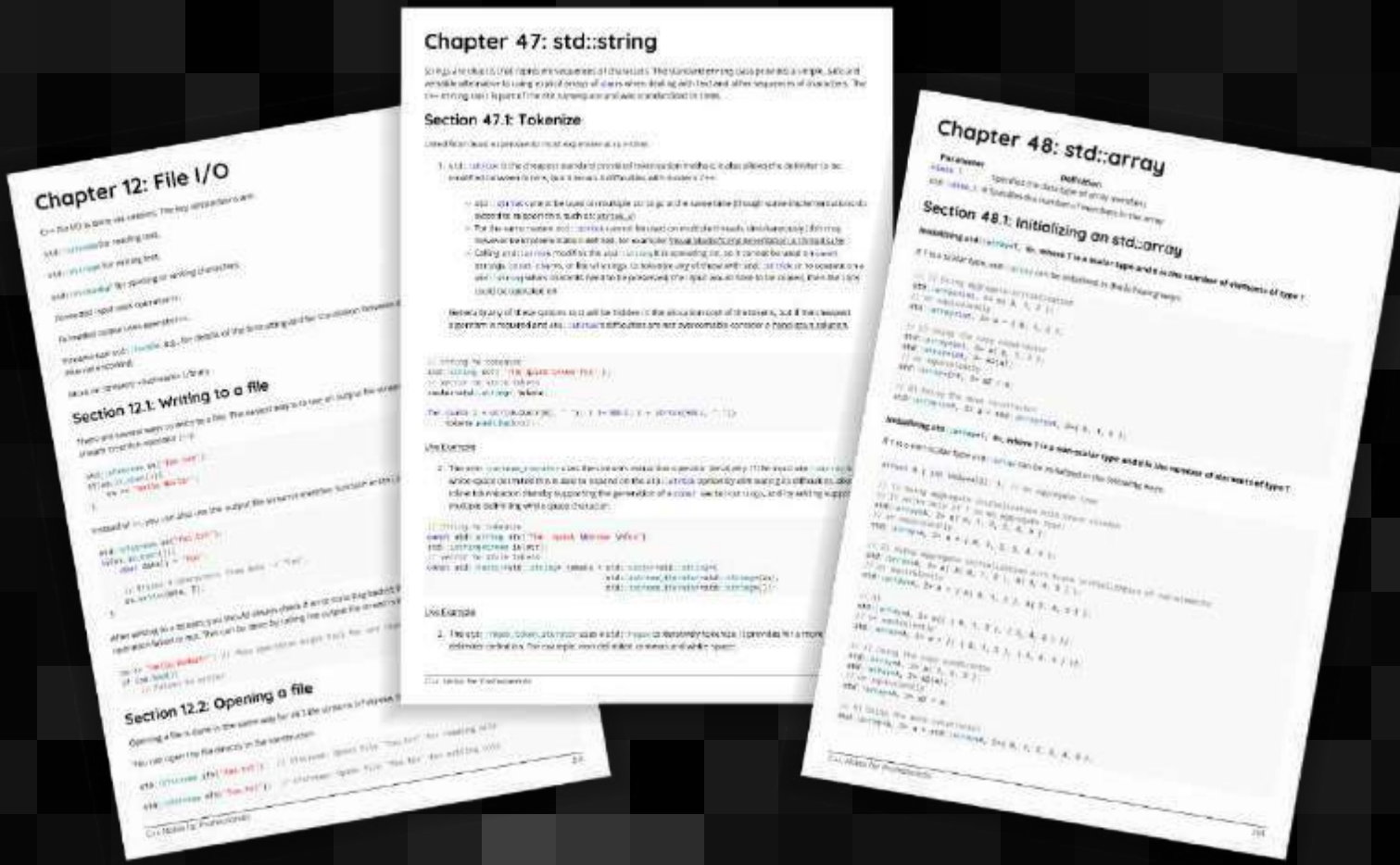


C++

Notes for Professionals



600+ pages
of professional hints and tricks

Contents

About	1
Chapter 1: Getting started with C++	2
Section 1.1: Hello World	2
Section 1.2: Comments	3
Section 1.3: The standard C++ compilation process	5
Section 1.4: Function	5
Section 1.5: Visibility of function prototypes and declarations	8
Section 1.6: Preprocessor	9
Chapter 2: Literals	11
Section 2.1: this	11
Section 2.2: Integer literal	11
Section 2.3: true	12
Section 2.4: false	13
Section 2.5: nullptr	13
Chapter 3: operator precedence	14
Section 3.1: Logical && and operators: short-circuit	14
Section 3.2: Unary Operators	15
Section 3.3: Arithmetic operators	15
Section 3.4: Logical AND and OR operators	16
Chapter 4: Floating Point Arithmetic	17
Section 4.1: Floating Point Numbers are Weird	17
Chapter 5: Bit Operators	18
Section 5.1: - bitwise OR	18
Section 5.2: ^ - bitwise XOR (exclusive OR)	18
Section 5.3: & - bitwise AND	20
Section 5.4: << - left shift	20
Section 5.5: >> - right shift	21
Chapter 6: Bit Manipulation	23
Section 6.1: Remove rightmost set bit	23
Section 6.2: Set all bits	23
Section 6.3: Toggling a bit	23
Section 6.4: Checking a bit	23
Section 6.5: Counting bits set	24
Section 6.6: Check if an integer is a power of 2	25
Section 6.7: Setting a bit	25
Section 6.8: Clearing a bit	25
Section 6.9: Changing the nth bit to x	25
Section 6.10: Bit Manipulation Application: Small to Capital Letter	26
Chapter 7: Bit fields	27
Section 7.1: Declaration and Usage	27
Chapter 8: Arrays	28
Section 8.1: Array initialization	28
Section 8.2: A fixed size raw array matrix (that is, a 2D raw array)	29
Section 8.3: Dynamically sized raw array	29
Section 8.4: Array size: type safe at compile time	30
Section 8.5: Expanding dynamic size array by using std::vector	31

Section 8.6: A dynamic size matrix using <code>std::vector</code> for storage	32
Chapter 9: Iterators	35
Section 9.1: Overview	35
Section 9.2: Vector Iterator	38
Section 9.3: Map Iterator	38
Section 9.4: Reverse Iterators	39
Section 9.5: Stream Iterators	40
Section 9.6: C Iterators (Pointers)	40
Section 9.7: Write your own generator-backed iterator	41
Chapter 10: Basic input/output in c++	43
Section 10.1: user input and standard output	43
Chapter 11: Loops	44
Section 11.1: Range-Based For	44
Section 11.2: For loop	46
Section 11.3: While loop	48
Section 11.4: Do-while loop	49
Section 11.5: Loop Control statements : Break and Continue	50
Section 11.6: Declaration of variables in conditions	51
Section 11.7: Range-for over a sub-range	52
Chapter 12: File I/O	54
Section 12.1: Writing to a file	54
Section 12.2: Opening a file	54
Section 12.3: Reading from a file	55
