

BLOOD BANK MANAGEMENT SYSTEM

C Programming Major Project

CSEG1032

University of Petroleum and Energy Studies

School of Computer Science

Student: **Vedansh Pandey**

SAP ID: **590025371**

Instructor: **Dr. Tanu Singh**

Submission Date: **November 30, 2025**

GitHub Repository:

<https://github.com/vedanshpandey4800-cmyk/BLOOD-BANK-MANAGEMENT-SYSTEM>

ABSTRACT

This project implements a console-based Blood Bank Management System in C programming language, demonstrating mastery of all 6 course units (variables, arrays, structures, functions, strings, I/O).

KEY FEATURES:

- Add Donor: Complete profile (ID, Name, Blood Group, Phone, Age, Address)**
- View Donors: Professional formatted table display**
- Request Blood: Instant emergency blood group matching**
- Menu System: User-friendly 4-option interface**

TECHNICAL SPECIFICATIONS:

- Data Structure: struct Donor array (MAX_DONORS = 100)**
- String Matching: strcmp() for blood group search**
- Compilation: gcc -o bloodbank main.c**

3. PROBLEM DEFINITION

PROBLEM STATEMENT:

Manual blood bank operations suffer from:

ISSUE	IMPACT
? No digital database	Complete data loss
? Manual donor search	5-10 minutes delay
? No emergency system	Life-threatening

PROJECT OBJECTIVES:

- 1. Fast donor registration (<30 seconds)**
- 2. Instant blood group matching (<1 second)**
- 3. Professional table-formatted display**
- 4. Robust error handling and input validation**

4. SYSTEM DESIGN

4.1 ALGORITHM

ALGORITHM: BloodBankManagementSystem()

BEGIN

1. Initialize:

donors[MAX_DONORS] ← empty array
donorCount ← 0

2. WHILE (choice ≠ 4) DO

2.1 DISPLAY Menu:

- 1. Add Donor**
- 2. View Donors**
- 3. Request Blood**
- 4. Exit**

2.2 READ choice FROM user

2.3 SWITCH (choice)

- CASE 1: CALL addDonor()**
- CASE 2: CALL displayDonors()**
- CASE 3: CALL requestBlood()**

CASE 4: BREAK
DEFAULT: DISPLAY "Invalid Choice!"
END SWITCH

3. DISPLAY "Thank You!"

4. END

END

ALGORITHM: BloodBankManagementSystem()

BEGIN

1. Initialize:

donors[MAX_DONORS] ← empty array

donorCount ← 0

2. WHILE (choice ≠ 4) DO

2.1 DISPLAY Menu:

1. Add Donor

2. View Donors

3. Request Blood

4. Exit

2.2 READ choice FROM user

2.3 SWITCH (choice)

CASE 1: CALL addDonor()

CASE 2: CALL displayDonors()

CASE 3: CALL requestBlood()

CASE 4: BREAK

DEFAULT: DISPLAY "Invalid Choice!"

END SWITCH

3. DISPLAY "Thank You!"

4. END

END

ALGORITHM: addDonor()

BEGIN

1. IF (donorCount ≥ MAX_DONORS) THEN

DISPLAY "Maximum limit reached!"

RETURN

2. ELSE

READ name INTO donors[donorCount].name

**READ bloodGroup INTO
donors[donorCount].bloodGroup**

```
    READ phone INTO
donors[donorCount].phone

    READ age INTO donors[donorCount].age

    READ address INTO
donors[donorCount].address
```

```
    donors[donorCount].id ← ++donorCount
    DISPLAY "Donor added successfully! ID: ",
donors[donorCount-1].id
```

```
3. END IF
END
```

ALGORITHM: requestBlood()

BEGIN

```
1. READ neededBloodGroup FROM user

2. found ← 0

3. FOR i ← 0 TO donorCount-1 DO
    IF (strcmp(donors[i].bloodGroup,
neededBloodGroup) == 0) THEN
        DISPLAY "Donor #", donors[i].id, ": ",
donors[i].name, " (", donors[i].phone, ")"
    found ← found + 1
```

