C programming Project

Submitted By : Shiva Vashisth

Sap Id : 590025394

Submitted To : Mr. Rahul Prashad

BRANCH : SCHOOL OF COMPUTER SCIENCE

BATCH NUMBER : 19

YEAR : 2025_26

1. Project Title

EMPLOYEE PAYROLL SYSTEM PROJECT

Using C Programming & File Handling

2. Introduction

The **Employee Payroll System** is a C-based application designed to manage essential salary details of employees in an organization. It allows the user to enter employee information, calculate salary components automatically, and display payroll details.

This project demonstrates how basic programming concepts in C such as **structures, arrays, functions, loops, and conditional statements** can be used to create a useful application.

3. Objective of the Project

The main objectives of the Employee Payroll System are:

- To store and manage employee details such as ID, Name, and Salary.
- To perform **salary calculations** automatically (HRA, DA, PF, Gross Salary).
- To allow the user to **add multiple employees** and **view payroll reports**.
- To provide a simple, menu-driven interface for easy user interaction.
- To demonstrate the usage of **structures and functions** in C programming.

4. Technologies Used

3. Technologies Used

Technology	Purpose
C Programming Language	Core logic and implementation
GCC Compiler / VS Code / CodeBlocks / Turbo C	Writing & executing the program
Standard Input/Output Library (stdio.h)	Reading input, printing output
User-defined Structures	Organizing employee data
Functions	Code reusability & modular programming

PROJECT FEATURES

1. Add New Employee

- Allows the user to enter basic employee information:
  - Employee ID
  - Employee Name
  - Basic Salary
- Automatically stores the data in an array of structures.

2. Automatic Salary Calculation

The system automatically calculates:

- **HRA (House Rent Allowance)** – 20% of basic salary
- **DA (Dearness Allowance)** – 15% of basic salary
- **PF (Provident Fund)** – 12% of basic salary
- **Gross Salary** using the formula:

Gross Salary = Basic + HRA + DA – PF

3. View All Employee Payrolls

- Displays a neat, formatted list of all stored employees.
- Shows:
  - ID
  - Name
  - Basic Salary
  - HRA
  - DA
  - PF
  - Gross Salary

4. Menu-Driven Interface

- Simple, user-friendly interface.
- The user can repeatedly choose options until they exit.
- Easy navigation without complex commands.

5. Storage for Multiple Employees

- Can store details of up to **50 employees** using an array.
- Ensures efficient memory usage with structures.

6. Error Handling

- Prevents adding employees beyond the maximum limit.

- Shows a message when no employees are available to display.
- Handles invalid menu choices.

7. Modular Code Design

- Uses functions like:
  - addEmployee()
  - viewEmployees()
- Makes the code readable, maintainable, and easy to extend.

8. Fast and Lightweight

- Works on any C compiler.
- No external dependencies.
- Uses simple logic, making it efficient and fast.

7. Explanation of Code (Very Simple Language)

#include <stdio.h>

#define MAX 50

- stdio.h is included to use input/output functions like printf() and scanf().
- MAX is defined as **50**, meaning the program can store details of up to 50 employees.

2. Structure Definition

```
struct Employee {
   int id;
   char name[30];
   float basicSalary;
   float houseRentAllowance;
   float dearnessAllowance;
   float providentFund;
```

```
    float grossSalary;
};
```

- A **structure** named Employee is created to group all the details related to one employee.
- It stores ID, Name, Salary Components, and Gross Salary.
- Using a structure makes the program organized and easy to manage.

3. Function Prototypes

```
void addEmployee(struct Employee emp[], int *count);
void viewEmployees(struct Employee emp[], int count);
```

- These tell the compiler that these functions exist.
- Helps in writing a clean, modular program.

4. Main Function

```
int main() {
    struct Employee emp[MAX];
    int count = 0;
    int choice;
```

- An array emp of size MAX is declared to store multiple employees.
- count keeps track of how many employees are added.

Menu Loop

```
while (1) {
    printf("\n===== EMPLOYEE PAYROLL SYSTEM =====\n");
    printf("1. Add Employee\n");
    printf("2. View Payroll\n");
    printf("3. Exit\n");
```

- while(1) creates an **infinite loop**, allowing the user to keep using the system until they choose to exit.

