Introduction to Computer Programming

- Week 10&11

- *Eng. Sylvain Manirakiza* -

November 23, 2025

Chap 6. Files Handling

- Week 10&11

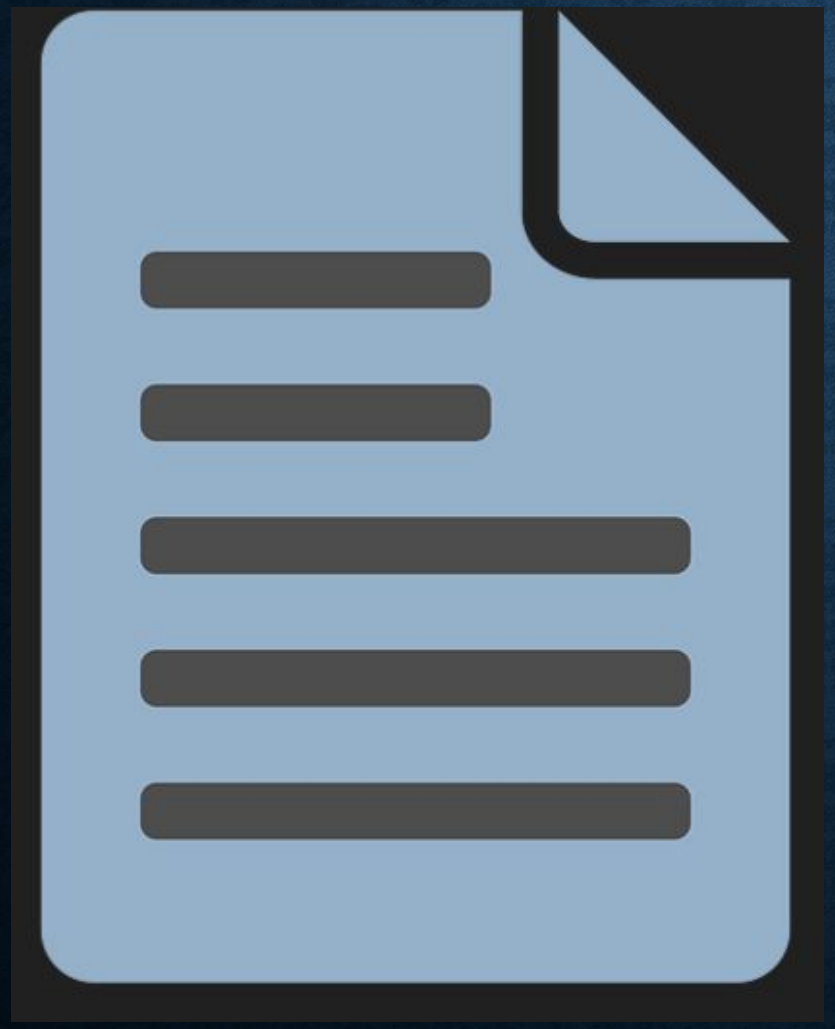
- *Eng. Sylvain Manirakiza* -

Sun - Nov 23rd, 2025

Recap of Chap 5: Pointers and Memory management

1. Pointers store memory addresses.
2. Dynamic memory allocation allows runtime memory management.
3. Proper memory management prevents leaks and errors.
4. Pointers enable powerful programming techniques.

What is a file?