END IF

4. END FOR

5. IF (found == 0) THEN

**DISPLAY "No ", neededBloodGroup, " donors
available!"**

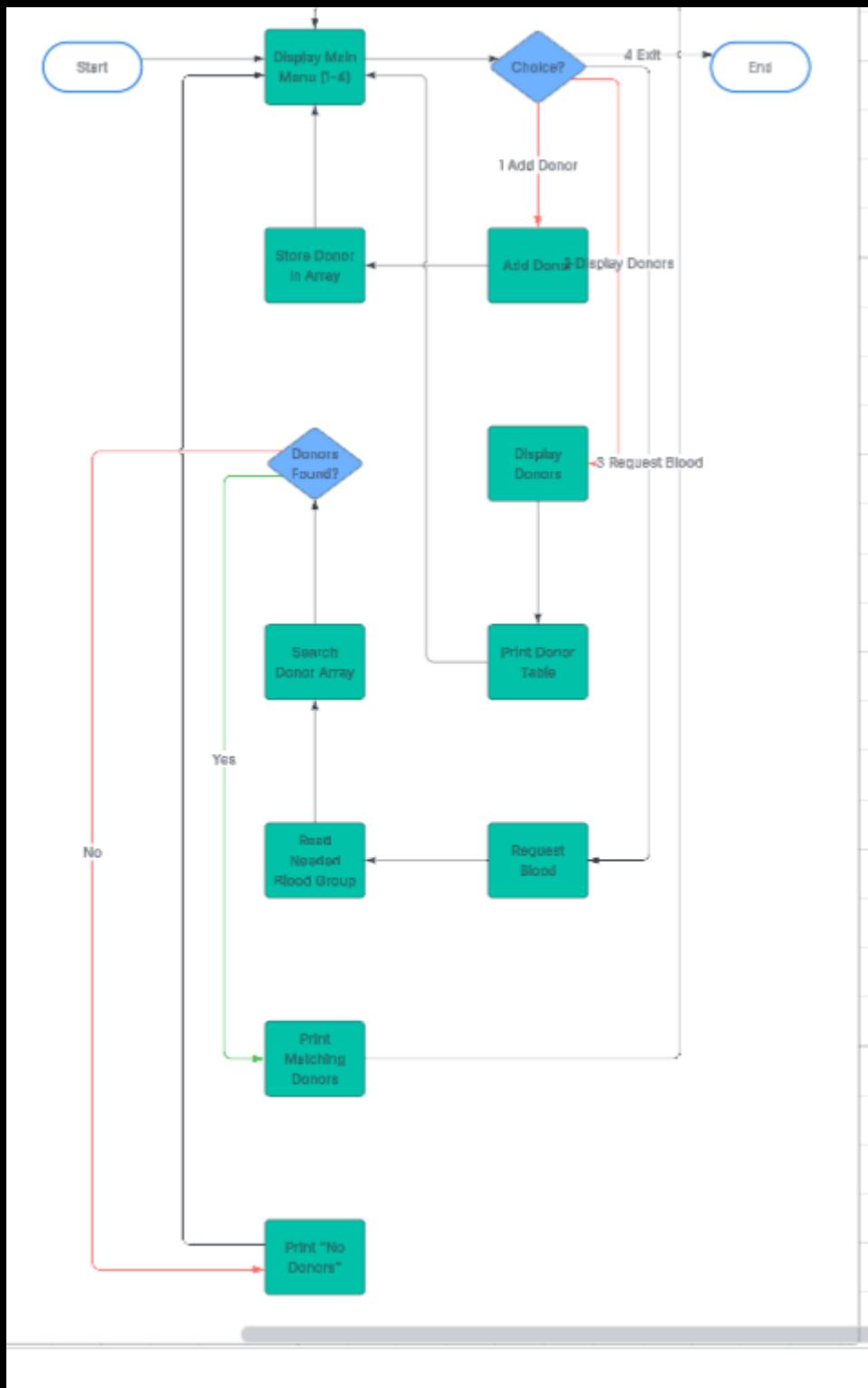
6. ELSE

**DISPLAY found, " ", neededBloodGroup, "
donors found!"**

7. END IF

END

4.2.FLOWCHART



5. IMPLEMENTATION DETAILS

5.1 DATA STRUCTURE

```
#define MAX_DONORS 100
```

```
struct Donor {  
    int id;  
    char name[50];  
    char bloodGroup[5]; // A+, O+, B+, etc.  
    char phone[15];  
    int age;  
    char address[100];  
};
```

```
struct Donor donors[MAX_DONORS];
```

```
int donorCount = 0;
```

