OOP 2021-2022

Εργασία Μαθήματος

Μωραΐτης Θεόδωρος sdi2000150

Ρούσσος Κωνσταντίνος sdi1800166

Εντολή μεταγλώττισης: **g++ main.cpp highway.cpp**

Στο πρόγραμμα χρησιμοποιούνται οι εξής κλάσεις:

Α) *Attiki_Odos*, η οποία περιέχει:

- έναν πίνακα με δείκτες στα τμήματα που αποτελούν τον αυτοκινητόδρομο (Segment **segments).
- τον συνολικό αριθμό των οχημάτων που βρίσκονται στον αυτοκινητόδρομο (int total_vehicles).
- το πλήθος των τμημάτων που αποτελούν τον αυτοκινητόδρομο (int NSegs).

Β) *Segment*, η οποία περιέχει:

- έναν δείκτη στην είσοδο του τρέχοντος τμήματος (Entrance *entrance).
- έναν πίνακα με δείκτες στα οχήματα που βρίσκονται στο τρέχον τμήμα του αυτοκινητόδρομου (Vehicle **vehicles).
- τη μέγιστη χωρητικότητα του τρέχοντος τμήματος (int capacity).
- έναν δείκτη στο προηγούμενο τμήμα του τρέχοντος (int *prev_seg).
- έναν δείκτη στο επόμενο τμήμα του τρέχοντος (int *next_seg).
- τον αριθμό του τρέχοντος τμήματος (int seg_id).
- τον αριθμό των οχημάτων που βρίσκονται στο συγκεκριμένο τμήμα (int vehicles_seg_sum).
- το πλήθος των τμημάτων που αποτελούν τον αυτοκινητόδρομο (int NSegs).
- το ποσοστό των οχημάτων, δοσμένο απ'τον χρήστη, που γίνεται ready για exit ή pass στην επόμενη κατάσταση (int Percent).

Γ) *Entrance*, η οποία περιέχει:

- το όνομα του κόμβου (string name).

- έναν πίνακα με δείκτες στα διόδια του κόμβου (Toll **tolls).

- το σύνολο των διοδίων με υπάλληλο του κόμβου (int ptoll_num).

- το σύνολο των ηλεκτρονικών διοδίων του κόμβου (int etoll_num).

- τον συνολικό αριθμό των διοδίων του κόμβου (int toll_num).

- την σταθερά, δοσμένη απ'τον χρήστη, σχετικά με το πόσα οχήματα μπορούν να εξυπηρετηθούν στα διόδια (int K).

- την σταθερά, δοσμένη απ'τον χρήστη, σχετικά με το πόσα οχήματα μπορούν να εξυπηρετηθούν στα διόδια (int Kconst). (Η διαφορά των K και Kconst εξηγείται στις ιδιαιτερότητες του προγράμματος)

- μία ένδειξη σχετικά με το αν υπάρχουν καθυστερήσεις ή όχι (bool delays).

Δ) η υπερκλάση *Toll*, η οποία περιέχει:

- έναν vector με δείκτες στα οχήματα που περιμένουν να εξυπηρετηθούν στο συγκεκριμένο διόδιο (vector <Vehicle*> vehicles).

- τον αριθμό του συγκεκριμένου διοδίου (int toll_id).

- το σύνολο των οχημάτων που περιμένουν να εξυπηρετηθούν στο συγκεκριμένο διόδιο (int vehicles_toll_sum).

- την σταθερά, δοσμένη απ'τον χρήστη, σχετικά με το πόσα οχήματα μπορούν να εξυπηρετηθούν στα διόδια (int K).

- έναν δείκτη στο όχημα που είναι έτοιμο να εισέλθει στο τμήμα του αυτοκινητόδρομου που ακολουθεί τον κόμβο στον οποίο βρίσκεται (Vehicle *veh_to_transfer).

Ε) *Physical_Toll*: υποκλάση της Toll.

ΣΤ) *Electronic_Toll*: υποκλάση της Toll.

Ζ) *Vehicle*, η οποία περιέχει:

- την έξοδο στην οποία κατευθύνεται το όχημα (int exit_node).

