

Contents

1. Introduction to C	1-6
1.1 Design Methods	1
1.1.1 Top-Down Design	1
1.1.2 Bottom-Up Design	2
1.1.3 Modular Approach	2
1.2 Programming Languages	2
1.2.1 Low Level Languages	2
1.2.1.1 Machine Level Language	2
1.2.1.2 Assembly Language	2
1.2.2 High-Level Languages	3
1.3 Translators	3
1.4 History Of C	3
1.5 Characteristics Of C	4
1.6 Structure Of A C Program	4
1.7 Environment For C	5
1.7.1 Unix Environment	5
1.7.2 MS-DOS Environment	5
1.7.2.1 Command Line	5
1.7.2.2 Integrated Development Environment	6
2. Elements of C	7-16
2.1 C Character Set	7
2.1.1 Alphabets	7
2.1.2 Digits	7
2.1.3 Special characters	7
2.2 Execution Characters/Escape Sequences	8
2.3 Trigraph Characters	8
2.4 Delimiters	9
2.5 Reserved Words / Keywords	9
2.6 Identifiers	9
2.7 Data Types	10
2.8 Constants	10
2.8.1 Numeric Constants	11
2.8.1.1 Integer constant	11
2.8.1.2 Real (floating point) Constants	12
2.8.2 Character Constants	13
2.8.3 String Constants	13

2.8.4	Symbolic Constants	13
2.9	Variables	14
2.9.1	Declaration of Variables	14
2.9.2	Initialisation of Variables	14
2.10	Expressions	15
2.11	Statements	15
2.11.1	Expression Statement	15
2.11.2	Compound Statement	15
2.12	Comments	16
3.	Input-Output In C	17-31
3.1	Conversion Specifications	17
3.2	Reading Input Data	18
3.3	Writing Output Data	21
3.4	Formatted Input And Output	24
3.4.1	Format For Integer Input	24
3.4.2	Format For Integer Output	25
3.4.3	Format For Floating Point Numeric Input	26
3.4.4	Format For Floating Point Numeric Output	27
3.4.5	Format For String Input	27
3.4.6	Format For String Output	28
3.5	Suppression Character in scanf()	28
3.6	Character I/O	29
3.6.1	getchar() and putchar()	29
Exercise		29
Answers		31
4.	Operators And Expressions	32-57
4.1	Arithmetic Operators	32
4.1.1	Unary Arithmetic Operators	32
4.1.2	Binary Arithmetic Operators	32
4.2	Integer Arithmetic	33
4.3	Floating-Point Arithmetic	34
4.4	Mixed Mode Arithmetic	34
4.5	Assignment Operators	35
4.6	Increment And Decrement Operators	35
4.6.1	Prefix Increment / Decrement	36
4.6.2	Postfix Increment / Decrement	36
4.7	Relational Operators	37
4.8	Logical Or Boolean Operators	39
4.8.1	AND (&&) Operator	39
4.8.2	OR () Operator	40
4.8.3	NOT (!) Operator	40
4.9	Conditional Operator	41
4.10	Comma Operator	42
4.11	sizeof Operator	43

4.12	Type Conversion	44
4.12.1	Implicit Type Conversions	44
4.12.2	Automatic Conversions	44
4.12.3	Type Conversion In Assignment	45
4.12.4	Explicit Type Conversion Or Type Casting	46
4.13	Precedence And Associativity Of Operators	47
4.14	Role Of Parentheses In Evaluating Expressions	50
4.15	Order Of Evaluation Of Operands	53
	Exercise	53
	Programming Exercise	56
	Answers	56
5.	Control Statements	58-109
5.1	Compound Statement or Block	58
5.2	if...else	59
5.2.1	Nesting of if...else	61
5.2.2	else if Ladder	63
5.3	Loops	65
5.3.1	while loop	65
5.3.2	do...while loop	69
5.3.3	for loop	71
5.3.4	Nesting Of Loops	75
5.3.5	Infinite Loops	77
5.4	break statement	78
5.5	continue statement	80
5.6	goto	82
5.7	switch	84
5.8	Some Additional Problems	90
5.9	Pyramids	99
	Exercise	103
	Programming Exercise	108
	Answers	109
6.	Functions	110-157
6.1	Advantages Of Using Functions	110
6.2	Library Functions	110
6.3	User-Defined Functions	111
6.4	Function Definition	112
6.5	Function Call	113
6.6	Function Declaration	114
6.7	return statement	116
6.8	Function Arguments	118
6.9	Types Of Functions	120
6.9.1	Functions With No Arguments And No Return Value-	120
6.9.2	Function With No Arguments But A Return Value	121
6.9.3	Function With Arguments But No Return Value	121