Input and Switch-Case

```
scanf("%d", &choice);
```

```
switch (choice) {
   case 1: addEmployee(emp, &count); break;
   case 2: viewEmployees(emp, count); break;
   case 3: return 0;
   default: printf("Invalid choice! Try again.\n");
}
```

- Takes user choice and performs the selected operation.
- return 0 exits the program.

5. addEmployee() Function

```
void addEmployee(struct Employee emp[], int *count) {
```

- Takes the employee array and count pointer.
- Uses pointer *count so that changes update the original count.

Check Maximum Limit

```
if (*count >= MAX) {
   printf("Cannot add more employees.\n");
   return;
}
```

- Prevents memory overflow by limiting employees to 50.

Reading Employee Data

```
scanf("%d", &emp[*count].id);
scanf("%s", emp[*count].name);
scanf("%f", &emp[*count].basicSalary);
```

- Stores ID, Name, and Basic Salary in the emp[count] structure.

Salary Calculations

```
emp[*count].houseRentAllowance = emp[*count].basicSalary * 0.20;
emp[*count].dearnessAllowance  = emp[*count].basicSalary * 0.15;
```

emp[*count].providentFund = emp[*count].basicSalary * 0.12;

- Calculates:
    - HRA = 20%
    - DA = 15%
    - PF = 12%

Gross Salary Calculation

emp[*count].grossSalary = emp[*count].basicSalary
+ emp[*count].houseRentAllowance
+ emp[*count].dearnessAllowance
- emp[*count].providentFund;

- Gross Salary = Basic + HRA + DA – PF

Increase Employee Count

(*count)++;

- Updates the number of stored employees.

6. viewEmployees() Function

void viewEmployees(struct Employee emp[], int count) {

- Displays details of all employees saved in the array.

Check for Empty List

```
if (count == 0) {
   printf("No employees to display.\n");
   return;
}
```

- If no employee is added, it stops.

Display Loop

```
for (int i = 0; i < count; i++) {
   printf("ID       : %d\n", emp[i].id);
   printf("Name     : %s\n", emp[i].name);
```

printf("basicSalary   : %.2f\n", emp[i].basicSalary);

- Loops through all employees and prints their details.
- Shows Basic Salary, HRA, DA, PF, and Gross Salary in a neatly formatted way.

8. Output Screenshots (Explain in Text)

===== EMPLOYEE PAYROLL SYSTEM =====

1. Add Employee

2. View Payroll

3. Exit

Enter choice:

- This is the main menu.
- The user must enter a number (1, 2, or 3) to continue.

Enter Employee ID: 101

Enter Employee Name: Rahul

Enter basicSalary Salary: 15000

Employee added successfully!

The program asks for employee details:

ID

Name

Basic Salary

After entering the values, the salary components (HRA, DA, PF) and Gross Salary are automatically calculated.

Finally, it shows “Employee added successfully!”

Then the menu appears again.

3️⃣ If the user selects Option 2 – “View Payroll”

yaml

Copy code

===== PAYROLL DETAILS =====

Employee 1

ID : 101

Name : Rahul

basicSalary : 15000.00

houseRentAllowance (20%) : 3000.00

dearnessAllowance (15%) : 2250.00

providentFund (12%) : 1800.00

grossSalary = 18450.00

This output shows all salary information of each employee.

The program displays:

✓ Employee ID

✓ Employee Name

✓ Basic Salary

✓ HRA (20%)

✓ DA (15%)

✓ PF deduction (12%)

✓ Gross Salary (Final Salary)

If more employees were added, the program prints each employee one by one in the same format.

4️⃣ If the user selects Option 3 – “Exit”

Copy code

Exiting...

The program closes successfully.

5️⃣ If the user enters an invalid choice

powershell

Copy code

Invalid choice! Try again.

The menu appears again and the user must enter a correct option.

9. Conclusion

The Employee Payroll System is a simple yet effective project that demonstrates real-world application of C programming concepts. It teaches:

- How to use structures to handle complex data.
- How to build a menu-driven program.
- How to apply functions for better code organization.
- How salary computations can be automated using formulas.