- το τμήμα του αυτοκινητόδρομου στο οποίο βρίσκεται στην τρέχουσα κατάσταση (int inside_seg).

- μία ένδειξη σχετικά με το αν είναι έτοιμο να κάνει exit ή pass στην επόμενη κατάσταση (bool ready).

- τον αριθμό πινακίδας του οχήματος (string lp).

Εντολή εκτέλεσης: ./**a.out** (τα ορίσματα δίνονται απ'τον χρήστη κατά την εκτέλεση της main)

Attiki_Odos constructor

Αφού ο χρήστης συμπληρώσει τα στοιχεία που του ζητούνται, καλείται ο constructor της Αττικής Οδού, με ορίσματα τα στοιχεία αυτά (int Nsegs, int K, int Percent, int capacity), όπου το capacity αντιστοιχεί στον πίνακα με τις χωρητικότητες (int capacity[ ]). Αφού αρχικοποιηθούν οι τιμές των μεταβλητών της κλάσης, δημιουργούνται τα τμήματα του αυτοκινητοδρόμου και καλείται ο constructor του κάθε τμήματος με ορίσματα τον αριθμό του τμήματος (int i), την χωρητικότητα του τμήματος (int capacity[i]), το προηγούμενο τμήμα και το επόμενο τμήμα (Segment * και τα δύο), τον αριθμό των οχημάτων που μπορούν να εξυπηρετηθούν στα διόδια (int K), το ποσοστό των οχημάτων που θα μεταβεί στην κατάσταση ready (int Percent) και τέλος το πλήθος των τμημάτων του αυτοκινητόδρομου (int NSegs).

->Το προηγούμενο και το επόμενο τμήμα αρχικοποιούνται με τιμή NULL στον constructor, καθώς τα τμήματα αυτά δεν έχουν δημιουργηθεί ακόμα. Μόλις τελειώσει η διαδικασία δημιουργίας των τμημάτων καλούνται οι Segment::set_previous(Segment *) και Segment::set_next(Segment *), για να δώσουν τις κατάλληλες τιμές στις μεταβλητές αυτές. Οι συνθήκες εξετάζουν αν βρισκόμαστε στο πρώτο, στο τελευταίο ή σε κάποιο ενδιάμεσο τμήμα, καθώς επίσης αν το πλήθος των τμημάτων είναι μόνο ένα.

Τέλος, καλείται η get_total_vehicles(), η οποία προσθέτει τα οχήματα του κάθε τμήματος και επιστρέφει το συνολικό τους πλήθος. Το πλήθος των οχημάτων του τμήματος ανακτάται απ΄την συνάρτηση Segment::get_no_of_vehicles().

Segment constructor

Ο constructor της Αττικής Οδού κάλεσε τον constructor του κάθε τμήματός του. Ο constructor αυτός, αφού αρχικοποιήσει και αυτός με τη σειρά του τις μεταβλητές, δημιουργεί τον κόμβο εισόδου του και καλεί τον constructor του, με ορίσματα τον αριθμό του τρέχοντος τμήματος (int seg_id), την χωρητικότητα του τμήματος (int capacity), τον αριθμό των οχημάτων που μπορούν να εξυπηρετηθούν στα διόδια (int K) και των πλήθος των τμημάτων του αυτοκινητόδρομου (int NSegs). Τέλος δημιουργεί τον πίνακα των οχημάτων του τμήματος και δημιουργεί και βάζει μερικά οχήματα μέσα σε αυτόν μέσω της συνάρτησης create_vehicles().

->Η Segment::create_vehicles() καλείται μόνο στην δημιουργία της κλάσης Segment. Αρχικά, δίνουμε στην μεταβλητή vehicle_seg_sum (πλήθος οχημάτων του τμήματος) μία τυχαία τιμή η οποία κυμαίνεται από 0 έως capacity/2, ώστε να μην γεμίσουμε το τμήμα απ'το καλημέρα και δημιουργήσουμε μποτιλιάρισμα στην υλοποίησή μας. Έπειτα γεμίζουμε τον πίνακα των οχημάτων με οχήματα, δημιουργώντας τα και καλώντας τον constructor τους με ορίσματα τον αριθμό πινακίδας τους (μέσω της συνάρτησης string gen_random_plate()), την έξοδο στην οποία προορίζονται (μέσω της συνάρτησης int get_exit()) και τον αριθμό του τρέχοντος τμήματος στο οποίο κινούνται (int this->seg_id).

