1. Simple Name Database Management Program in C

Overview

This C program demonstrates a basic implementation of a name database management system. It allows the user to perform several operations on the database, including inputting new names, deleting existing names, printing all names, and sorting the names in alphabetical order. The program uses a static memory allocation approach, meaning that the size of the database (number of names and the length of each name) is fixed and determined at compile time.

Program Structure

The program consists of the following main components:

1. **Global Constants and Variables:**
    - #define cols 20: Defines the maximum number of characters in each name.
    - #define rows 5: Defines the maximum number of names that can be stored.
    - int cnt=0: A counter to keep track of the number of names currently in the database.
2. **Function Declarations:**
    - void print_menu(): Displays the menu options to the user.
    - void input(char (*)[]): Adds a new name to the database.
    - void delete(char (*)[]): Deletes a name from the database based on the user's input.
    - void print(char (*)[]): Prints all the names currently in the database.
    - void sort(char (*)[]): Sorts the names in alphabetical order.
3. **Main Function:**
    - Initializes the 2D array names to store the names.
    - Uses a while(1) loop to repeatedly show the menu and execute the selected operation.
    - Handles user input and calls the corresponding function based on the user's choice.
4. **Function Definitions:**
    - **print_menu()**: Prints the menu options for the user.
    - **input(char (*p)[cols])**: Prompts the user to input a new name and adds it to the names array if there is space.
    - **delete(char (*p)[cols])**: Prompts the user to input an index and deletes the name at that index if it is valid.
    - **print(char (*p)[cols])**: Prints all the names in the names array.
    - **sort(char (*p)[cols])**: Sorts the names in the names array using the bubble sort algorithm.

Memory Allocation

- **Static Memory Allocation:** The program uses static memory allocation to manage the names database. The size of the array names is fixed with 5 rows and 20 columns (char names[rows][cols]). This means that the program can store up to 5 names, each up to 19 characters long (plus the null terminator).

Detailed Function Descriptions

print_menu()

This function displays the menu options for the user to choose from. It does not take any parameters and does not return any values.

input(char (*p)[cols])

This function allows the user to input a new name into the database.

- **Parameters:** A pointer to the 2D array names.
- **Functionality:** Checks if there is space in the database, prompts the user for a name, and adds it to the names array. It also ensures that the name does not exceed the allocated memory by replacing the newline character with a null terminator.
- **Limitations:** If the database is full, it informs the user and does not add the new name.

delete(char (*p)[cols])

This function deletes a name from the database based on the user's input index.

- **Parameters:** A pointer to the 2D array names.
- **Functionality:** Checks if there are any names to delete, prompts the user for the index of the name to delete, and removes the name by shifting subsequent names up one position.
- **Limitations:** If the index is invalid or the database is empty, it informs the user.

print(char (*p)[cols])

This function prints all the names currently in the database.

- **Parameters:** A pointer to the 2D array names.
- **Functionality:** Iterates through the names array and prints each name along with its index.

sort(char (*p)[cols])

This function sorts the names in the database in alphabetical order using the bubble sort algorithm.

- **Parameters:** A pointer to the 2D array names.
- **Functionality:** Compares adjacent names and swaps them if they are out of order. Repeats this process until the entire array is sorted.

Sample Program Execution

Here is a sample interaction with the program:

```
***Name Data Base ***
i : input
d : delete
p : print
s : sort
q : quit
...Enter choice: i
Enter name:
Alice
Name inserted successfully...!!!

***Name Data Base ***
i : input
d : delete
p : print
s : sort
q : quit
...Enter choice: p
Name 0: Alice

***Name Data Base ***
i : input
d : delete
p : print
s : sort
q : quit
...Enter choice: i
Enter name:
Bob
Name inserted successfully...!!!

***Name Data Base ***
i : input
d : delete
p : print
s : sort
q : quit
...Enter choice: s
Sorted the Data Base...
```

```
***Name Data Base ***
i : input
d : delete
p : print
s : sort
q : quit
...Enter choice: p
Name 0: Alice
Name 1: Bob

***Name Data Base ***
i : input
d : delete
p : print
s : sort
q : quit
...Enter choice: q
***Thanks for using name database***
```

Conclusion

This program is a simple yet effective way to demonstrate the concepts of static memory allocation, array manipulation, and basic operations on a database. It provides a clear example of how to manage a fixed-size collection of data in C.

