ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος
Δομές Δεδομένων (C++)

Αριθμός εργασίας – Τίτλος εργασίας	***COVID-19 Application – Εργασία 2 Ατόμων***
Ονόματα φοιτητών	Μιχάλης Στυλιανίδης Κωνσταντίνος Καλογερόπουλος
Αρ. Μητρώων	Π19165 Π19057
Ημερομηνία παράδοσης	16/5/2020

1 Εισαγωγή

Για την παρούσα εργασία έχει υλοποιηθεί ένα πρόγραμμα που προσομοιάζει μια εφαρμογή κινητών (smartphones) στην οποία οι χρήστες δίνουν με την συγκατάθεση τους την τοποθεσία οπού βρίσκονται, την κατάσταση την κατάσταση της υγείας τους και μετά από κάποιες λειτουργίες που πραγματοποιούνται, οι χρήστες έχουν την δυνατότητα να γνωρίζουν εάν βρέθηκε κοντά τους φορέας του COVID-19.

2 Περιγραφή του προγράμματος

Το πρόγραμμα υλοποιείται με την χρήση απλών συνδεδεμένων λιστών. Η χρήση της λίστας προσομοιάζει την τροχιά του χρήστη, η οποία περιέχει την ημέρα(Day), τις συντεταγμένες(x,y) που βρίσκεται μέσα στον χώρο, την ώρα σε μορφή ΩΡΕΣ/ΛΕΠΤΑ/ΔΕΥΤΕΡΟΛΕΠΤΑ και την κατάσταση της υγείας του χρήστη. Η κίνηση των χρηστών γίνεται σε ένα δισδιάστατο χώρο (DxD όπου D η πλευρά του πλέγματος), οπού δημιουργείται τυχαία με την τιμή της πλευράς να κυμαίνεται από 50 έως 100. Η πάροδος του χρόνου υλοποιήθηκε με την χρήση ενός επαναληπτικού βρόγχου και ενός μετρητή ο οποίος αυξάνεται ανά 30 για να προσομοιωθεί η πάροδος των 30 δευτερολέπτων. Στο σενάριο αυτό, όλες οι τυχαίες τοποθεσίες που δημιουργούνται βρίσκονται εντός της παραπάνω τετράγωνης περιοχής και επίσης θεωρείται ότι όλοι οι χρήστες κινούνται πεζοί και με ταχύτητες που κυμαίνονται μεταξύ 3 και 6 km/h.

2.1 Δομή του προγράμματος

Το πρόγραμμα είναι χωρισμένο στα έξι ακόλουθα αρχεία.

2.1.1 Main.cpp

Το αρχείο περιλαμβάνει την συνάρτηση main στην οποία εκτελείται η ροή του προγράμματος και κάποιες επιπλέον βοηθητικές συναρτήσεις.

2.1.2 User.h

Το αρχείο αυτό περιλαμβάνει το struct User, το οποίο ορίζει τα χαρακτηρίστηκα ενός χρήστη συγκεκριμένα τις συντεταγμένες του, την ώρα σε μορφή ωρα:λεπτα:δευτερολεπτα και την κατάσταση μόλυνσης του χρήστη. Επίσης, περιέχει την συνάρτηση setInfectionStatus() η οποία επιλέγει με 20% πιθανότητα εάν ο χρήστης έχει μολυνθεί από τον Covid-19 και την συνάρτηση displayUserData() η οποία εμφανίζει όλα τα στοιχεία του χρήστη που αναφέρθηκαν πιο πάνω.

2.1.3 LinkedList.h

Το αρχείο αυτό ορίζεται ως header file και περιλαμβάνει το struct node το οποίο ορίζει την απλά συνδεδεμένη λίστα και όλες τις συναρτήσεις επί αυτής.

2.1.4 LinkedList.cpp

Το αρχείο αυτό περιλαμβάνει την υλοποίηση των συναρτήσεων της λίστας.

