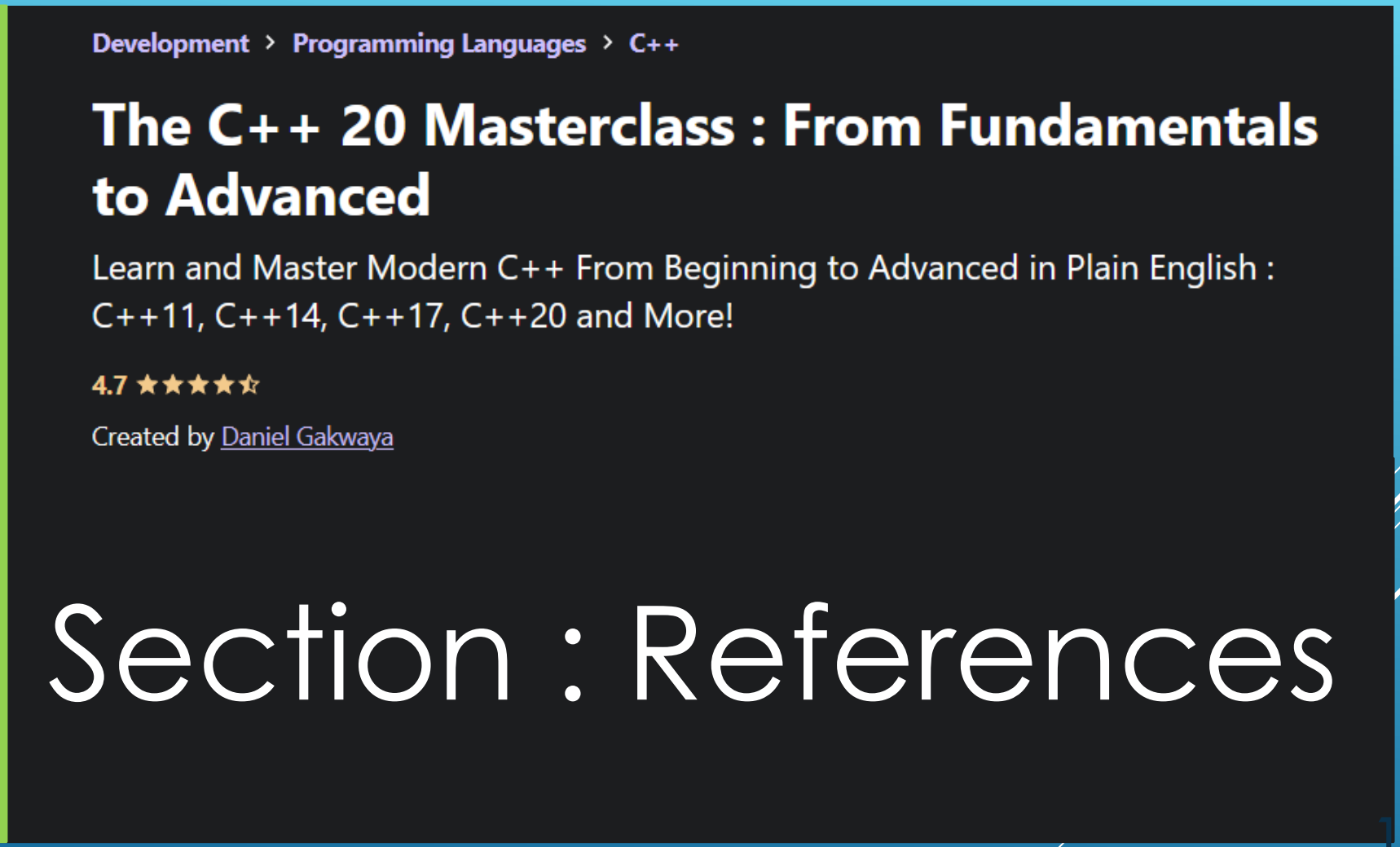
Slides



Section : References

Slide intentionally left empty

References : Introduction


int
var
0x12ab
33


int
var
0x12ab
33
var_alias

Slide intentionally left empty

Declaring and Using References


int
var
0x12ab
33


int
var
0x12ab
33
var_alias

Declaring and using references

```
int int_value {45};
double double_value{33.65};

int& reference_to_int_value_1{int_value};  // Assign throught initialization
int& reference_to_int_value_2 = int_value; // Assign throught assignment
double& reference_to_double_value_1 {double_value};

//You have to declare and initialize in one statement
//int& some_reference; // Error

std::cout << "int_value : " << int_value << std::endl;
std::cout << "double_value : " << double_value << std::endl;
std::cout << "reference_to_int_value_1 : " << reference_to_int_value_1 << std::endl;
std::cout << "reference_to_int_value_2 : " << reference_to_int_value_2 << std::endl;
std::cout << "reference_to_double_value1 : "  << reference_to_double_value_1 << std::endl;
std::cout << "&int_value : " << &int_value << std::endl;
std::cout << "&double_value : " << &double_value << std::endl;
std::cout << "&reference_to_int_value_1 : " << &reference_to_int_value_1 << std::endl;
std::cout << "&reference_to_int_value_2 : " << &reference_to_int_value_2 << std::endl;
std::cout << "&reference_to_double_value_1 : " << &reference_to_double_value_1 << std::endl;
