

!= (Logical operator not)	
<i>lecture01</i>	p. 23
" (Double quote)	
<i>lecture01</i>	p. 15
#define	
<i>lecture01</i>	p. 16,18
<i>lecture09</i>	p. 3,4
#ifndef	
<i>lecture09</i>	p. 3
#include	
<i>lecture01</i>	p. 16,20
<i>lecture02</i>	p. 7
<i>lecture09</i>	p. 3
%p	
<i>lecture09</i>	p. 22
& (bit and)	
<i>lecture01</i>	p. 23
& operator (address)	
<i>lecture01</i>	p. 10
<i>lecture03</i>	p. 4,6,8
&& (Logical operator and)	
<i>lecture01</i>	p. 23
' (Single quote)	
<i>lecture01</i>	p. 15
* operator (dereferencing)	
<i>lecture01</i>	p. 10
<i>lecture03</i>	p. 7
- (arrow) reference to structure	
<i>lecture03</i>	p. 19,20
. (dot) reference to structure	
<i>lecture03</i>	p. 16,20
.h File	
<i>lecture01</i>	p. 16
.hpp file	
<i>lecture08</i>	p. 14
2-3-4 Tree	
<i>lecture06</i>	p. 20
\0	
<i>lecture01</i>	p. 14,15
<i>lecture02</i>	p. 10-12
\n	
<i>lecture02</i>	p. 10

__FILE__	
<i>lecture09</i>	p. 5,6
__LINE__	
<i>lecture09</i>	p. 5,6
((Curly brackets)	
<i>lecture01</i>	p. 22
(bit or)	
<i>lecture01</i>	p. 23
(Logical operator or)	
<i>lecture01</i>	p. 23

A

Abstract class	
<i>lecture12</i>	p. 10
accept()	
<i>lecture11</i>	p. 4
Address	
<i>lecture01</i>	p. 8-10,12,13
Address of variable	
<i>lecture03</i>	p. 4
Adelson-Velsky, Georgy	
<i>lecture06</i>	p. 17
Aggregate	
<i>lecture12</i>	p. 5
Aggregate vs multiple inheritance	
<i>lecture12</i>	p. 11
Algorithm	
<i>lecture01</i>	p. 12
<i>lecture12</i>	p. 27
Alignment of structures	
<i>lecture03</i>	p. 18
And (logical operator)	
<i>lecture01</i>	p. 23
Angle brackets vs double quotes for header files	
<i>lecture02</i>	p. 7
Architecture	
<i>lecture01</i>	p. 10
argc	
<i>lecture02</i>	p. 2
<i>lecture03</i>	p. 14,15
<i>lecture03</i>	p. 13
Argument passed as reference	

<i>lecture08</i>	p. 16
Argument passed by reference	
<i>lecture08</i>	p. 17
argv[]	
<i>lecture02</i>	p. 2
<i>lecture03</i>	p. 13-15
Array	
<i>lecture01</i>	p. 10,12-14
<i>lecture03</i>	p. 6
<i>lecture05</i>	p. 17-21
<i>lecture06</i>	p. 1,9,10
<i>lecture08</i>	p. 20
Array in C and in Java	
<i>lecture03</i>	p. 12
Array of strings	
<i>lecture03</i>	p. 12,13
Array of structures	
<i>lecture03</i>	p. 16
Array vs pointer	
<i>lecture03</i>	p. 6,8,9,12
Array – multidimensional	
<i>lecture03</i>	p. 13
Array: returned by a function	
<i>lecture04</i>	p. 11-13
Array:Pointer	
<i>lecture05</i>	p. 2
Arrow reference to structure field	
<i>lecture03</i>	p. 19,20
ASCII	
<i>lecture01</i>	p. 9
ASCII table	
<i>lecture01</i>	p. 22
assert	
<i>lecture02</i>	p. 4
Assigning address to pointer	
<i>lecture03</i>	p. 6
Assignment	
<i>lecture01</i>	p. 22,23
<i>lecture02</i>	p. 2
Assignment of objects of derived classes	
<i>lecture12</i>	p. 7,8
Assignment operator	
<i>lecture10</i>	p. 18-20
ATK	
<i>lecture09</i>	p. 18

atof()	
<i>lecture03</i>	p. 1
atoi()	
<i>lecture03</i>	p. 1
atol()	
<i>lecture03</i>	p. 1
autoconf	
<i>lecture09</i>	p. 11
automake	
<i>lecture09</i>	p. 11
autoscan	
<i>lecture09</i>	p. 10
Autotools	
<i>lecture09</i>	p. 9-11
AVL tree	
<i>lecture06</i>	p. 17-19
B	
B-Tree	
<i>lecture06</i>	p. 20-23
<i>lecture07</i>	p. 2
Bad path	
<i>lecture12</i>	p. 16
Balanced tree	
<i>lecture06</i>	p. 17-19
Beeper program	
<i>lecture11</i>	p. 16
Bell Labs	
<i>lecture01</i>	p. 6,7
Berkeley Software Distribution (BSD)	
<i>lecture07</i>	p. 2
Berners-Lee, Tim	
<i>lecture11</i>	p. 7
Binary file	
<i>lecture04</i>	p. 2
Binary search	
<i>lecture06</i>	p. 9,10,13,14
Binary tree	
<i>lecture06</i>	p. 14-17
bind()	
<i>lecture11</i>	p. 4
Bit	
<i>lecture01</i>	p. 8