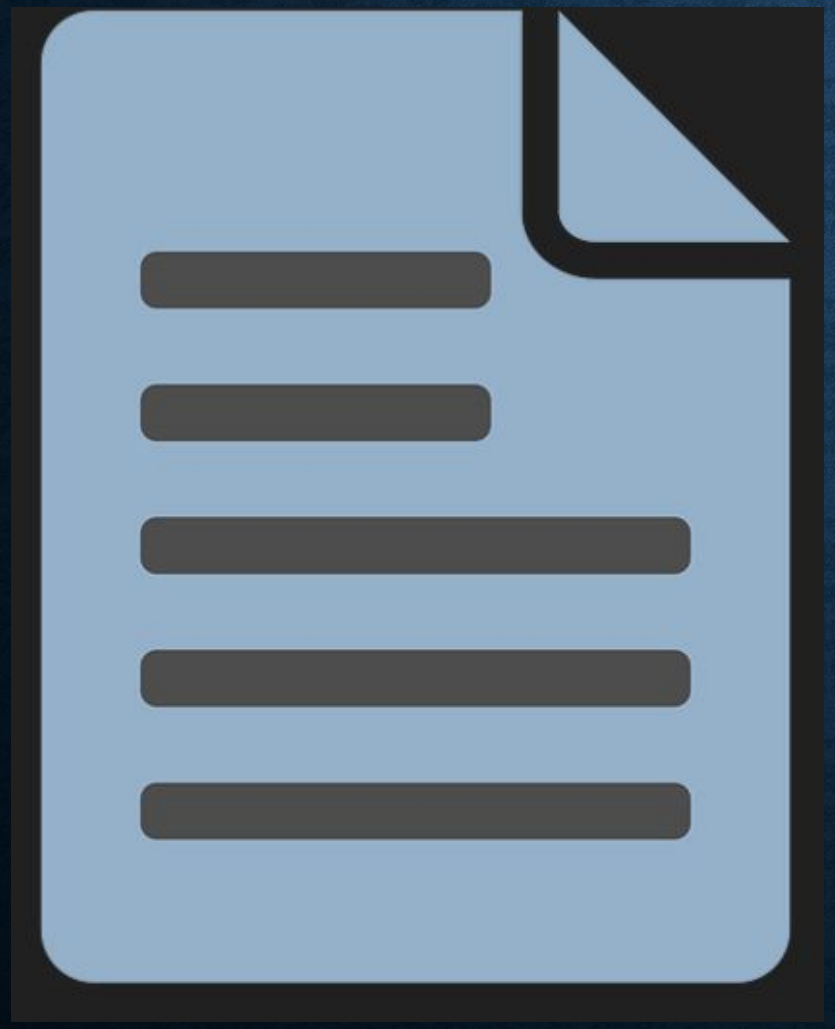


A named collection of data **stored permanently on a disk** (like a hard drive or flash drive), outside the program's memory.

What is File Management in Programming?

- File management in programming, also known as file handling, refers to the process of **creating**, **opening**, **reading, writing**, **Appending**, and **closing** files within a program.
- This is a fundamental aspect of programming that allows applications to interact with persistent storage, such as hard drives, to store and retrieve data

