Implement a simple database using memory mapped files

Igor Półchłopek

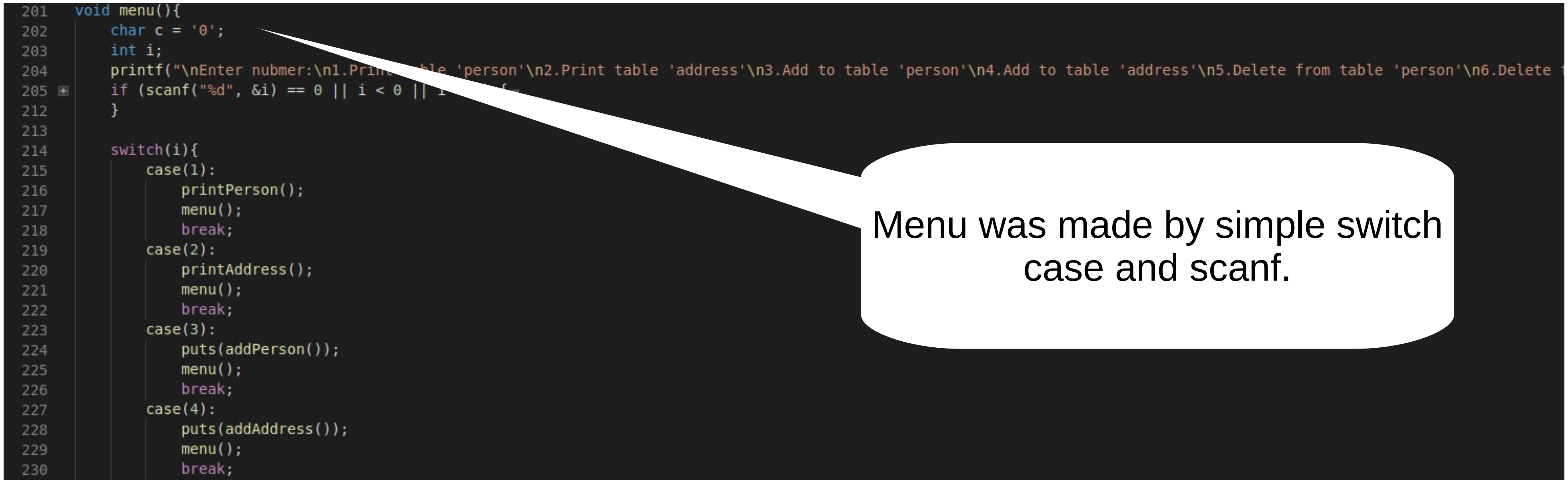
The project description

- A program that make a simple database using memory mapped files.
- Program can print, add, delete and replace record from tables(files) in database.
- Everything is controlled by console interface.

The challenge

- Support memory mapped files as I/O for my simple database is little tricky at beginning.
- Deleting records wasn`t that obvious and I wasted time on trying make delete process too complicated.

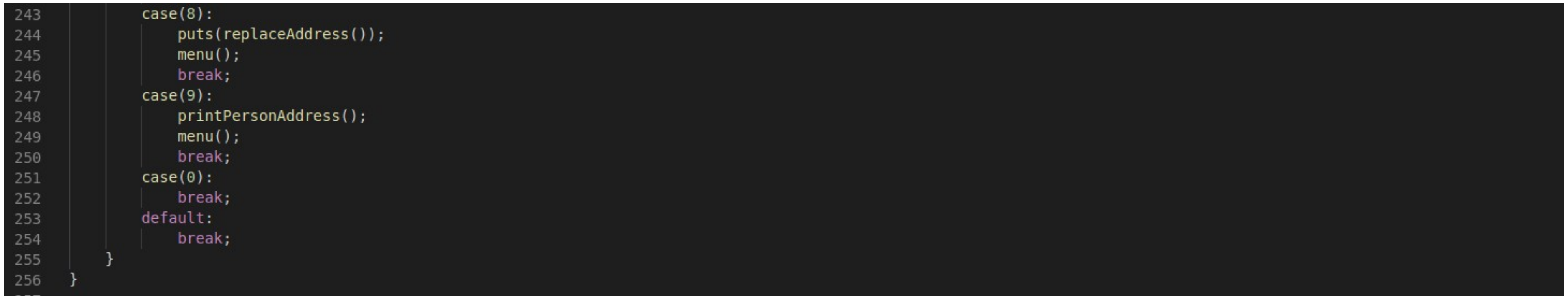
The solution



```c
void menu(){
    char c = '0';
    int i;
    printf("\nEnter nubmer:\n1.Print table 'person'\n2.Print table 'address'\n3.Add to table 'person'\n4.Add to table 'address'\n5.Delete from table 'person'\n6.Delete f
    if (scanf("%d", &i) == 0 || i < 0 || i [illegible] {…
    }

    switch(i){
        case(1):
            printPerson();
            menu();
            break;
        case(2):
            printAddress();
            menu();
            break;
        case(3):
            puts(addPerson());
            menu();
            break;
        case(4):
            puts(addAddress());
            menu();
            break;
```

