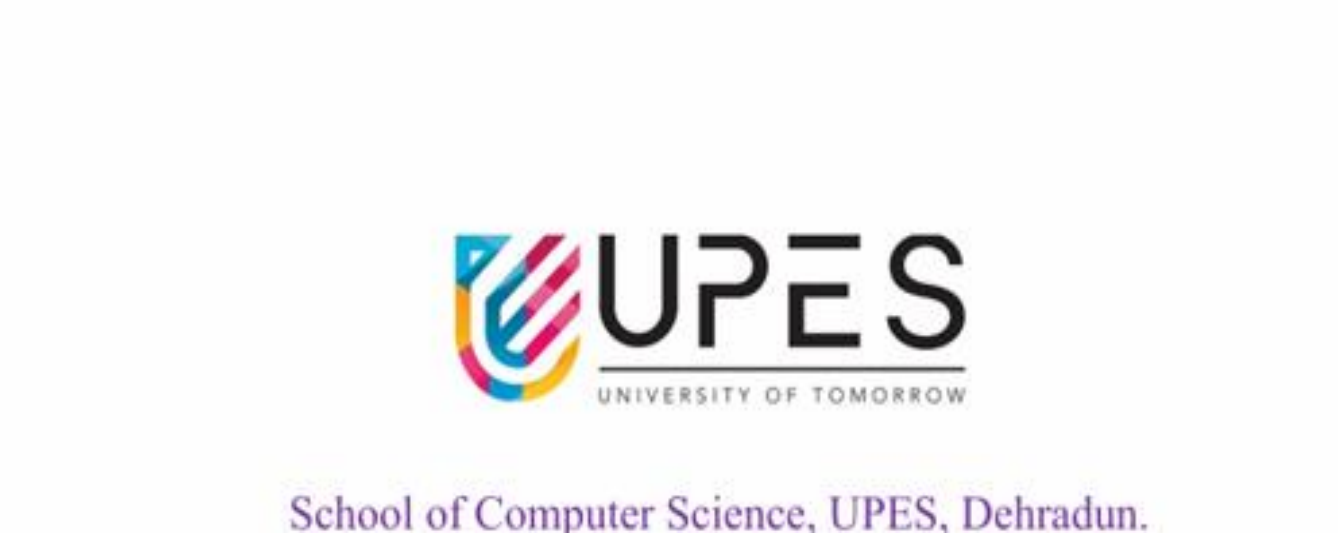


Modern Periodic Table Console Application

BTech CSE, UPES Dehradun

Submitted by:

Dikshita Choudhary

SAP ID :590026654

Course: C Programming

Date: November 29, 2025

Abstract

This project’s all about building a simple console app in C that gives you the full modern periodic table.The program packs in data for all 118 elements—atomic numbers, names, symbols, electron configurations—you name it.You can search for elements by

atomic number, name, or symbol, and it doesn't matter how you type it in. Want to see the whole table? That's there too. The app uses structures and arrays, and it shows off file handling, string tricks, modular programming, and menus that make sense. It's perfect for learning or quick reference.All the main features work smoothly, with input checks and clean, readable output.

Problem Definition

If you've studied chemistry, you know the pain—sometimes you just need quick look at the periodic table, but dragging around a giant chart or trying to find a stable internet connection is a hassle. This project fixes that.It's a lightweight, offline console program that lets you look up any element or scroll through the whole table, right from your computer.

Input:

Pick an option from the menu (1-4), then enter either a name, symbol, or atomic number (1-118).

```
PS C:\Users\ASUS\Desktop\Sem 1\Dikshita> gcc try.c
PS C:\Users\ASUS\Desktop\Sem 1\Dikshita> ./a.exe
Welcome to the Modern Periodic Table!

============================
     Modern Periodic Table
============================
1. Search by Atomic Number
2. Search by Name or Symbol
3. List all elements
4. Exit
Enter your choice: 1
Enter atomic number (1-118): 6
```

Output:

You get detailed info for the element—name, symbol, atomic number, electron configuration—or you see the entire periodic table. If you mess up the input, it tells you.

Just keep in mind: it’s fixed at 118 elements, runs in the console, and doesn’t use any file I/O.

System Design

Algorithm / Pseudocode

MAIN PROGRAM:

1. Load up an array with all 118 elements.2. Show a welcome menu.

3. While the user wants to keep going:

   a. Display the options (1: Search by atomic number, 2: Search by name or symbol, 3: List all, 4: Exit)

   b. Get the user’s choice.c. If they pick 1, search by atomic number.

   d. If they pick 2, search by name or symbol (case-insensitive).

   e. If they pick 3, show all elements.