5.2 KEY FUNCTIONS

```
void addDonor() {  
    // Input validation + array storage  
    donors[donorCount].id = ++donorCount;  
    printf("✓ Donor added! ID: %d\n", donors[donorCount-1].id);  
}
```

```
void requestBlood() {
```

```
    char needed[5];
```

```

scanf("%s", needed);

// Loop through array, strcmp() matching

// Display matching donors

}

```

6.

TEST CASE TABLE

TEST CASE ID	DESCRIPTION	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	REMA
01	ADD NEW DONOR	1 RAM O+ 323322 25 DELHI	DONOR “RAM” ADDED	DONOR “RAM” ADDED	ID GENE RATED
02	VIEW DONOR LIST	2	“RAM” “O+” ID-01	“RAM” “O+” ID-01	TABLE FORM ATTIN
03	REQUEST DONOR	3 O+	RAM	RAM	
04	EXIT	4	EXIT	EXIT	

6.2.SCREENSHOTS

```
C:\Users\vedansh pandey\Pictures\Screenshots>cd BLOOD-BANK-MANAGEMENT-SYSTEM-main
C:\Users\vedansh pandey\Pictures\Screenshots\BLOOD-BANK-MANAGEMENT-SYSTEM-main>gcc -o bloodbank.exe main.c
C:\Users\vedansh pandey\Pictures\Screenshots\BLOOD-BANK-MANAGEMENT-SYSTEM-main>bloodbank.exe
≡frq BLOOD BANK MANAGEMENT SYSTEM ≡frq
1. ≈Pò Add Donor
2. ≈fōï View Donors
3. ≈fÜç Request Blood
4. ≈fÆj Exit
≈Pñ Choice (1-4): 1

≈Pñ Name: SIDDHARTH
≈Pñ Blood Group: O+
≈Pñ Phone: 2332132
≈Pñ Age: 22
≈Pñ Address: LUCKNOW
≈Eà Donor 'SIDDHARTH' (ID: 1) ADDED!
```

```
≈Pñ Name: ABHIRAJ
≈Pñ Blood Group: AB
≈Pñ Phone: 332433
≈Pñ Age: 21
≈Pñ Address: PUNJAB
≈Eà Donor 'ABHIRAJ' (ID: 2) ADDED!
```

[Press Enter]2

```
≈Eà DONOR 'ABHIRAJ' (ID: 2) ADDED!
```

[Press Enter]2

```
1. ≈Pò Add Donor
2. ≈fōï View Donors
3. ≈fÜç Request Blood
4. ≈fÆj Exit
≈Pñ Choice (1-4): 2
```

≈fōï DONOR LIST (2):

ID	Name	Blood	Phone	Age
1	SIDDHARTH	O+	2332132	22
2	ABHIRAJ	AB	332433	21

[Press Enter]

```
1. ↵Pò Add Donor
2. ↵fôï View Donors
3. ↵fÜç Request Blood
4. ↵Fç Exit
↑Pñ Choice (1-4): 3
```

```
=fÜç EMERGENCY BLOOD REQUEST:
↑Pñ Needed Blood Group: 0-
```

```
=föì 0- DONORS:
↑¥î NO 0- DONORS!
```

7.CONCLUSION & FUTURE WORK

7.1 CONCLUSION

THIS PROJECT SUCCESSFULLY DEMONSTRATES THE USE OF FUNCTIONS AND STRUCTURES IN C. THIS PROJECT ALSO HELP IN SOLVING THE REALWORLD PROBLEM OF BLOOD DONOR REQUIREMENT

7.2 FUTURE ENHANCEMENTS:

- File I/O for data persistence
- Advanced search by name/location
- Blood inventory management
- SMS notification system

8. REFERENCES

1. Kernighan, B.W., Ritchie, D.M. The C Programming Language (2nd Edition)
2. UPES CSEG1032 Course Material
3. GitHub Example Repository:
<https://github.com/aalavandhaann/Major-Project-Example.git>