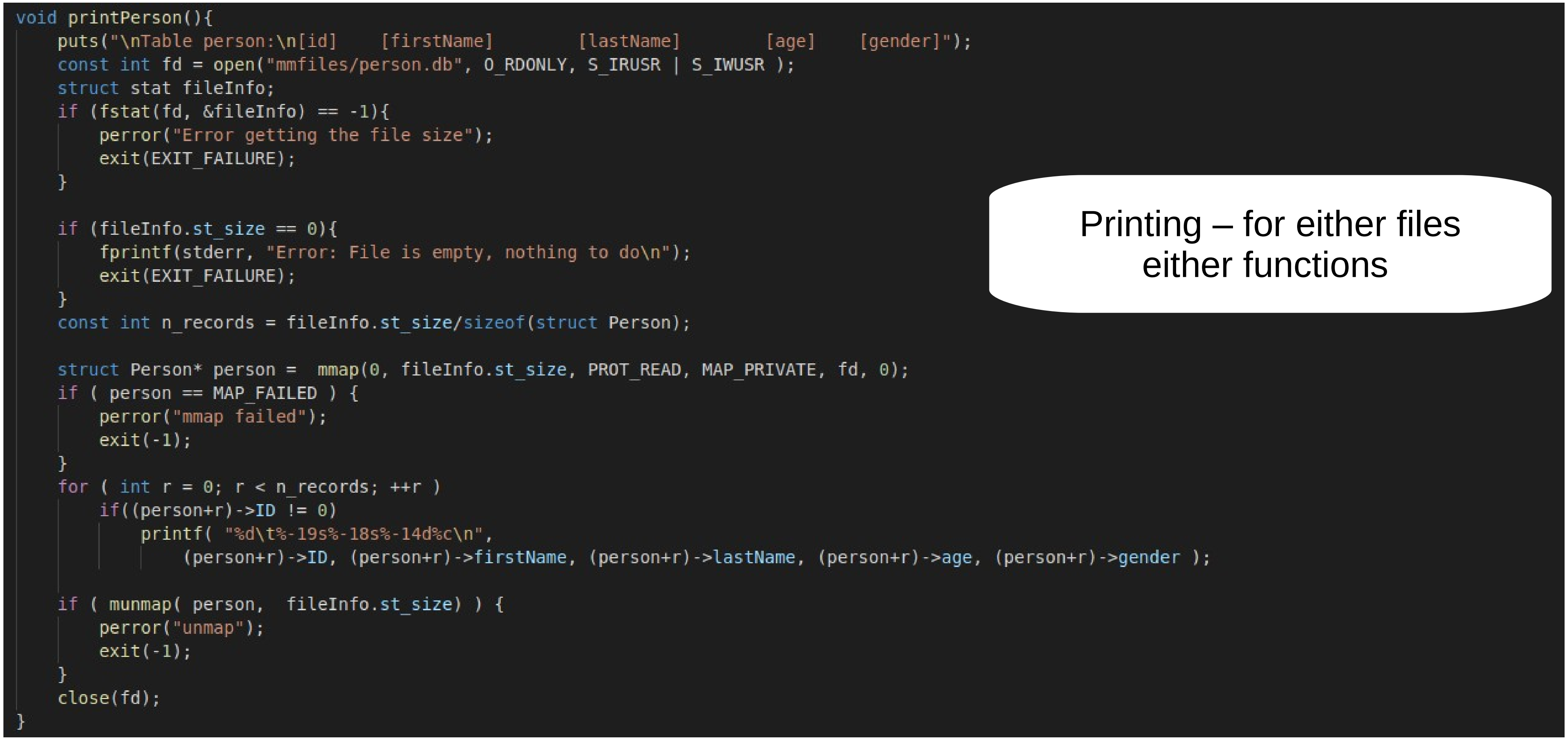
Menu was made by simple switch case and scanf.

⋮



```c
        case(8):
            puts(replaceAddress());
            menu();
            break;
        case(9):
            printPersonAddress();
            menu();
            break;
        case(0):
            break;
        default:
            break;
    }
}
```