std::cout << "sizeof(int) : " << sizeof(int) << std::endl;
std::cout << "sizeof(int&) : " << sizeof(int&) << std::endl;
std::cout << "sizeof(reference_to_int_value_1) : " << sizeof(reference_to_int_value_1) << std::endl;
```

Modify data through reference

```
//Modify through reference : changes reflect to original variable
std::cout << std::endl;
std::cout << "Modifying data through reference : " << std::endl;

reference_to_int_value_1 = 55;

//Print out after modification of int_value
std::cout << "int_value : " << int_value << std::endl;
std::cout << "double_value : " << double_value << std::endl;
std::cout << "reference_to_int_value_1 : " << reference_to_int_value_1 << std::endl;
std::cout << "reference_to_int_value_2 : " << reference_to_int_value_2 << std::endl;
std::cout << "reference_to_double_value1 : "  << reference_to_double_value_1 << std::endl;
std::cout << "&int_value : " << &int_value << std::endl;
std::cout << "&double_value : " << &double_value << std::endl;
std::cout << "&reference_to_int_value_1 : " << &reference_to_int_value_1 << std::endl;
std::cout << "&reference_to_int_value_2 : " << &reference_to_int_value_2 << std::endl;
std::cout << "&reference_to_double_value_1 : " << &reference_to_double_value_1 << std::endl;
std::cout << "sizeof(int) : " << sizeof(int) << std::endl;
std::cout << "sizeof(int&) : " << sizeof(int&) << std::endl;
std::cout << "sizeof(reference_to_int_value_1) : " << sizeof(reference_to_int_value_1) << std::endl;
```