->Η string gen_random_plate() είναι μία συνάρτηση που μέσω δυο rand() επιλέγει τρία τυχαία κεφαλαία γράμματα και τέσσερεις τυχαίους αριθμούς, ώστε να δημιουργήσει έναν μοναδικό αριθμό πινακίδων.

->Η Segment::get_exit() επιλέγει τυχαία έναν αριθμό απ'το 0 έως το NSegs. Ωστόσο, η έξοδος πρέπει να βρίσκεται μετά το τμήμα στο οποίο δημιουργείται το όχημα. Για το λόγο αυτό, υπάρχει μια while, στόχος της οποίας είναι να καλεί την rand() ξανά και ξανά έως ότου ο τυχαίος αυτός αριθμός να είναι μεγαλύτερος ή ίσος με το τρέχον τμήμα του αυτοκινητόδρομου.

Vehicle constructor

Ο constructor των οχημάτων αρχικοποιεί τις μεταβλητές με βάση τα ορίσματα και δίνει την τιμή false στην μεταβλητή ready, αφού το όχημα αρχικά δεν είναι έτοιμο να κάνει pass ή exit. Η συνθήκη if υπάρχει μόνο για την εκτύπωση μηνυμάτων.

Entrance constructor

Το κάθε τμήμα κάλεσε τον constructor του κόμβου εισόδου του, το οποίο αρχικοποίησε τις μεταβλητές του με βάση τα ορίσματα που του δόθηκαν. Το πλήθος των διοδίων με υπάλληλο και των ηλεκτρονικών πήρε την τιμή 0 και κλήθηκε η create_tolls(int, int, int) για την δημιουργία των διοδίων. Μετά την επιστροφή της συνάρτησης, αλλάζει η τιμή του πλήθους των διοδίων και η delays αποκτά την τιμή false ως ένδειξη της απουσίας καθυστερήσεων.

->Έχουμε δύο μεταβλητές K. Την K και την Kconst. Η διαφορά του είναι ότι η K όταν εισέρχονται λιγότερα οχήματα αυξάνεται κατά 1 και όταν εισέρχονται λιγότερα μειώνεται κατά 1. Η Kconst απ'την άλλη, παραμένει σταθερή και χρησιμοποιείται σε υπολογισμούς.

->Η Entrance::create_tolls(int, int, int) παίρνει σαν ορίσματα την χωρητικότητα, τον αριθμό του τμήματος στο οποίο ανήκει η είσοδος και το πλήθος των τμημάτων. Για τους υπολογισμούς μας χρησιμοποιούμε το ½ του Kconst, και για αυτό ελέγχουμε να έχουμε αριθμούς μεγαλύτερους του μηδενός. Σε κάθε είσοδο θέλουμε τουλάχιστον δύο διόδια (βάση λογικής δεν μπορεί ένα διόδιο να εξυπηρετήσει ολόκληρο κόμβο, όσο μικρή διέλευση και να έχει) και γι'αυτό ελέγχουμε το αποτέλεσμα της tollstocreate.

->Τα διόδια δημιουργούνται ως εξής. Το αποτέλεσμα της tollstocreate αν είναι άρτιος αριθμός, τότε το 80% θα γίνει διόδια με υπάλληλο και το άλλο 20% θα γίνει ηλεκτρονικά. Αν είναι περιττός, τότε το 60% θα γίνει με υπάλληλο και το 40% ηλεκτρονικά. Σε κάθε περίπτωση θα δημιουργηθεί το δίοδιο που θέλουμε(με υπάλληλο ή ηλεκτρονικό), θα κληθεί ο ανάλογος constructor και θα αυξηθεί ο μετρητής του αντίστοιχου διοδίου, καθώς και ο συνολικός μετρητής διοδίων.