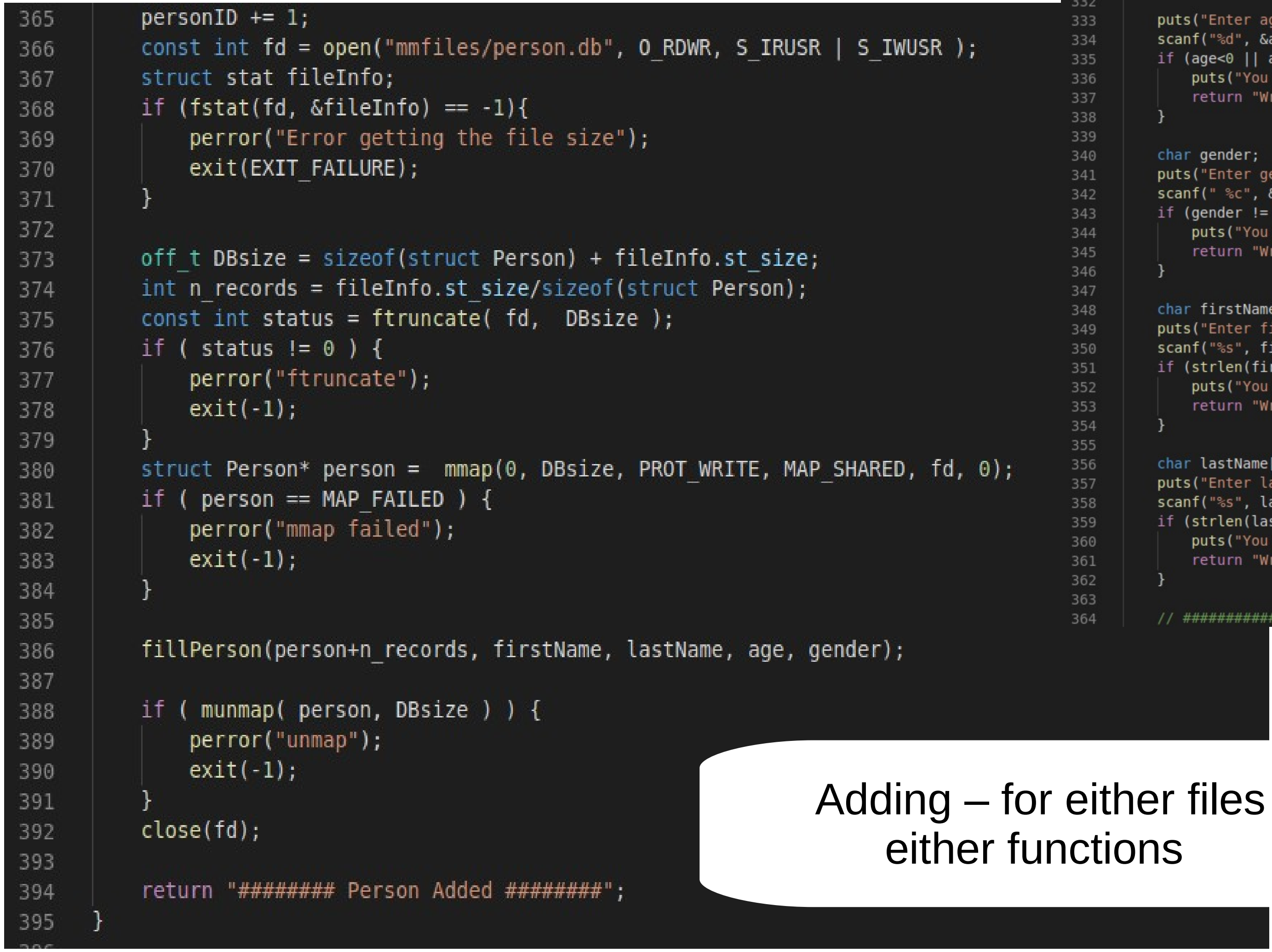
The solution



```c
void printPerson(){
    puts("\nTable person:\n[id]    [firstName]        [lastName]        [age]    [gender]");
    const int fd = open("mmfiles/person.db", O_RDONLY, S_IRUSR | S_IWUSR );
    struct stat fileInfo;
    if (fstat(fd, &fileInfo) == -1){
        perror("Error getting the file size");
        exit(EXIT_FAILURE);
    }

    if (fileInfo.st_size == 0){
        fprintf(stderr, "Error: File is empty, nothing to do\n");
        exit(EXIT_FAILURE);
    }
    const int n_records = fileInfo.st_size/sizeof(struct Person);