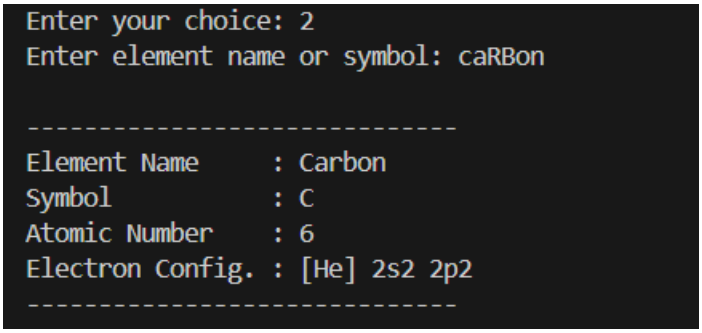
   f. If they pick 4, exit the program nicely.

   g. Anything else—let them know it’s not a valid option.SEARCH BY ATOMIC NUMBER:

- Loop through each element. If the atomic number matches, show the details and you’re done. If not, say it wasn’t found.

SEARCH BY NAME/SYMBOL:

- Go through each element and check if the name or symbol matches (doesn’t matter how you type it). If it does, show the details. If not, say it wasn’t found.



DISPLAY ELEMENT:

- Print the name, symbol, atomic number, and electron configuration in a neat block.### Data Flow

User Input → Menu Handler → Search Functions → PeriodicTable Array → Formatted Output

↓

Error Messages ← Validation Checks

Key Data Structure

```
typedef struct {
    int Atomic_Number;
    char Name[20];
    char Symbol[3];
    char Electron_Config[50];
} Element;
```

Implementation Details

Main Components

1. Global Data Structure : There's a hardcoded array called PeriodicTable[118] that holds all the element info for reliability.

2. Core Functions :

- printElementInfo(Element E): Neatly prints the details for a single element.

- searchByAtomicNumber(int number): Checks the array for the right atomic number.

- strCaseCmp(const char *s1, const char *s2): Custom function to compare strings, ignoring case.

- searchByNameOrSymbol(char input[]): Looks for a match in either the name or symbol.

- listAllElements(): Runs through all 118 elements and lists them.3. **Key Code Snippet – Case-Insensitive Search:

```
int strCaseCmp(const char *s1, const char *s2) {
    while (*s1 && *s2) {
        if (tolower(*s1) != tolower(*s2))
            return tolower(*s1) - tolower(*s2);
        s1++; s2++;
    }
    return tolower(*s1) - tolower(*s2);
}
```

4. Input Handling : For numbers, "scanf()" does the job. For strings, it uses "fgets()" and strips out the newline with "strcspn()" so you don't get weird input bugs.

5. Modular Design : Every big task is its own function, so the code stays tidy and easy to follow.

Test Cases

Test Case -01 (Carbon Search):

Enter atomic number (1-118): 6

Element Name : Carbon

Symbol : C

Atomic Number : 6

Electron Config : [He] 2s2 2p2

Sample Console Output

Test Case -01 (Carbon Search):

Enter atomic number (1-118): 6

Element Name : Carbon

Symbol : C

Atomic Number : 6

Electron Config. : [He] 2s2 2p2

Test Case -02 (Case Insensitive):

Enter element name or symbol: HYDROGEN

Element Name : Hydrogen

Symbol : H

Atomic Number : 1

Electron Config.: 1s1

```
Enter your choice: 1
Enter atomic number (1-118): 6

------------------------------
Element Name     : Carbon
Symbol           : C
Atomic Number    : 6
Electron Config. : [He] 2s2 2p2
------------------------------

============================
     Modern Periodic Table
============================
1. Search by Atomic Number
2. Search by Name or Symbol
3. List all elements
4. Exit
Enter your choice: 2
Enter element name or symbol: c

------------------------------
Element Name     : Carbon
Symbol           : C
Atomic Number    : 6
Electron Config. : [He] 2s2 2p2
------------------------------
```

Conclusion

This project nailed what it set out to do: a working, easy-to-use periodic table app that puts solid C programming skills on display.It covers the basics like structures, arrays, handling strings, breaking things into functions, and interactive input/output. The case-insensitive search helps a lot, and the menu is simple enough that anyone can use it for quick chemistry lookups.What went well:

- All 118 elements included
- Search works great, no errors
- The console interface looks greate
- Errors are handled carefully

Appendix - Complete Source Code

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

/*
 * Structure definition for each element of the Periodic Table.
 * Stores atomic number, name, symbol, and electron configuration.
 */
typedef struct
{
    int Atomic_Number;
```

```
    char Name[20];
    char Symbol[3];
    char Electron_Config[50];
}
Element;

// Array holding details of all 118 elements in the periodic table.

