Intro. Computing with the C Programming Language

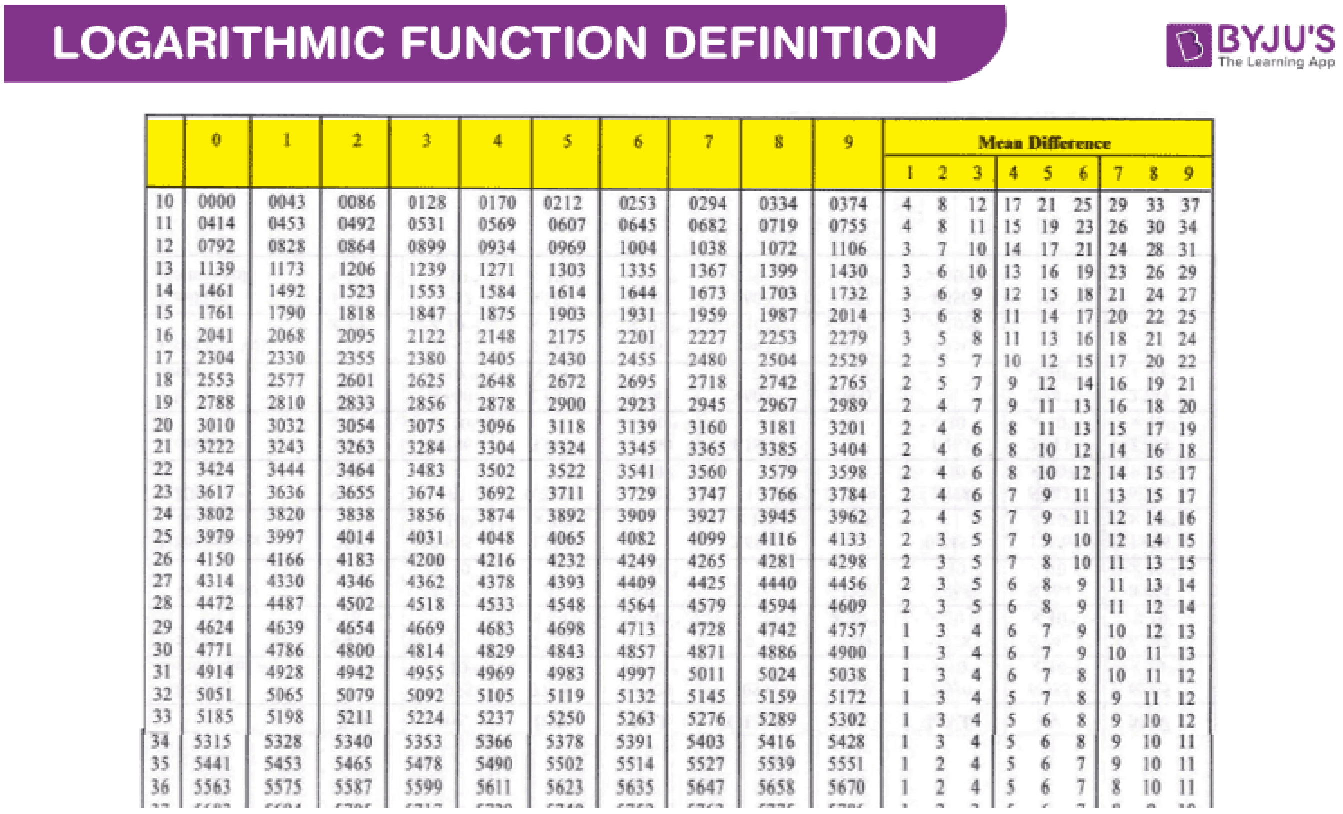
Iteration

Shin Hong

25 September 2023

Iteration

- Repeating similar tasks without making errors is something that computers do well, and people do poorly
  - e.g., generating tabular data