Bit operators		C++ initialization	
<i>lecture01</i>	p. 23	<i>lecture10</i>	p. 4
Block		C++ rules	
<i>lecture01</i>	p. 22	<i>lecture12</i>	p. 32
Block of instructions		C++ vs Java	
<i>lecture01</i>	p. 15	<i>lecture08</i>	p. 18-20
Boolean		<i>lecture10</i>	p. 13,14
<i>lecture01</i>	p. 10,11,21	<i>lecture12</i>	p. 9,10,20,21
Box-Müller		C11	
<i>lecture03</i>	p. 3	<i>lecture01</i>	p. 7
break		C89	
<i>lecture01</i>	p. 24	<i>lecture01</i>	p. 7
BSD (Berkeley Software Distribution)		C99	
<i>lecture07</i>	p. 2	<i>lecture01</i>	p. 7
<i>lecture10</i>	p. 22	Cairo	
Built-in functions		<i>lecture09</i>	p. 18
<i>lecture02</i>	p. 5,8	Calling functions	
Byte		<i>lecture04</i>	p. 8
<i>lecture01</i>	p. 8	calloc()	
Byte address		<i>lecture04</i>	p. 18
<i>lecture12</i>	p. 18	Canonical class	
		<i>lecture10</i>	p. 6,7,9,10,15,18,19
C		case	
		<i>lecture01</i>	p. 24
C and inheritance		Case	
<i>lecture12</i>	p. 11-13	<i>lecture02</i>	p. 11
C environment		Case insensitive comparison	
<i>lecture01</i>	p. 7	<i>lecture02</i>	p. 13
C program structure		Casting	
<i>lecture01</i>	p. 18	<i>lecture12</i>	p. 31
C standard library		catch	
<i>lecture02</i>	p. 8	<i>lecture08</i>	p. 16
C vs Java		Catching errors	
<i>lecture01</i>	p. 6,12-14,17,18	<i>lecture08</i>	p. 11,12
<i>lecture02</i>	p. 5	CFLAGS	
<i>lecture03</i>	p. 12,20	<i>lecture04</i>	p. 5
<i>lecture04</i>	p. 7,17	Changing case	
<i>lecture05</i>	p. 1,2,16	<i>lecture02</i>	p. 11
<i>lecture06</i>	p. 23	char	
<i>lecture09</i>	p. 15-17	<i>lecture01</i>	p. 10,11
C++		Character classification	
<i>lecture08</i>	p. 10-13	<i>lecture02</i>	p. 10,11
C++ constructor		Character encoding	
<i>lecture10</i>	p. 4	<i>lecture01</i>	p. 15,22
		Character Encoding	

<i>lecture02</i>	p. 17	<i>lecture01</i>	p. 22,23
Character encoding		<i>lecture02</i>	p. 2
<i>lecture02</i>	p. 15-18	Compiler	
Chinese characters		<i>lecture01</i>	p. 16,17,19,20
<i>lecture02</i>	p. 14,16,17	Compiling a C program	
cin		<i>lecture01</i>	p. 17
<i>lecture08</i>	p. 15	Compiling on Linux	
CJK		<i>lecture01</i>	p. 17
<i>lecture02</i>	p. 15	Condition	
class		<i>lecture01</i>	p. 21
<i>lecture08</i>	p. 18	Condition variable	
<i>lecture09</i>	p. 15	<i>lecture14</i>	p. 2-4
<i>lecture12</i>	p. 19	Conditional compiling	
Class (canonical)		<i>lecture09</i>	p. 6,9
<i>lecture10</i>	p. 6,7,9,10,15,18,19	<i>lecture10</i>	p. 1
Class naming rules		configure	
<i>lecture10</i>	p. 3,4	<i>lecture09</i>	p. 10,11
Class template		conio.h	
<i>lecture12</i>	p. 22-25	<i>lecture09</i>	p. 8
Classes		connect()	
<i>lecture08</i>	p. 17,18	<i>lecture11</i>	p. 2-4
<i>lecture10</i>	p. 3	Constants	
Classification of characters		<i>lecture01</i>	p. 16,18
<i>lecture02</i>	p. 10,11	Constructor	
Clean shutdown		<i>lecture08</i>	p. 18
<i>lecture13</i>	p. 1	Constructor (copy)	
close()		<i>lecture10</i>	p. 10-12,14,15
<i>lecture11</i>	p. 3	Constructor (default)	
Code		<i>lecture10</i>	p. 7-9
<i>lecture01</i>	p. 8	cons_cast	
codepoint		<i>lecture12</i>	p. 31
<i>lecture02</i>	p. 15	Container	
<i>lecture02</i>	p. 15	<i>lecture12</i>	p. 26
Collection		Coplien, Jim	
<i>lecture08</i>	p. 19	<i>lecture10</i>	p. 6,7,9,10,15,18,19
Collections		Copy (shallow vs deep)	
<i>lecture05</i>	p. 16	<i>lecture10</i>	p. 14
Command-line parameters		Copy constructor	
<i>lecture02</i>	p. 2	<i>lecture10</i>	p. 10-12,14,15
<i>lecture03</i>	p. 13-15	Copy operator	
Comparison of Data structures		<i>lecture10</i>	p. 19,20
<i>lecture07</i>	p. 3-5	Core dump	
Comparison of strings		<i>lecture11</i>	p. 14
<i>lecture02</i>	p. 12,13	Course expectations	
Comparison operators		<i>lecture01</i>	p. 2

Course notes	
<i>lecture01</i>	p. 3
Course Organization	
<i>lecture01</i>	p. 6
Course overview	
<i>lecture14</i>	p. 4-8
Course schedule	
<i>lecture01</i>	p. 1
cout	
<i>lecture08</i>	p. 14,15
Craftsmanship	
<i>lecture01</i>	p. 5
Cryptography	
<i>lecture06</i>	p. 12
ctime()	
<i>lecture03</i>	p. 2,3,21
ctype.h	
<i>lecture02</i>	p. 10
<i>lecture02</i>	p. 11
Curly brackets	
<i>lecture01</i>	p. 15,22
Cygwin	
<i>lecture01</i>	p. 2
c_str()	
<i>lecture12</i>	p. 29
 D	
Daemon	
<i>lecture13</i>	p. 4,8
daemon()	
<i>lecture13</i>	p. 8
Dahl, Ole-Johan	
<i>lecture08</i>	p. 12
Data	
<i>lecture01</i>	p. 8
<i>lecture05</i>	p. 16,17
Data structure	
<i>lecture05</i>	p. 17
Data structure functions	
<i>lecture06</i>	p. 23
<i>lecture07</i>	p. 2,3
Data structures	
<i>lecture05</i>	p. 16,17,21,22

<i>lecture06</i>	p. 1-13
<i>lecture07</i>	p. 4,5
Data structures comparison	
<i>lecture07</i>	p. 3-5
Data types	
<i>lecture01</i>	p. 10-12
Database	
<i>lecture06</i>	p. 20,23
<i>lecture07</i>	p. 5
ddd	
<i>lecture09</i>	p. 22
Debugging	
<i>lecture09</i>	p. 22
Declaration	
<i>lecture05</i>	p. 1
Declaration of pointer	
<i>lecture03</i>	p. 5,6
Declaration of variable	
<i>lecture01</i>	p. 9
Deep copy	
<i>lecture10</i>	p. 14,15
Default constructor	
<i>lecture10</i>	p. 7-9
Default destructor	
<i>lecture10</i>	p. 10
Default parameters	
<i>lecture08</i>	p. 16
Degenarated binary tree	
<i>lecture06</i>	p. 17
delete	
<i>lecture08</i>	p. 15
Deleting a file	
<i>lecture04</i>	p. 2
deque	
<i>lecture12</i>	p. 26
Dereferencing	
<i>lecture03</i>	p. 7,8,19,20
Derived class	
<i>lecture12</i>	p. 4
Destructor	
<i>lecture08</i>	p. 18
Destructor (default)	
<i>lecture10</i>	p. 10
Direct access	
<i>lecture04</i>	p. 2