2.1.5 COVID19_Operations.h

Στο αρχείο αυτό ορίζονται κάποιες global μεταβλητές και οι τέσσερις συναρτήσεις που ζήτουνται.

2.1.6 COVID19_Operations.cpp

Στο αρχείο αυτό υλοποιούνται οι συναρτήσεις που ορίστηκαν στο συνώνυμο header file.

2.2 Απλά συνδεδεμένη λίστα

Η δομή που χρησιμοποιήθηκε για την αποθήκευση της τροχιάς του κάθε χρήστη είναι η απλά συνδεδεμένη λίστα.

```
struct User {
    // Ταυτότητα χρήστη
    int id;
    // Ωρα
    int hours;
    int minutes;
    int seconds;
    // Συντεταγμένες
    int x;
    int y;
    // Κατάσταση μόλυνσης
    bool infected;
```

Εικόνα 1 Η δομή User είναι το data μέρος του κόμβου .

2.2.1 void llInit (listPtr* head);

```
void llInit(listPtr* head) {
    *head = nullptr;
}
```

Εικόνα 2 Σε αυτό το σημείο αρχικοποιείται στην μνήμη η λίστα.

2.2.2 User llData (listPtr p);

```
User llData(listPtr ptr) {
    return ptr->data;
}
```

Εικόνα 3 Η συνάρτηση επιστρέφει το περιεχόμενο(τα δεδομένα) ενός κόμβου.

2.2.3 int llInsertEnd (listPtr* head, User user);

```
int llInsertEnd(listPtr* head, User user) {
    listPtr newNode;

    newNode = (listNode*)malloc(sizeof(listNode));
    if (!newNode) {
        cout << "Memory couldn't be allocated" << endl;
        return false;
    }
    newNode->data = user;

    newNode->next = nullptr;
    if (*head == nullptr) {
        *head = newNode;
    } else {
        listPtr temp = *head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    return true;
}
```

Εικόνα 4 Προσθέτει άλλο ένα κόμβο στο τέλος της λίστας.

2.2.4 int llInsertAfter (listPtr p, User user);

```
int llInsertAfter(listPtr p, User user) {
    listPtr newNode;

    // Δέσμευση μνήμης για νέο κόμβο
    newNode = (listNode*)malloc(sizeof(listNode));
    // Έλεγχος επιτυχίας δέσμευσης μνήμης
    if (!newNode) {
        cout << "Memory couldn't be allocated" << endl;
        return false;
    }
    // Προσθήκη του user στο data
    newNode->data = user;

    newNode->next = p->next; // Το next του νέου κόμβου δείχνει εκεί όπου έδειχνε το p->next
    p->next = newNode; // Και το p->next τώρα δείχνει στο νέο κόμβο
    return true;
}
```

Εικόνα 5 Η προσθήκη(insert) ενός στοιχείου, γίνεται μετά από έναν ενδιάμεσο κόμβο(στον οποίο δείχνει ο p).

Αρχικά δημιουργεί έναν νέο κόμβο και θέτει τα δεδομένα σε αυτόν, έπειτα θέτει τον next του νέου κόμβου να δείχνει στον next του ενδιάμεσου κόμβου και τέλος θέτει το next του ενδιάμεσου κόμβου να δείχνει στο νέο κόμβο.
Παράλληλα πραγματοποιείται ένας τυπικός έλεγχος για την επιτυχή δέσμευση μνήμης.

2.2.5 int llDeleteStart (listPtr* head, User* user);

```
int llDeleteStart(listPtr* head, User* user) {
    listPtr current;

    if (*head==nullptr) {
        cout << "Δεν υπάρχει στοιχείο στη λίστα για διαγραφή" << endl;
        return false;
    }

    current = *head;
    *user = current->data;

    *head = (*head)->next;
    free(current);
    return true;
}
```

Εικόνα 6 Η συνάρτηση δέχεται ως όρισμα την κεφαλή μιας λίστας και διαγράφει τον πρώτο κόμβο της λίστας.