6.9.4	Function With Arguments And Return Value	123
6.10	More About Function Declaration	124
6.11	Declaration Of Functions With No Arguments	124
6.12	If Declaration Is Absent	125
6.13	Order Of Evaluation Of Function Arguments	125
6.14	main() Function	125
6.15	Library Functions	126
6.16	Old Style Of Function Declaration	126
6.17	Old Style Of Function Definition	126
6.18	Local, Global And Static Variables	130
6.18.1	Local Variables	130
6.18.2	Global Variables	131
6.18.3	Static Variables	132
6.19	Recursion	132
6.19.1	Tower Of Hanoi	136
6.19.2	Advantages And Disadvantages Of Recursion	139
6.19.3	Local Variables In Recursion	139
6.20	Some Additional Problems	140
Exercise		149
Programming Exercise		155
Answers		156

7. Arrays 158-195

7.1	One Dimensional Array	158
7.1.1	Declaration of 1-D Array	158
7.1.2	Accessing 1-D Array Elements	159
7.1.3	Processing 1-D Arrays	160
7.1.4	Initialization of 1-D Array	162
7.1.5	1-D Arrays And Functions	165
7.1.5.1	Passing Individual Array Elements to a Function	165
7.1.5.2	Passing whole 1-D Array to a Function	165
7.2	Two Dimensional Array	167
7.2.1	Declaration and Accessing Individual Elements of a 2-D array	167
7.2.2	Processing 2-D Arrays	168
7.2.3	Initialization of 2-D Arrays	169
7.3	Arrays With More Than Two Dimensions	173
7.3.1	Multidimensional Array And Functions	174
7.4	Introduction To Strings	175
7.4.1	Input and output of strings	175
7.5	Some Additional Problems	175
Exercise		191
Programming Exercise		193
Answers		194

8. Pointers 196-252

8.1	About Memory	196
-----	--------------	-----

8.2	Address Operator	197
8.3	Pointers Variables	197
8.3.1	Declaration Of Pointer Variables	198
8.3.2	Assigning Address To Pointer Variables	198
8.3.3	Dereferencing Pointer Variables	199
8.4	Pointer Arithmetic	201
8.5	Precedence Of Dereferencing Operator And Increment/Decrement Operators	204
8.6	Pointer Comparisons	206
8.7	Pointer To Pointer	206
8.8	Pointers and One Dimensional Arrays	208
8.9	Subscripting Pointer Variables	211
8.10	Pointer to an Array	212
8.11	Pointers And Two Dimensional Arrays	213
8.12	Subscripting Pointer To An Array	216
8.13	Pointers And Three Dimensional Arrays	217
8.14	Pointers And Functions	219
8.15	Returning More Than One Value From A Function	221
8.16	Function Returning Pointer	222
8.17	Passing a 1-D Array to a Function	223
8.18	Passing a 2-D Array to a Function	225
8.19	Array Of Pointers	227
8.20	void Pointers	229
8.21	Dynamic Memory Allocation	231
8.21.1	malloc()	231
8.21.2	calloc()	233
8.21.3	realloc()	233
8.21.4	free()	234
8.21.5	Dynamic Arrays	235
8.22	Pointers To Functions	238
8.22.1	Declaring A Pointer To A Function	239
8.22.2	Calling A Function Through Function Pointer	240
8.22.3	Passing a Function's Address as an Argument to Other Function	240
8.22.4	Using Arrays Of Function Pointers	242
Exercise		244
Answers		251
9.	Strings	253-287
9.1	String Constant or String Literal	253
9.2	String Variables	255
9.3	String Library Functions	257
9.3.1	strlen()	257
9.3.2	strcmp()	258
9.3.3	strcpy()	259
9.3.4	strcat()	261
9.4	String Pointers	262
9.5	Array Of Strings Or Two Dimensional Array Of Characters	264

9.6	Array Of Pointers To Strings	267
9.7	sprintf()	272
9.8	sscanf()	273
9.9	Some Additional Problems	274
	Exercise	280
	Programming Exercise	284
	Answers	286