Directory operations	
<i>lecture04</i>	p. 3
dirent.h	
<i>lecture04</i>	p. 3
Distribution	
<i>lecture03</i>	p. 3
do ... while	
<i>lecture02</i>	p. 1
Dot reference to structure field	
<i>lecture03</i>	p. 16,20
double	
<i>lecture01</i>	p. 12
Double quote	
<i>lecture01</i>	p. 15
Double quotes vs angle brackets for header files	
<i>lecture02</i>	p. 7
Doubly linked list	
<i>lecture06</i>	p. 8
Dumping a binary file	
<i>lecture04</i>	p. 3
Dynamic analysis:gdb	
<i>lecture09</i>	p. 22
Dynamic analysis:Valgrind	
<i>lecture09</i>	p. 22
Dynamic data structures	
<i>lecture07</i>	p. 5
Dynamic memory	
<i>lecture04</i>	p. 16-18
<i>lecture05</i>	p. 1,2,17-20
Dynamic memory example	
<i>lecture04</i>	p. 19

E

Eclipse	
<i>lecture09</i>	p. 22
EDP	
<i>lecture05</i>	p. 16
Electric-Fence	
<i>lecture09</i>	p. 22
Electronic Data Processing	
<i>lecture05</i>	p. 16
ELLEMENTEL	

<i>lecture12</i>	p. 32
else	
<i>lecture01</i>	p. 21,23
else if	
<i>lecture01</i>	p. 23
Encapsulation	
<i>lecture08</i>	p. 18
<i>lecture10</i>	p. 3
Encoding	
<i>lecture01</i>	p. 9
<i>lecture02</i>	p. 15
End-of-string marker	
<i>lecture01</i>	p. 14,15
EOF	
<i>lecture02</i>	p. 9,10
Epoch	
<i>lecture03</i>	p. 2
errno	
<i>lecture03</i>	p. 1
<i>lecture09</i>	p. 15
errno.h	
<i>lecture03</i>	p. 1
<i>lecture09</i>	p. 15
Error checking	
<i>lecture02</i>	p. 2-4
<i>lecture03</i>	p. 1
Error management	
<i>lecture02</i>	p. 4,5
Exam	
<i>lecture01</i>	p. 3
Exam dates	
<i>lecture01</i>	p. 3
Example of pointer usage	
<i>lecture03</i>	p. 8
Example: day of the week when you were born	
<i>lecture03</i>	p. 21-23
Example: linked list	
<i>lecture06</i>	p. 4-6
Exams	
<i>lecture01</i>	p. 2-4
Exception	
<i>lecture02</i>	p. 4,5
<i>lecture12</i>	p. 14-16
Exception (uncaught)	
<i>lecture12</i>	p. 17

Exceptions	
<i>lecture08</i>	p. 16
exec()	
<i>lecture13</i>	p. 6
<i>lecture13</i>	p. 7,8
exec1()	
<i>lecture13</i>	p. 7
Executable	
<i>lecture01</i>	p. 16
execv()	
<i>lecture13</i>	p. 7
Expectations	
<i>lecture01</i>	p. 2
Exponent	
<i>lecture01</i>	p. 12
Exponential distribution	
<i>lecture03</i>	p. 3
extern	
<i>lecture04</i>	p. 7
<i>lecture04</i>	p. 7
<i>lecture09</i>	p. 14

F

Factorial	
<i>lecture05</i>	p. 14
fclose()	
<i>lecture03</i>	p. 25
feof()	
<i>lecture04</i>	p. 1
ferror()	
<i>lecture04</i>	p. 1
fflush()	
<i>lecture09</i>	p. 22
fgetc()	
<i>lecture02</i>	p. 9
<i>lecture04</i>	p. 1
fgets()	
<i>lecture01</i>	p. 16
<i>lecture02</i>	p. 10
<i>lecture04</i>	p. 14
fgets():Return value	
<i>lecture03</i>	p. 1
FIFO	

<i>lecture06</i>	p. 8
<i>lecture07</i>	p. 4
FILE	
<i>lecture03</i>	p. 26
FILE *	
<i>lecture03</i>	p. 25
Files	
<i>lecture03</i>	p. 24-26
Final exam	
<i>lecture01</i>	p. 3
First In First Out	
<i>lecture06</i>	p. 8
float	
<i>lecture01</i>	p. 12
flock()	
<i>lecture04</i>	p. 2
Flow control	
<i>lecture01</i>	p. 21,23,24
<i>lecture02</i>	p. 1
fopen()	
<i>lecture03</i>	p. 25,26
<i>lecture10</i>	p. 21
for	
<i>lecture02</i>	p. 1
fork()	
<i>lecture13</i>	p. 4,5,7
Formatted input and output	
<i>lecture02</i>	p. 10
fprintf()	
<i>lecture02</i>	p. 10
<i>lecture03</i>	p. 26
fputc()	
<i>lecture02</i>	p. 9
<i>lecture04</i>	p. 1
fputs()	
<i>lecture02</i>	p. 10
<i>lecture03</i>	p. 26
fread()	
<i>lecture04</i>	p. 1
Free Software Foundation (FSF)	
<i>lecture09</i>	p. 9
free()	
<i>lecture04</i>	p. 19
<i>lecture04</i>	p. 18,21-23
Freeing a binary tree	

<i>lecture06</i>	p. 16
friend	
<i>lecture10</i>	p. 18
fseek()	
<i>lecture04</i>	p. 2
FSF	
<i>lecture09</i>	p. 9
ftok()	
<i>lecture13</i>	p. 10,11
Function call	
<i>lecture04</i>	p. 8-12
<i>lecture10</i>	p. 2
Function declaration	
<i>lecture02</i>	p. 6,7
Function identification	
<i>lecture02</i>	p. 5,6
Function nesting	
<i>lecture01</i>	p. 15
Function object	
<i>lecture12</i>	p. 27-31
Function pointer	
<i>lecture06</i>	p. 23,24
Function pointers	
<i>lecture09</i>	p. 16,17
Function prototype	
<i>lecture01</i>	p. 16
<i>lecture02</i>	p. 7
Function template	
<i>lecture12</i>	p. 18-21,25
Function vs method	
<i>lecture10</i>	p. 16
Function: Pointers as argument	
<i>lecture04</i>	p. 13,14
<i>lecture05</i>	p. 2,3
Function: returning an array	
<i>lecture04</i>	p. 11-13
Functional programming	
<i>lecture12</i>	p. 30
Functions	
<i>lecture04</i>	p. 8
Functions, nesting	
<i>lecture02</i>	p. 6
Functor	
<i>lecture12</i>	p. 27-31
fwrite()	

