



```

20 * - From this point to the rest of the files are the components for
21 the data table lib we made it's
22
23 * - totally customizable, if you don't like something about this lib
24 you can change it.
25
26 * - It's totally type safe and easy to use.
27
28 */
29
30 export type DataTableContextType<Column extends Record<string,
31 unknown>> = {
32   pagination: TablePaginationStateType;
33   setPagination: React.Dispatch<React.
34     SetStateAction<TablePaginationStateType>>;
35   selection: TableSelectionStateType;
36   setSelection: React.Dispatch<React.
37     SetStateAction<TableSelectionStateType>>;
38   search: TableSearchStateType;
39   setSearch: React.Dispatch<React.
40     SetStateAction<TableSearchStateType>>;
41   columnsView: ColumnsViewStateType<Column>[] | undefined;
42   setColumnsView: React.Dispatch<React.
43     SetStateAction<ColumnsViewStateType<Column>>>> | undefined;
44   order: OrderStateType[];
45   setOrder: React.Dispatch<React.SetStateAction<OrderStateType>>>;
46   filterBy: FilterByType;
47   setFilterBy: React.Dispatch<React.SetStateAction<FilterByType>>>;
48 };
49
50 // table-1.tsx
51
52 // tabs
53
54 // textarea
55
56 // toast
57
58 // toggle
59
60 // toggle-group
61
62 // tooltip
63
64 // index.ts
65
66 // registry-schema
67
68 // registry-vi
69
70 // registry-vi-components
71
72 // accordion
73
74 // badge
75
76 // button
77
78 // somer
79
80 // table
81
82 // index.ts
83
84 // table.constants.ts
85
86 // table.hook.tsx
87
88 // table.lib.ts
89
90 // table.tsx
91
92 // table.types.ts
93
94 // tabs
95
96 // textarea

```

```

+ 12 // printf("%s", line);\n
+ 11 // }\n
+ 10\n
① 9 // NOTE MAKING FOR LOOP.\n
+ 8 // for (int i = 0; i < 5; i++) {\n
+ 7 // printf("%d\\n", i);\n
+ 6 // }\n
+ 5\n
① 4 // NOTE MAKING MIN MAX FUNCTION.\n
+ 3 int first = 1;\n
+ 2 int min, max, val;\n
+ 1\n
+ 40 while (scanf("%d", &val) != EOF) {\n
+ 1\n
+ 2     if (first || val > max)\n
+ 3     |     max = val;\n
+ 4     if (first || val < min)\n
+ 5     |     min = val;\n
+ 6     first = 0;\n
+ 7\n
+ 8 }\n
+ 9 printf("%d", min);\n
+ 10 printf("%d", max);\n
+ 11\n
+ 12 return 0;\n
+ 13 }\n

```

→ ch\_1 git:(master) × gcc ./index.c -o index && ./index

3  
3  
3  
3  
3  
3  
3  
3  
3  
3

● NORMAL +40 ~3 index.c c 75% 40:33

<EARNING/c\_learning/ch\_1/index.c" 53L, 900B written

🔍 /index

~/w1ldduck/LOW LEVEL LEARNING/c\_learning/ch\_1

```

9 #include <stdio.h>
8 #include <string.h>
+ 7 #include <wchar.h>
+ 6
+ 5 void read_file(char _file_name[]) {
+ 4     FILE *file;
+ 3     file = fopen(_file_name, "r");
+ 2     char text[30];
+ 1
+ 10     printf("reading %s\n", _file_name);
+ 1
+ 2     for (int i = 0; fgets(text, 30, file) != NULL; ++i) {
+ 3         printf("%d %s", i, text);
+ 4     }
+ 5
+ 6     char word;
+ 7     int word_count = 0;
+ 8
+ 9     for (int i = 0; (word = getw(file)) != EOF; i++) {
+ 10         if (word == "*" || word == *"\t" || word == *"\n") {
+ 11             ++word_count;
+ 12         }
+ 13     }
+ 14     printf("\nthe words count for '%s' is %d word\n", _file_name,
+ 15         word_count);

```

→ ch\_2 git:(master) × gcc ./index.c -o index && ./index

THIS IS CHAPTER 2 OF C

reading index.ts

```

0 export type User = {
1   name: string;
2   age: number;
3   gender: "male" | "female";
4
5 };

```

the words count for `index.ts` is 0 word

→ ch\_2 git:(master) ×



```

8 #include <stdio.h>
7 #include <string.h>
6
+ 5 void read_file(char _file_name[]) {
+   4   FILE *file;
+   3   file = fopen(_file_name, "r");
+   2   char text[30];
+   1
+ 9   printf("%s\n", _file_name);
+   1
+ 2   for (int i = 0; fgets(text, 30, file) !=
+     NULL; i++) {
+ 3     | printf("%d %s", i, text);
+ 4   }
+ 5 }

```

→ ch\_2 git:(master) × gcc ./index.c -o index && ./index

THIS IS CHAPTER 2 OF C

index.ts

```

0 export type User = {
1   name: string;
2   age: number;
3   gender: "male" | "female";
4
5 };

```

→ ch\_2 git:(master) ×