10. Structure And Union**288-333**

10.1	Defining a Structure	288
10.2	Declaring Structure Variables	289
	10.2.1 With Structure Definition	289
	10.2.2 Using Structure Tag	289
10.3	Initialization Of Structure Variables	290
10.4	Accessing Members of a Structure	290
10.5	Assignment of Structure Variables	292
10.6	Storage of Structures in Memory	292
10.7	Size of Structure	293
10.8	Array of Structures	293
10.9	Arrays Within Structures	295
10.10	Nested Structures (Structure Within Structure)	296
10.11	Pointers to Structures	298
10.12	Pointers Within Structures	299
10.13	Structures And Functions	299
	10.13.1 Passing Structure Members As Arguments	299
	10.13.2 Passing Structure Variable As Argument	300
	10.13.3 Passing Pointers To Structures As Arguments	301
	10.13.4 Returning A Structure Variable From Function	302
	10.13.5 Returning A Pointer To Structure From A Function	303
	10.13.6 Passing Array Of Structures As Argument	303
10.14	Self Referential Structures	309
10.15	Linked List	309
	10.15.1 Traversing a Linked List	311
	10.15.2 Searching in a Linked List	311
	10.15.3 Insertion into a Linked List	311
	10.15.4 Insertion in the Beginning	312
	10.15.5 Insertion in Between or at the end	312
	10.15.6 Deletion From A Linked List	313
	10.15.7 Deletion of First Node	313
	10.15.8 Deletion of a Node in Between or at the End	314
	10.15.9 Creation Of List	314
	10.15.10 Reversing A Linked List	318
10.16	union	321
10.17	typedef	326
	Exercise	329
	Programming Exercise	332
	Answers	332

11. Files	334-376
11.1 Text And Binary Modes	334
11.2 Concept Of Buffer	335
11.3 Opening a File	335
11.3.1 Errors in Opening Files	337
11.4 Closing a File	337
11.5 End of File	338
11.6 Structure of a General File Program	338
11.7 Predefined File Pointers	339
11.8 Character I/O	339
11.8.1 fputc()	339
11.8.2 fgetc()	340
11.8.3 getc() and putc()	340
11.9 Integer I/O	341
11.9.1 putw()	341
11.9.2 getw()	341
11.10 String I/O	342
11.10.1 fputs()	342
11.10.2 fgets()	342
11.11 Formatted I/O	343
11.11.1 fprintf()	343
11.11.2 fscanf()	344
11.12 Block Read / Write	345
11.12.1 fwrite()	345
11.12.2 fread()	347
11.13 Random Access To File	348
11.13.1 fseek()	349
11.13.2 ftell()	350
11.13.3 rewind()	351
11.14 Other File Functions	362
11.14.1 feof()	362
11.14.2 perror()	363
11.14.3 clearerr()	364
11.14.4 perror()	364
11.14.5 rename()	364
11.14.6 unlink()	365
11.14.7 remove()	365
11.14.8 fflush()	366
11.14.9 tmpfile()	366
11.14.10 tmpnam()	366
11.14.11 freopen()	367
11.15 Command Line Arguments	367
11.16 Some Additional Problems	368
Exercise	374
Programming Exercise	375
Answers	376

12. The C Preprocessor	377-406
12.1 #define	378
12.2 Macros with Arguments	379
12.3 Nesting in Macros	381
12.4 Problems with Macros	382
12.5 Macros Vs Functions	385
12.6 Generic Functions	386
12.7 #undef	387
12.8 Stringizing Operator (#)	387
12.9 Token Pasting Operator(##)	388
12.10 Including Files	389
12.11 Conditional Compilation	389
12.11.1 #if And #endif	390
12.11.2 #else and #elif	390
12.11.3 defined Operator	393
12.11.4 #ifdef and #ifndef	393
12.11.5 Writing Portable Code	395
12.11.6 Debugging	396
12.11.7 Commenting A Part Of Code	397
12.11.8 Other Uses of conditional compilation	397
12.12 Predefined Macro Names	398
12.13 #line	399
12.14 #error	399
12.15 Null Directive	400
12.16 #pragma	400
12.17 How to see the code expanded by the Preprocessor	401
Exercise	401
Answers	405
13. Operations on Bits	407-432
13.1 Bitwise AND (&)	408
13.2 Bitwise OR ()	408
13.3 Bitwise XOR (^)	409
13.4 One's Complement (~)	410
13.5 Bitwise Left Shift (<<)	411
13.6 Bitwise Right Shift (>>)	411
13.7 Multiplication and Division by 2 using shift operators	412
13.8 Masking	413
13.8.1 Masking Using Bitwise AND	413
13.8.2 Masking Using Bitwise OR	415
13.8.3 Masking Using Bitwise XOR	415
13.8.4 Switching off Bits Using Bitwise AND and Complement Operator	416
13.9 Some additional Problems	418
13.10 Bit Fields	426
Exercise	429
Answers	431