<i>lecture04</i>	p. 1
G	
g++	
<i>lecture08</i>	p. 15
Garbage collector	
<i>lecture04</i>	p. 19
<i>lecture08</i>	p. 19
Gateway	
<i>lecture10</i>	p. 25
gcc	
<i>lecture01</i>	p. 17
<i>lecture01</i>	p. 7
gcd()	
<i>lecture04</i>	p. 8,9
Generic class	
<i>lecture12</i>	p. 22-25
Generic function	
<i>lecture12</i>	p. 17-21,25
Generic tree	
<i>lecture12</i>	p. 22,23
getaddrinfo()	
<i>lecture11</i>	p. 3,8
getchar()	
<i>lecture02</i>	p. 9
getenv()	
<i>lecture13</i>	p. 7
getopt()	
<i>lecture03</i>	p. 15
getpid()	
<i>lecture11</i>	p. 11
getppid()	
<i>lecture11</i>	p. 11
gets()	
<i>lecture02</i>	p. 10
Git	
<i>lecture09</i>	p. 21
Glib	
<i>lecture07</i>	p. 3
<i>lecture09</i>	p. 18
Global variable	
<i>lecture03</i>	p. 1
<i>lecture04</i>	p. 15

<i>lecture09</i>	p. 13,15,16
gmtime()	
<i>lecture03</i>	p. 21
Gnome	
<i>lecture07</i>	p. 3
Gnome Tool Kit (GTK)	
<i>lecture09</i>	p. 18-20
GNU	
<i>lecture07</i>	p. 3
GNU autotools	
<i>lecture09</i>	p. 9-11
GNU Tool Kit (GTK)	
<i>lecture12</i>	p. 11,13
Good path	
<i>lecture12</i>	p. 16
Gosling, James	
<i>lecture12</i>	p. 10
Grades	
<i>lecture01</i>	p. 4
grep	
<i>lecture13</i>	p. 3
GTK (Gnome Tool Kit)	
<i>lecture09</i>	p. 18-20
GTK (GNU Tool Kit)	
<i>lecture12</i>	p. 11,13
gtk.h	
<i>lecture09</i>	p. 18
GtkWidget	
<i>lecture09</i>	p. 18,19
GTK_WINDOW	
<i>lecture09</i>	p. 19

H

Hanoi (towers of)	
<i>lecture05</i>	p. 15
Hash function	
<i>lecture06</i>	p. 11,12
Hash table	
<i>lecture06</i>	p. 11-13
<i>lecture07</i>	p. 2
Hashmap	
<i>lecture12</i>	p. 26
head	

<i>lecture04</i>	p. 3
Head of list	
<i>lecture06</i>	p. 1
Header file	
<i>lecture01</i>	p. 16
<i>lecture02</i>	p. 7
<i>lecture09</i>	p. 1,2,15
Heap	
<i>lecture01</i>	p. 8
<i>lecture04</i>	p. 17
Help on functions	
<i>lecture02</i>	p. 5
Hiding	
<i>lecture12</i>	p. 8
History of C	
<i>lecture01</i>	p. 7
Hoare, Antony	
<i>lecture05</i>	p. 6
Honesty	
<i>lecture01</i>	p. 5
HTTP	
<i>lecture11</i>	p. 7-10
HTTPCnx	
<i>lecture11</i>	p. 10
httpd	
<i>lecture11</i>	p. 7
I	
if	
<i>lecture01</i>	p. 21,23
Implementation	
<i>lecture12</i>	p. 5
In-memory database	
<i>lecture07</i>	p. 5
Information	
<i>lecture05</i>	p. 16,17
Information Technology	
<i>lecture05</i>	p. 16
Inheritance	
<i>lecture12</i>	p. 4-10
Inheritance (multiple)	
<i>lecture12</i>	p. 10,11
Inheritance (multiple) vs aggregate	

<i>lecture12</i>	p. 11
Inheritance and C	
<i>lecture12</i>	p. 11-13
Initialization of pointer	
<i>lecture03</i>	p. 7,8
Initialization of structure	
<i>lecture03</i>	p. 16
Initializing members	
<i>lecture12</i>	p. 3
Input/Output	
<i>lecture02</i>	p. 9,10
Insertion in a binary tree	
<i>lecture06</i>	p. 15,16
int	
<i>lecture01</i>	p. 11
integer operations	
<i>lecture01</i>	p. 11
Inter Process Communication (IPC)	
<i>lecture13</i>	p. 9-12
Interface	
<i>lecture12</i>	p. 5
iostream	
<i>lecture08</i>	p. 14
IPC	
<i>lecture13</i>	p. 9,10
isalnum()	
<i>lecture02</i>	p. 11
isalpha()	
<i>lecture02</i>	p. 11
isdigit()	
<i>lecture02</i>	p. 11
islower()	
<i>lecture02</i>	p. 11
ISO	
<i>lecture02</i>	p. 16
isprint()	
<i>lecture02</i>	p. 11
ispunct()	
<i>lecture02</i>	p. 11
isspace()	
<i>lecture02</i>	p. 11
isupper()	
<i>lecture02</i>	p. 11
IT	
<i>lecture05</i>	p. 16

Iterator	
<i>lecture12</i>	p. 26

J

Java vs C	
<i>lecture01</i>	p. 6,12-14,17,18
<i>lecture02</i>	p. 5
<i>lecture03</i>	p. 12,20
<i>lecture04</i>	p. 7
java vs C	
<i>lecture04</i>	p. 17
Java vs C	
<i>lecture05</i>	p. 1,2,16
<i>lecture06</i>	p. 23
<i>lecture09</i>	p. 15-17
Java vs C++	
<i>lecture08</i>	p. 18-20
<i>lecture10</i>	p. 13,14
<i>lecture12</i>	p. 9,10,20,21