This project is ideal for beginners learning C and serves as a strong foundation for more advanced management systems

11. References

- C Programming by Yashwant Kanetkar
- Online C documentation
- File Handling tutorials

CODE :

```
#include <stdio.h>

#define MAX 50

// Structure to store employee details
struct Employee {
   int id;
   char name[30];
   float basicSalary;     // basicSalary salary
   float houseRentAllowance;       // House Rent Allowance
   float dearnessAllowance;        // Dearness Allowance
   float providentFund;        // Provident Fund
   float grossSalary;     // grossSalary
};

int main();
```

```
// Function prototypes
void addEmployee(struct Employee emp[], int *count);
void viewEmployees(struct Employee emp[], int count);

// Main Program
int main() {
    struct Employee emp[MAX];
    int count = 0;
    int choice;

    while (1) {
        printf("\n===== EMPLOYEE PAYROLL SYSTEM =====\n");
        printf("1. Add Employee\n");
        printf("2. View Payroll\n");
        printf("3. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addEmployee(emp, &count);
                break;

            case 2:
                viewEmployees(emp, count);
                break;

            case 3:
                printf("Exiting...\n");
                return 0;

            default:
                printf("Invalid choice! Try again.\n");
        }
    }
}
```

```
// Function to add a new employee
void addEmployee(struct Employee emp[], int *count) {
    if (*count >= MAX) {
        printf("Cannot add more employees.\n");
        return;
    }

    printf("\nEnter Employee ID: ");
    scanf("%d", &emp[*count].id);

    printf("Enter Employee Name: ");
    scanf("%s", emp[*count].name);

    printf("Enter basicSalary Salary: ");
    scanf("%f", &emp[*count].basicSalary);

    // Salary calculations
    emp[*count].houseRentAllowance = emp[*count].basicSalary * 0.20;     // 20% of
basicSalary
    emp[*count].dearnessAllowance  = emp[*count].basicSalary * 0.15;     // 15% of
basicSalary
    emp[*count].providentFund  = emp[*count].basicSalary * 0.12;     // 12% of basicSalary
    emp[*count].grossSalary = emp[*count].basicSalary + emp[*count].houseRentAllowance
+ emp[*count].dearnessAllowance - emp[*count].providentFund;

    (*count)++;

    printf("Employee added successfully!\n");
}

// Function to view payroll of all employees
void viewEmployees(struct Employee emp[], int count) {
    if (count == 0) {
        printf("No employees to display.\n");
        return;
    }
```

```
    printf("\n===== PAYROLL DETAILS =====\n");

    for (int i = 0; i < count; i++) {
        printf("\nEmployee %d\n", i + 1);
        printf("ID        : %d\n", emp[i].id);
        printf("Name      : %s\n", emp[i].name);
        printf("basicSalary     : %.2f\n", emp[i].basicSalary);
        printf("houseRentAllowance (20%%) : %.2f\n", emp[i].houseRentAllowance);
        printf("dearnessAllowance (15%%)  : %.2f\n", emp[i].dearnessAllowance);
        printf("providentFund (12%%)  : %.2f\n", emp[i].providentFund);
        printf("grossSalary = %.2f\n", emp[i].grossSalary);
    }
}
```

OUTPUT :

Step : 1

. Program Starts

The program begins execution and enters an infinite loop.
It first displays the main menu:

===== EMPLOYEE PAYROLL SYSTEM =====

1. Add Employee

2. View Payroll

3. Exit

Enter choice:

Step : 2

The program will ask:

Enter Employee ID:

→ user inputs an integer

Enter Employee Name:

→ user inputs a string (name)

Enter basicSalary Salary:

→ user enters a float (basic salary)

3. Program Calculates Salary Components

For the given basic salary (let's call it **B**):

- House Rent Allowance (HRA) = 20% of B
- Dearness Allowance (DA) = 15% of B
- Provident Fund (PF) = 12% of B
- Gross Salary = B + HRA + DA – PF

After this calculation, it prints:

Employee added successfully!

Then returns to the main menu again:

===== EMPLOYEE PAYROLL SYSTEM =====

1. Add Employee

2. View Payroll

3. Exit

Enter choice:

☑ **4. If user enters choice 2 (View Payroll)**

If no employee was added:

No employees to display.

If employees exist, it displays payroll details like this:

===== PAYROLL DETAILS =====

Employee 1

ID : 101

Name : Rohan

basicSalary : 30000.00

houseRentAllowance (20%) : 6000.00

dearnessAllowance (15%) : 4500.00

providentFund (12%) : 3600.00

grossSalary = 34900.00

If multiple employees were added, it prints details for each one sequentially.

☑ **5. If user enters choice 3 (Exit)**

The program prints:

Exiting...

And then terminates.

☑ **6. If user enters any other number**

Example: 8
The program prints:

Invalid choice! Try again.

And returns to main menu.