1.Simple Name Database Management Program in C :

```
int main()
{
        char names[rows][cols];
        char choice;
        while(1)
        {
                print_menu();
                __fpurge(stdin);
                scanf("%c",&choice);
                switch(choice)
                {
                        case 'i' : input(names);break;
                        case 'd' : delete(names);break;
                        case 'p' : print(names);break;
                        case 's' : sort(names);break;
                        case 'q' : printf("***Thanks for using name database***\n");
                                   return 0;
                        default : printf("Invalid choice...!!!\n");
                }
        }
        return 0;
}
```

Complete all the functions…,

2. Name Database Management Program in C Using Dynamic Memory Allocation

Overview

This C program demonstrates a dynamic implementation of a name database management system. It allows the user to perform several operations on the database, including inputting new names, deleting existing names, printing all names, and sorting the names in alphabetical order. The program uses dynamic memory allocation, meaning the size of the database can grow or shrink at runtime as needed.

Program Structure

The program consists of the following main components:

1. **Global Variables**:
   - int cnt = 0: A counter to keep track of the number of names currently in the database.
2. **Function Declarations**:
   - void print_menu(): Displays the menu options to the user.
   - void* input(char (*)[]): Adds a new name to the database.
   - void* delete(char (*)[]): Deletes a name from the database based on the user's input.
   - void sort(char (*)[]): Sorts the names in alphabetical order.
   - void print(char (*)[]): Prints all the names currently in the database.
3. **Main Function**:
   - Initializes the names pointer to NULL to store the names.
   - Uses a while (1) loop to repeatedly show the menu and execute the selected operation.
   - Handles user input and calls the corresponding function based on the user's choice.
4. **Function Definitions**:
   - print_menu(): Prints the menu options for the user.
   - input(char (*)[]): Prompts the user to input a new name and dynamically allocates memory to store it.
   - delete(char (*)[]): Prompts the user to input an index and deletes the name at that index if it is valid.
   - sort(char (*)[]): Sorts the names in the names array using the bubble sort algorithm.
   - print(char (*)[]): Prints all the names in the names array.

3. Name Database Management Program in C Using Dynamic Memory Allocation and Double Pointer

Overview

This C program demonstrates a dynamic implementation of a name database management system. It allows the user to perform several operations on the database, including inputting new names, deleting existing names, printing all names, sorting the names in alphabetical order, and finding specific names. The program uses dynamic memory allocation, meaning the size of the database can grow or shrink at runtime as needed.

Program Structure

The program consists of the following main components:

1. **Global Variables**:
   - int cnt = 0: A counter to keep track of the number of names currently in the database.
2. **Function Declarations**:
   - void print_menu(): Displays the menu options to the user.
   - void* input(char **): Adds a new name to the database.
   - void print(char **): Prints all the names currently in the database.
   - void* delete(char **): Deletes a name from the database based on the user's input.
   - void sort(char **): Sorts the names in alphabetical order.
   - void find(char **): Finds a specific name in the database.
   - void* getString(): Reads a string from the user.
3. **Main Function**:
   - Initializes the names pointer to NULL to store the names.
   - Uses a while (1) loop to repeatedly show the menu and execute the selected operation.
   - Handles user input and calls the corresponding function based on the user's choice.
4. **Function Definitions**:
   - print_menu(): Prints the menu options for the user.
   - input(char **): Prompts the user to input a new name and dynamically allocates memory to store it.
   - print(char **): Prints all the names in the names array.
   - delete(char **): Prompts the user to input an index and deletes the name at that index if it is valid.
   - sort(char **): Sorts the names in the names array using the bubble sort algorithm.
   - find(char **): Searches for a specific name in the database and prints its index if found.
   - getString(): Reads a string from the user input.