K

K&R	
<i>lecture01</i>	p. 6
Kernighan (Brian)	
<i>lecture01</i>	p. 6
Key/Value store	
<i>lecture12</i>	p. 17
key_t	
<i>lecture13</i>	p. 11
<i>lecture13</i>	p. 10,12
kill()	
<i>lecture11</i>	p. 13
<i>lecture11</i>	p. 15

L

Lab2 hints	
<i>lecture05</i>	p. 3-5
Labs	
<i>lecture01</i>	p. 3,4

Landis, Evgenii	
<i>lecture06</i>	p. 17
Last In First Out	
<i>lecture06</i>	p. 7,8
ld	
<i>lecture01</i>	p. 20
libpthread	
<i>lecture13</i>	p. 15
Library file	
<i>lecture04</i>	p. 6
LIFO	
<i>lecture06</i>	p. 7,8
<i>lecture07</i>	p. 4
Linked list	
<i>lecture05</i>	p. 22
<i>lecture06</i>	p. 1-10,13
<i>lecture07</i>	p. 1
Linker	
<i>lecture01</i>	p. 16,17,19,20
<i>lecture04</i>	p. 7-9
Linux	
<i>lecture01</i>	p. 2
List	
<i>lecture08</i>	p. 19
list	
<i>lecture12</i>	p. 26
listen()	
<i>lecture11</i>	p. 4
Listener	
<i>lecture10</i>	p. 25
localtime()	
<i>lecture03</i>	p. 21
Locking a file	
<i>lecture04</i>	p. 2
Logical operators	
<i>lecture01</i>	p. 23
long	
<i>lecture01</i>	p. 11
<i>lecture01</i>	p. 11
Loop	
<i>lecture02</i>	p. 1

M

MAC address	
<i>lecture10</i>	p. 27
Macro	
<i>lecture09</i>	p. 4,5
main()	
<i>lecture01</i>	p. 16
make	
<i>lecture01</i>	p. 7
<i>lecture04</i>	p. 5,6
<i>lecture04</i>	p. 4-6
<i>lecture09</i>	p. 9
Makefile	
<i>lecture04</i>	p. 5,6
malloc()	
<i>lecture04</i>	p. 18-20
<i>lecture05</i>	p. 20
<i>lecture09</i>	p. 16
man	
<i>lecture02</i>	p. 5
<i>lecture10</i>	p. 20
<i>lecture10</i>	p. 21
Map	
<i>lecture12</i>	p. 26
Marker (end-of-string)	
<i>lecture01</i>	p. 14,15
Mathematical functions	
<i>lecture01</i>	p. 19,20
Mathematical functions:Compiler	
<i>lecture01</i>	p. 19
Mathematical Induction	
<i>lecture05</i>	p. 10
Mathematical induction	
<i>lecture05</i>	p. 8-10
Matrix example	
<i>lecture09</i>	p. 2,3
Maurolico, Francisco	
<i>lecture05</i>	p. 9
MD5	
<i>lecture06</i>	p. 12
memory	
<i>lecture01</i>	p. 8
<i>lecture01</i>	p. 8
Memory address	
<i>lecture01</i>	p. 9,10
Memory leak	

<i>lecture04</i>	p. 22
Mercurial	
<i>lecture09</i>	p. 21
Message nesting	
<i>lecture10</i>	p. 27
Message queue	
<i>lecture13</i>	p. 9-11,13
Method	
<i>lecture06</i>	p. 24
Method definition	
<i>lecture08</i>	p. 18
Method hiding	
<i>lecture12</i>	p. 8
Method overloading	
<i>lecture12</i>	p. 8,9
Method overriding	
<i>lecture12</i>	p. 8
Method vs function	
<i>lecture10</i>	p. 16
Methods	
<i>lecture08</i>	p. 17
<i>lecture09</i>	p. 1
Methods in structures	
<i>lecture08</i>	p. 18
Midcourse exam	
<i>lecture01</i>	p. 3
MidCourse Exam	
<i>lecture08</i>	p. 1-10
Mixing C++ and C	
<i>lecture10</i>	p. 1,2
mktime()	
<i>lecture03</i>	p. 21
msgctl()	
<i>lecture13</i>	p. 11
msgget()	
<i>lecture13</i>	p. 10
msgrcv()	
<i>lecture13</i>	p. 10
msgsnd()	
<i>lecture13</i>	p. 10
Multi-threading	
<i>lecture04</i>	p. 15,16
Multidimensional array	
<i>lecture03</i>	p. 13
Multiple inclusions	

<i>lecture09</i>	p. 3
Multiple inheritance	
<i>lecture12</i>	p. 10,11
Multiple inheritance vs aggregate	
<i>lecture12</i>	p. 11
Multiple processors	
<i>lecture13</i>	p. 13,14
Multitasking	
<i>lecture13</i>	p. 13-15
Multithreading	
<i>lecture11</i>	p. 6
Mutex	
<i>lecture14</i>	p. 2,3

N

Name of variable	
<i>lecture01</i>	p. 9
namespace	
<i>lecture08</i>	p. 14
<i>lecture12</i>	p. 1-3
Naming a structure	
<i>lecture03</i>	p. 16,17
Naming of classes, members and methods	
<i>lecture10</i>	p. 3,4
ndbm.h	
<i>lecture12</i>	p. 17
Nested structures	
<i>lecture12</i>	p. 12
Nesting functions	
<i>lecture02</i>	p. 6
Network programming	
<i>lecture10</i>	p. 22-28
<i>lecture11</i>	p. 1-6
Networks	
<i>lecture10</i>	p. 27,28
new	
<i>lecture08</i>	p. 15
<i>lecture09</i>	p. 16
nm	
<i>lecture09</i>	p. 14
Node	
<i>lecture05</i>	p. 21,22
Non binary tree	

<i>lecture06</i>	p. 20-23
Normal distribution	
<i>lecture03</i>	p. 3
Not (logical operator)	
<i>lecture01</i>	p. 23
NULL	
<i>lecture02</i>	p. 10,13
<i>lecture03</i>	p. 1,7
Nygaard, Kristen	
<i>lecture08</i>	p. 12
Nygaard, Kirsten	
<i>lecture08</i>	p. 12

O

Object	
<i>lecture08</i>	p. 19
Object creation/destruction	
<i>lecture10</i>	p. 4-6
Object modelling	
<i>lecture12</i>	p. 31
Object Oriented Programming	
<i>lecture09</i>	p. 17
Object reference	
<i>lecture08</i>	p. 19
Object-Oriented Programming	
<i>lecture06</i>	p. 24
od	
<i>lecture04</i>	p. 3
Operating system	
<i>lecture10</i>	p. 20,21
operator	
<i>lecture08</i>	p. 14,15
Operator (assignment)	
<i>lecture10</i>	p. 18-20
Operator (copy)	
<i>lecture10</i>	p. 19,20
Operator as function	
<i>lecture10</i>	p. 17-19
Operator as method	
<i>lecture10</i>	p. 17-19
Operator overloading	
<i>lecture10</i>	p. 15,16
operator()	

