<b>tan</b> <b>tanf</b> (C++11) <b>tanl</b> (C++11)	computes tangent ( $\tan x$ ) (function)
<b>asin</b> <b>asinf</b> (C++11) <b>asinl</b> (C++11)	computes arc sine ( $\arcsin x$ ) (function)
<b>acos</b> <b>acosf</b> (C++11) <b>acosl</b> (C++11)	computes arc cosine ( $\arccos x$ ) (function)
<b>atan</b> <b>atanf</b> (C++11) <b>atanl</b> (C++11)	computes arc tangent ( $\arctan x$ ) (function)
<b>atan2</b> <b>atan2f</b> (C++11) <b>atan2l</b> (C++11)	arc tangent, using signs to determine quadrants (function)

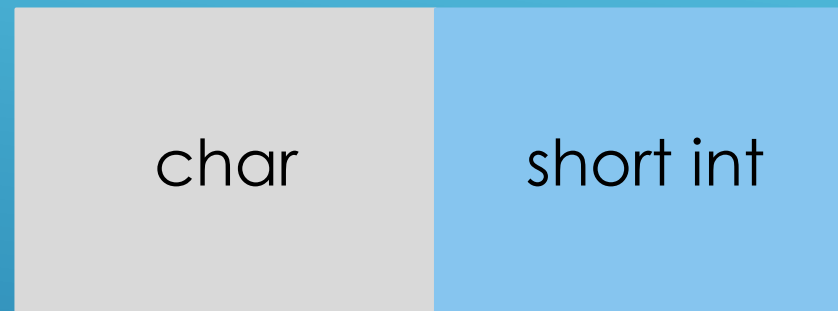
Slide intentionally left empty

# Weird Integral Types

103

Integral types less than 4 bytes in size  
don't support arithmetic operations





105

```
short int var1 {10};
short int var2 {20};

char var3 {40};
char var4 {50};

std::cout << "size of var1 : " << sizeof(var1) << std::endl;
std::cout << "size of var2 : " << sizeof(var2) << std::endl;
std::cout << "size of var3 : " << sizeof(var3) << std::endl;
std::cout << "size of var4 : " << sizeof(var4) << std::endl;

auto result1 = var1 + var2 ;
auto result2 = var3 + var4;

std::cout << "size of result1 : " << sizeof(result1) << std::endl;
std::cout << "size of result2 : " << sizeof(result2) << std::endl;
```

```
short int var1 {10};  
short int var2 {20};
```

```
char var3 {40};  
char var4 {50};
```

```
std::cout << "size of var1 : " << sizeof(var1) << std::endl;  
std::cout << "size of var2 : " << sizeof(var2) << std::endl;  
std::cout << "size of var3 : " << sizeof(var3) << std::endl;  
std::cout << "size of var4 : " << sizeof(var4) << std::endl;
```

```
auto result1 = var1 + var2 ;  
auto result2 = var3 + var4;
```

Conversion to int

```
std::cout << "size of result1 : " << sizeof(result1) << std::endl;  
std::cout << "size of result2 : " << sizeof(result2) << std::endl;
```

The same behavior is present on other operators like bitwise shift operators (  $\gg$  and  $\ll$  ) .

# Operations on data : Summary

109

## Operations on Data

- . Introduction
- . Basic Operations
- . Precedence and Associativity
- . Prefix-Postfix Increment-Decrement
- . Compound Assignment Operators
- . Relational operators
- . Logical Operators
- . Output Formatting
- . Numeric Limits
- . Math Functions
- . Weird Integral Types

Slide intentionally left empty