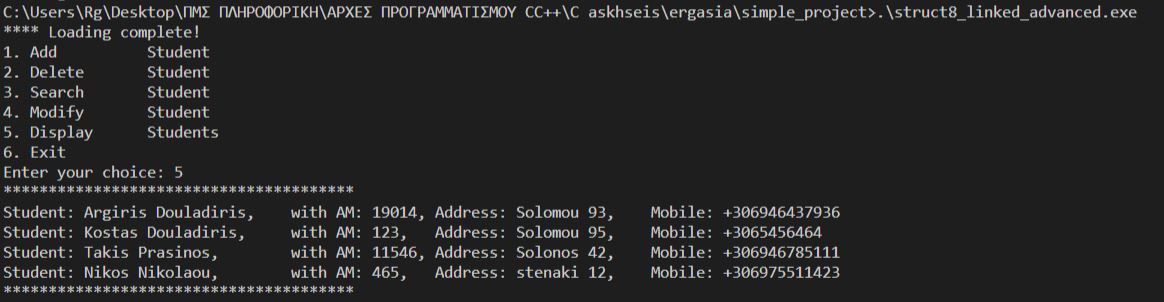
Σημείωση: Κάνοντας zoom στο word οι εικόνες φαίνονται καλύτερα.

Στο αρχείο με τις συναρτήσεις έχω ορίσει την: extern struct student\* head για να έχω πρόσβαση στον global pointer struct student\* head του main script.

1. Κατά την εκκίνηση του προγράμματος εκτελείται η συνάρτηση void load() και εάν υπάρχει κάποιο αρχείο .dat, το διαβάζει, και τυπώνει τόσα αστεράκια (π.χ. \*\*\*) όσες είναι και οι εγγραφές στο αρχείο. Κάθε αστεράκι τυπώνεται μετά το πέρας δημιουργίας κάθε κόμβου. (Τελείως διακοσμητική η προβολή των \*\*\*).