<i>lecture12</i>	p. 27,29
Or (logical operator)	
<i>lecture01</i>	p. 23
Order	
<i>lecture05</i>	p. 20,21
<i>lecture07</i>	p. 5
Orderly termination	
<i>lecture13</i>	p. 1
ostream	
<i>lecture10</i>	p. 17
Output overloading	
<i>lecture10</i>	p. 17
Over-engineering	
<i>lecture07</i>	p. 5
Overflow	
<i>lecture02</i>	p. 12
Overloading	
<i>lecture02</i>	p. 5
<i>lecture08</i>	p. 16
<i>lecture10</i>	p. 2
<i>lecture12</i>	p. 8,19
Overloading output operator	
<i>lecture10</i>	p. 17
Overriding	
<i>lecture12</i>	p. 8
Overriding (template)	
<i>lecture12</i>	p. 25
Overview	
<i>lecture14</i>	p. 4-8

P

Pango	
<i>lecture09</i>	p. 18
Parent class	
<i>lecture12</i>	p. 4
Parent process	
<i>lecture11</i>	p. 12
Pascal, Blaise	
<i>lecture05</i>	p. 9
PATH	
<i>lecture13</i>	p. 7
pclose()	
<i>lecture13</i>	p. 2,3

perror()		<i>lecture12</i>	p. 19
<i>lecture03</i>	p. 1	printf()	
Persistence		<i>lecture02</i>	p. 8,10
<i>lecture07</i>	p. 6	Priorities	
pid_t		<i>lecture06</i>	p. 8
<i>lecture11</i>	p. 11	private	
Pipe		<i>lecture12</i>	p. 5,6
<i>lecture02</i>	p. 9	<i>lecture12</i>	p. 6,7
Pivot		Process	
<i>lecture05</i>	p. 6-8	<i>lecture10</i>	p. 20,21
Pointer		<i>lecture11</i>	p. 11,12
<i>lecture01</i>	p. 10	<i>lecture13</i>	p. 1,14
<i>lecture03</i>	p. 4-8,19,20	Process id	
<i>lecture05</i>	p. 1,2	<i>lecture11</i>	p. 11
Pointer arithmetic		Project	
<i>lecture03</i>	p. 10,11	<i>lecture09</i>	p. 1
Pointer on a function		Propagation of exception	
<i>lecture06</i>	p. 23,24	<i>lecture12</i>	p. 16,17
Pointer on structure		protected	
<i>lecture03</i>	p. 19,20	<i>lecture12</i>	p. 6,7
Pointer to a file		Protocol	
<i>lecture03</i>	p. 25	<i>lecture10</i>	p. 25
Pointer vs array		<i>lecture11</i>	p. 7
<i>lecture03</i>	p. 6,8,9,12	Prototype (function)	
Pointers		<i>lecture01</i>	p. 16
<i>lecture04</i>	p. 11,12	<i>lecture02</i>	p. 7
Pointers as arguments to a function		ps	
<i>lecture04</i>	p. 13,14	<i>lecture11</i>	p. 12,13
<i>lecture05</i>	p. 2,3	<i>lecture13</i>	p. 2,3
Pointers as parameters		pthread.h	
<i>lecture06</i>	p. 2,3	<i>lecture13</i>	p. 15
Pointers to functions		pthread_create()	
<i>lecture09</i>	p. 16,17	<i>lecture13</i>	p. 16
popen()		pthread_exit()	
<i>lecture13</i>	p. 2,3	<i>lecture13</i>	p. 16
Port		pthread_join()	
<i>lecture10</i>	p. 25,26	<i>lecture13</i>	p. 16
<i>lecture11</i>	p. 4	<i>lecture14</i>	p. 2
Portability		public	
<i>lecture09</i>	p. 6-9	<i>lecture08</i>	p. 18
pptx		<i>lecture12</i>	p. 6,7
<i>lecture04</i>	p. 3	putchar()	
Preprocessor		<i>lecture02</i>	p. 9
<i>lecture01</i>	p. 16-18,20	puts()	
<i>lecture09</i>	p. 3-6,8,9	<i>lecture02</i>	p. 10

Q

Quality	
<i>lecture01</i>	p. 5
Queue	
<i>lecture12</i>	p. 26
<i>lecture13</i>	p. 15
Quick-sort	
<i>lecture05</i>	p. 6-8,11-14
Quiz 1	
<i>lecture07</i>	p. 6-8

R

Race condition	
<i>lecture13</i>	p. 15
Radix	
<i>lecture01</i>	p. 12
random()	
<i>lecture03</i>	p. 2,3
read()	
<i>lecture11</i>	p. 3,5
Reading ZIP or XML	
<i>lecture04</i>	p. 3
realloc()	
<i>lecture04</i>	p. 18
<i>lecture05</i>	p. 19,20
Recursion	
<i>lecture05</i>	p. 10-15
<i>lecture06</i>	p. 5,6
Recursion vs loops	
<i>lecture05</i>	p. 14
recv()	
<i>lecture11</i>	p. 3-5
Reentrant	
<i>lecture13</i>	p. 15
Reference	
<i>lecture03</i>	p. 7
Reference to structure filed	
<i>lecture03</i>	p. 16
Return value	
<i>lecture02</i>	p. 3,4,8

Return value from main()	
<i>lecture01</i>	p. 16
Ritchie (Dennis)	
<i>lecture01</i>	p. 6,7
Ritchie, Dennis	
<i>lecture01</i>	p. 6
<i>lecture14</i>	p. 5,8
Robustness	
<i>lecture01</i>	p. 5
Root	
<i>lecture06</i>	p. 14
Rounding error	
<i>lecture01</i>	p. 12
Router	
<i>lecture10</i>	p. 25

