1. Hello01.c

This program shows the basic shape of a C program. It has a main function, which is where everything starts running. Inside it, the code uses printf to show the message "hello world!!" on the screen.

2. Hello02.c

Here, an integer variable named n is created and given a value. The program then prints that value by using %d inside printf, which lets you display the number as part of a sentence.

3. Control01.c

This file shows how C handles different loop structures. It uses for, while, and do-while loops to print number sequences. It also includes an if-else section to show how the program changes its behavior based on a condition.

4. Types01.c

This program focuses on how integer division works. When two integers are divided, C removes the decimal part and keeps only the whole number. The output shows both numbers and the truncated result.

5. Types02.c

This example uses the long int type, which can store bigger numbers than a regular int. It also shows that you must use %ld in printf when you want to print a long int value.

6. Types03.c

This program introduces floating-point numbers with the float type. It performs a calculation with decimals and prints the result using %f, which is the correct format for floating-point output.

7. Types04.c

Here, the program uses the double type, which gives more accuracy than float. It performs division and prints the more precise value using %lf in printf.

8. Memory01.c

This code prints the values of an int and a long int. It also uses sizeof to show how many bytes of memory each of these data types takes up inside the computer.

9. Memory02.c

This program explains pass-by-value. When a value is sent to a function, only a copy is passed. So even if the function changes the value inside itself, the original value in main stays the same.

10. Memory04.c

This file shows pass-by-reference. Instead of sending a value, the program sends the memory address using a pointer. Because of that, the function can directly change the original variable by accessing its memory.

11. Structs01.c

This example creates a struct, which groups related variables together into a single data type. It shows how to get the values inside the struct using the dot operator, and how to access them using a pointer with the arrow operator.

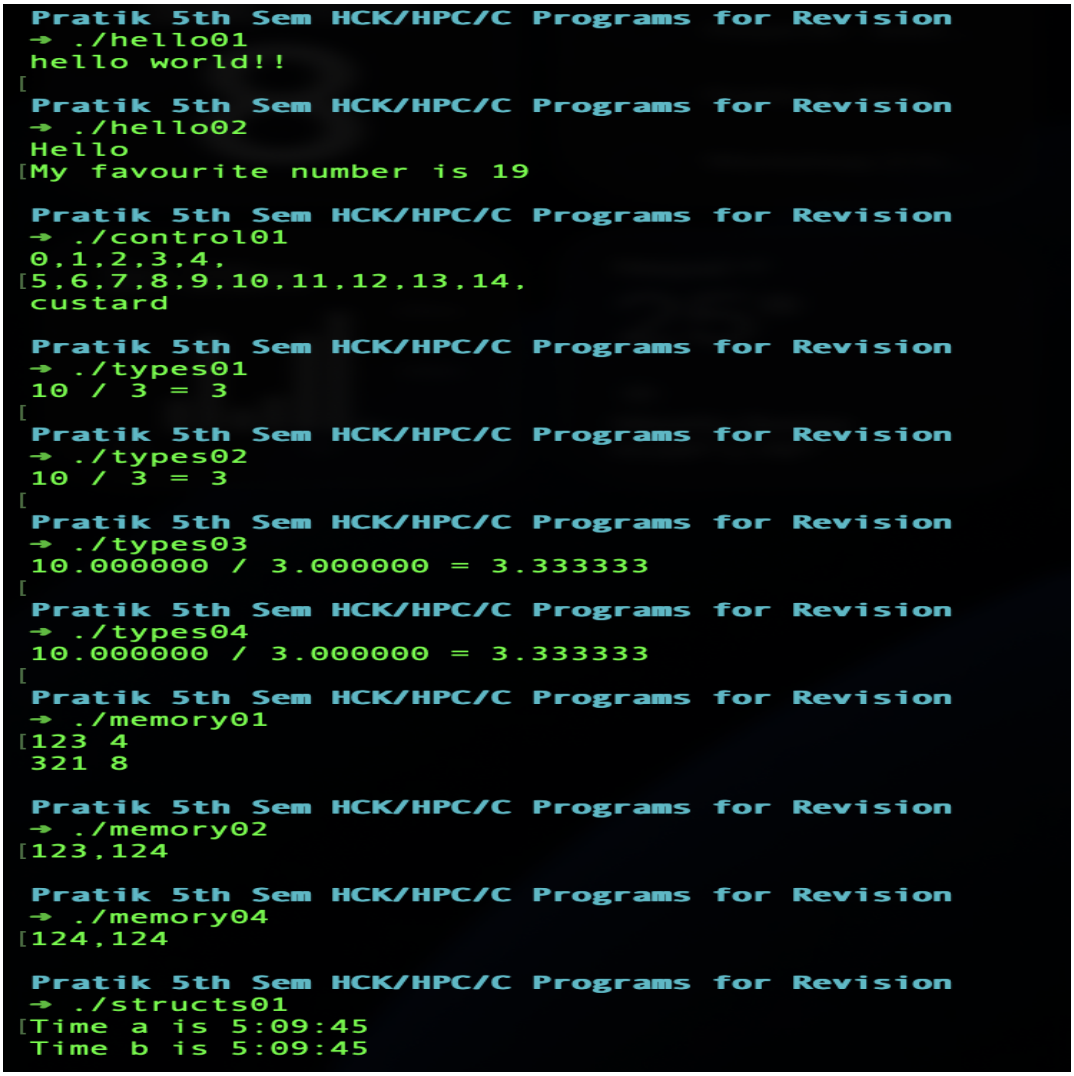
12. Memory10.c

This program uses malloc to create an integer array while the program is running. It also prints the size of the pointer and compares it to the size of the actual data type being pointed to.

13. Memory11.c

In this program, an array is created using dynamic memory allocation. Pointer arithmetic is used to fill the array and read its elements. A second pointer is moved through the memory block, showing how arrays can be handled directly through pointer movement in C.

Appendix



```
Pratik 5th Sem HCK/HPC/C Programs for Revision
➜ ./memory02
[123,124

Pratik 5th Sem HCK/HPC/C Programs for Revision
➜ ./memory04
[124,124

Pratik 5th Sem HCK/HPC/C Programs for Revision
➜ ./structs01
[Time a is 5:09:45
Time b is 5:09:45

Pratik 5th Sem HCK/HPC/C Programs for Revision
➜ ./memory10
[8
4
0,0
1,2
2,4
3,6
4,8
5,10
6,12
7,14
8,16
9,18

Pratik 5th Sem HCK/HPC/C Programs for Revision
➜ ./memory11
[0,0
1,2
2,4
3,6
4,8
5,10
6,12
7,14
8,16
9,18
```