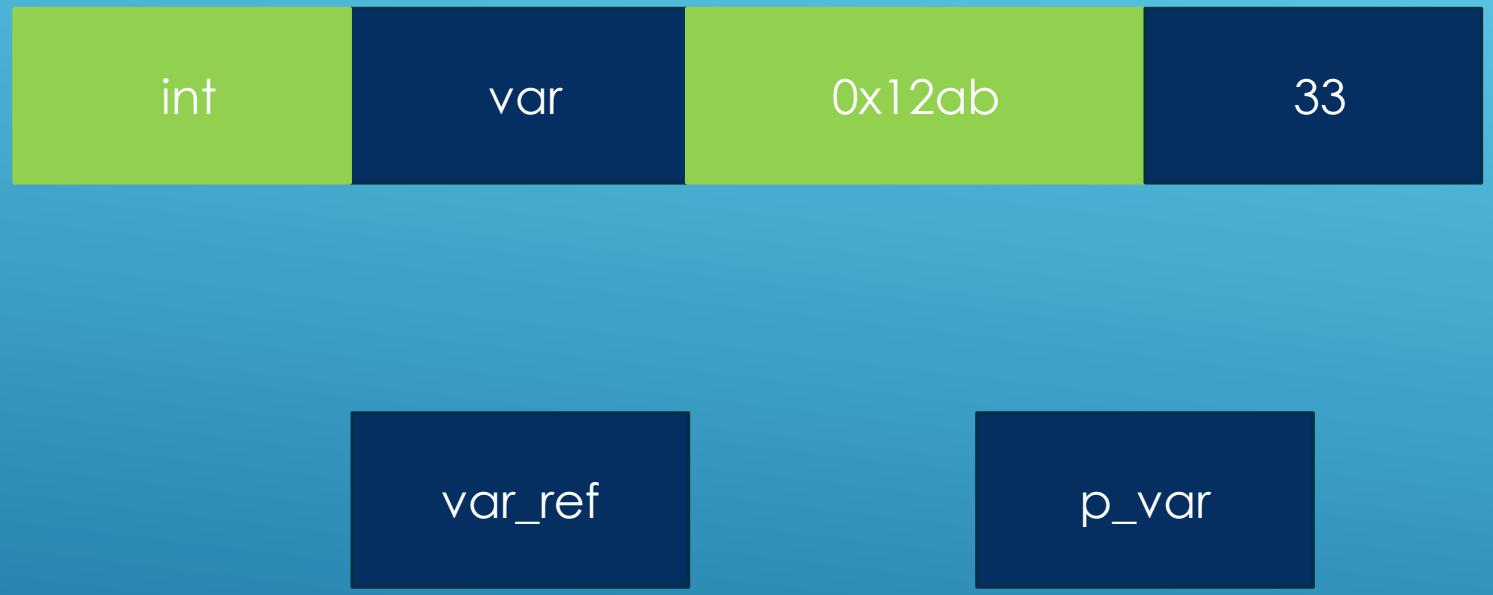
Modify data through original variable

```
//Modifying data directly : changes reflect even in references
std::cout << std::endl;
std::cout << "Modifying data directly : " << std::endl;
double_value = 9.99;

//Print out after modification of double_value
std::cout << "int_value : " << int_value << std::endl;
std::cout << "double_value : " << double_value << std::endl;
std::cout << "reference_to_int_value_1 : " << reference_to_int_value_1 << std::endl;
std::cout << "reference_to_int_value_2 : " << reference_to_int_value_2 << std::endl;
std::cout << "reference_to_double_value1 : "  << reference_to_double_value_1 << std::endl;
std::cout << "&int_value : " << &int_value << std::endl;
std::cout << "&double_value : " << &double_value << std::endl;
std::cout << "&reference_to_int_value_1 : " << &reference_to_int_value_1 << std::endl;
std::cout << "&reference_to_int_value_2 : " << &reference_to_int_value_2 << std::endl;
std::cout << "&reference_to_double_value_1 : " << &reference_to_double_value_1 << std::endl;
std::cout << "sizeof(int) : " << sizeof(int) << std::endl;
std::cout << "sizeof(int&) : " << sizeof(int&) << std::endl;
std::cout << "sizeof(reference_to_int_value_1) : " << sizeof(reference_to_int_value_1) << std::endl;
```

Slide intentionally left empty

Comparing References to Pointers


int
var
0x12ab
33


int
var
0x12ab
33
var_ref
p_var

References	Pointers
• Don't use dereferencing for reading and writing • Can't be changed to reference something else • Must be initialized at declaration	• Must go through dereference operator to read/write through pointed to value • Can be changed to point somewhere else • Can be declared un-initialized (will contain garbage addresses)