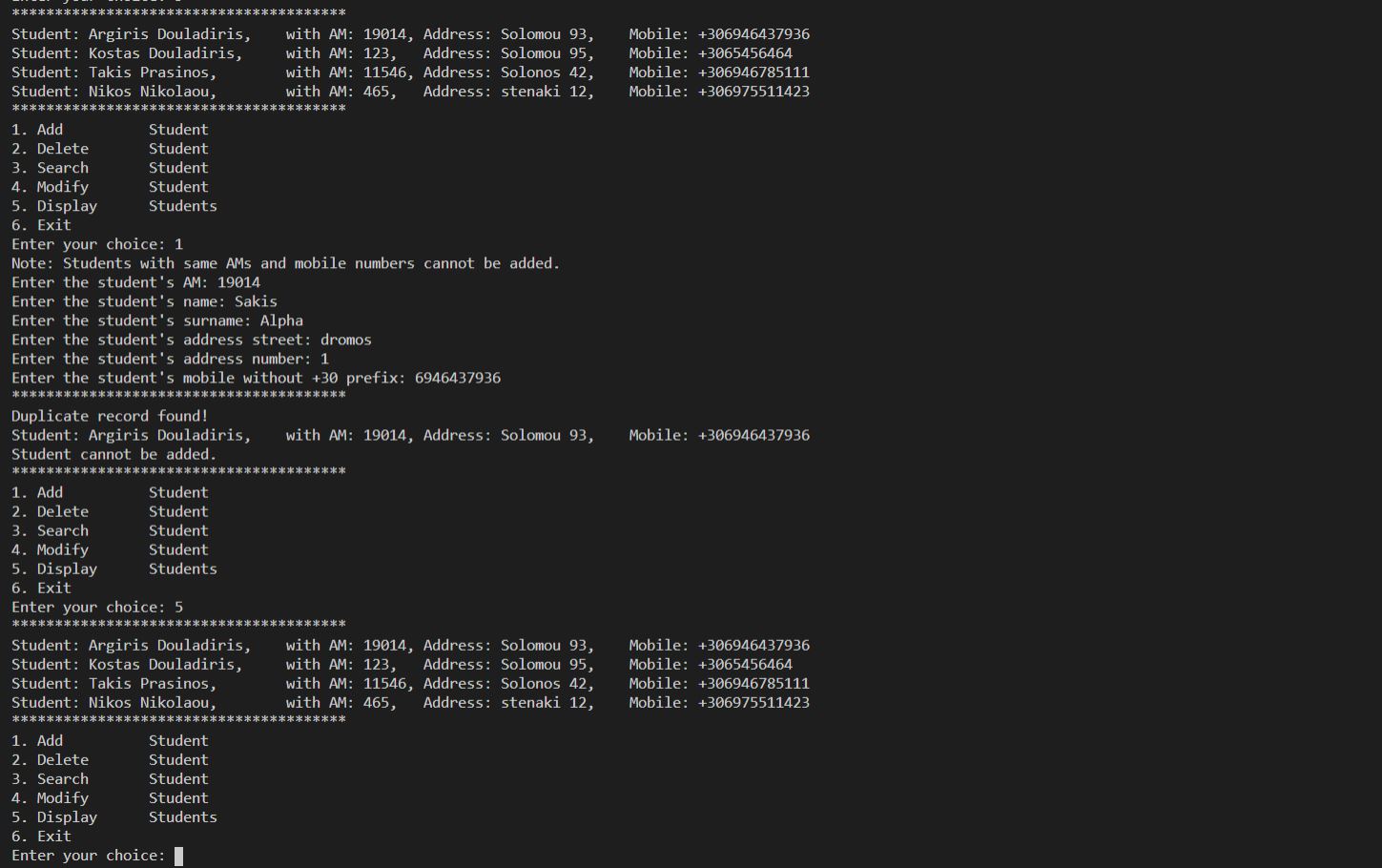
Η συνάρτηση λειτουργεί δημιουργώντας μια καινούρια λίστα κατά τη διάρκεια του τρέχων προγράμματος (during runtime) και κάθε φορά που δημιουργείται ένας νέος κόμβος αντιγράφει τον τύπο δεδομένων struct student που διαβάζει με τη χρήση της fread(). Ο πρώτος κόμβος στο αρχείο .dat είναι ο τελευταίος κόμβος που δημιουργήθηκε. Δηλαδή, αν έχω μια λίστα με κόμβους 3->2->1->NULL τους αποθηκέυω με σειρά 3, 2, 1.