2.2.6 int llDeleteAfter (listPtr prev, User* user);

```
int llDeleteAfter(listPtr prev, User* user) {
    listPtr current;

    if (prev->next==nullptr) {
        cout << "Δεν υπάρχει κόμβος μετά τον δοθέν για διαγραφή";
        return false;
    }

    current = prev->next;
    *user = current->data;

    prev->next = current->next;
    free(current);
    return true;
}
```

Εικόνα 7 Η διαγραφή(delete) ενός στοιχείου, γίνεται μετά από κάποιο υφιστάμενο στοιχείο της λίστας.

Παράλληλα, πραγματοποιείται έλεγχος εγκυρότητας για το αν υπάρχει κόμβος μετά από αυτόν που δόθηκε και εμφανίζει κατάλληλο μήνυμα σε περίπτωση που δεν υπάρχει.

2.2.7 void llDisplay (listPtr head);

```
void llDisplay(listPtr head) {
    listPtr current;

    current = head;
    while (current != nullptr) {
        (current->data).displayUserData();
        current = current->next;
    }
}
```

Εικόνα 8 Η συνάρτηση αυτή εμφανίζει το περιεχόμενο της λίστας.

```
void displayUserData() const {
    cout << "User " << id <<  " -> Coordinates: x = " << x << " y = " << y << ". Time ";
    cout << hours << ":" << minutes << ":" << seconds << boolalpha << " Infection status: " << infected << endl;
}
```

Εικόνα 9 Η συνάρτηση αυτή μορφοποιεί τα δεδομένα του struct και καλείται από την llDisplay(βλ. εικόνα 8) για την εμφάνιση των κόμβων της λίστας.

2.2.8 void llDestroy (listPtr* head);

```
void llDestroy(listPtr* head) {
    listPtr ptr;

    while (*head != nullptr) {
        ptr = *head;
        *head = (*head)->next;
        free(ptr);
    }
}
```

Εικόνα 10 Διαγράφει όλη την λίστα.

2.3 Βασικές λειτουργίες του προγράμματος

2.3.1 void repair (listPtr userTrajectory);

Πολλές φορές στις κινητές συσκευές υπάρχουν απώλειες της μετάδοσης του στίγματος GPS με αποτέλεσμα μερικοί σημεία του χώρου να λείπουν από την ροή του χρόνου. Η συνάρτηση αυτή καλείται να ανιχνεύσει αυτά τα σημεία και να συμπληρώσει τους κόμβους που λείπουν από την λίστα.

Για την λειτουργία της παραπάνω συνάρτησης δημιουργήθηκαν τεχνητά <<χαμένα>> στίγματα δεδομένων μέσα στον χρόνο της εκτέλεσης του προγράμματος ώστε να επιτελείται ο σκοπός κατασκευής της.

Για το σκοπό αυτό δημιουργήθηκε η συνάρτηση gpsWorked() όπως περιγράφεται στην παρακάτω εικόνα και έχει 10% πιθανότητες να επιστρέψει false οπότε με τον έλεγχο της τιμής που επιστρέφει, επιλέγουμε κάποια δεδομένα να μην εισάγονται στην λίστα .

```
bool gpsWorked() {
    // 10% πιθανότητα να μη δουλέψει το gps και να χαθεί μία χρονική στιγμή
    int random = rand() % 10;
    return random != 1;
}
```

Εικόνα 11 Δημιουργία της συνάρτησης

```
if (gpsWorked()) { // Η gpsWorked επιστρέφει true αν το gps δούλεψε, αλλιώς false
    // Εισαγωγή στη λίστα του συγκεκριμένου χρήστη της συγκεκριμένης μέρας
    // της πληροφορίας περί απόστασης, χρόνου, id και κατάστασης μόλυνσης
    llInsertEnd(&Users[day][userNum], user);
}
```

Εικόνα 12 Έλεγχος εισαγωγής δεδομένων με βάση την συνάρτηση gpsworked()