    struct Person* person =  mmap(0, fileInfo.st_size, PROT_READ, MAP_PRIVATE, fd, 0);
    if ( person == MAP_FAILED ) {
        perror("mmap failed");
        exit(-1);
    }
    for ( int r = 0; r < n_records; ++r )
        if((person+r)->ID != 0)
            printf( "%d\t%-19s%-18s%-14d%c\n",
                (person+r)->ID, (person+r)->firstName, (person+r)->lastName, (person+r)->age, (person+r)->gender );

    if ( munmap( person,  fileInfo.st_size) ) {
        perror("unmap");
        exit(-1);
    }
    close(fd);
}
```

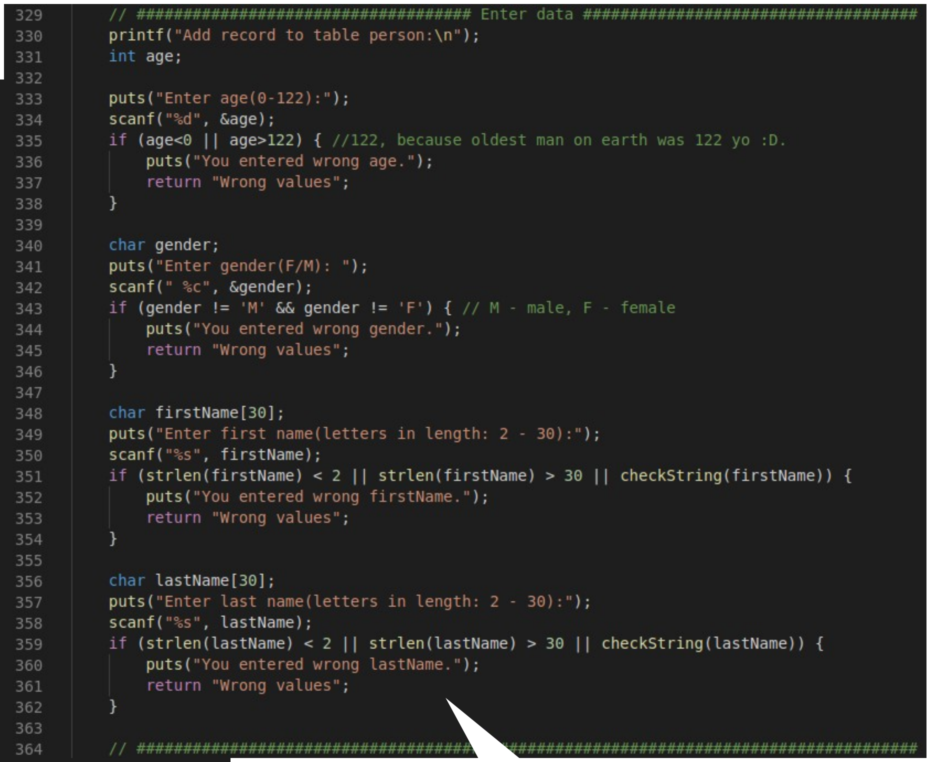
Printing – for either files
either functions

