ΠΕΚΟΣ ΕΥΑΓΓΕΛΟΣ 1115201400157
ΜΕΝΤΖΕΛΟΣ ΗΛΙΑΣ 1115201400106
ΠΑΝΤΕΛΙΑΔΗ ΕΥΦΡΟΣΥΝΗ 1115201400301

3η Ασκηση ΥΣΒΔ 2017-2018

Η υλοποιήση της εργάσιας έγινε σε γλώσσα C και βρίσκεται στο φάκελο sorted_file_64.
Υλοποιήσαμε όλα τα ζητούμενα της εκφώνησης και είναι πλήρως λειτουργική. Παρακάτω, ακολουθεί μια σύντομη περιγραφή των δομών και των συναρτήσεων που χρησιμοποιούνται.

ΔΟΜΕΣ

Υλοποιήθηκε μια δομή heap, ώστε να βρίσκουμε την ελάχιστη από τις bufferSize-1 εγγραφές, η οποία θα γραφτεί στο block του νέου αρχείου. Οι συναρτήσεις που την αποτελούν είναι :

```
void update_heap(heap * my_heap, heap_node * node);

void add_heap(heap * my_heap, heap_node * node);

int is_empty_heap(heap * my_heap);

void delete_heap(heap ** my_heap);
```

ΣΥΝΑΡΤΗΣΕΙΣ

- **int Compare(Record* record1,Record* record2,int fieldNo)**: Χρησιμοποιείται για την σύγκριση δύο εγγραφών με βάση το fieldNo
- **void Sort_Each_BufferSize_Blocks(int fd_temp , int copied_blocks , int fieldNo, int bufferSize):** Χρησιμοποιείται για το πρώτο βήμα της ταξινόμησης. Ταξινομούμε επιτόπου τα blocks του αρχείου σύμφωνα με το bufferSize. Γι’ αυτό το βήμα της ταξινόμησης υλοποιήσαμε και κάναμε χρήση της quicksort (**void quicksort(char ** data, int low, int high, int fieldNo)** στην οποία αντίστοιχα χρησιμοποιούμε την **int partition(char ** data, int low, int high, int fieldNo)** η οποία κάνει τους απαραίτητους δισχωρισμός για την quicksort. Συμπεριφερόμαστε στα bufferSize blocks σαν να είναι ενιαίος πίνακας και έτσι η ταξινόμηση έχει μικρές διαφορές με μία κανονική QuickSort σε έναν πίνακα.
- **void Copy_Block(int fd , int fd_temp):** Χρησιμοποιείται για να αντιγράψουμε το αρχικό ΜΗ ταξινομημένο αρχείο (που προκύπτει από την SR_InsertEntry(int fileDesc, Record record)) σε ένα προσωρινό, έτσι ώστε σε αυτό να αρχίσουμε την ταξινόμηση.

- **char * sort_by_level(int file_id, int level, int fieldNo, int bufferSize):** Εδώ, γίνονται τα επόμενα επίπεδα της ταξινόμησης. Δημιουργεί ένα νέο αρχείο όπου τα blocks είναι ταξινομημένα. Πιο συγκεκριμένα, πρόκειται για μια επανάληψη όπου φτιάχνει ενα προσωρινό αρχειο και έχει bufferSize – 1 blocks από τα οποία διάβαζει κάθε φορά το μικρότερο στοιχείο με τη χρήση του heap και γράφει στο νέο αρχείο. Έτσι σε κάθε επανάληψη ταξινομούνται (bufferSize – 1)* blocks μέχρι να ταξινομηθούν όλα.
- **void SR_ReadRecord(void * data, Record * record) :** Διαβάζει μια εγγραφή από το block
- **void SR_PrintRecords(Record record) :** Γράφει μια εγγραφή στο block
- **void SR_PrintRecords(Record record) :** Εκτυπώνει μια εγγραφή από το block

Όλες οι άλλες συναρτήσεις, εκτός από την SR_SortedFile, είναι ίδιες με την 1η άσκηση του μαθήματος.