2.3.2 int findCrowdedPlaces (int day, int startSeconds, int endSeconds, int squareRegionOfInterest, int minimumStayDuration, listPtr users[][usersNumber]);

Η συνάρτηση αυτή επιστρέφει το πλήθος των χρηστών που βρέθηκαν εντός μίας τετραγωνικής περιοχής μία συγκεκριμένη ημέρα, εντός ενός χρονικού διαστήματος και παρέμειναν στην περιοχή τουλάχιστον για κάποια ώρα. Με αυτό τον τρόπο, οι αρχές μπορούν να εντοπίζουν προβληματικές περιοχές ώστε να λάβουν τα κατάλληλα μέτρα.

Περιληπτικά, επιστρέφει το πλήθος των χρηστών που βρέθηκαν σε ένα μια δοθέντα περιοχή ,σε και για, κάποιο χρονικό διάστημα μέσα στην μέρα.

2.3.3 bool possibleCOVID_19Infection (listPtr userTrajectory, int day, listPtr allUsers[][usersNumber])

Η συνάρτηση αυτή θα παίρνει ως είσοδο την τροχιά ενός χρήστη για μία συγκεκριμένη ημέρα, θα την συγκρίνει με τις τροχιές των ασθενών και θα επιστρέφει TRUE αν ο χρήστης βρέθηκε εντός ακτίνας R από τον ασθενή για διάστημα τουλάχιστον T1 λεπτών της ώρας και το πολύ T2 λεπτά αργότερα από τη στιγμή που πέρασε ο ασθενής. Π.χ. αν Day= 4/8/2020, R=2m, T1=15 λεπτά και T2= 460 λεπτά (4 ώρες), αν την 4/8/2020 ο χρήστης πέρασε σε ένα μέτρο απόσταση 2 ώρες μετά από τον σημείο που βρέθηκε ένας ασθενής για κορωνοϊό και έμεινε στην περιοχή για 30 λεπτά τότε πρέπει να ειδοποιείται (TRUE).

3 Επίδειξη της λύσης

Σε αυτό το σημείο, παρουσιάζεται η εκτέλεση (run) του προγράμματος.

3.1.1 void repair (listPtr userTrajectory);

Η συνάρτηση αυτή προσθέτει τους κατάλληλους κόμβους στο <τυχαία> ορισμένο χρονικό διάστημα που λείπουν.

```
User 1 -> Coordinates: x = 63 y = 33. Time 0:0:30 Infection status: false
User 1 -> Coordinates: x = 61 y = 33. Time 0:1:0 Infection status: false
User 1 -> Coordinates: x = 59 y = 33. Time 0:1:30 Infection status: false
User 1 -> Coordinates: x = 57 y = 33. Time 0:2:0 Infection status: false
User 1 -> Coordinates: x = 55 y = 33. Time 0:2:30 Infection status: false
User 1 -> Coordinates: x = 55 y = 31. Time 0:3:0 Infection status: false
User 1 -> Coordinates: x = 55 y = 29. Time 0:3:30 Infection status: false
User 1 -> Coordinates: x = 55 y = 27. Time 0:4:0 Infection status: false
User 1 -> Coordinates: x = 55 y = 25. Time 0:4:30 Infection status: false
User 1 -> Coordinates: x = 55 y = 23. Time 0:5:0 Infection status: false
User 1 -> Coordinates: x = 55 y = 21. Time 0:5:30 Infection status: false
User 1 -> Coordinates: x = 55 y = 19. Time 0:6:0 Infection status: false
User 1 -> Coordinates: x = 55 y = 18. Time 0:6:30 Infection status: false
User 1 -> Coordinates: x = 55 y = 15. Time 0:7:0 Infection status: false
User 1 -> Coordinates: x = 55 y = 14. Time 0:7:30 Infection status: false
User 1 -> Coordinates: x = 55 y = 14. Time 0:8:0 Infection status: false
User 1 -> Coordinates: x = 55 y = 14. Time 0:8:30 Infection status: false
User 1 -> Coordinates: x = 57 y = 14. Time 0:9:0 Infection status: false
User 1 -> Coordinates: x = 59 y = 14. Time 0:9:30 Infection status: false
User 1 -> Coordinates: x = 61 y = 14. Time 0:10:0 Infection status: false
User 1 -> Coordinates: x = 61 y = 14. Time 0:10:30 Infection status: false
User 1 -> Coordinates: x = 63 y = 16. Time 0:11:0 Infection status: false
User 1 -> Coordinates: x = 63 y = 18. Time 0:11:30 Infection status: false
```

