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: Enums and type aliases

Slides

Slide intentionally left empty

Enums And Type Aliases: Introduction



unsigned long long int huge_num {18'446'744'073'709'551ull};

Slide intentionally left empty

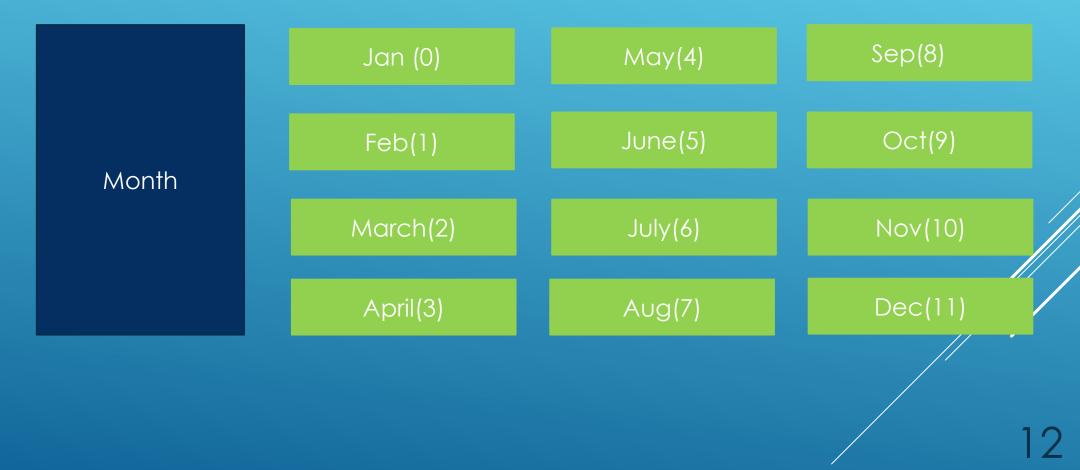
Enum Classes



enum Declaration

Using the enum

Each enumeration is represented by an integral value under the hood



Customize integral values

```
enum class Month {
    Jan = 1 , Feb, Mar, Apr ,
    May, Jun, Jul, Aug = 18,
    Sep, Oct, Nov, Dec
};
```

Can even have negative integral values associated with an enumerator

Multiple enumerators for the same value

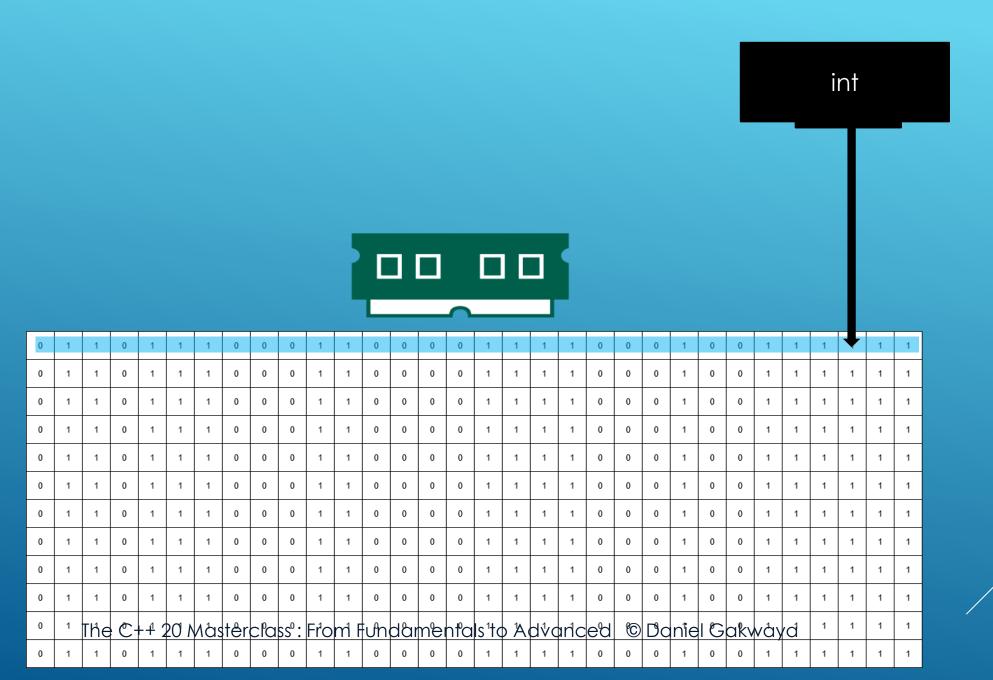
```
enum class Month {
        Jan=1 ,January =1, Feb=2,February =2, Mar,March=3, Apr ,
        May, Jun, Jul, Aug,
        Sep, Oct, Nov, Dec
};
```

Enumerators in terms of others

```
enum class Month {
    Jan = 1, January = Jan,
    Feb , February = Feb,
    Mar , March = Mar,
    Apr , April = Apr,
    May,
    Jun, June = Jun,
    Jul, July = Jul,
    Aug, August = Aug,
    Sep, September = Sep,
    Oct, October = Oct,
    Nov, November = Nov,
    Dec, December = Dec
};
```

Size of an enum

int is the default type associated with enums in C++



4 bytes

-2,147,483,648 ~ 2,147,483,647

Compiler error

```
enum class Month {
    Jan = -30, January = Jan,
    Feb , February = Feb,
    Mar , March = Mar,
    Apr , April = Apr,
    May,
    Jun, June = Jun,
    Jul, July = Jul,
    Aug, August = Aug,
    Sep, September = Sep,
    Oct, October = Oct,
    Nov = 2'147'483'647, November = Nov,
    Dec, December = Dec
};
```