    // Populate with tuples: {atomic number, name, symbol, electron configuration}

Element PeriodicTable[118] =
{
    {1, "Hydrogen", "H", "1s1"},
    {2, "Helium", "He", "1s2"},
    {3, "Lithium", "Li", "[He] 2s1"},
    {4, "Beryllium", "Be", "[He] 2s2"},
    {5, "Boron", "B", "[He] 2s2 2p1"},
    {6, "Carbon", "C", "[He] 2s2 2p2"},
    {7, "Nitrogen", "N", "[He] 2s2 2p3"},
    {8, "Oxygen", "O", "[He] 2s2 2p4"},
    {9, "Fluorine", "F", "[He] 2s2 2p5"},
    {10, "Neon", "Ne", "[He] 2s2 2p6"},
    {11, "Sodium", "Na", "[Ne] 3s1"},
    {12, "Magnesium", "Mg", "[Ne] 3s2"},
    {13, "Aluminium", "Al", "[Ne] 3s2 3p1"},
    {14, "Silicon", "Si", "[Ne] 3s2 3p2"},
    {15, "Phosphorus", "P", "[Ne] 3s2 3p3"},
```

{16, "Sulfur", "S", "[Ne] 3s2 3p4"},

{17, "Chlorine", "Cl", "[Ne] 3s2 3p5"},

{18, "Argon", "Ar", "[Ne] 3s2 3p6"},

{19, "Potassium", "K", "[Ar] 4s1"},

{20, "Calcium", "Ca", "[Ar] 4s2"},

{21, "Scandium", "Sc", "[Ar] 3d1 4s2"},

{22, "Titanium", "Ti", "[Ar] 3d2 4s2"},

{23, "Vanadium", "V", "[Ar] 3d3 4s2"},

{24, "Chromium", "Cr", "[Ar] 3d5 4s1"},

{25, "Manganese", "Mn", "[Ar] 3d5 4s2"},

{26, "Iron", "Fe", "[Ar] 3d6 4s2"},

{27, "Cobalt", "Co", "[Ar] 3d7 4s2"},

{28, "Nickel", "Ni", "[Ar] 3d8 4s2"},

{29, "Copper", "Cu", "[Ar] 3d10 4s1"},

{30, "Zinc", "Zn", "[Ar] 3d10 4s2"},

{31, "Gallium", "Ga", "[Ar] 3d10 4s2 4p1"},

{32, "Germanium", "Ge", "[Ar] 3d10 4s2 4p2"},

{33, "Arsenic", "As", "[Ar] 3d10 4s2 4p3"},

{34, "Selenium", "Se", "[Ar] 3d10 4s2 4p4"},

{35, "Bromine", "Br", "[Ar] 3d10 4s2 4p5"},

{36, "Krypton", "Kr", "[Ar] 3d10 4s2 4p6"},

{37, "Rubidium", "Rb", "[Kr] 5s1"},

{38, "Strontium", "Sr", "[Kr] 5s2"},

{39, "Yttrium", "Y", "[Kr] 4d1 5s2"},

{40, "Zirconium", "Zr", "[Kr] 4d2 5s2"},

{41, "Niobium", "Nb", "[Kr] 4d4 5s1"},

{42, "Molybdenum", "Mo", "[Kr] 4d5 5s1"},

{43, "Technetium", "Tc", "[Kr] 4d5 5s2"},

{44, "Ruthenium", "Ru", "[Kr] 4d7 5s1"},

{45, "Rhodium", "Rh", "[Kr] 4d8 5s1"},

{46, "Palladium", "Pd", "[Kr] 4d10"},

{47, "Silver", "Ag", "[Kr] 4d10 5s1"},

{48, "Cadmium", "Cd", "[Kr] 4d10 5s2"},

{49, "Indium", "In", "[Kr] 4d10 5s2 5p1"},

{50, "Tin", "Sn", "[Kr] 4d10 5s2 5p2"},

{51, "Antimony", "Sb", "[Kr] 4d10 5s2 5p3"},

{52, "Tellurium", "Te", "[Kr] 4d10 5s2 5p4"},

{53, "Iodine", "I", "[Kr] 4d10 5s2 5p5"},

{54, "Xenon", "Xe", "[Kr] 4d10 5s2 5p6"},

{55, "Cesium", "Cs", "[Xe] 6s1"},

{56, "Barium", "Ba", "[Xe] 6s2"},

{57, "Lanthanum", "La", "[Xe] 5d1 6s2"},

{58, "Cerium", "Ce", "[Xe] 4f1 5d1 6s2"},

{59, "Praseodymium", "Pr", "[Xe] 4f3 6s2"},

{60, "Neodymium", "Nd", "[Xe] 4f4 6s2"},

{61, "Promethium", "Pm", "[Xe] 4f5 6s2"},

{62, "Samarium", "Sm", "[Xe] 4f6 6s2"},

{63, "Europium", "Eu", "[Xe] 4f7 6s2"},

{64, "Gadolinium", "Gd", "[Xe] 4f7 5d1 6s2"},

{65, "Terbium", "Tb", "[Xe] 4f9 6s2"},

{66, "Dysprosium", "Dy", "[Xe] 4f10 6s2"},

{67, "Holmium", "Ho", "[Xe] 4f11 6s2"},

{68, "Erbium", "Er", "[Xe] 4f12 6s2"},

{69, "Thulium", "Tm", "[Xe] 4f13 6s2"},

{70, "Ytterbium", "Yb", "[Xe] 4f14 6s2"},

{71, "Lutetium", "Lu", "[Xe] 4f14 5d1 6s2"},

{72, "Hafnium", "Hf", "[Xe] 4f14 5d2 6s2"},

{73, "Tantalum", "Ta", "[Xe] 4f14 5d3 6s2"},

{74, "Tungsten", "W", "[Xe] 4f14 5d4 6s2"},

{75, "Rhenium", "Re", "[Xe] 4f14 5d5 6s2"},

{76, "Osmium", "Os", "[Xe] 4f14 5d6 6s2"},

{77, "Iridium", "Ir", "[Xe] 4f14 5d7 6s2"},

{78, "Platinum", "Pt", "[Xe] 4f14 5d9 6s1"},

{79, "Gold", "Au", "[Xe] 4f14 5d10 6s1"},

{80, "Mercury", "Hg", "[Xe] 4f14 5d10 6s2"},

{81, "Thallium", "Tl", "[Xe] 4f14 5d10 6s2 6p1"},

{82, "Lead", "Pb", "[Xe] 4f14 5d10 6s2 6p2"},

{83, "Bismuth", "Bi", "[Xe] 4f14 5d10 6s2 6p3"},

{84, "Polonium", "Po", "[Xe] 4f14 5d10 6s2 6p4"},

{85, "Astatine", "At", "[Xe] 4f14 5d10 6s2 6p5"},

{86, "Radon", "Rn", "[Xe] 4f14 5d10 6s2 6p6"},

{87, "Francium", "Fr", "[Rn] 7s1"},

{88, "Radium", "Ra", "[Rn] 7s2"},

{89, "Actinium", "Ac", "[Rn] 6d1 7s2"},

{90, "Thorium", "Th", "[Rn] 6d2 7s2"},

{91, "Protactinium", "Pa", "[Rn] 5f2 6d1 7s2"},

{92, "Uranium", "U", "[Rn] 5f3 6d1 7s2"},

{93, "Neptunium", "Np", "[Rn] 5f4 6d1 7s2"},

{94, "Plutonium", "Pu", "[Rn] 5f6 7s2"},

{95, "Americium", "Am", "[Rn] 5f7 7s2"},

{96, "Curium", "Cm", "[Rn] 5f7 6d1 7s2"},

