Project Report: UrduIO — Custom Input/Output Header in C

***DAY 1(Adding likhde and parhle)***  
The goal of this mini-project was to create a custom C header file that redefines standard I/O functions (printf and scanf) into user-defined Urdu-style commands:  
- likhde() → for output (replaces printf)  
- parhle() → for input (replaces scanf)  
  
We aimed to make C behave more like Python in terms of syntax simplicity, so users could write:

* age = parhle("Apni age batao: ");  
  likhde("Tumhari age hai: ", age);

instead of the traditional:

* printf("Apni age batao: ");  
  scanf("%d", &age);  
  printf("Tumhari age hai: %d", age);

# Challenges Faced

C’s input/output system heavily relies on format specifiers like %d and %s. We wanted to remove those entirely — similar to Python’s print function. However, C doesn’t support runtime type inference.

C doesn’t support function overloading (same function name, different argument types) natively. That made it difficult to have a single likhde() function handle integers, floats, and strings.

Dev-C++ uses an older GCC build which doesn’t always play nicely with \_Generic, the C11 feature used for type detection. This caused linker and semantic errors.

# Solutions & Techniques Used

We created a header called urduio.h, which included all our custom functions marked as static to avoid linker conflicts.

We introduced several variants (likhde2, likhde2s, likhde2f) to mimic overloading and handle multiple data types.

# Example Code

#include "ccp headers/urduio.h"  
  
int main(void) {  
 int age;  
 char name[30];  
  
 age = parhle("Apni age batao: ");  
 likhde2("Tumhari age hai:", age);  
  
 parhle\_str("Apna naam batao: ", name, sizeof(name));  
 likhde2s("Tumhara naam hai:", name);  
  
 return 0;  
}

**Output**:

Apni age batao: 19  
Tumhari age hai: 19  
Apna naam batao: Ali  
Tumhara naam hai: Ali

# Debugging

We faced several warnings and linker errors during development, mostly related to implicit declarations and cached object files in Dev-C++. Using static inline functions and the Rebuild All option fixed most of these issues.

• Add colorized terminal output.

• Support multiple chained arguments in likhde().