Custom type

```
enum class Month : unsigned char {
    Jan = 0 , January = Jan
    Feb , February = Feb,
   Mar, March = Mar,
    Apr , April = Apr ,
   May,
    Jun, June = Jun,
    Jul, July = Jul,
    Aug, August = Aug,
    Sep, September = Sep,
    Oct, October = Oct,
    Nov, November = Nov,
    Dec, December = Dec
};
```

Custom type

```
enum class Month : unsigned char {
    Jan = 0, January = Jan,
    Feb , February = Feb,
   Mar, March = Mar,
    Apr , April = Apr ,
   May,
    Jun, June = Jun,
    Jul, July = Jul,
    Aug, August = Aug,
    Sep, September = Sep,
    Oct, October = Oct,
    Nov, November = Nov,
    Dec, December = Dec
};
```

Compiler Error

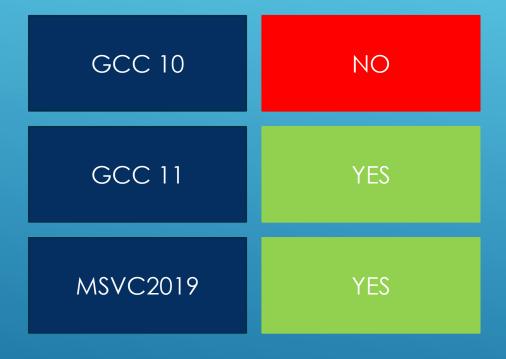
```
enum class Month : unsigned char {
    Jan = 0 , January = Jan
    Feb , February = Feb,
    Mar , March = Mar,
    Apr , April = Apr,
    May,
    Jun, June = Jun,
    Jul, July = Jul,
    Aug, August = Aug,
    Sep, September = Sep,
    Oct, October = Oct,
    Nov = 300, November = Nov,
    Dec, December = Dec
};
```

Slide intentionally left empty

using enum

```
enum class Month {
    Jan , Feb, Mar, Apr ,
   May, Jun, Jul, Aug,
   Sep, Oct, Nov, Dec
};
std::string_view moth_to_string(Month month){
    switch(month){
        case Month::Jan : return "January";
        case Month::Feb : return "February";
        case Month::Mar : return "March";
        case Month::Apr : return "April";
        case Month::May : return "May";
        case Month::Jun : return "June";
        case Month::Jul : return "July";
        case Month::Aug : return "Aug";
        case Month::Sep : return "Sep";
        case Month::Oct : return "Oct";
        default : return "None";
```

```
enum class Month {
    Jan , Feb, Mar, Apr ,
   May, Jun, Jul, Aug,
   Sep, Oct, Nov, Dec
};
std::string_view moth_to_string(Month month){
    switch(month){
        using enum Month;
       case Jan : return "January";
        case Feb : return "February";
       case Mar : return "March";
        case Apr : return "April";
       case May : return "May";
        case Jun : return "June";
        case Jul : return "July";
        case Aug : return "August";
        case Sep : return "September";
        case Oct : return "October";
        case Nov : return "November";
        case Dec : return "December";
```



Slide intentionally left empty

Old Enums

```
enum Direction {
    TopLeft,
    TopRight,
    Center,
    BottomLeft,
    BottomRight
};
enum Tool {
    Pen, Ikaramu = Pen,
   Marker,
    Eraser,
    Rectangle,
    Circle,
    PaintBucket
};
```

Legacy enums implicitly transform to int, which is good for std::cout, but that introduces the problem that we can compare different enum types which is very confusing.

Legacy Enum Problems

```
enum Direction {
    TopLeft, TopRight, Center, BottomLeft, BottomRight
};
enum Tool {
    Pen, Marker, Eraser, Rectangle, Circle, PaintBucket
};
Direction direction{BottomLeft};
Tool tool {Pen};
std::cout << std::boolalpha;</pre>
std::cout << "tool > direction : " << (tool > direction) << std::endl;</pre>
std::cout << "direction : " << direction << std::endl;</pre>
std::cout << "Tool : " << tool << std::endl;</pre>
```

Slide intentionally left empty

Type Aliases

unsigned long long int huge_num {18'446'744'073'709'551ull};

Type alias

```
using HugeInt = unsigned long long int;
HugeInt huge_num {18'446'744'073'709'551ull};
std::cout << "huge_num : " << huge_num << std::endl;
HugeInt other_huge_num {18'112'334'073'709'752ull};
std::cout << "other_huge_num : " << other_huge_num << std::endl;</pre>
```

typedef

```
//typedef (older) syntax : Not recommended in new code
typedef unsigned long long int HugeInt;
HugeInt huge_num {18'446'744'073'709'551ull};
std::cout << "huge_num : " << huge_num << std::endl;</pre>
```

Slide intentionally left empty

Enums And Type Aliases: Summary



Using the enum

unsigned long long int huge_num {18'446'744'073'709'551ull};

Type alias

```
using HugeInt = unsigned long long int;
HugeInt huge_num {18'446'744'073'709'551ull};
std::cout << "huge_num : " << huge_num << std::endl;
HugeInt other_huge_num {18'112'334'073'709'752ull};
std::cout << "other_huge_num : " << other_huge_num << std::endl;</pre>
```

Slide intentionally left empty