

FINAL PROJECT HANDbook

Programming in C

Survival Guide For DJJJJ



March 23, 2025

DJJJ

NSCC Ecampus Programing Year 1

**Names:** Dawson Brown, Judah Csanyi, Joshua Leslie, Jeremy Paruch

**Student Numbers:** W0468898, W0509673,W0228010, W0222971

**Due Date:** April 14, 2025

**Course:** PROG 2007 – Programing In C

**Instructor:** Nadia Gouda

Table of Contents

[Programming in C 3](#_Toc195451477)

[Project Overview 3](#_Toc195451478)

[The Presentation 3](#_Toc195451479)

[What do we need for this project 4](#_Toc195451480)

[How the heck do we do this 7](#_Toc195451481)

[The Group Policy (Ethos) 9](#_Toc195451482)

[Commitment to Participation 9](#_Toc195451483)

[Respect and Communication 9](#_Toc195451484)

[Accountability 9](#_Toc195451485)

[Quality 9](#_Toc195451486)

[Support 9](#_Toc195451487)

[The Group Policy Continued (Ethos) 10](#_Toc195451488)

[Conflict Resolution 10](#_Toc195451489)

[Transparency 10](#_Toc195451490)

[Work-Life Balance 10](#_Toc195451491)

[Decision-Making Process 10](#_Toc195451492)

[Who is taking care of what? 11](#_Toc195451493)

[Dawson 11](#_Toc195451494)

[Jeremy 11](#_Toc195451495)

[Josh 11](#_Toc195451496)

[Judah 11](#_Toc195451497)

[Speculation of Project Development 12](#_Toc195451498)

[Week 1: 12](#_Toc195451499)

[Week 2: 12](#_Toc195451500)

[Week 3: 12](#_Toc195451501)

[Week 4 April 13-16) 12](#_Toc195451502)

[Project Documentation Notes 13](#_Toc195451503)

[April 01 13](#_Toc195451504)

[April 04 13](#_Toc195451505)

[April 07 13](#_Toc195451506)

# Programming in C

## Project Overview

We must create a Smart Business Management System for a client looking to automate their operations.

The client is a mid-sized retail store that struggles with:

Inventory Management (Tracking stock, adding/removing items)

Employee Payroll (Calculating salaries, handling overtime)

Customer Transactions (Processing purchases and discounts)

Your team needs to develop a C-based software that automates these tasks efficiently while using the core concepts learned in this course.

Each team must assign roles (e.g., Lead Developer, UI Designer, File Handling Expert, Testing & Documentation).

**We MUST submit a Project report (5–7 pages), Power Point Presentation**

## **The Presentation**

Will be going over the demonstration of the program itself. The Presentation is 10 Minutes long,

Explaining:

Why it should be our system

How it improves business efficiency

Demonstrate the key features

Team roles and contributions

# What do we need for this project

Functionality & Implementation (40 marks)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Excellent (10)** | **Good (7-9)** | **Fair (4-6)** | **Poor (0-3)** |
| **User Registration & Login System** | Fully functional login system for customers and employees with secure password handling. | Login system works but lacks security features (e.g., weak password validation). | Basic login system, but missing key features like user roles. | No functional login system implemented. |
| **Inventory & Sales Management** | Items can be added, removed, updated, and displayed correctly. Sales records stored. | Inventory and sales system works but has minor bugs. | Limited inventory functionality (e.g., only adding items works). | Inventory system missing or non-functional. |
| **Employee Payroll System** | Payroll calculations include hours worked, overtime, and bonuses. | Payroll system calculates basic salaries but lacks advanced features (overtime, bonuses). | Payroll system implemented but contains errors or incomplete logic. | Payroll system missing or does not function correctly. |
| **File Handling & Data Persistence** | Data is saved and retrieved using files. Program resumes with stored data. | File handling works, but lacks full persistence (e.g., only some data is saved). | Partial file handling; some features do not persist. | No file handling or data is lost when program exits. |