Εικόνα 13

Όπως παρατηρείται στην εικόνα 13, τα στίγματα GPS (συντεταγμένες x,y) έχουν συμπληρωθεί ορθά από την συνάρτηση repair() και δεν υπάρχουν χρονικά διαστήματα μεγαλύτερα των 30 χωρίς να έχει καταγραφεί η κίνηση του χρήστη.

3.1.2 int findCrowdedPlaces (int day, int startSeconds, int endSeconds, int squareRegionOfInterest, int minimumStayDuration, listPtr users[][usersNumber]);

Η συνάρτηση εκτελείται με την επιλογή που δίνεται από τον χρήστη στο τέλος κάθε ημέρας.

```
---------------------- DAY 1 ----------------------


Do you want to check for crowded places? Press 1 for yes, or 0 for no
1
Based on the provided data there were 4 people in the given area

---------------------- DAY 2 ----------------------


Do you want to check for crowded places? Press 1 for yes, or 0 for no
1
Based on the provided data there were 3 people in the given area

---------------------- DAY 3 ----------------------


Do you want to check for crowded places? Press 1 for yes, or 0 for no
1
Based on the provided data there were 4 people in the given area
```

Εικόνα 14

Όπως παρατηρείται στην εικόνα 14, το πρόγραμμα ρωτάει τον χρήστη αν θέλει να ελέγξει κάποια περιοχή για συγχρωτισμό και όταν ο χρήστης δίνει την τιμή 1, εκτελείται η συνάρτηση findCrowdedPlaces() και εμφανίζει τον αριθμό των ατόμων που βρέθηκαν στην ίδια περιοχή στο δοθέν χρονικό διάστημα και χώρο.

3.1.3 bool possibleCOVID_19Infection (listPtr userTrajectory, int day, listPtr allUsers[][usersNumber])

Η συνάρτηση αυτή στο τέλος κάθε ημέρας εκτυπώνει μήνυμα για τους χρήστες που βρέθηκαν κοντά σε κρούσμα κορωνοϊού. Λαμβάνει κάποιες συγκεκριμένες παραμέτρους που με βάση αυτές κρίνει το κατά πόσο ο χρήστης θα λάβει μήνυμα κινδύνου.

```
---------------------- DAY 1 ----------------------

None was dangerously close to a COVID-19 patient today
---------------------- DAY 2 ----------------------

User 5 was dangerously close to a COVID-19 patient and he may be sick
User 6 was dangerously close to a COVID-19 patient and he may be sick
User 22 was dangerously close to a COVID-19 patient and he may be sick
---------------------- DAY 3 ----------------------

User 11 was dangerously close to a COVID-19 patient and he may be sick
User 16 was dangerously close to a COVID-19 patient and he may be sick
---------------------- DAY 4 ----------------------

User 2 was dangerously close to a COVID-19 patient and he may be sick
---------------------- DAY 5 ----------------------

None was dangerously close to a COVID-19 patient today
---------------------- DAY 6 ----------------------

User 18 was dangerously close to a COVID-19 patient and he may be sick
User 25 was dangerously close to a COVID-19 patient and he may be sick
---------------------- DAY 7 ----------------------

User 11 was dangerously close to a COVID-19 patient and he may be sick
```

Εικόνα 15

Το μήνυμα που εμφανίζεται, παρατηρείται στην παραπάνω εικόνα. (Εικονα 15)