```
    {97, "Berkelium", "Bk", "[Rn] 5f9 7s2"},
    {98, "Californium", "Cf", "[Rn] 5f10 7s2"},
    {99, "Einsteinium", "Es", "[Rn] 5f11 7s2"},
    {100, "Fermium", "Fm", "[Rn] 5f12 7s2"},
    {101, "Mendelevium", "Md", "[Rn] 5f13 7s2"},
    {102, "Nobelium", "No", "[Rn] 5f14 7s2"},
    {103, "Lawrencium", "Lr", "[Rn] 5f14 7s2 7p1"},
    {104, "Rutherfordium", "Rf", "[Rn] 5f14 6d2 7s2"},
    {105, "Dubnium", "Db", "[Rn] 5f14 6d3 7s2"},
    {106, "Seaborgium", "Sg", "[Rn] 5f14 6d4 7s2"},
    {107, "Bohrium", "Bh", "[Rn] 5f14 6d5 7s2"},
    {108, "Hassium", "Hs", "[Rn] 5f14 6d6 7s2"},
    {109, "Meitnerium", "Mt", "[Rn] 5f14 6d7 7s2"},
    {110, "Darmstadtium", "Ds", "[Rn] 5f14 6d9 7s1"},
    {111, "Roentgenium", "Rg", "[Rn] 5f14 6d10 7s1"},
    {112, "Copernicium", "Cn", "[Rn] 5f14 6d10 7s2"},
    {113, "Nihonium", "Nh", "[Rn] 5f14 6d10 7s2 7p1"},
    {114, "Flerovium", "Fl", "[Rn] 5f14 6d10 7s2 7p2"},
    {115, "Moscovium", "Mc", "[Rn] 5f14 6d10 7s2 7p3"},
    {116, "Livermorium", "Lv", "[Rn] 5f14 6d10 7s2 7p4"},
    {117, "Tennessine", "Ts", "[Rn] 5f14 6d10 7s2 7p5"},
    {118, "Oganesson", "Og", "[Rn] 5f14 6d10 7s2 7p6"}
};

