

DETAILED CONTENTS**with STEPS**

Fourth Edition

1. INTRODUCTION	1		
(a) Programming paradigms	1		
1.1. History of C++	6		
1.2. C++ features	7		
1.3. C++ vs C	8		
1.4. Program development	12		
1.5. General structure of C++ program	13		
1.6. Steps in executing a C++ program	18		
C++ compilers	20		
C++ IDE	20		
1.7. Compiling and running C++ program			
in VS code	21		
Errors	28		
Debugging	30		
1.8. Compile and run C++ program in			
Turbo C++	34		
1.9. Advantages and disadvantages of			
C++	35		
1.10. Applications of C++	37		
2. C++ FUNDAMENTALS	41		
2.1. Character set	41		
2.2. Tokens	42		
2.2.1. Identifiers	43		
2.2.2. Keywords	44		
2.2.3. Literals	45		
2.2.3.1. Some other ways to define constants	48		
(a) Manifest constants using			
#define directive	48		
(b) const qualifier	49		
2.3. Variable	50		
2.4. Data types	52		
2.5. Type casting	59		
		1. Implicit type casting	60
		2. Explicit type casting	61
2.6. Operators	61		
2.7. Expression	71		
2.7. Precedence and associativity of opera-			
tors	73		
2.8. Statement	75		
2.8.1. Flow control statements	76		
2.8.1.1. Selection control statements	76		
2.8.1.1.1. if-else	76		
Nested if-else	77		
If-else-if ladder	78		
2.8.1.1.2. Switch	79		
2.8.1.2. Looping statements	81		
2.8.1.2.1. while loop	81		
2.8.1.2.2. do-while loop	83		
2.8.1.2.3. For loop	84		
2.8.1.3. Jump statements	87		
(a) break statement	87		
(b) continue statement	88		
(c) return statement	88		
(d) goto statement	89		
(e) assert statement	89		
static_assert	90		
2.9. Basic Input and Output	91		
3. FUNCTIONS	107		
3.1. Function	108		
Need of function	108		
3.1.1. Advantages of functions	110		
3.2. Classification of functions	110		
3.2.1. Library functions	111		
3.2.2. User defined functions	116		
3.2.2.1. Function declaration	116		
3.2.2.2. Function definition	118		
return statement	118		
Parameter list	120		
3.2.2.3. Function call	121		
3.2.2.3.1. Passing arguments	122		
3.2.2.3.1.1. Pass by value	122		

3.2.2.3.1.2	Pass by address	123			
3.2.2.3.1.3	Pass by reference using reference				
	Variable	125			
	Reference variable	125			
3.2.2.3.2	Default argument	131			
3.3	Inline functions	135			
	Need of inline function	136			
3.4	Function overloading	138			
3.5	main() function	143			
3.6	storage for functions	146			
3.7	Recursion	147			
	Advantages	151			
	Disadvantages	151			
3.8	storage class specifier	152			
3.9	constexpr function	159			
3.10	Lambda expression/function	160			
3.11	auto and decltype	165			
	decltype	166			
4.	ARRAYS AND STRINGS	176			
4.1	Arrays - Introduction	177			
4.1.1	Declaration of arrays	177			
4.1.2	Initialization of arrays	180			
	(a) Initializing arrays declared statically	180			
	(b) Initializing arrays declared dynamically	182			
4.1.3	Accessing the elements	183			
4.2	Parallel arrays	187			
4.3	For each loop	188			
4.4	Multi-dimensional arrays	189			
4.4.1	2-Dimensional arrays	189			
	Declaring 2-D array dynamically	193			
4.5	Arrays and functions	194			
4.5.1	Passing 1-D array to a function	194			
4.5.2	Passing multi-dimensional array to a function	196			
4.5.3	Returning array from a function	197			
4.6	std::array and std::vector	199			
	std::vector	202			
4.7	Strings	206			
	(a) Declaring and initializing a				
	string	206			
	(b) Reading a string	209			
	(c) Displaying a string	211			
	(d) Traversing a string	212			
	(e) Passing string to a function	213			
4.7.1	String manipulation	214			
	(a) Determining string length	214			
	(b) Concatenating strings	215			
	(c) Copying a string	216			
	(d) Comparing a string	217			
	(e) Reverse a string	218			
	(f) Find a string and replace it	220			
	(g) Get a substring	222			
	(h) Converting std::string to array of characters	224			
	(i) Converting std::string to numbers and vice-versa	224			
	(j) Some other operations	226			
4.7.2	Array of strings	227			
5.	POINTERS, STRUCTURES AND UNIONS	236			
5.1	Understanding pointers	236			
	(a) Address of operator	237			
	(b) Pointer declaration	237			
	(c) Initialization of pointer variable	238			
	(d) Dereference operator (*)	238			
	(e) Using pointers	239			
	(f) Types of pointers	240			
5.2	Pointer to pointer	246			
5.3	Pointer and constants	247			
5.4	Operations on pointers	249			
	(a) Adding/Subtracting an integer to/from a pointer	249			
	(b) Incrementing/decrementing a pointer	250			
	(c) Subtracting one pointer variable from another	250			
	(d) Comparing pointers	251			
5.4.1	Invalid operations on pointers	251			
5.5	Pointer notation and Arrays	253			
5.5.1	Pointer to array	256			
5.5.2	Array of pointers	257			