The solution



```c
    personID += 1;
    const int fd = open("mmfiles/person.db", O_RDWR, S_IRUSR | S_IWUSR );
    struct stat fileInfo;
    if (fstat(fd, &fileInfo) == -1){
        perror("Error getting the file size");
        exit(EXIT_FAILURE);
    }

    off_t DBsize = sizeof(struct Person) + fileInfo.st_size;
    int n_records = fileInfo.st_size/sizeof(struct Person);
    const int status = ftruncate( fd,  DBsize );
    if ( status != 0 ) {
        perror("ftruncate");
        exit(-1);
    }
    struct Person* person =  mmap(0, DBsize, PROT_WRITE, MAP_SHARED, fd, 0);
    if ( person == MAP_FAILED ) {
        perror("mmap failed");
        exit(-1);
    }

    fillPerson(person+n_records, firstName, lastName, age, gender);

    if ( munmap( person, DBsize ) ) {
        perror("unmap");
        exit(-1);
    }
    close(fd);

    return "######## Person Added ########";
}
```

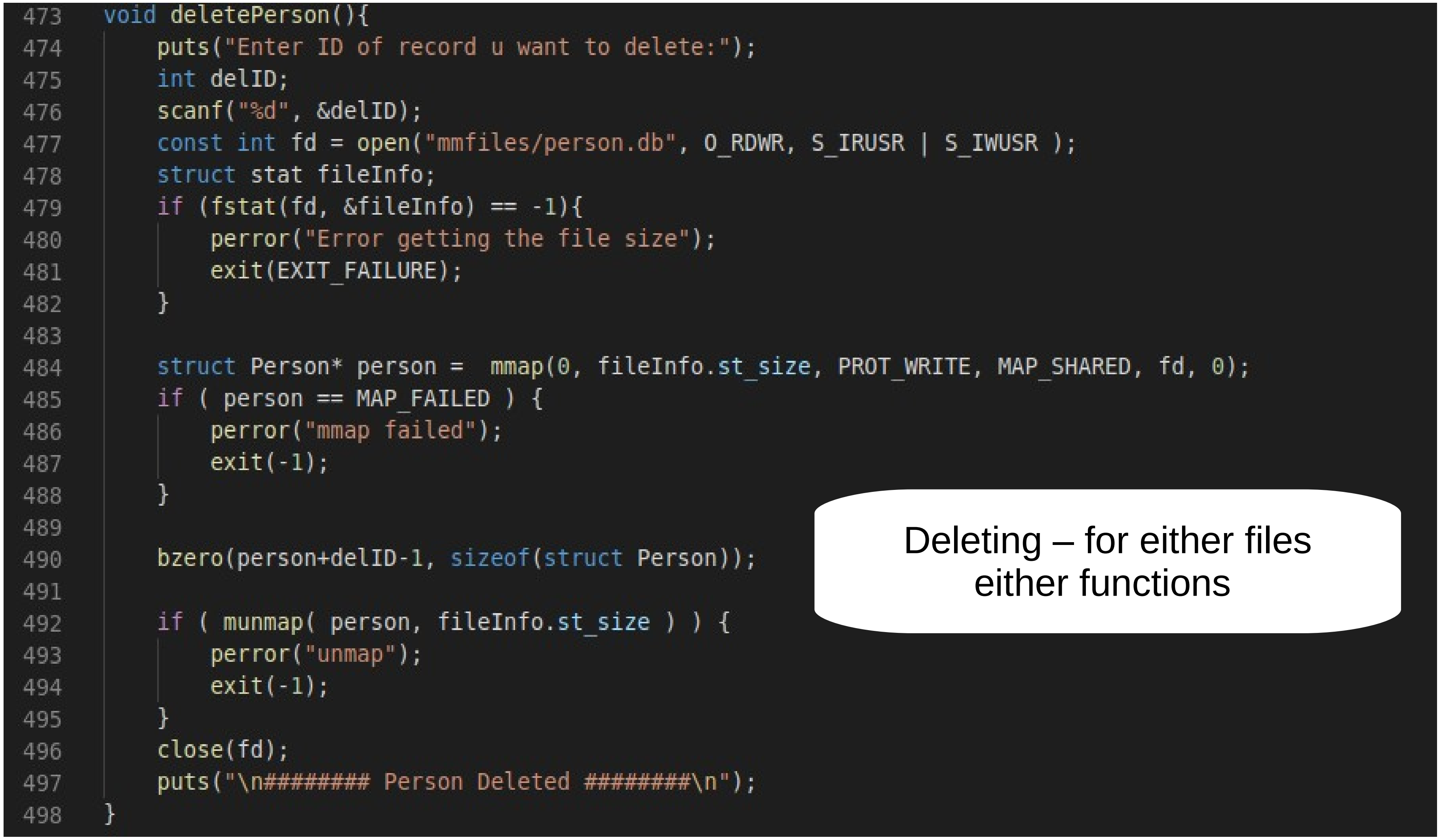
Adding – for either files either functions



```c
    // #################################### Enter data ####################################
    printf("Add record to table person:\n");
    int age;

    puts("Enter age(0-122):");
    scanf("%d", &age);
    if (age<0 || age>122) { //122, because oldest man on earth was 122 yo :D.
        puts("You entered wrong age.");
        return "Wrong values";
    }

    char gender;
    puts("Enter gender(F/M): ");
    scanf(" %c", &gender);
    if (gender != 'M' && gender != 'F') { // M - male, F - female
        puts("You entered wrong gender.");
        return "Wrong values";
    }

    char firstName[30];
    puts("Enter first name(letters in length: 2 - 30):");
    scanf("%s", firstName);
    if (strlen(firstName) < 2 || strlen(firstName) > 30 || checkString(firstName)) {
        puts("You entered wrong firstName.");
        return "Wrong values";
    }

    char lastName[30];
    puts("Enter last name(letters in length: 2 - 30):");
    scanf("%s", lastName);
    if (strlen(lastName) < 2 || strlen(lastName) > 30 || checkString(lastName)) {
        puts("You entered wrong lastName.");
        return "Wrong values";
    }

    // ##########################################################################
```