14. Miscellaneous Features In C	433-463
14.1 Enumeration	433
14.2 Storage Classes	437
14.2.1 Automatic	437
14.2.2 External	439
14.2.3 Static	442
14.2.3.1 Local Static Variables	442
14.2.3.2 Global Static Variables	443
14.2.4 Register	444
14.3 Storage Classes in Functions	445
14.4 Linkage	445
14.5 Memory During Program Execution	445
14.6 const	447
14.7 volatile	449
14.8 Functions With Variable Number Of Arguments	450
14.8.1 Passing Variable Number of Arguments To Another Function	454
14.9 lvalue and rvalue	456
14.10 Compilation And Execution of C Programs	457
14.10.1 Preprocessor	457
14.10.2 Compiler	457
14.10.3 Assembler	457
14.10.4 Linker	457
Exercise	458
Answers	463

15. Building project and creation of library	464-486
15.1 Requirement Analysis	464
15.2 Top Level Design	465
15.3 Detail Design	465
15.4 Coding	468
15.4.1 Dtmanip.h	468
15.4.2 Datefmt.c	469
15.4.3 Valid.c	469
15.4.4 Leap.c	470
15.4.5 Julian.c	470
15.4.6 Weekday.c	471
15.4.7 Cmpdate.c	472
15.4.8 Diffymd.c	472
15.4.9 Diffdays.c	473
15.4.10 Addyear.c	474
15.4.11 Subyear.c	474
15.4.12 Addmonth.c	475
15.4.13 Submonth.c	475
15.4.14 Adddays.c	476
15.4.15 Subdays.c	477
15.4.16 Main.c	477

15.5	Building Project in Turbo C	481
15.6	Testing	481
15.7	Creation Of Library And Using it in your Program in Turbo C	482
15.7.1	Deletion of a Module From Library	482
15.7.2	Getting Modules From Library	483
15.7.3	Changing Version Of Module In Library	483
15.8	Building Project On Unix	483
15.8.1	Writing Makefile	484
15.8.2	Building Project With Make	486
15.9	Creation Of Library And Using In Your Program in Unix	486
16.	Code Optimization in C	487-494
16.1	Optimization is a Technique	487
16.2	Optimization With Tool	487
16.3	Optimization With Loop	487
16.3.1	Loop Unrolling	487
16.3.2	Avoiding Calculations In Loops	488
16.4	Fast Mathematics	488
16.4.1	Avoid Unnecessary Integer Division	488
16.4.2	Multiplication And Division by Power Of 2	488
16.5	Simplifying Expressions	489
16.6	Declare prototypes for Functions	489
16.7	Better Way Of Calling Function	489
16.8	Prefer int to char or short	489
16.9	Use of Register Variables	490
16.10	Optimization With Switch Statement	490
16.11	Avoid Pointer Dereference	492
16.12	Prefer Pre Increment/Decrement to Post Increment/Decrement	492
16.13	Prefer Array to Memory Allocation	492
16.14	Use Array Style Code Rather Than Pointer	492
16.15	Expression Order Understanding	492
16.16	Declaration Of Local Function	493
16.17	Breaking Loop With Parallel Coding	493
16.18	Trick to use Common Expression	493
16.19	Declaring Local Variables Based On Size	494
16.20	Prefer Integer Comparison	494
16.21	Avoid String Comparison	494
17.	C and Assembly Interaction	495-504
17.1	Inline Assembly Language	495
17.2	Linking Of Two Assembly Files	498
17.2.1	Memory Models	500
17.2.2	C And Segments in Library	500
17.3	Linking Assembly Procedure in C Program	502