LOGARITHMIC FUNCTION DEFINITION

	0	1	2	3	4	5	6	7	8	9	Mean Difference								
											1	2	3	4	5	6	7	8	9
10	0000	0043	0086	0128	0170	0212	0253	0294	0334	0374	4	8	12	17	21	25	29	33	37
11	0414	0453	0492	0531	0569	0607	0645	0682	0719	0755	4	8	11	15	19	23	26	30	34
12	0792	0828	0864	0899	0934	0969	1004	1038	1072	1106	3	7	10	14	17	21	24	28	31
13	1139	1173	1206	1239	1271	1303	1335	1367	1399	1430	3	6	10	13	16	19	23	26	29
14	1461	1492	1523	1553	1584	1614	1644	1673	1703	1732	3	6	9	12	15	18	21	24	27
15	1761	1790	1818	1847	1875	1903	1931	1959	1987	2014	3	6	8	11	14	17	20	22	25
16	2041	2068	2095	2122	2148	2175	2201	2227	2253	2279	3	5	8	11	13	16	18	21	24
17	2304	2330	2355	2380	2405	2430	2455	2480	2504	2529	2	5	7	10	12	15	17	20	22
18	2553	2577	2601	2625	2648	2672	2695	2718	2742	2765	2	5	7	9	12	14	16	19	21
19	2788	2810	2833	2856	2878	2900	2923	2945	2967	2989	2	4	7	9	11	13	16	18	20
20	3010	3032	3054	3075	3096	3118	3139	3160	3181	3201	2	4	6	8	11	13	15	17	19
21	3222	3243	3263	3284	3304	3324	3345	3365	3385	3404	2	4	6	8	10	12	14	16	18
22	3424	3444	3464	3483	3502	3522	3541	3560	3579	3598	2	4	6	8	10	12	14	15	17
23	3617	3636	3655	3674	3692	3711	3729	3747	3766	3784	2	4	6	7	9	11	13	15	17
24	3802	3820	3838	3856	3874	3892	3909	3927	3945	3962	2	4	5	7	9	11	12	14	16
25	3979	3997	4014	4031	4048	4065	4082	4099	4116	4133	2	3	5	7	9	10	12	14	15
26	4150	4166	4183	4200	4216	4232	4249	4265	4281	4298	2	3	5	7	8	10	11	13	15
27	4314	4330	4346	4362	4378	4393	4409	4425	4440	4456	2	3	5	6	8	9	11	13	14
28	4472	4487	4502	4518	4533	4548	4564	4579	4594	4609	2	3	5	6	8	9	11	12	14
29	4624	4639	4654	4669	4683	4698	4713	4728	4742	4757	1	3	4	6	7	9	10	12	13
30	4771	4786	4800	4814	4829	4843	4857	4871	4886	4900	1	3	4	6	7	9	10	11	13
31	4914	4928	4942	4955	4969	4983	4997	5011	5024	5038	1	3	4	6	7	8	10	11	12
32	5051	5065	5079	5092	5105	5119	5132	5145	5159	5172	1	3	4	5	7	8	9	11	12
33	5185	5198	5211	5224	5237	5250	5263	5276	5289	5302	1	3	4	5	6	8	9	10	12
34	5315	5328	5340	5353	5366	5378	5391	5403	5416	5428	1	3	4	5	6	8	9	10	11
35	5441	5453	5465	5478	5490	5502	5514	5527	5539	5551	1	2	4	5	6	7	9	10	11
36	5563	5575	5587	5599	5611	5623	5635	5647	5658	5670	1	2	4	5	6	7	8	10	11

Iteration

- Repeating similar tasks without making errors is something that computers do well, and people do poorly
  - e.g., generating tabular data

- Beside recursion, C provides iteration with loop-statements to write repetitive programs
  - while statement
  - for statement

While statement

```
1   void countdown (int n) {
2       while (n > 0) {
3           printf(“%d\n”, n) ;
4           n = n - 1 ;
5       }
6       printf(“blastoff!\n”) ;
7   }
```

- The while statement is executed as follows:
  - evaluate the loop condition (Line 2)
  - if the condition is evaluated to true, execute the loop body (Lines 3—4)
  - if false, execute the next statement (Line 6)

- A while statement is to perform loops until the specified condition is unsatisfied

Loop condition

- The loop body should update variables to eventually make the loop condition to be false
- Infinite loop may be resulted if the loop body is not guaranteed to falsify the loop condition
- example.

```
void Sequence (int n)
{
    while (n != 1)
    {
        printf ("%i\n", n);
        if (n%2 == 0)         /* n is even */
        {
            n = n / 2;
        }
        else                  /* n is odd */
        {
            n = n*3 + 1;
        }
    }
}
```

Ex. Log Table

- `log_table.c`
  - change the log base into 2
  - limit the digits to the right of decimal point

Loop Body

- We can declare a local variable inside of a loop body
  - such a local variable cannot be accessed outside of the loop body

- We can put a loop statement to the body of another loop (i.e., nested loop)
  - a nested loop can access all local variables of its outer loop

- example
  - `multiplication_table.c`
  - `multiplication_table2.c`

For Statement

- A for-statement specifies the initialization and the update of an iterator explicitly
  - example.

```
int i, sum = 0 ;
for (i = 1 ; i <= 10 ; i = i + 1) {
    sum = sum + i ;
}
```

```
int i, sum = 0 ;
i = 1 ;
while (i <= 10) {
    sum = sum + i ;
    i = i + 1 ;
}
```

Encapsulation and Generalization by Function

- By defining a piece of code as a function, we can make the code less complex and more reusable
  - by giving a name to a sequence of statements, the program can be easier to read and debug
  - by dividing a program into functions, you can each part in isolation and compose these into a whole
  - well-defined functions are useful for many programs

- Generalization: make the logic more general such that it can be reused in multiple situations

- Encapsulation: wrap up the code such that a caller can use the logic without considering the detail