Info from user console input

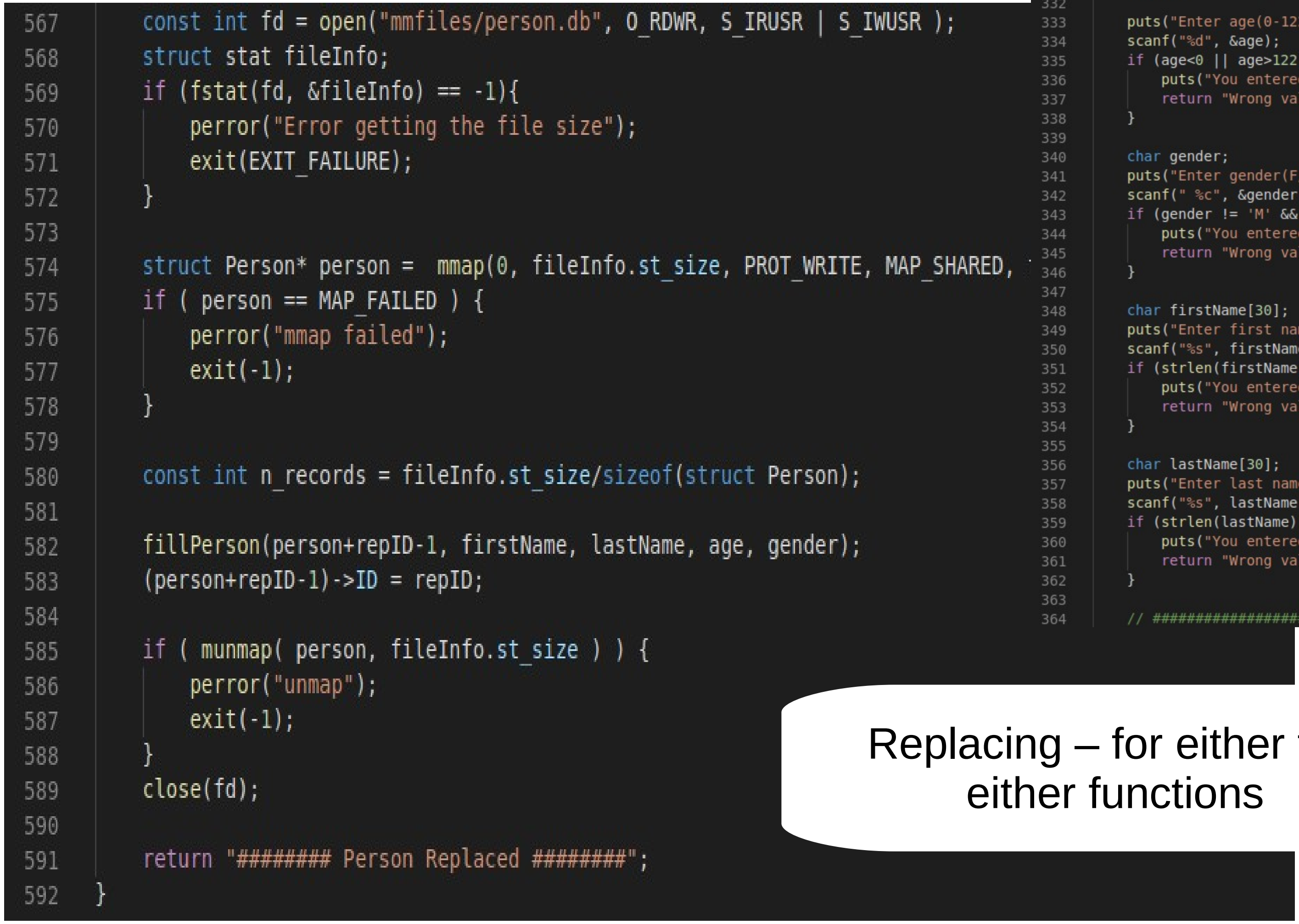
The solution



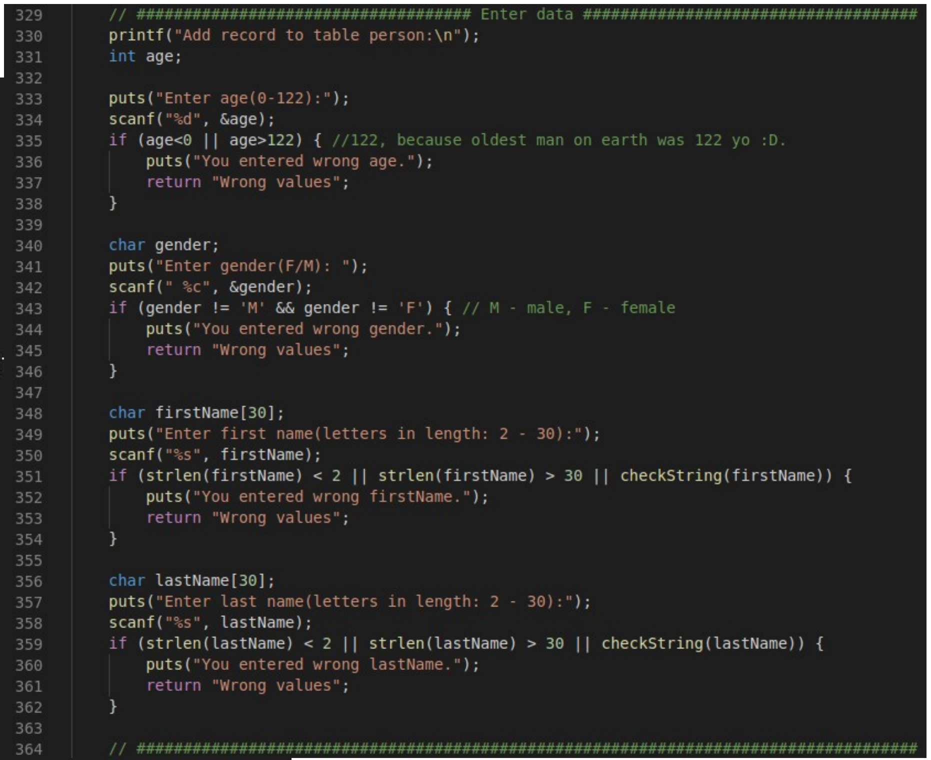
```c
void deletePerson(){
    puts("Enter ID of record u want to delete:");
    int delID;
    scanf("%d", &delID);
    const int fd = open("mmfiles/person.db", O_RDWR, S_IRUSR | S_IWUSR );
    struct stat fileInfo;
    if (fstat(fd, &fileInfo) == -1){
        perror("Error getting the file size");
        exit(EXIT_FAILURE);
    }

    struct Person* person =  mmap(0, fileInfo.st_size, PROT_WRITE, MAP_SHARED, fd, 0);
    if ( person == MAP_FAILED ) {
        perror("mmap failed");
        exit(-1);
    }

    bzero(person+delID-1, sizeof(struct Person));

    if ( munmap( person, fileInfo.st_size ) ) {
        perror("unmap");
        exit(-1);
    }
    close(fd);
    puts("\n######## Person Deleted ########\n");
}
```