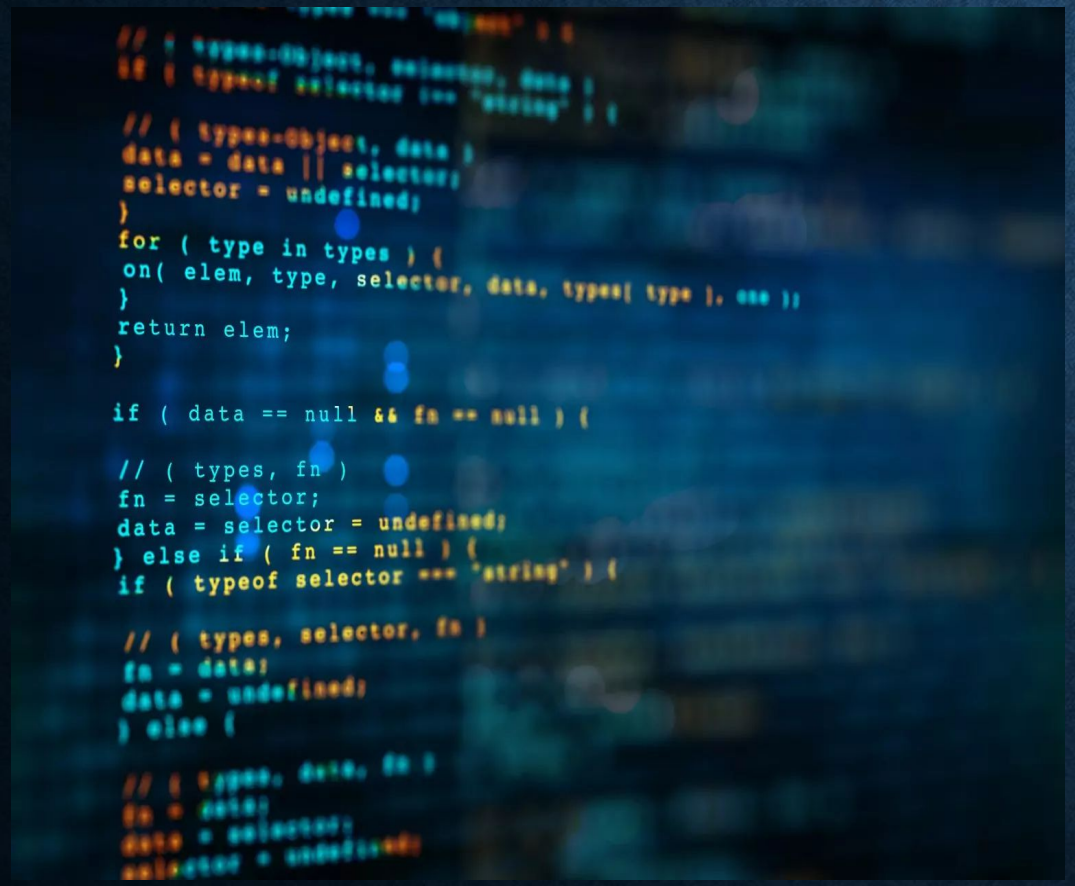
Files vs. Variables



Unlike variables (which disappear when the program ends), a file:

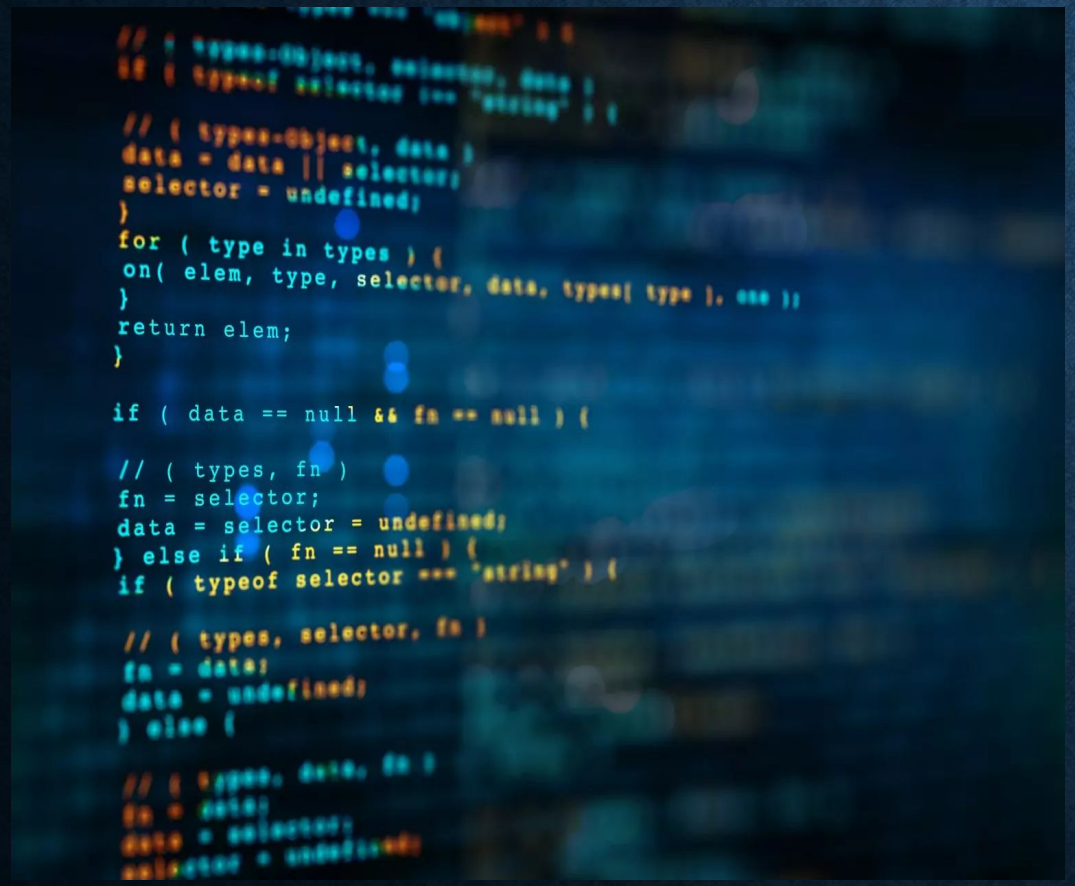
- Keeps the data after the program stops
- Can be reused, shared, and opened any time

Why Do Programmers Work with Files?



Purpose	Example
Store user data	Save names, scores, or preferences
Log program activity	Store user login records
Read configuration	Load settings from a file
Transfer data	Export reports, import databases

Importance of File Management



1. **Permanent storage**: Unlike variables (RAM), file data is stored permanently.
2. **Data sharing:** Share data between programs or sessions.
3. **Large data handling:** Store large amounts of information efficiently.