Section 12.4: Opening modes	57
Section 12.5: Reading an ASCII file into a <code>std::string</code>	58
Section 12.6: Writing files with non-standard locale settings	59
Section 12.7: Checking end of file inside a loop condition, bad practice?	60
Section 12.8: Flushing a stream	61
Section 12.9: Reading a file into a container	61
Section 12.10: Copying a file	62
Section 12.11: Closing a file	62
Section 12.12: Reading a `struct` from a formatted text file	63
Chapter 13: C++ Streams	65
Section 13.1: String streams	65
Section 13.2: Printing collections with <code>iostream</code>	66
Chapter 14: Stream manipulators	68
Section 14.1: Stream manipulators	68
Section 14.2: Output stream manipulators	73
Section 14.3: Input stream manipulators	75
Chapter 15: Flow Control	77
Section 15.1: case	77
Section 15.2: switch	77
Section 15.3: catch	77
Section 15.4: throw	78
Section 15.5: default	79
Section 15.6: try	79
Section 15.7: if	79
Section 15.8: else	80
Section 15.9: Conditional Structures: if, if..else	80

Section 15.10: goto	81
Section 15.11: Jump statements : break, continue, goto, exit	81
Section 15.12: return	84
Chapter 16: Metaprogramming	86
Section 16.1: Calculating Factorials	86
Section 16.2: Iterating over a parameter pack	88
Section 16.3: Iterating with std::integer_sequence	89
Section 16.4: Tag Dispatching	90
Section 16.5: Detect Whether Expression is Valid	90
Section 16.6: If-then-else	92
Section 16.7: Manual distinction of types when given any type T	92
Section 16.8: Calculating power with C++11 (and higher)	93
Section 16.9: Generic Min/Max with variable argument count	94
Chapter 17: const keyword	95
Section 17.1: Avoiding duplication of code in const and non-const getter methods	95
Section 17.2: Const member functions	96
Section 17.3: Const local variables	97
Section 17.4: Const pointers	97
Chapter 18: mutable keyword	99
Section 18.1: mutable lambdas	99
Section 18.2: non-static class member modifier	99
Chapter 19: Friend keyword	101
Section 19.1: Friend function	101
Section 19.2: Friend method	102
Section 19.3: Friend class	102
Chapter 20: Type Keywords	104
Section 20.1: class	104
Section 20.2: enum	105
Section 20.3: struct	106
Section 20.4: union	106
Chapter 21: Basic Type Keywords	108
Section 21.1: char	108
Section 21.2: char16_t	108
Section 21.3: char32_t	108
Section 21.4: int	108
Section 21.5: void	108
Section 21.6: wchar_t	109
Section 21.7: float	109
Section 21.8: double	109
Section 21.9: long	109
Section 21.10: short	110
Section 21.11: bool	110
Chapter 22: Variable Declaration Keywords	111
Section 22.1: decltype	111
Section 22.2: const	111
Section 22.3: volatile	112
Section 22.4: signed	112
Section 22.5: unsigned	112
Chapter 23: Keywords	114