Declaration and reading

```
//Declare pointer and reference

double double_value {12.34};

double& ref_double_value {double_value}; // Reference to double_value

double* p_double_value {&double_value}; //Pointer to double_value


//Reading
std::cout << "double_value : " << double_value << std::endl;
std::cout << "ref_double_value : " << ref_double_value << std::endl;
std::cout << "p_double_value : " << p_double_value << std::endl;
std::cout << "*p_double_value : " << *p_double_value << std::endl;
```

Writing

```
//Writting through pointer
std::cout << std::endl;
std::cout << "Writting through pointer : " << std::endl;

*p_double_value = 15.44;

std::cout << "double_value : " << double_value << std::endl;
std::cout << "ref_double_value : " << ref_double_value << std::endl;
std::cout << "p_double_value : " << p_double_value << std::endl;
std::cout << "*p_double_value : " << *p_double_value << std::endl;

//Writting through reference
std::cout << std::endl;
std::cout << "Writting through reference : " << std::endl;

ref_double_value = 18.44;

std::cout << "double_value : " << double_value << std::endl;
std::cout << "ref_double_value : " << ref_double_value << std::endl;
std::cout << "p_double_value : " << p_double_value << std::endl;
std::cout << "*p_double_value : " << *p_double_value << std::endl;
```

Can't make a reference refer to something else

```
double double_value {12.34};

double& ref_double_value {double_value}; // Reference to double_value

double other_double_value{100.23};

//This works, but it doesn't make ref_double_value reference other_double_value
//it merely changes the value referenced by ref_double_value to 100.23
//Visualize this in slides.
ref_double_value = other_double_value;


//If you change ref_double_value now, other_double_value stays the same
//proving that ref_double_value is not referencing other_double_value.

ref_double_value = 333.33;
```

A pointer can point somewhere else

```
//A pointer can point somewhere else
std::cout << std::endl;
std::cout << "A pointer can point somewhere else : " << std::endl;

p_double_value = & other_double_value;

std::cout << "double_value : " << double_value << std::endl;
std::cout << "ref_double_value : " << ref_double_value << std::endl;
std::cout << "p_double_value : " << p_double_value << std::endl;
std::cout << "*p_double_value : " << *p_double_value << std::endl;
std::cout << "other_double_value : " << other_double_value << std::endl;

std::cout << std::endl;
std::cout << "Changing the now pointed to value : " << std::endl;

*p_double_value = 555.66;

std::cout << "double_value : " << double_value << std::endl;
std::cout << "ref_double_value : " << ref_double_value << std::endl;
std::cout << "p_double_value : " << p_double_value << std::endl;
std::cout << "*p_double_value : " << *p_double_value << std::endl;
std::cout << "other_double_value : " << other_double_value << std::endl;
```