Deleting – for either files either functions

The solution



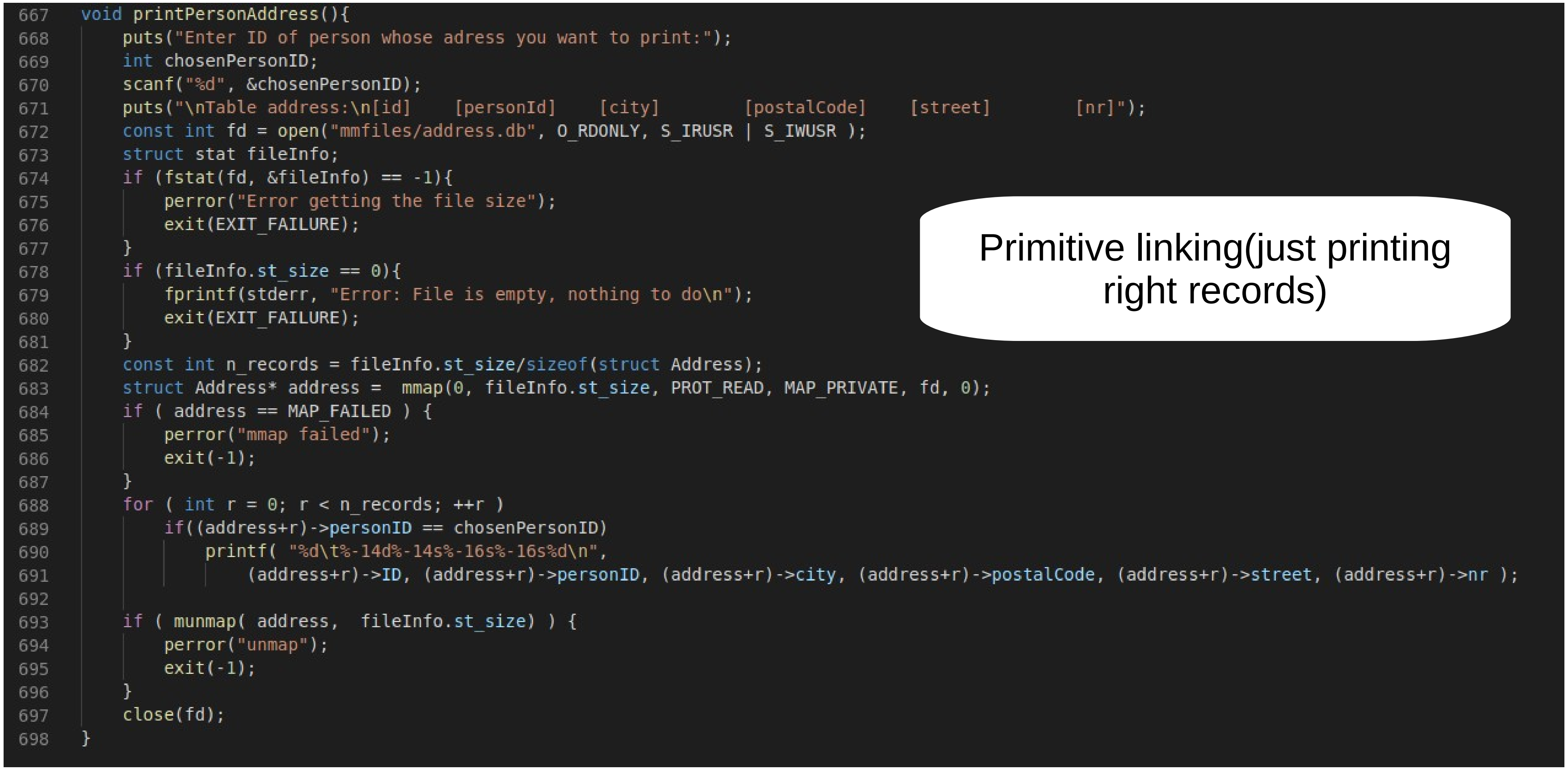
```c
    const int fd = open("mmfiles/person.db", O_RDWR, S_IRUSR | S_IWUSR );
    struct stat fileInfo;
    if (fstat(fd, &fileInfo) == -1){
        perror("Error getting the file size");
        exit(EXIT_FAILURE);
    }

    struct Person* person =  mmap(0, fileInfo.st_size, PROT_WRITE, MAP_SHARED,
    if ( person == MAP_FAILED ) {
        perror("mmap failed");
        exit(-1);
    }

    const int n_records = fileInfo.st_size/sizeof(struct Person);

    fillPerson(person+repID-1, firstName, lastName, age, gender);
    (person+repID-1)->ID = repID;

    if ( munmap( person, fileInfo.st_size ) ) {
        perror("unmap");
        exit(-1);
    }
    close(fd);

    return "######## Person Replaced ########";
}
```



```c
    // #################################### Enter data ####################################
    printf("Add record to table person:\n");
    int age;

    puts("Enter age(0-122):");
    scanf("%d", &age);
    if (age<0 || age>122) { //122, because oldest man on earth was 122 yo :D.
        puts("You entered wrong age.");
        return "Wrong values";
    }

    char gender;
    puts("Enter gender(F/M): ");
    scanf(" %c", &gender);
    if (gender != 'M' && gender != 'F') { // M - male, F - female
        puts("You entered wrong gender.");
        return "Wrong values";
    }

    char firstName[30];
    puts("Enter first name(letters in length: 2 - 30):");
    scanf("%s", firstName);
    if (strlen(firstName) < 2 || strlen(firstName) > 30 || checkString(firstName)) {
        puts("You entered wrong firstName.");
        return "Wrong values";
    }

    char lastName[30];
    puts("Enter last name(letters in length: 2 - 30):");
    scanf("%s", lastName);
    if (strlen(lastName) < 2 || strlen(lastName) > 30 || checkString(lastName)) {
        puts("You entered wrong lastName.");
        return "Wrong values";
    }

    // ####################################################################################
```