Οπότε έφτιαξα δύο pointers tail και head τύπου struct student. Δείχνω με τον head την πρώτη εγγραφή (η οποία είναι και η μεταγενέστερη) και με tail δείχνω κάθε παλαιότερο κόμβο που μπαίνει στο τέλος καθώς διαβάζω το αρχείο .dat. Έτσι η νεότερη χρονικά εγγραφή βρίσκεται στην κορυφή της λίστας (head).

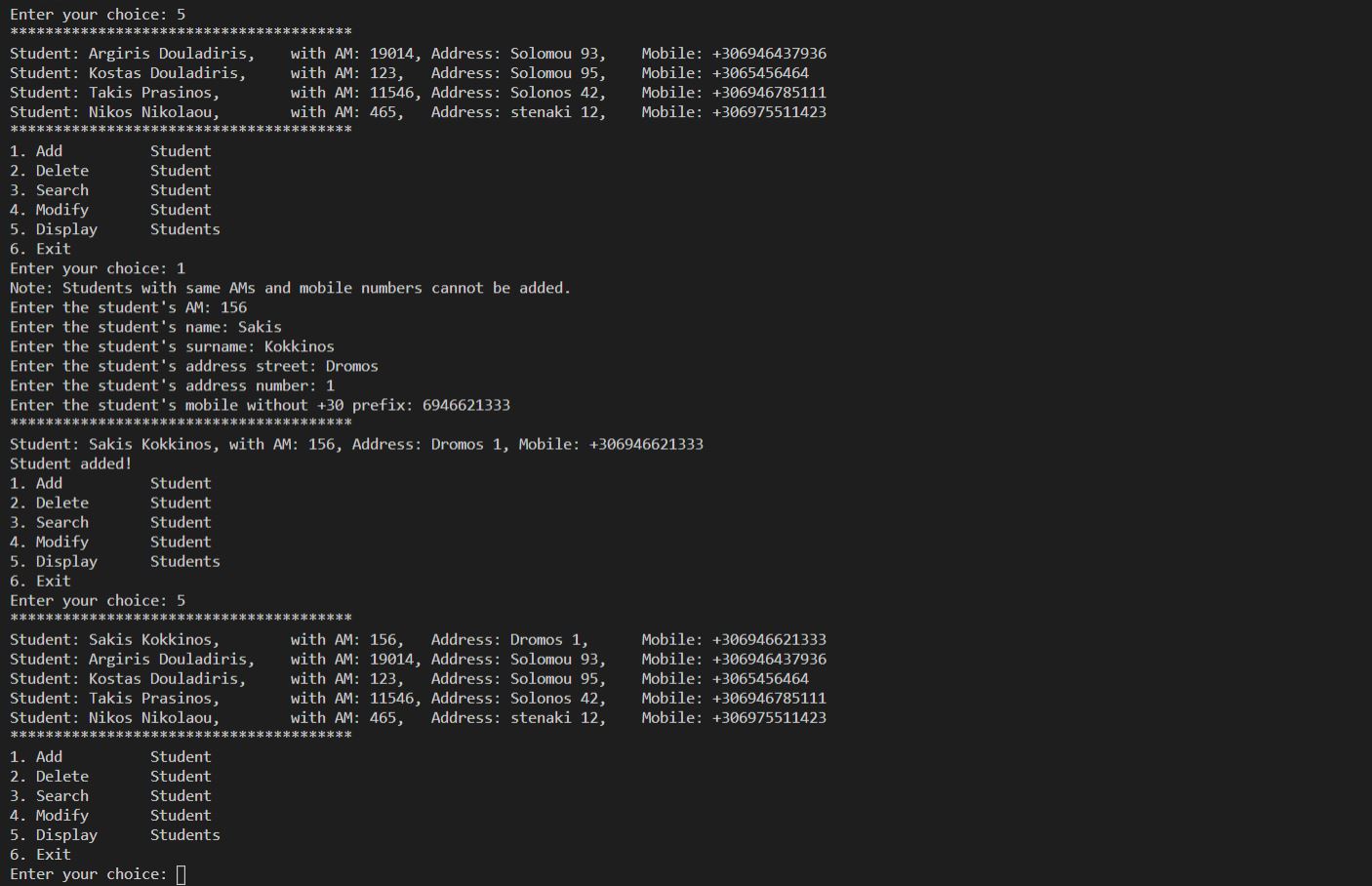


Παραπάνω βλέπουμε και τη συνάρτηση display() η οποία δημιουργεί έναν pointer τύπου struct student που δείχνει στο head και διατρέχω τη λίστα μου κάνοντας το τρέχων node ίσο με node->next μέχρι να βρω τον τερματικό κόμβο του οποίου node->next είναι ίσο με NULL. Σε κάθε προσπέλαση των κόμβων, τυπώνω τα πεδία του κόμβου.

2. Καλώντας τη συνάρτηση add() πρόσθεσα τη συνάρτηση check\_duplicate\_record() που ελέγχει αν εισάγουμε εγγραφή με τα ίδια στοιχεία (στο παρόν έθεσα τα am και mobile να είναι πάντοτε διαφορετικά).