Section 23.1: asm	114
Section 23.2: Different keywords	114
Section 23.3: typename	118
Section 23.4: explicit	119
Section 23.5: sizeof	119
Section 23.6: noexcept	120
Chapter 24: Returning several values from a function	122
Section 24.1: Using std::tuple	122
Section 24.2: Structured Bindings	123
Section 24.3: Using struct	124
Section 24.4: Using Output Parameters	125
Section 24.5: Using a Function Object Consumer	126
Section 24.6: Using std::pair	127
Section 24.7: Using std::array	127
Section 24.8: Using Output Iterator	127
Section 24.9: Using std::vector	128
Chapter 25: Polymorphism	129
Section 25.1: Define polymorphic classes	129
Section 25.2: Safe downcasting	130
Section 25.3: Polymorphism & Destructors	131
Chapter 26: References	133
Section 26.1: Defining a reference	133
Chapter 27: Value and Reference Semantics	134
Section 27.1: Definitions	134
Section 27.2: Deep copying and move support	134
Chapter 28: C++ function "call by value" vs. "call by reference"	138
Section 28.1: Call by value	138
Chapter 29: Copying vs Assignment	140
Section 29.1: Assignment Operator	140
Section 29.2: Copy Constructor	140
Section 29.3: Copy Constructor Vs Assignment Constructor	141
Chapter 30: Pointers	143
Section 30.1: Pointer Operations	143
Section 30.2: Pointer basics	143
Section 30.3: Pointer Arithmetic	145
Chapter 31: Pointers to members	147
Section 31.1: Pointers to static member functions	147
Section 31.2: Pointers to member functions	147
Section 31.3: Pointers to member variables	148
Section 31.4: Pointers to static member variables	148
Chapter 32: The This Pointer	150
Section 32.1: this Pointer	150
Section 32.2: Using the this Pointer to Access Member Data	152
Section 32.3: Using the this Pointer to Differentiate Between Member Data and Parameters	152
Section 32.4: this Pointer CV-Qualifiers	153
Section 32.5: this Pointer Ref-Qualifiers	156
Chapter 33: Smart Pointers	158
Section 33.1: Unique ownership (std::unique_ptr)	158
Section 33.2: Sharing ownership (std::shared_ptr)	159

Section 33.3: Sharing with temporary ownership (std::weak_ptr)	161
Section 33.4: Using custom deleters to create a wrapper to a C interface	163
Section 33.5: Unique ownership without move semantics (auto_ptr)	164
Section 33.6: Casting std::shared_ptr pointers	166
Section 33.7: Writing a smart pointer: value_ptr	166
Section 33.8: Getting a shared_ptr referring to this	168
Chapter 34: Classes/Structures	170
Section 34.1: Class basics	170
Section 34.2: Final classes and structs	170
Section 34.3: Access specifiers	171
Section 34.4: Inheritance	172
Section 34.5: Friendship	174
Section 34.6: Virtual Inheritance	175
Section 34.7: Private inheritance: restricting base class interface	176
Section 34.8: Accessing class members	177
Section 34.9: Member Types and Aliases	178
Section 34.10: Nested Classes/Structures	182
Section 34.11: Unnamed struct/class	186
Section 34.12: Static class members	187
Section 34.13: Multiple Inheritance	191
Section 34.14: Non-static member functions	192
Chapter 35: Function Overloading	195
Section 35.1: What is Function Overloading?	195
Section 35.2: Return Type in Function Overloading	196
Section 35.3: Member Function cv-qualifier Overloading	196
Chapter 36: Operator Overloading	199
Section 36.1: Arithmetic operators	199
Section 36.2: Array subscript operator	200
Section 36.3: Conversion operators	201
Section 36.4: Complex Numbers Revisited	202
Section 36.5: Named operators	206
Section 36.6: Unary operators	208
Section 36.7: Comparison operators	209
Section 36.8: Assignment operator	210
Section 36.9: Function call operator	211
Section 36.10: Bitwise NOT operator	211
Section 36.11: Bit shift operators for I/O	212
Chapter 37: Function Template Overloading	213
Section 37.1: What is a valid function template overloading?	213
Chapter 38: Virtual Member Functions	214
Section 38.1: Final virtual functions	214
Section 38.2: Using override with virtual in C++11 and later	214
Section 38.3: Virtual vs non-virtual member functions	215
Section 38.4: Behaviour of virtual functions in constructors and destructors	216
Section 38.5: Pure virtual functions	217
Chapter 39: Inline functions	220
Section 39.1: Non-member inline function definition	220
Section 39.2: Member inline functions	220
Section 39.3: What is function inlining?	220
Section 39.4: Non-member inline function declaration	221

This is a sample, click download link to get the full Tutorial

CLICK BELOW