/*
 * Prints the details of an element, nicely formatted.
 */
```

```
void printElementInfo(Element E)
{
    printf("\n-----------------------------\n");
    printf("Element Name     : %s\n", E.Name);
    printf("Symbol           : %s\n", E.Symbol);
    printf("Atomic Number    : %d\n", E.Atomic_Number);
    printf("Electron Config. : %s\n", E.Electron_Config);
    printf("-----------------------------\n");
}

/*
 * Search for an element by its atomic number.
 * Returns 1 if found, 0 if not found.
 */
int searchByAtomicNumber(int number)
{
    for (int i = 0; i < 118; i++)
    {
        if (PeriodicTable[i].Atomic_Number == number)
        {
            printElementInfo(PeriodicTable[i]);
            return 1;
        }
    }
    // Element not found case
    return 0;
}
```

```
/*
 * Case-insensitive comparison of two strings.
 * Returns 0 if same, non-zero if different.
 */
int strCaseCmp(const char *s1, const char *s2)
{
    while (*s1 && *s2)
        {
        if (tolower(*s1) != tolower(*s2))
            return tolower(*s1) - tolower(*s2);
        s1++;
        s2++;
    }
    // Handles difference in string length too
    return tolower(*s1) - tolower(*s2);
}


/*
 * Search for an element by its name or symbol.
 * Returns 1 if found, 0 if not found.
 */
int searchByNameOrSymbol(char input[])
{
    for (int i = 0; i < 118; i++) {
        if (strCaseCmp(input, PeriodicTable[i].Name) == 0 || strCaseCmp(input,
PeriodicTable[i].Symbol) == 0)
```

```
        {
            printElementInfo(PeriodicTable[i]);
            return 1;
        }
    }
    // In case no matching element is found
    return 0;
}

/*
 * List all elements in the periodic table.
 * Displays atomic number, symbol, and name.
 */
void listAllElements()
{
    printf("\nList of Elements (Atomic Number : Symbol - Name):\n");
    for (int i = 0; i < 118; i++) {
        printf("%3d : %3s - %s\n", PeriodicTable[i].Atomic_Number,
PeriodicTable[i].Symbol, PeriodicTable[i].Name);
    }
}

/*
 * Main entry point of the program.
 * Displays menu, takes user input, and calls requested features.
 */
int main()
```

```
{
    int choice;
    printf("Welcome to the Modern Periodic Table!\n");

    // Main interactive loop
    while (1)
        {
        printf("\n============================\n");
        printf("     Modern Periodic Table    \n");
        printf("============================\n");
        printf("1. Search by Atomic Number\n");
        printf("2. Search by Name or Symbol\n");
        printf("3. List all elements\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar(); // Consume leftover newline

        if (choice == 1)
        {
            int num;
            printf("Enter atomic number (1-118): ");
            scanf("%d", &num);
            getchar(); // Clear input buffer

            // If element not found, inform user
```

```
    if (!searchByAtomicNumber(num)) {
        printf("No element found with atomic number %d\n", num);
    }
} else if (choice == 2)
{
    char input[50];
    printf("Enter element name or symbol: ");
    fgets(input, sizeof(input), stdin);
    input[strcspn(input, "\n")] = 0; // Remove newline char from input

    // Notify if search fails
    if (!searchByNameOrSymbol(input)) {
        printf("No element found with name or symbol '%s'\n", input);
    }
}
else if (choice == 3)
{
    listAllElements(); // Display the full table
}
else if (choice == 4)
{
    printf("Thank you for using the Modern Periodic Table. Goodbye!\n");
    break;
}
else
{
    // Handle invalid choices
```

```
            printf("Invalid choice. Please enter 1 to 4.\n");
        }
    }

    return 0;          // Successful program termination
}
```