Types of File Access

Access Mode	Description	Common Use
Read (r)	Open a file for reading only	Load existing data
Write (w)	Creates or replaces a file	Create or reset a file
Append(a)	Add new data without erasing	Add logs or new entries

File Handling Operations - General syntax - Pseudocode

Open File → Open<filename> for <mode

Modes:

- Read → Input
- Write → Output
- Append → appending

Closing File → Close<filename>

Example - Open

```
Open "students.txt" For Input
Open "report.txt" For Output
Open "log.txt" For Append
```

Example - Writing to a File

```
Start
  Open "marks.txt" For Output
Write "Student: Aline, Score: 85" To
"marks.txt"
Close "marks.txt"
End
```

Example - Reading from a File

```
Start
  Open "marks.txt" For Input
Read line From "marks.txt"
Output line
Close "marks.txt"
End
```

Example - Reading from a File

Output

```
Name: Aline, Score: 85
```

File Reading – Line by Line or Until EOF

```
Start
Open "marks.txt" For Input

While NOT EOF("marks.txt") Do
    Read line From "marks.txt"
    Output line
End While
Close "marks.txt"
End
```

Appending to a File

What is Appending? - Appending means adding data to the end of an existing file without deleting what's already there.

Pseudocode Example: Appending to a File

```
Start
  Open "log.txt" For Append
  Write "User logged in at 10:30 AM" to “log.text”
  Close “logFile”
End
```

Output in file (after multiple runs):

Output

```
User logged in at 10:30 AM
User logged in at 11:45 AM
```

📌 Appending is useful for logs, activity tracking, or adding entries.

File Error Handling and Best Practices

Common Errors	Cause	Example
File not found	Trying to read a non-existent file	Open "data.txt" For Reading
Permission denied	Trying to write to a locked file	OS/user restrictions
Not closing file	Can cause memory/resource issues	Forgetting Close()

Best Practice

- Always **check if file exists** before reading.
- Always **close the file** after operations.
- Use **clear file names** and **correct file modes**.
- Use EndOfFile(file) when reading in a loop.
- When writing, confirm whether you want to **overwrite** or **append.**

How This Connects with Previous Chapters

Chapter	How it connects to File Management
Variables	You use variables to store file content temporarily
Control Structures	Use loops to read all lines until EndOfFile()
Modular Programming	You can write functions to read/write files
Pointers & Memory	Files interact with memory indirectly - what you read is stored in memory, and what you write is transferred from memory

Recap/Summary of File Management

What You've Learned:

- How to open, read, write, and append to a file
- When to use different file access modes
- How to read files line by line using a loop
- The importance of closing files and handling errors

Exercices

Activity #1: Writing Student Records

Write pseudocode to:

- Ask the user for a student's name and score.
- Save the record to grades.txt (e.g., "Alice - 92").
- Repeat for 3 students.

Exercices

Activity #2: Reading and Printing Records

- Open *grades.txt* for reading.
- Read all student records line by line.
- Print each student's name and score.

Exercices

Activity #3: Append a New Record

- Open the same file in append mode.
- Add one new student record to the file.
- Then read and display all records.

Answers to the exercices

Activity #1: Collect 3 student records and save them

```
Open "grades.txt" For Output      // create or overwrite
Set count ← 1
While count ≤ 3 Do
    Output "Enter student name:"
    Input name
    Output "Enter student score:"
    Input score
Write name & " - " & score To "grades.txt"
    count ← count + 1
End While
Close "grades.txt"
```

Activity #1: Example of Output

Student Records
Alice - 92
Brian - 85
Cynthia - 78

Activity 2: Read and display records

```
Open "grades.txt" For Input

Output "Student Records (Initial):"
While NOT EOF("grades.txt") Do
    Read line From "grades.txt"
    Output line
End While

Close "grades.txt"
```

Activity 3: Append one more student record to the same file

```
Open "grades.txt" For Append

Output "Enter NEW student name to append:"
Input newName
Output "Enter NEW student score:"
Input newScore
Write newName & " - " & newScore To "grades.txt"
Close "grades.txt"
```

Activity 3&4: Read all records again and show

```
Open "grades.txt" For Input

Output "Student Records (After Append):"
While NOT EOF("grades.txt") Do
    Read line From "grades.txt"
    Output line
End While

Close "grades.txt"
```

Activity 4: Output example - After Append

Student Records
Alice - 92
Brian - 85
Cynthia - 78
David - 88

More exercises on File management

In your respective groups, please find the time to discuss and collaborate to solve these exercise(use that link to access them). It will help you to master this chapter. *No submission is required.*

End