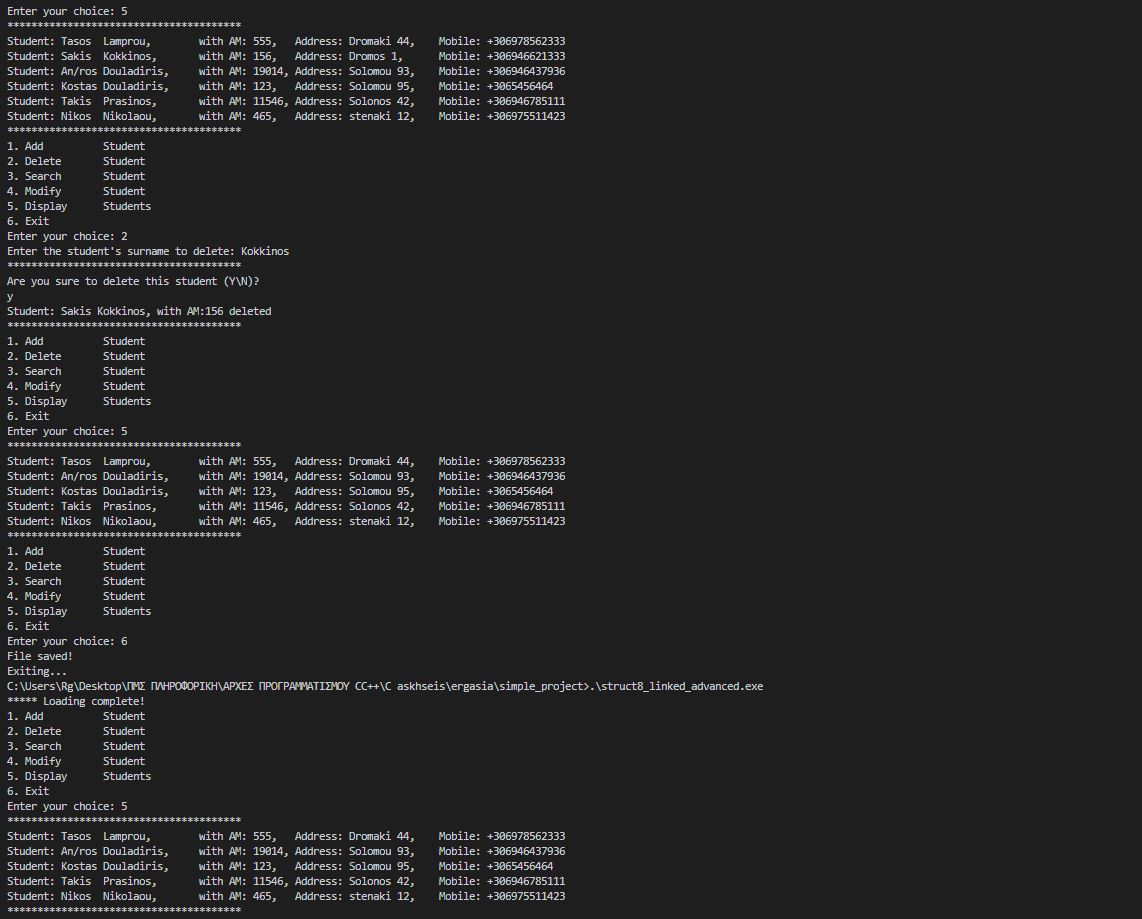
Στη συνέχεια η συνάρτηση add() δημιουργεί έναν κόμβο στην αρχή της λίστας (head), αντιγράφει τα πεδία int am, char stname[20], char stsurname[20], char address[30], char mobile[13] με ανάθεση και χρήση συναρτήσεων strcat και strcpy. Η λίστα είναι μια απλή διασυνδεδεμένη λίστα.



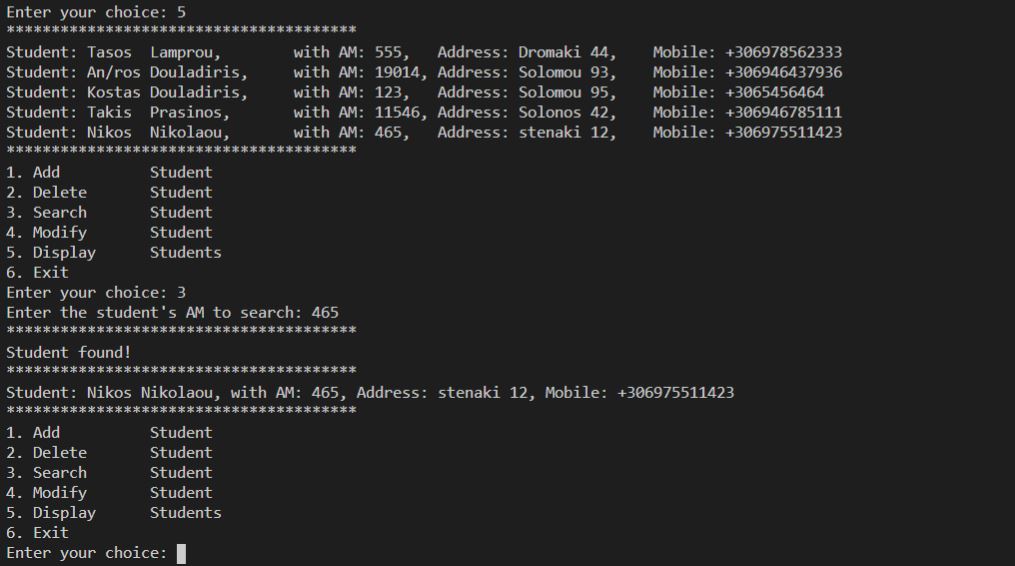
3. Καλώντας τη συνάρτηση delete() κάνω προσπέλαση σε κάθε κόμβο και με τη χρήση της strcmp βρίσκω το πεδίο stsurname του κόμβου που θέλω να διαγράψω. Στη συνέχεια κάνω το node\_previous->next να δείχνει node->next και κάνω free τον κόμβο που διαγράφω. Παρακάτω φαίνεται ότι ο χρήστης:

"Student: Sakis Kokkinos, with AM:156 deleted" διαγράφτηκε.

Και το επιβεβαιώνουμε τρέχοντας την display() και αφού κλείσουμε και ανοίξουμε το πρόγραμμα.



4. Καλώντας τη συνάρτηση search(), κάνω προσπέλαση τους κόμβους έναν-έναν έχοντας δημιουργήσει έναν pointer r τύπου struct student, ο οποίος αρχικά δείχνει στην αρχή της λίστας head και στη συνέχεια παίρνει τις τιμές του node->next μέχρι να βρει την τιμή NULL. Θα σταματήσει όταν βρει το πεδίο που έχω ορίσει να συγκρίνει (ακέραιος ΑΜ) και θα τυπώσει στην κονσόλα τα πεδία του τρέχοντος κόμβου.



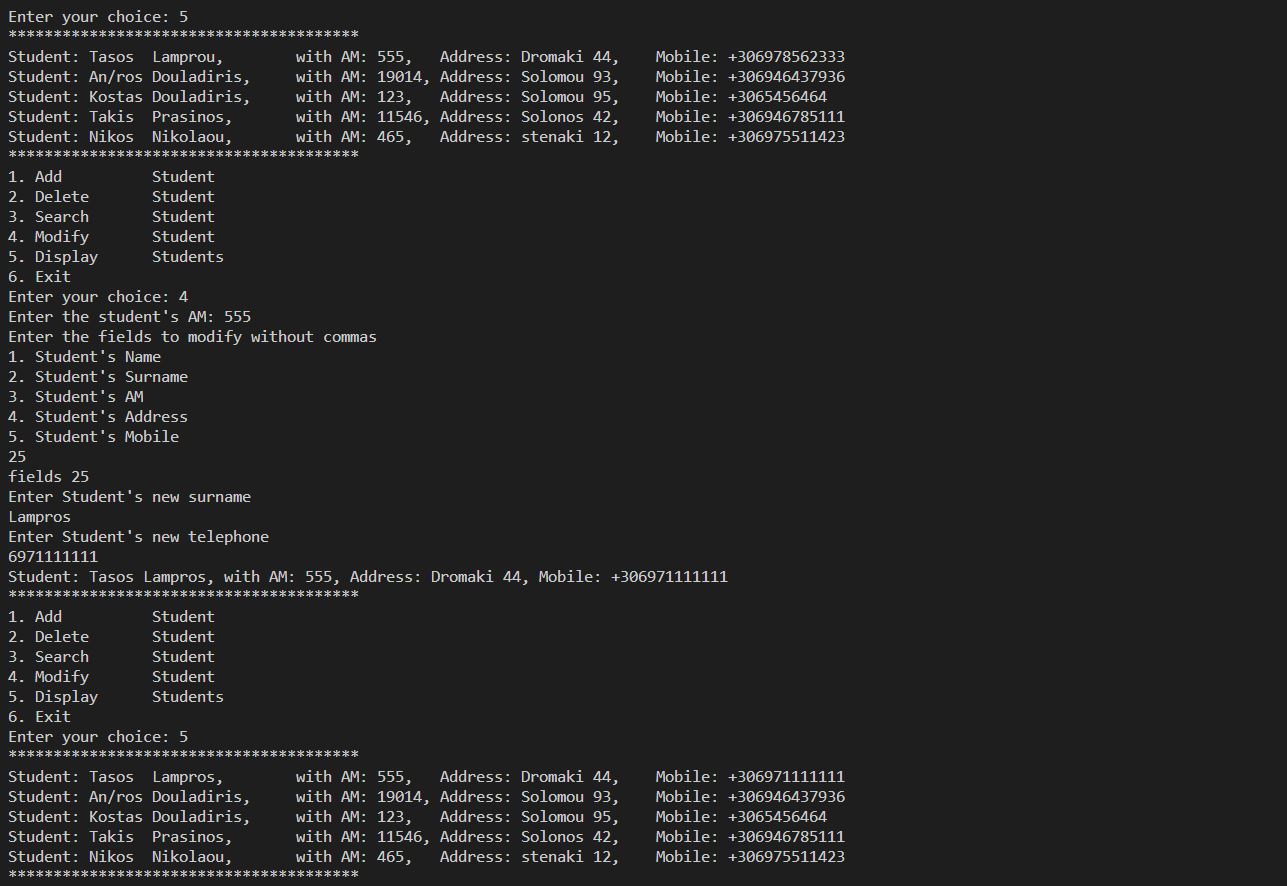
5. Καλώντας τη συνάρτηση modify(), κάνω προσπέλαση τους κόμβους έναν-έναν μέχρι να βρω το πεδίο που αναζητώ. Στη συνέχεια αποθηκεύω τον αριθμό των πεδίων που θέλω να αλλάξω σε έναν πίνακα χαρακτήρων fields και ύστερα για κάθε πεδίο που θέλω να αλλάξω, χρησιμοποιώ ανάθεση και συναρτήσεις strcat και strcpy.