Code Quality & Efficiency (25 marks)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Excellent (10)** | **Good (7-9)** | **Fair (4-6)** | **Poor (0-3)** |
| **Code Readability & Structure** | Well-structured, modular code with meaningful function names and comments. | Code is mostly readable, with some lack of structure. | Code is functional but poorly structured and difficult to follow. | Code is disorganized with little to no structure or comments. |
| **Use of Functions & Modularity** | Efficient use of functions and header files to organize the program. | Uses functions but lacks modularity (e.g., excessive use of global variables). | Some functions used, but most of the logic is in main(). | No meaningful functions; everything written in main(). |
| **Use of Pointers & Dynamic Memory Allocation** | Proper memory management (malloc(), free()) used effectively. | Dynamic memory used but minor memory leaks present. | Basic use of pointers; some incorrect memory management. | No dynamic memory allocation or incorrect pointer usage. |
| **Error Handling & Security** | Robust input validation and protection against invalid inputs. | Basic input validation with minor security issues. | Limited error handling, program crashes on invalid inputs. | No error handling; program easily crashes. |

📌 Presentation & Client Pitch (20 marks)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Excellent (10)** | **Good (7-9)** | **Fair (4-6)** | **Poor (0-3)** |
| **Convincing Business Case** | Clear explanation of benefits for the client; persuasive pitch. | Good explanation, but lacks a strong business argument. | Some benefits explained, but lacks persuasion. | No clear business case presented. |
| **Live Demonstration** | Smooth, well-organized demo covering all key features. | Demo is functional but has minor errors. | Demo presented, but with major issues or missing features. | No working demo provided |

**Team Collaboration & Documentation (15 marks)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Excellent (5)** | **Good (4)** | **Fair (2-3)** | **Poor (0-1)** |
| **Teamwork & Role Assignment** | Clear role distribution, all members contributed. | Most members contributed, but roles were unclear. | Unequal workload; some members contributed less. | Poor teamwork or one person did all the work. |
| **Project Report & Code Documentation** | Clear, well-written report and code comments explaining logic. | Report and comments are mostly clear but missing some details. | Report and comments are present but lack clarity. | Report missing or very unclear. |
| **On-Time Submission** | Submitted before the deadline. | Submitted on the deadline. | Submitted late by 1-2 days. | Submitted late by more than 2 days. |

# How the heck do we do this

Module 1-3: Data Types, Operators, and Flow Control

User Registration & Login System

* Customers and employees should have separate logins.
* Use arrays & strings to store names and passwords.

Discount System for Customers

Use conditional statements to apply a discount if a customer’s total purchase is above $100

Loops, Arrays, Pointers, and Strings

Inventory Management

* Use loops and arrays to track items.
* Allow users to add, remove, or update stock levels.

Payroll System for Employees

* Calculate salaries based on hours worked and overtime pay.
* Implement a structure (struct) to store employee details.

Dynamic Memory, Sorting, and Functions

Use dynamic memory (malloc()) to store transaction history.

Implement a sorting algorithm to display top-selling products.

File Handling: Saving and Retrieving Data

* Store all customer transactions and inventory data in a file.
* Allow the system to load data from files when restarted.

Module 10-12: Preprocessor Directives, Functions, and Security

Modularized Code

* Split your program into multiple functions for readability.
* Use header files (.h) to organize functions properly.

Preprocessor Directives & Security Features

Use conditional compilation to handle different environments.

Prevent invalid inputs using error handling techniques

**Presentation Requirements (Convincing the Client!)**

Each team must deliver a 10-minute presentation explaining:

Why the client should choose your system

How your system improves business efficiency

Live demonstration of key features

Team roles and contributions

# The Group Policy (Ethos)

## Commitment to Participation

We will make an effort to contribute equally to our projects by attending meetings and fulfilling our responsibilities in tasks.