5.6	Handling strings using pointers	258	6.1.2	Object	307
5.7	Pointers as function arguments	260	6.2	Accessing class members	310
5.8	Pointers and Functions	260	6.3	Passing objects as arguments	315
5.8.1	Function returning pointers	260	6.4	Returning object from a function	320
5.8.2	Pointer to a function or function pointer	261	6.5	const object and const member function	321
	(a) Calling function using pointer to a function	262	6.6	static members of a class	323
	(b) Array of pointers to functions	263		(a) Static data member	323
	(c) Passing function as argument to a function	264		(b) Static member function	326
5.9	Complex Pointer declarations	266	6.7	Dynamic object	328
5.10	Type aliasing : using & typedef	267		(a) Creation and deletion	329
5.11	Enumerations	269		(b) Creating a dynamic object using reference variable	331
5.12	Structure	274		(c) Pointer to array of objects	332
	(a) Declaring structure variables	275		(d) Array of pointer to objects	334
	(b) Initializing structure member(s)	276	6.8	Anonymous class and object	336
	(c) Accessing structure members	277	6.9	Global class and Local class	338
5.12.1	Array of Structures	279	6.10	Local, Global and Static objects	341
	(a) Declaration	279	6.11	Nested class	343
	(b) Initialization	280	6.12	Singleton class	345
5.12.2	Nesting of structure	281	6.13	Using header files for class	347
	(a) Declaration	281	6.14	Encapsulation	349
	(b) Initialization	282			
	(c) Accessing members	283	7. CONSTRUCTOR AND DE- STRUCTOR	362	
5.12.3	Structure and pointers	284	7.1	Constructor	363
	Creating structure dynamically	286		(a) Default constructor	365
	Self-referential structure	287		(b) Parameterized constructor	366
5.12.4	Structures and functions	289		(i) Constructor for array of objects	367
	(a) Passing individual member	289		(ii) Constructor with default arguments	371
	(b) Passing entire structure to a function & returning	290		(iii) Constructor overloading	373
5.12.5	Some important points of structures	291		(iv) Constructor delegation	374
5.13	Union	292		(v) Member Initializer list	375
	(a) Declaring union	293		(vi) Initialization at run time using constructor	378
	(b) Declaring union variable	293	7.1.1	Private constructor	380
	(c) Accessing union variable	294	7.1.2	Copy constructor	381
	(d) Initializing a union	294		Explicit copy constructor	384
6. OBJECTS AND CLASSES			7.1.3	Some important points for constructors	386
		301	7.2	Destructor	388
6.1	Objects and classes	301	7.3	Friend function	390
6.1.1	Class	302	7.4	Friend class	396