S

scanf()	
<i>lecture01</i>	p. 16
<i>lecture02</i>	p. 3,4,10
<i>lecture04</i>	p. 14
SCCS	
<i>lecture09</i>	p. 21
Schedule	
<i>lecture01</i>	p. 1
Search	
<i>lecture06</i>	p. 9,10
<i>lecture07</i>	p. 5
search.h	
<i>lecture07</i>	p. 2
<i>lecture12</i>	p. 22
Searching	
<i>lecture12</i>	p. 27
Self-managing list	
<i>lecture06</i>	p. 8
Semaphore	
<i>lecture13</i>	p. 9,11,12
semctl()	
<i>lecture13</i>	p. 12
semget()	
<i>lecture13</i>	p. 12
Semi-colon	
<i>lecture01</i>	p. 15

semop()		<i>lecture13</i>	p. 12	<i>lecture13</i>	p. 13
send()		<i>lecture11</i>	p. 5	Signal handler	
Serve		<i>lecture11</i>	p. 3,4	<i>lecture11</i>	p. 15
Server		<i>lecture11</i>	p. 4	signal()	
Set		<i>lecture13</i>	p. 14,15	<i>lecture11</i>	p. 15
setlocale		<i>lecture12</i>	p. 26	<i>lecture11</i>	p. 16
setlocale()		<i>lecture02</i>	p. 15	signal.h	
set_terminate()		<i>lecture03</i>	p. 3	<i>lecture11</i>	p. 13,15
SHA1		<i>lecture12</i>	p. 17	Signals	
Shallow copy		<i>lecture06</i>	p. 12	<i>lecture11</i>	p. 13,14
Shared library		<i>lecture10</i>	p. 14	<i>lecture13</i>	p. 1
Shared memory		<i>lecture04</i>	p. 7	signed	
shmat()		<i>lecture13</i>	p. 9,11,14	<i>lecture01</i>	p. 11,12
shmctl()		<i>lecture13</i>	p. 11	SIGSTOP	
shmget()		<i>lecture13</i>	p. 11	<i>lecture11</i>	p. 15
short		<i>lecture01</i>	p. 11	sig_t	
Side-effects		<i>lecture09</i>	p. 5	<i>lecture11</i>	p. 15
sigaction()		<i>lecture11</i>	p. 16	Simula	
SIGCHLD		<i>lecture13</i>	p. 5	<i>lecture08</i>	p. 12
SIGINT		<i>lecture13</i>	p. 1	Single quote	
SIGKILL		<i>lecture11</i>	p. 15	<i>lecture01</i>	p. 15
Signal				sizeof()	
				<i>lecture03</i>	p. 6,13
				Socket	
				<i>lecture11</i>	p. 1-3
				socket()	
				<i>lecture11</i>	p. 2
				Sorting	
				<i>lecture05</i>	p. 6-8,11-14,19
				<i>lecture12</i>	p. 27
				Source control	
				<i>lecture09</i>	p. 20,21
				Specialization	
				<i>lecture12</i>	p. 5
				Specialization (template)	
				<i>lecture12</i>	p. 25
				Splitting code	
				<i>lecture09</i>	p. 11,12
				sscanf()	
				<i>lecture01</i>	p. 16
				Stack	
				<i>lecture01</i>	p. 8
				<i>lecture04</i>	p. 9-12
				<i>lecture12</i>	p. 26
				Stack trace	

<i>lecture12</i>	p. 16
Stallman, Richard	
<i>lecture09</i>	p. 9
Standard C++ library	
<i>lecture08</i>	p. 15
Standard Template Library (STL)	
<i>lecture12</i>	p. 25-31
static	
<i>lecture04</i>	p. 7,16
<i>lecture04</i>	p. 16
<i>lecture09</i>	p. 14
<i>lecture09</i>	p. 13,14
Static analysis:oclint	
<i>lecture09</i>	p. 22
Static function	
<i>lecture09</i>	p. 14
Static variable	
<i>lecture04</i>	p. 16
static_cast	
<i>lecture12</i>	p. 31
std	
<i>lecture08</i>	p. 14
stderr	
<i>lecture02</i>	p. 9
<i>lecture02</i>	p. 9
<i>lecture03</i>	p. 1
<i>lecture10</i>	p. 1
stdin	
<i>lecture02</i>	p. 9
<i>lecture02</i>	p. 9,10
<i>lecture03</i>	p. 24,25
stdio.h	
<i>lecture03</i>	p. 25
stdlib.h	
<i>lecture03</i>	p. 1
<i>lecture04</i>	p. 18
<i>lecture11</i>	p. 12
stdout	
<i>lecture02</i>	p. 9,10
<i>lecture02</i>	p. 9,10
<i>lecture03</i>	p. 24,25
<i>lecture10</i>	p. 1
Stepanov, Alexander	
<i>lecture12</i>	p. 26
STL (Standard Template Library)	

<i>lecture12</i>	p. 25-31
Strategy	
<i>lecture06</i>	p. 7,8
strcasecmp()	
<i>lecture02</i>	p. 13
strcat()	
<i>lecture02</i>	p. 12
strchr()	
<i>lecture02</i>	p. 13
strcmp()	
<i>lecture02</i>	p. 12,13
strcpy()	
<i>lecture02</i>	p. 12
strdup()	
<i>lecture04</i>	p. 18
<i>lecture05</i>	p. 18
Stream	
<i>lecture02</i>	p. 9
Stream redirection	
<i>lecture03</i>	p. 24
strerror()	
<i>lecture03</i>	p. 1
String	
<i>lecture01</i>	p. 10,14,15
string	
<i>lecture08</i>	p. 15
String array	
<i>lecture03</i>	p. 12,13
String comparison	
<i>lecture02</i>	p. 12,13
String conversion to number	
<i>lecture03</i>	p. 1
String declaration	
<i>lecture03</i>	p. 11
String search	
<i>lecture02</i>	p. 13
string.h	
<i>lecture02</i>	p. 11-13
<i>lecture04</i>	p. 18
Strings	
<i>lecture02</i>	p. 11-13
<i>lecture03</i>	p. 1
strlen()	
<i>lecture02</i>	p. 11
strncasecmp()	

<i>lecture02</i>	p. 13
strncat()	
<i>lecture02</i>	p. 12
strncmp()	
<i>lecture02</i>	p. 12,13
strncpy()	
<i>lecture02</i>	p. 12
Strong typing	
<i>lecture12</i>	p. 18
Stroustrup, Bjarne	
<i>lecture08</i>	p. 11-14
<i>lecture10</i>	p. 13
strchr()	
<i>lecture02</i>	p. 13
strsep()	
<i>lecture02</i>	p. 14
<i>lecture12</i>	p. 28
strstr()	
<i>lecture02</i>	p. 13
strtod()	
<i>lecture03</i>	p. 1
strtok()	
<i>lecture02</i>	p. 13,14
strtok_r()	
<i>lecture12</i>	p. 28
strtol()	
<i>lecture03</i>	p. 1
Struct	
<i>lecture03</i>	p. 20
struct	
<i>lecture03</i>	p. 15,17,20,23
<i>lecture03</i>	p. 16-20
<i>lecture05</i>	p. 17
<i>lecture09</i>	p. 2
struct (C++)	
<i>lecture08</i>	p. 17
struct addrinfo	
<i>lecture11</i>	p. 2,3
struct tm	
<i>lecture03</i>	p. 21
Structure alignment	
<i>lecture03</i>	p. 18
Structure and pointer	
<i>lecture03</i>	p. 19,20
Structure initialization	

