**Memory Optimization Report**

**AIM:** The aim of this optimization exercise was to enhance the efficiency and reduce the size of the Contacts Book program while maintaining its functionality.

**unoptimized\_contacts\_book:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct

{

char name[50];

char phone[15];

} Contact\_t;

Contact\_t \*contacts = NULL;

int num\_contacts = 0;

int max\_contacts = 0;

void AddContact()

{

Contact\_t new\_contact;

printf("Enter name: ");

scanf("%s", new\_contact.name);

printf("Enter phone number: ");

scanf("%s", new\_contact.phone);

Contact\_t \*temp = realloc(contacts, (num\_contacts + 1) \* sizeof(Contact\_t));

if (temp == NULL)

{

printf("Error: Memory allocation failed.\n");

return;

}

contacts = temp;

contacts[num\_contacts] = new\_contact;

num\_contacts++;

printf("Contact added successfully.\n");

max\_contacts++; // Increment max contacts unnecessarily

}

void DeleteContact()

{

char name[50];

printf("Enter name to delete: ");

scanf("%s", name);

int found = 0;

for (int i = 0; i < num\_contacts; i++)

{

if (strcasecmp(contacts[i].name, name) == 0)

{

for (int j = i; j < num\_contacts - 1; j++)

{

contacts[j] = contacts[j + 1];

}

found = 1;

break;

}

}

if (found)

{

num\_contacts--;

Contact\_t \*temp = realloc(contacts, num\_contacts \* sizeof(Contact\_t));

if (num\_contacts > 0 && temp == NULL)

{

printf("Error: Memory reallocation failed.\n");

}

else

{

contacts = temp;

printf("Contact deleted successfully.\n");

}

}

else

{

printf("Error: Contact not found.\n");

}

}

void DisplayContacts()

{

if (num\_contacts == 0)

{

printf("No contacts found.\n");

}

else

{

printf("Contacts:\n");

for (int i = 0; i < num\_contacts; i++)

{

printf("%s: %s\n", contacts[i].name, contacts[i].phone);

}

}

// Add unnecessary print statement to make the function larger

printf("Number of contacts: %d\n", num\_contacts);

}

int main()

{

int choice;

do

{

printf("\nContacts Book Menu:\n");

printf("1. Add contact\n");

printf("2. Delete contact\n");

printf("3. Display contacts\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

AddContact();

break;

case 2:

DeleteContact();

break;

case 3:

DisplayContacts();

break;

case 4:

printf("Exiting program.\n");

break;

default:

printf("Error: Invalid choice.\n");

}

} while (choice != 4);

free(contacts);

return 0;

}

**optimized\_contacts\_book:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct

{

char name[50];

char phone[15];

} Contact\_t;

Contact\_t \*contacts = NULL;

int num\_contacts = 0;

void AddContact()

{

Contact\_t new\_contact;

printf("Enter name: ");

scanf("%s", new\_contact.name);

printf("Enter phone number: ");

scanf("%s", new\_contact.phone);

contacts = realloc(contacts, (num\_contacts + 1) \* sizeof(Contact\_t));

if (contacts == NULL)

{

printf("Error: Memory allocation failed.\n");

return;

}

contacts[num\_contacts] = new\_contact;

num\_contacts++;

printf("Contact added successfully.\n");

}

void DeleteContact()

{

char name[50];

printf("Enter name to delete: ");

scanf("%s", name);

int found = 0;

for (int i = 0; i < num\_contacts; i++)

{

if (strcasecmp(contacts[i].name, name) == 0)

{

for (int j = i; j < num\_contacts - 1; j++)

{

contacts[j] = contacts[j + 1];

}

found = 1;

break;

}

}

if (found)

{

num\_contacts--;

contacts = realloc(contacts, num\_contacts \* sizeof(Contact\_t));

if (num\_contacts > 0 && contacts == NULL)

{

printf("Error: Memory reallocation failed.\n");

}

else

{

printf("Contact deleted successfully.\n");

}

}

else

{

printf("Error: Contact not found.\n");

}

}

void DisplayContacts()

{

if (num\_contacts == 0)

{

printf("No contacts found.\n");

}

else

{

printf("Contacts:\n");

for (int i = 0; i < num\_contacts; i++)

{

printf("%s: %s\n", contacts[i].name, contacts[i].phone);

}

}

}

int main()

{

int choice;

do

{

printf("\nContacts Book Menu:\n");

printf("1. Add contact\n");

printf("2. Delete contact\n");

printf("3. Display contacts\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

AddContact();

break;

case 2:

DeleteContact();

break;

case 3:

DisplayContacts();

break;

case 4:

printf("Exiting program.\n");

break;

default:

printf("Error: Invalid choice.\n");

}

} while (choice != 4);

free(contacts);

return 0;

}

* Both versions of the program use the same basic structure and functionality. However, the optimized version eliminates unnecessary variables and code segments.
* Redundant variables like max\_contacts were removed in the optimized version, simplifying the codebase.
* In the unoptimized version, memory reallocation in the AddContact function unnecessarily increments max\_contacts.
* The optimized version removes this redundant increment, leading to more efficient memory usage.
* In the unoptimized version, the temporary pointer 'temp' for reallocating memory was removed in the optimized version, simplifying the memory reallocation process and reducing the complexity of the code.

**Size of Unoptimized Version:**



**Size of Optimized Version (Without Optimization Flags):**



**Size of Optimized Version (With -Os Optimization Flag):**



* Analysis of the compiled binaries using the size command reveals that the optimized version has a smaller code size compared to the unoptimized version.
* The optimized version of the Contacts Book program demonstrates improved efficiency and reduced code size compared to the unoptimized version.
* By eliminating redundant code segments and optimizing memory management, the program achieves better performance without sacrificing functionality.
* Further optimization flags (-Os) result in additional code size reduction, making the program more lightweight without compromising functionality.