## Respect and Communication

We aim to promote open and respectful communication so that every team member feels comfortable sharing their ideas, concerns, and feedback.

## Accountability

Each of us is accountable for completing our assigned tasks in a timely manner. If an unexpected situation arises, it’s essential to inform the team early and collaborate on finding a solution.

## Quality

Our goal is to deliver high-quality work instead of hurrying, as rushing could lead to errors.

## Support

We will actively support each other by sharing our knowledge and offering help when someone is facing difficulties.

# The Group Policy Continued (Ethos)

## Conflict Resolution

In the event of disagreements, team members should tackle conflicts with a focus on solutions and aim for a compromise; if this fails, faculty intervention may be necessary.

## Transparency

We will keep each other informed and updated on our progress regarding the projects.

## Work-Life Balance

We will strive to avoid overworking ourselves and acknowledge the significance of taking breaks.

## Decision-Making Process

We will conduct group votes before making decisions. The majority will decide; in the case of a tie, the designated tiebreaker will make the final decision on the course of action.

# Who is taking care of what?

## Dawson

Explanation for presentation with Jeremy/Bug Catcher

## Jeremy

Explanation with presentation with Dawson/Bug Catcher

## Josh

Lead Programmer

## Judah

Bug Catcher/Login System

# Speculation of Project Development

## Week 1:

Day 1: Kickoff, assign tasks, create shared task board/tracker

Day 2-3 Work on OOP

Day 4-5: continue on OOP for two hours, switch to starting client side programming for two hours

Day 6-7: OOP build continues, assign light tasks for Linux project and C project

## Week 2:

Day 8-10: Finish Project 1 Proto Type (April 2nd)

Day 11-14 Full Sprint Client Side Development (Target prototype April 05)

Parllel Coding/set up for C Project

## Week 3:

Days 15-16 Push C project to April 09 deadline

Days 17-18 Work on Linux Project (april 12 deadline)

Day 19-20 (Review ALL PROTOTYPES, Start presentation prep)

## Week 4 April 13-16)

Refine All Projects

Fix Bugs

Finish Documentation

Rehearse Presentations

|  |  |  |
| --- | --- | --- |
| Project | Prototype Deadlines | Buffer/Polish |
| OOP | April 02 | April 07-13 |
| Client Side | April 05 | April 07-13 |
| C Project | April 09 | April 10-13 |
| Linux | April 12 | April 13-16 |

# Project Documentation Notes

## April 01

Basic menu was established, inventory add items and inventory process sales. Followed shortly. Tasks broken down into several functions, shortly there after. Testing commenced.

Figure :Proof of Example: Menu

A screen shot of a computer program

AI-generated content may be incorrect.

## April 04

Inventory display functions were established and tested.

Figure : Example Inventory Display Functions

A screen shot of a computer program

AI-generated content may be incorrect.

## April 07

* Added employee functions. payroll systems established and tested.

Figure :Examples of Employee and Payroll Systems

A screen shot of a computer code

AI-generated content may be incorrect.

* File reading and file loading were completed on the same day

Figure :Example of File Saving and Reading

A screen shot of a computer code

AI-generated content may be incorrect.

Judah completed the clear screen function, Jeremy established a working pause function, and the login was finalized by Judah. Appropriate testing conducted and concluded,

Figure :Proof of Clear Screen and Press enter

A computer screen shot of a computer code

AI-generated content may be incorrect.

## April 13.

Finalizing Filing system, final inspection and testing before April 14th’s presentation day.

Figure : Finalizing Code Part 1

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Finalizing Code Part 2

A screen shot of a computer screen

AI-generated content may be incorrect.

Figure : Finalizing Code Part 3

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Finalizing Code Part 4

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Finalizing Code Part 5

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Finalizing Code Part 6

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Finalizing Code Part 7

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Finalizing Code Part 8

A screen shot of a computer program

AI-generated content may be incorrect.