<i>lecture03</i>	p. 16
Structure naming	
<i>lecture03</i>	p. 16,17
Structures	
<i>lecture03</i>	p. 15-20
Subprocess	
<i>lecture13</i>	p. 4-7
Subversion	
<i>lecture09</i>	p. 21
super()	
<i>lecture12</i>	p. 4
switch	
<i>lecture01</i>	p. 24
Synchronization	
<i>lecture13</i>	p. 15
<i>lecture14</i>	p. 2-4
System call	
<i>lecture10</i>	p. 21
System calls	
<i>lecture10</i>	p. 20
System V	
<i>lecture10</i>	p. 22
system()	
<i>lecture11</i>	p. 12,13
<i>lecture13</i>	p. 2

T

Tail pointer	
<i>lecture06</i>	p. 7,8
TCP	
<i>lecture11</i>	p. 7
TCP/IP	
<i>lecture10</i>	p. 25,26
TCPAcceptor	
<i>lecture11</i>	p. 4,5
TCPConnector	
<i>lecture11</i>	p. 4,5,10
TCPStream	
<i>lecture11</i>	p. 4,5,10
tdelete()	
<i>lecture12</i>	p. 22
template	
<i>lecture12</i>	p. 19,20

Template (Class)		<i>lecture02</i>	p. 11
<i>lecture12</i>	p. 23,24	Tools	
Template (class)		<i>lecture09</i>	p. 1
<i>lecture12</i>	p. 22,24,25	toupper()	
Template (function)		<i>lecture02</i>	p. 11
<i>lecture12</i>	p. 18-21,25	Towers of Hanoi	
Template specialization		<i>lecture05</i>	p. 15
<i>lecture12</i>	p. 25	Tree	
Templates		<i>lecture06</i>	p. 13,20-23
<i>lecture12</i>	p. 20,21	<i>lecture07</i>	p. 1,2,4
Testing		<i>lecture12</i>	p. 22,23
<i>lecture10</i>	p. 1	try	
tfind()		<i>lecture08</i>	p. 16
<i>lecture12</i>	p. 22	tsearch()	
this		<i>lecture12</i>	p. 22
<i>lecture10</i>	p. 4	twalk()	
Thomson (Ken)		<i>lecture12</i>	p. 22
<i>lecture01</i>	p. 7	typedef	
Thomson, Ken		<i>lecture03</i>	p. 17
<i>lecture01</i>	p. 6	<i>lecture09</i>	p. 2
<i>lecture14</i>	p. 5	typename	
Thread		<i>lecture12</i>	p. 20
<i>lecture13</i>	p. 14-16	Typing	
<i>lecture14</i>	p. 1-4	<i>lecture12</i>	p. 17
Threads			
<i>lecture09</i>	p. 16		
<i>lecture11</i>	p. 6		
throw			
<i>lecture08</i>	p. 16		
<i>lecture12</i>	p. 15		
Time functions			
<i>lecture03</i>	p. 2,20,21		
time()			
<i>lecture03</i>	p. 2,21		
time.h			
<i>lecture03</i>	p. 21		
<i>lecture03</i>	p. 2,21		
timegm()			
<i>lecture03</i>	p. 21		
time_t			
<i>lecture03</i>	p. 21		
<i>lecture03</i>	p. 2		
Tokenizing			
<i>lecture02</i>	p. 13,14		
tolower()			

U

Uncaught exception		<i>lecture12</i>	p. 16,17
Unexpected exception		<i>lecture12</i>	p. 16
Unicode		<i>lecture02</i>	p. 15,17,18
union		<i>lecture03</i>	p. 23
<i>lecture03</i>	p. 24	unistd.h	
<i>lecture09</i>	p. 8	UNIX	
Unix		<i>lecture01</i>	p. 6
<i>lecture01</i>	p. 7	Unix pipe	
<i>lecture14</i>	p. 5		

<i>lecture02</i>	p. 9
unlink()	
<i>lecture04</i>	p. 2
unsigned	
<i>lecture01</i>	p. 11,12
UTF-16	
<i>lecture02</i>	p. 17
UTF-32	
<i>lecture02</i>	p. 17
UTF-8	
<i>lecture02</i>	p. 15,18

V

Variable declaration	
<i>lecture01</i>	p. 9
Variable name	
<i>lecture01</i>	p. 9
Variable number of parameters	
<i>lecture02</i>	p. 6
vector	
<i>lecture08</i>	p. 19,20
<i>lecture12</i>	p. 26,29
virtual	
<i>lecture12</i>	p. 10
<i>lecture12</i>	p. 10
Virtual machine	
<i>lecture09</i>	p. 7
Virtual method	
<i>lecture12</i>	p. 10
Visual C++	
<i>lecture09</i>	p. 22
Visual Studio	
<i>lecture01</i>	p. 7
void	
<i>lecture04</i>	p. 13
void*	
<i>lecture04</i>	p. 18
Von Neumann (John)	
<i>lecture01</i>	p. 8
Von Neumann, John	
<i>lecture04</i>	p. 9

W

wait()	
<i>lecture13</i>	p. 6
waitpid()	
<i>lecture13</i>	p. 6
Walking a binary tree	
<i>lecture06</i>	p. 16
Wall gcc flag	
<i>lecture09</i>	p. 22
wchar	
<i>lecture02</i>	p. 14,15
while	
<i>lecture02</i>	p. 1
Wide char	
<i>lecture02</i>	p. 14,15
Wikipedia reference for operators	
<i>lecture10</i>	p. 15
wine	
<i>lecture09</i>	p. 7
write()	
<i>lecture11</i>	p. 3,5

X

Xcode	
<i>lecture01</i>	p. 7
<i>lecture09</i>	p. 22
XML	
<i>lecture04</i>	p. 3

Y

Yhread	
<i>lecture13</i>	p. 15

Z

ZIP	
<i>lecture04</i>	p. 3
Zombie process	