Replacing – for either files
either functions

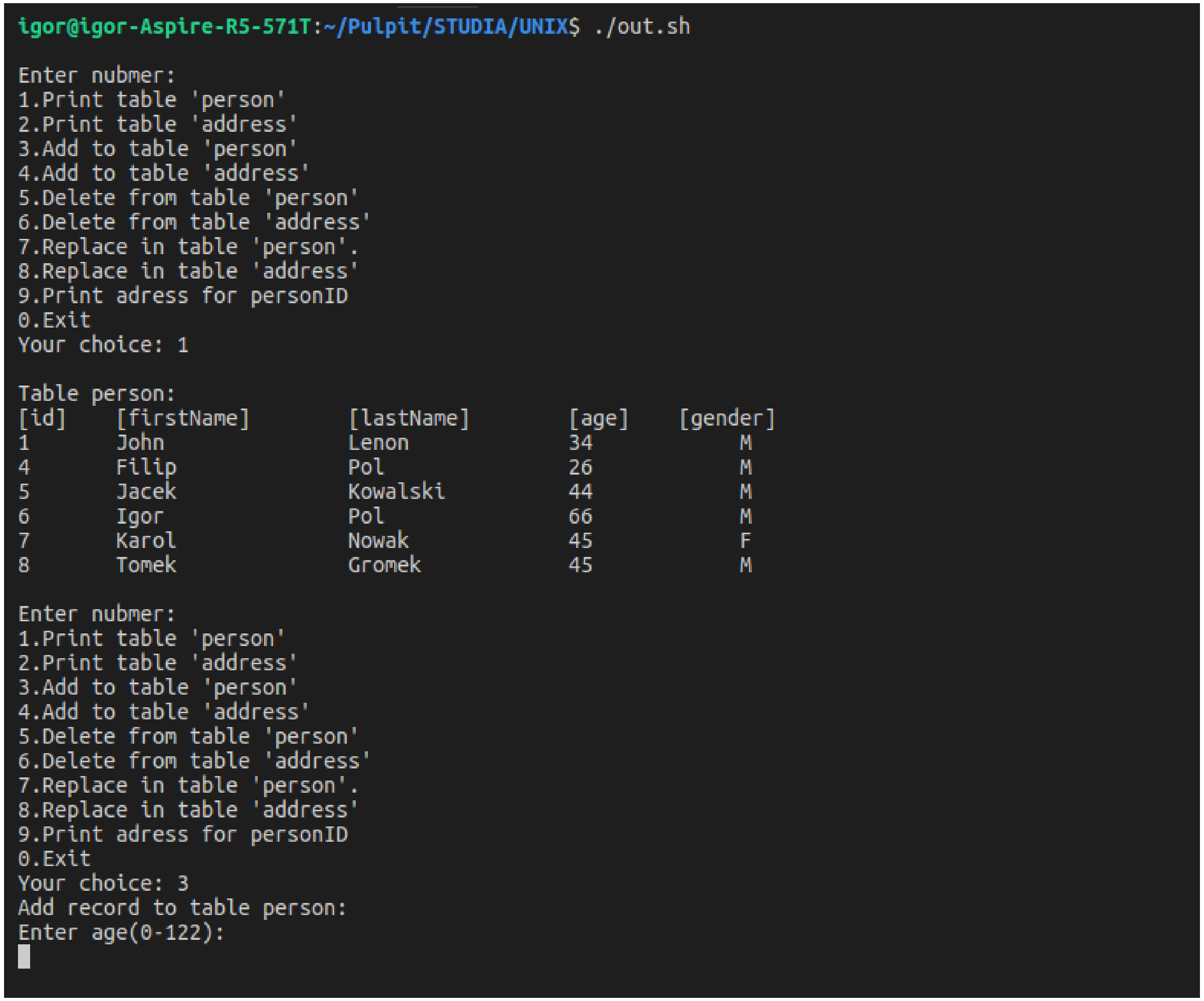
The solution



```c
void printPersonAddress(){
    puts("Enter ID of person whose adress you want to print:");
    int chosenPersonID;
    scanf("%d", &chosenPersonID);
    puts("\nTable address:\n[id]    [personId]    [city]        [postalCode]    [street]        [nr]");
    const int fd = open("mmfiles/address.db", O_RDONLY, S_IRUSR | S_IWUSR );
    struct stat fileInfo;
    if (fstat(fd, &fileInfo) == -1){
        perror("Error getting the file size");
        exit(EXIT_FAILURE);
    }
    if (fileInfo.st_size == 0){
        fprintf(stderr, "Error: File is empty, nothing to do\n");
        exit(EXIT_FAILURE);
    }
    const int n_records = fileInfo.st_size/sizeof(struct Address);
    struct Address* address =  mmap(0, fileInfo.st_size, PROT_READ, MAP_PRIVATE, fd, 0);
    if ( address == MAP_FAILED ) {
        perror("mmap failed");
        exit(-1);
    }
    for ( int r = 0; r < n_records; ++r )
        if((address+r)->personID == chosenPersonID)
            printf( "%d\t%-14d%-14s%-16s%-16s%d\n",
                (address+r)->ID, (address+r)->personID, (address+r)->city, (address+r)->postalCode, (address+r)->street, (address+r)->nr );

    if ( munmap( address,  fileInfo.st_size) ) {
        perror("unmap");
        exit(-1);
    }
    close(fd);
}
```

Primitive linking(just printing right records)

Result



```
igor@igor-Aspire-R5-571T:~/Pulpit/STUDIA/UNIX$ ./out.sh

Enter nubmer:
1.Print table 'person'
2.Print table 'address'
3.Add to table 'person'
4.Add to table 'address'
5.Delete from table 'person'
6.Delete from table 'address'
7.Replace in table 'person'.
8.Replace in table 'address'
9.Print adress for personID
0.Exit
Your choice: 1

Table person:
[id]    [firstName]        [lastName]         [age]    [gender]
1       John               Lenon              34            M
4       Filip              Pol                26            M
5       Jacek              Kowalski           44            M
6       Igor               Pol                66            M
7       Karol              Nowak              45            F
8       Tomek              Gromek             45            M

Enter nubmer:
1.Print table 'person'
2.Print table 'address'
3.Add to table 'person'
4.Add to table 'address'
5.Delete from table 'person'
6.Delete from table 'address'
7.Replace in table 'person'.
8.Replace in table 'address'
9.Print adress for personID
0.Exit
Your choice: 3
Add record to table person:
Enter age(0-122):
```

Hardest issue

- Handling memory mapped files was necessary for this project.
- A lot time I spend on interface and user input verification
- Hardest was implementing „delete”. It wasn`t obvious to do that way as I did.