References are somewhat like const pointers

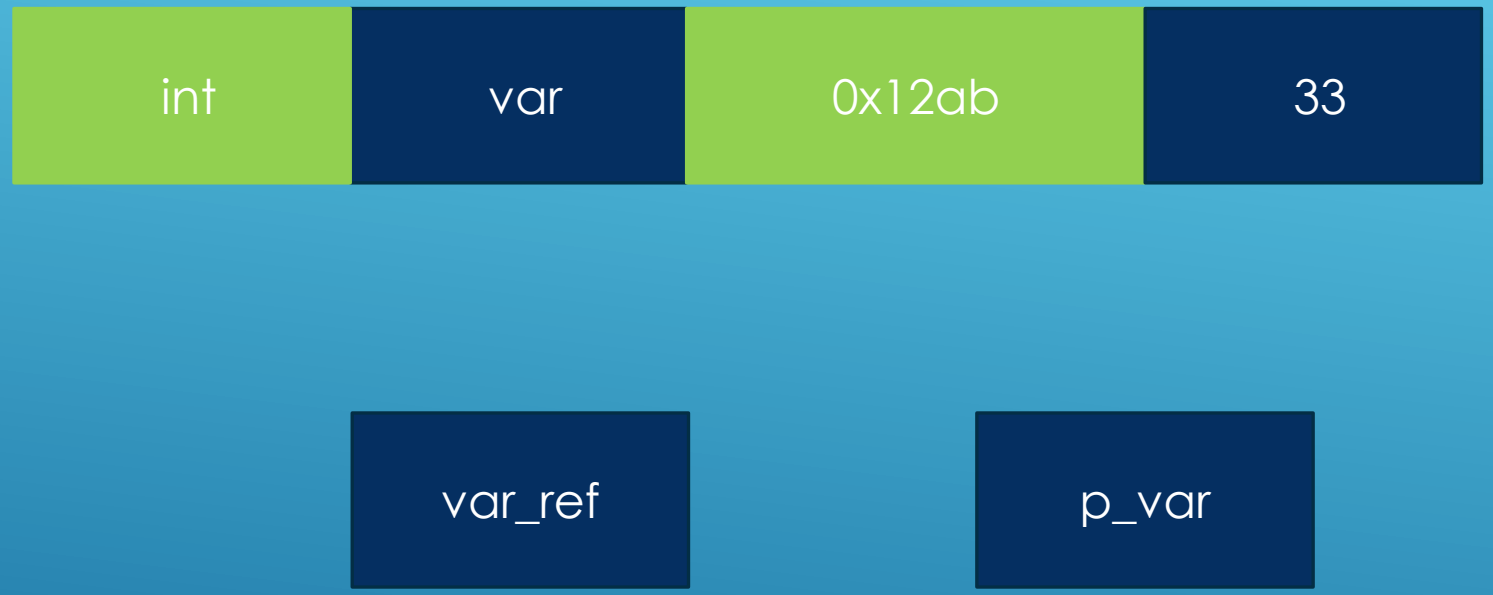
```
//References behave like constant pointers, but they have
//a much friendlier syntax as they don't require dereferencing
//to read and write through referenced data.

double *const const_p_double_value {&double_value};

const_p_double_value = &other_double_value;// Error
```

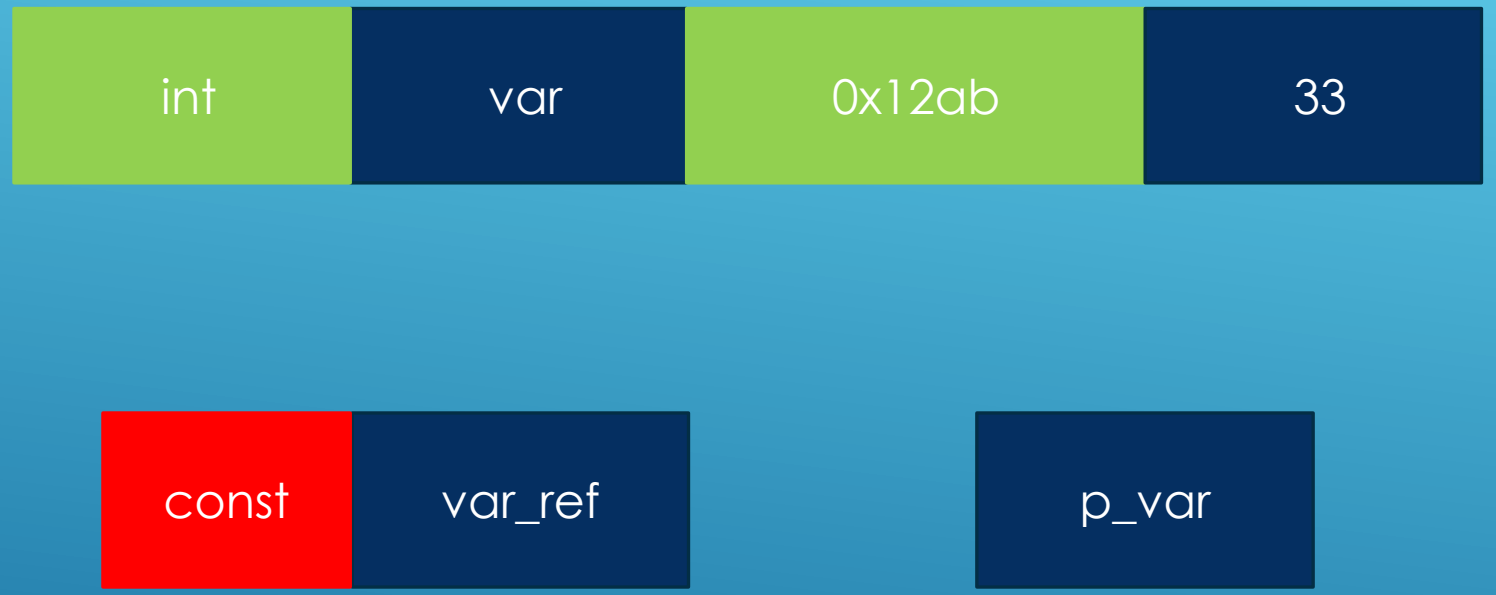
There is not syntax to modify what is referred to by a reference

Slide intentionally left empty

References and const


int
var
0x12ab
33


int
var
0x12ab
33
var_ref
p_var


int
var
0x12ab
33
const
var_ref
p_var

Non const reference

```
//Non const reference
std::cout << std::endl;
std::cout << "Non const reference : " << std::endl;
int age {27};
int& ref_age{age};

std::cout << "age : " << age << std::endl;
std::cout << "ref_age : " << ref_age << std::endl;


//Can modify original variable through reference

std::cout << std::endl;
std::cout << "Modify original variable through reference : " << std::endl;

ref_age++; //Mofify through reference

std::cout << "age : " << age << std::endl;
std::cout << "ref_age : " << ref_age << std::endl;
```

const reference

```
//const reference

std::cout << std::endl;
std::cout << "Const references : " << std::endl;
age = 30;
const int& const_ref_age{age};

std::cout << "age : " << age << std::endl;
std::cout << "const_ref_age : " << const_ref_age << std::endl;

//Try to modify throug const reference
const_ref_age = 31; // Error
```

Duplicate const reference behavior with pointers

```
//Can achieve the same thing as const ref with pointer : const pointer to const data
//Remember that a reference by default is just like a const pointer. All we need
//to do is make the const pointer point to const data

const int* const const_ptr_to_const_age{&age};

*const_ptr_to_const_age = 32; // Error
```

No such thing

```
const int& const weird_ref_age{age}; // Error
```

const applies to reference variable name. Not to original variable

```
//const reference

std::cout << std::endl;
std::cout << "Const references : " << std::endl;
age = 30;
const int& const_ref_age{age};

std::cout << "age : " << age << std::endl;
std::cout << "const_ref_age : " << const_ref_age << std::endl;

//Try to modify throug const reference
const_ref_age = 31; // Error
```

Slide intentionally left empty

Range based for loop with references

```cpp
int scores[] {1,2,3,4,5,6,7,8,9,10};

//Printing positions
std::cout << std::endl;

std::cout << "Scores : ";
for ( auto score : scores){
    std::cout << " " << score ;
}
std::cout << std::endl;
```

Modify array data in a range based for loop[FAIL]

```
//Modification attempt
for ( auto score : scores){
    score = score*10; // Modifies copy of value in scores
}

//Print out
std::cout << "Scores : ";
for ( auto score : scores){
    std::cout << " " << score ;
}
std::cout << std::endl;
```

References to the rescue!

```
for(auto& score : scores){
    score = score * 10;
}

//Print out
std::cout << "Scores : ";
for ( auto score : scores){
    std::cout << " " << score ;
}
std::cout << std::endl;
```

Slide intentionally left empty

References : Summary


int
var
0x12ab
33


int
var
0x12ab
33
var_alias

- Declaring and using references
- Pointers Vs References
- References and const