18. Library Functions

		505-521
18.1	Mathematical Functions	505
18.1.1	abs()	505
18.1.2	acos()	505
18.1.3	asin()	505
18.1.4	atan()	505
18.1.5	atan2()	505
18.1.6	cabs()	505
18.1.7	ceil()	505
18.1.8	cos()	505
18.1.9	cosh()	505
18.1.10	exp()	505
18.1.11	fabs()	505
18.1.12	floor()	505
18.1.13	fmod()	505
18.1.14	frexp()	505
18.1.15	ldexp()	505
18.1.16	log()	505
18.1.17	log10()	505
18.1.18	modf()	505
18.1.19	pow()	505
18.1.20	sin()	505
18.1.21	sinh()	505
18.1.22	sqrt()	505
18.1.23	tan()	505
18.1.24	tanh()	505
18.2	Character Type Functions	508
18.2.1	isalnum()	508
18.2.2	isalpha()	508
18.2.3	iscntrl()	508
18.2.4	isdigit()	508
18.2.5	isgraph()	508
18.2.6	islower()	508
18.2.7	isprint()	508
18.2.8	ispunct()	508
18.2.9	isspace()	508
18.2.10	isupper()	509
18.2.11	isxdigit()	509
18.2.12	tolower()	509
18.2.13	toupper()	509
18.3	String Manipulation Functions	509
18.3.1	strcat()	509
18.3.2	strchr()	509
18.3.3	strcmp()	510
18.3.4	strcpy()	510
18.3.5	strcspn()	510

521	18.3.6	strlen()	510
505	18.3.7	strncat()	510
505	18.3.8	strcmp()	511
505	18.3.9	strcpy()	511
505	18.3.10	strpbrk()	512
505	18.3.11	strrchr()	512
505	18.3.12	strspn()	512
505	18.3.13	strstr()	512
506	18.4	Input/Output Functions	513
506	18.4.1	access()	513
506	18.4.2	chmod()	513
506	18.4.3	clearerr()	514
506	18.4.4	close()	514
506	18.4.5	creat()	514
506	18.4.6	fclose()	514
506	18.4.7	feof()	514
507	18.4.8	ferror()	514
507	18.4.9	fflush()	515
507	18.4.10	fgetc()	515
507	18.4.11	fgets()	515
507	18.4.12	fileno()	515
507	18.4.13	fopen()	515
507	18.4.14	fprintf()	515
507	18.4.15	fputc()	516
507	18.4.16	fputs()	516
507	18.4.17	fread()	516
508	18.4.18	fputchar()	516
508	18.4.19	fscanf()	516
508	18.4.20	fseek()	516
508	18.4.21	fstat()	517
508	18.4.22	ftell()	517
508	18.4.23	isatty()	517
508	18.4.24	open()	517
508	18.4.25	read()	518
508	18.4.26	remove()	518
508	18.4.27	rename()	518
509	18.4.28	setbuf()	518
509	18.4.29	sopen()	519
509	18.4.30	stat()	520
509	18.4.31	sprintf()	520
509	18.4.32	sscanf()	520
509	18.4.33	tell()	520
509	18.4.34	tmpfile()	520
510	18.4.35	tmpnam()	520
510	18.4.36	unlink()	521

Chapter 1

Introduction to C

Software is a collection of programs and a program is a collection of instructions given to the computer. Development of software is a stepwise process. Before developing a software, number of processes are done. The first step is to understand the user requirements. Problem analysis arises during the requirement phase of software development. Problem analysis is done for obtaining the user requirements and to determine the input and output of the program.

For solving the problem, an “algorithm” is implemented. Algorithm is a sequence of steps that gives method of solving a problem. This “algorithm” creates the logic of program. On the basis of this “algorithm”, program code is written. The steps before writing program code are as-

User requirements



Problem analysis



Input and Output



Designing algorithm



Program coding

Process of program development

1.1 Design Methods

Designing is the first step for obtaining solution of a given problem. The purpose of designing is to represent the solution for the system. It is really difficult to design a large system because the complexity of system cannot be represented easily. So various methods have been evolved for designing.

1.1.1 Top-Down Design

Every system has several hierarchies of components. The top-level component represents the whole system. Top-Down design method starts from top-level component to lowest level (bottom) component. In this design method, the system is divided into some major components.