7.5	this pointer	398	9.1	Inheritance	456
8.	OPERATOR OVERLOADING & TYPE CONVERSION	409		Advantages of Inheritance	458
8.1	Operator overloading - Introduction	410		Disadvantages of Inheritance	459
	Ways of defining	410		Implementing Inheritance in C++	459
	Syntax	411	9.1.1	Modes of Inheritance	461
	List of operators	411	9.1.2	Constructors and Initialization	465
	Rules for operator overloading	412	9.1.3	Destructors and Inheritance	472
	Advantages and disadvantages	413	9.1.4	Name lookup and Overriding behaviour	473
8.1.1	Overloading binary arithmetic operators	414	9.1.5	Modifying base class accessibility from derived class	477
8.1.2	Overloading unary Increment/decre- ment operators	421	9.1.6	Types of inheritance	479
8.1.3	Overloading relational operators	423		(e) Multiple inheritance (C++ only)	482
8.1.4	Overloading logical operators (!, &&,) and unary +, -	427		Ambiguity in Multiple inheritance	483
8.1.5	Overloading assignment operator (=)	428		Resolving Diamond Problem :	
8.1.6	Overloading subscript ([])	429		Virtual base class	487
8.1.7	Overloading function call operator (()) - Functor	431	9.2	Dependency	487
8.1.8	Overloading address of operator	434	9.3	Association	488
8.1.9	Overloading dereference operator	434	9.4	Composition	489
8.1.10	Overloading new and delete operators	434	9.5	Aggregation	491
8.1.11	Overloading comma operator	438	9.6	Inheritance v/s Composition	494
8.2	Type conversion	439	10.	VIRTUAL FUNCTIONS	500
8.2.1	Primitive type to class type conversion	439	10.1	Polymorphism	500
8.2.2	User-defined type (class) to primitive type conversion	441		Binding	502
8.2.3	class type to another class type conversion	443		Types of polymorphism	503
	(a) Conversion function in the source class	443		Advantages and disadvantages	504
	(b) Conversion function in the destination class	445	10.2	Need of virtual function	504
9.	OBJECT RELATIONSHIPS : INHERITANCE AND COMPOSITION	456	10.3	Virtual functions	507
			10.4	Override and final specifier	511
			10.5	Covariant return types	514
			10.6	Virtual destructors	514
			10.7	Object slicing	516
			10.8	Pure virtual functions	517
			10.9	Virtual table and Virtual pointer	519
			10.10	Type casting in classes	521
			11.	C++ STREAMS AND FILE HANDLING	527
			11.1	C++ streams	527
				(b) Hierarchy of stream classes	529
			11.2	I/O operations	531
			11.2.1	Unformatted I/O	531

11.2.1.1	istream member functions	531			
11.2.1.2	ostream member functions	537			
11.2.2	Formatted I/O	538			
11.2.2.1	ios stream class functions and flags	538			
	(a) Member functions	538			
11.3	Overloading insertion and extraction operators	545			
11.4	C++ File streams	549			
11.4.1	File operations	550			
11.4.1.1	Opening a file	551			
11.4.1.1.1	Opening file using open() function	551			
	File modes	552			
11.4.1.1.2	Opening file using constructor	553			
11.4.1.2	Read/write to a file	555			
11.4.1.3	Closing a file	555			
11.4.2	Types of files : Text v/s Binary files	562			
11.4.2.1	I/O operations on binary files : read() and write()	563			
11.4.3	File position pointers	567			
11.4.3.1	Reposition file position pointers	567			
11.4.4	I/O Error handling	576			
11.4.5	Redirection	580			
	Redirection using streams	582			
11.4.6	Command line arguments	583			
11.4.7	Sending output to printer	585			
11.5	String streams	586			
12.	TEMPLATES	592			
12.1	Function template	593			
	(a) Function template with multiple template type	596			
	(b) Function template that uses parameters other than non-Template type parameters	597			
	(c) Overloading Function template	599			
	(d) Function template specialization	600			
	(e) Default value for function parameter	601			
	(f) Macro v/s Function template	602			
12.2	Class template	603			
	(a) Class template specialization	607			
	(b) Multiple template type parameter	609			
12.3	Points to remember	610			
13.	EXCEPTION HANDLING	615			
13.1	Exception handling mechanism using try/catch	616			
	(a) Throw and catch exceptions between functions	618			
	(b) Multiple catch blocks	619			
	(c) Catch all exceptions	620			
	(d) Rethrow an exception	621			
	(e) Throwing exception from constructor	622			
	(f) Stack unwinding	623			
13.2	Standard Exception hierarchy	624			
14.	STL	628			
14.1	STL	629			
	std::pair STL	629			
14.1.1	Containers	631			
	(a) Sequence containers	631			
	(b) Container adaptors	634			
	(c) Associative containers	637			
	(d) Unordered associative containers	640			
14.1.2	Iterator	640			
14.1.3	Algorithm	642			
14.1.4	Functors	643			
15.	LEFTOVERS	645			
15.1	Namespace	645			
	(a) Creating namespace	646			
	(b) Accessing namespace members	646			
15.2	Preprocessor directives	651			
	(a) File Inclusion	651			
	(b) Macro definition	653			
	(c) Conditional compilation	655			
	(d) Other directives	659			
	(e) Some pre-defined macros	660			