Παρακάτω βλέπουμε τις αλλαγές στην εγγραφή:

Student: Tasos Lamprou, with AM: 555, Address: Dromaki 44, Mobile: +306978562333

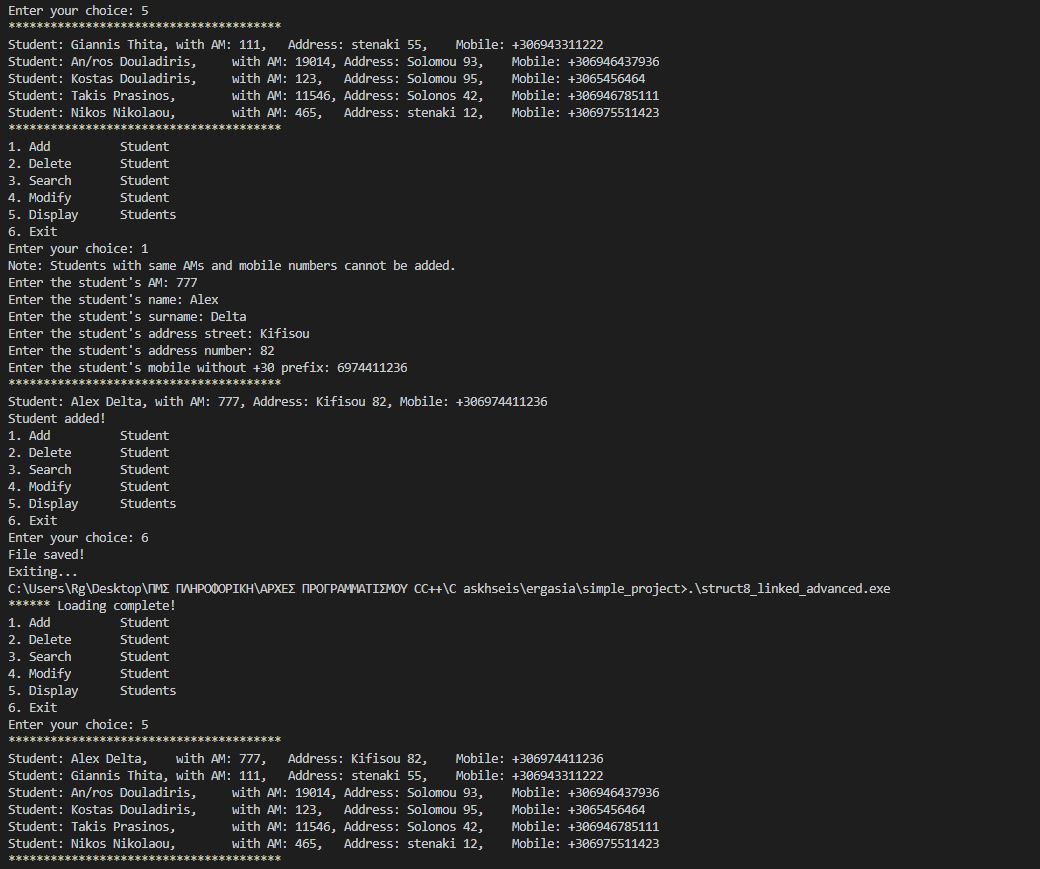
όπου επιλέγω να αλλάξω τα πεδία fields 25 (δηλαδή Student's new surname και Student's new telephone) και έτσι έχω την νέα εγγραφή:

Student: Tasos Lampros, with AM: 555, Address: Dromaki 44, Mobile: +306971111111

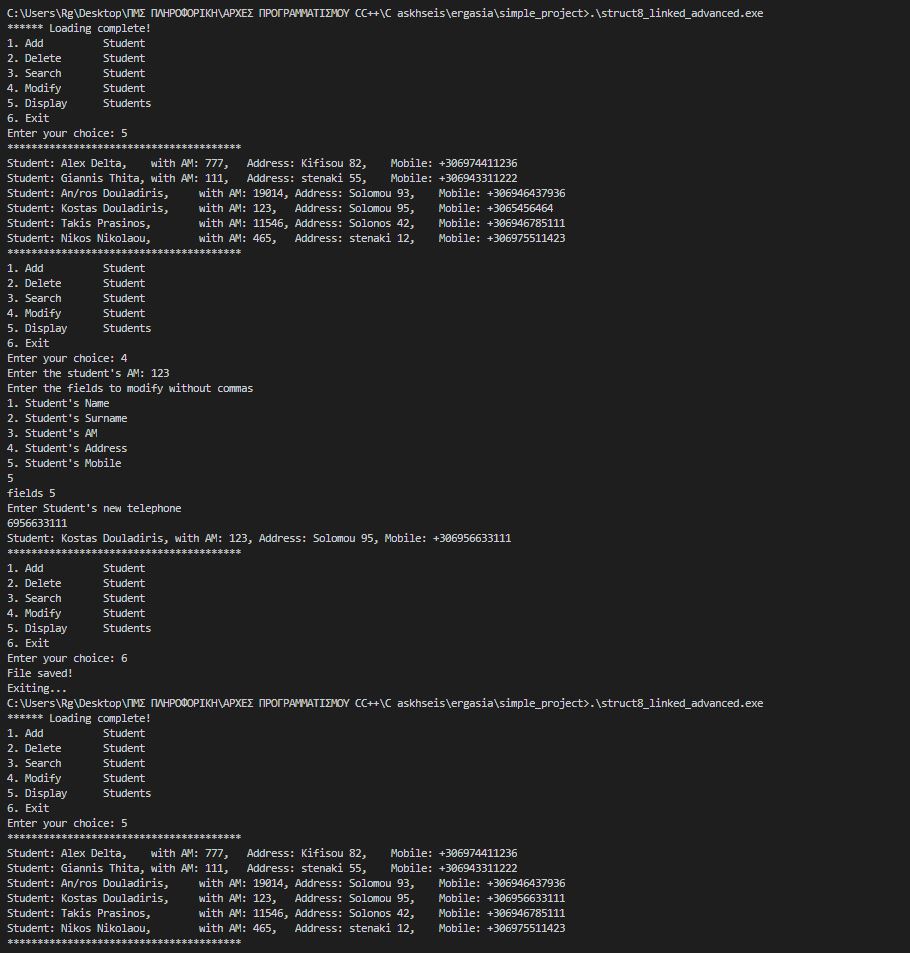


6. Τέλος, καλώντας τη συνάρτηση save(), ορίζω έναν pointer τύπου FILE ο οποίος δείχνει στο αρχείο ./student.dat και κατα την προσπέλαση των κόμβων της λίστας αποθηκεύει με τη χρήση της fwrite() κάθε κόμβο τύπου struct student σε δυαδική μορφή.

Είτε έχω add:



Είτε έχω modify:



Είτε έχω delete:

