References

```
/*

    Topics:
      . References:
        . Declaring and using references
        . Comparing pointers to references
        . References and const
        . Range based for loops

*/
```

References: Declaration

```cpp
int int_data{ 33 };
double double_data{ 55 };

// References
int &ref_int_data{ int_data };
double &ref_double_data{ double_data };

// Print stuff out
fmt::println("int_data: {}", int_data);
fmt::println("&int_data: {}", fmt::ptr(&int_data));
fmt::println("double_data: {}", double_data);
fmt::println("&double_data: {}", fmt::ptr(&double_data));

fmt::println("======================");

fmt::println("ref_int_data: {}", ref_int_data);
fmt::println("&ref_int_data: {}", fmt::ptr(&ref_int_data));
fmt::println("ref_double_data: {}", ref_double_data);
fmt::println("&ref_double_data: {}", fmt::ptr(&ref_double_data));

int_data = 111;
double_data = 67.2;

//Changes through references are visible through original data
ref_int_data = 1012;
ref_double_data = 1000.45;
```

References VS Pointers

```
// 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
fmt::println("double_value: {} ", double_value);
fmt::println("ref_double_value: {} ", ref_double_value);
fmt::println("p_double_value: {} ", fmt::ptr(p_double_value));
fmt::println("*p_double_value: {}", *p_double_value);

// Writting through pointer
*p_double_value = 15.44;

fmt::println( "double_value: {} ", double_value );
fmt::println( "ref_double_value {}: ", ref_double_value );
fmt::println( "p_double_value: {} ", fmt::ptr(p_double_value ));
fmt::println( "*p_double_value: {} ", *p_double_value );

//Writting through reference
ref_double_value = 18.44;

fmt::println( "double_value: {} ", double_value );
fmt::println( "ref_double_value: {}", ref_double_value );
fmt::println( "p_double_value: {}", fmt::ptr(p_double_value));
fmt::println( "*p_double_value: {}", *p_double_value );
```

References VS Pointers

```
double some_other_double{78.45};

//Make the reference reference something else.
ref_double_value = some_other_double;   // This is not changing the reference,
                    //  it is changing the value of the reference

fmt::println( "Making the reference reference something else..." );
fmt::println( "double_value: {} " , double_value );
fmt::println( "ref_double_value {}: " , ref_double_value );
fmt::println( "p_double_value: {} " , fmt::ptr(p_double_value));
fmt::println( "*p_double_value: {}" , *p_double_value );



//Make the pointer point to something else
p_double_value = &some_other_double;  // The pointer is trully pointing to something else
fmt::println( "Making the pointer point somewhere else..." );
fmt::println( "double_value: {}" , double_value );
fmt::println( "ref_double_value: {}" , ref_double_value );
fmt::println( "&double_value: {}" , fmt::ptr(&double_value )); // an address
fmt::println( "&ref_double_value: {}" , fmt::ptr(&ref_double_value)); // an address
fmt::println( "p_double_value: {}" , fmt::ptr(p_double_value)); //an address
fmt::println( "*p_double_value: {}" , *p_double_value );
```

References and const

```
fmt::println("Non const reference : ");
int age{ 27 };
const int &ref_age{ age };

fmt::println("age: {}", age);
fmt::println("ref_age: {}", ref_age);


//Can't modify original variable through reference
fmt::println("Modify original variable through reference : ");
//ref_age++; //Mofify through reference

fmt::println("age: {}", age);
fmt::println("ref_age: {}", ref_age);
```

Range based for loops with reference

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

// Printing before change

fmt::print("Scores : ");
for (auto score : scores) {
  fmt::print(" {}", score);
}
fmt::print("\n");


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

// Printing after change
fmt::print("Scores : ");
for (auto score : scores) {
  fmt::print(" {}", score);
}
fmt::print("\n");
```