->Η συνάρτηση tollstocreate=capacity/(Kconst/2), καθώς και τα ποσοστά 80% άρτιος και 60% περιττός είναι αποτέλεσμα διαδοχικών δοκιμών τιμών για την καλύτερη δημιουργία και κατανομή διοδίων.

Toll constructor

Αρχικοποιούνται οι μεταβλητές βάση ορισμάτων, το πλήθος των οχημάτων που περιμένουν στα διόδια αποκτά μια τυχαία τιμή συναρτήσει του K/2 και καλείται η create_vehicles(int ,int).

->Η Toll::create_vehicles(int, int) δημιουργεί οχήματα και καλείται ο constructor τους με ορίσματα την string gen_random_plate(), την int get_exit(int, int) και την τιμή -1 που δηλώνει ότι το όχημα βρίσκεται στα διόδια. Έπειτα καλείται η push_back(), ώστε να προστεθεί στο vector των οχημάτων.

->Η Toll::get_exit(int, int) λειτουργεί ακριβώς όπως και της Segment.

Operate

Η Attiki_Odos::operate() αρχικά θέτει μέσω της ready_vehicles() κάποια οχήματα σε ready, σε κάθε τμήμα του αυτοκινητόδρομου. Έπειτα καλεί την operate() του κάθε τμήματος, από το τελευταίο τμήμα προς την αρχή του αυτοκινητόδρομου. Στο τέλος εκτυπώνει, με τη βοήθεια της συνάρτησης int get_total_vehicles() το σύνολο των οχημάτων που περιέχει στο τέλος του κάθε κύκλου προσομοίωσης. Ο κύκλος προσομοίωσης τελειώνει με την κλήση της add_vehicles() από την main, που προσθέτει οχήματα στα διόδια της κάθε εισόδου.

->Η Segment::ready_vehicles() κάνει ready (Vehicle::set_ready(), όπου ready=true) ένα ποσοστό των οχημάτων, ξεκινώντας απ' την αρχή του πίνακα. Μία ιδιαιτερότητα της υλοποίησης είναι ότι αν δεν καταφέρουν να περάσουν όλα τα ready, τότε παραμένουν ready και συμπεριλαμβάνονται στην μαθηματική συνάρτηση: ready = get_no_of_vehicles() * ((float)Percent/100). Δηλαδή, στον επόμενο κύκλο δεν τα προσθέτουμε στο νέο αριθμό ready, αλλά χρησιμοποιούνται ώστε να βγει το νέο ready και εφόσον βρίσκονται στην αρχή αρχή (όπως θα εξηγήσουμε παρακάτω, στη sort_vehicles()) θα είναι σίγουρα ready στον επόμενο κύκλο.

->Η add_vehicles() καλεί για κάθε τμήμα την Segment::add_vehicles(int, int) με ορίσματα το πλήθος των τμημάτων και το τρέχων τμήμα. Η Segment::add_vehicles(int, int) με τη σειρά της καλεί την Entrance::add_vehicles(int, int) με τα ίδια ορίσματα, η οποία με τη σειρά της καλεί την Toll::add_vehicles(int, int, int, int) με έξτρα ορίσματα τον αριθμό των διοδίων με υπάλληλο και των ηλεκτρονικών.

->Η Toll::add_vehicles(int, int, int, int) δημιουργεί τυχαία οχήματα στα διόδια. Η αρχική if, εξυπηρετεί στην δημιουργία επαρκών οχημάτων, ώστε να μην έχουμε έλλειψη. Οι πράξεις και οι συντελεστές είναι αποτέλεσμα διάφορων δοκιμών. Στην συνέχεια στο vtemp vector που έχουμε δηλώσει προσθέτουμε νέα οχήματα και καλούμε την sort(vector <Vehicle*>) πριν τον διαγράψουμε.

->Η Toll::sort(vector <Vehicle*>) μας εγγυάται ότι σε κάθε εισαγωγή νέου οχήματος στα διόδια, το αμάξι που μπήκε πρώτο θα βγει και πρώτο. Αυτό επιτυγχάνεται μέσω ενός reverse του αρχικού, που το είχαμε ήδη κάνει reverse στην Toll::create_vehicles(int, int), έναν insert μεταξύ των δύο vectors και άλλη μια reverse στο τέλος για να φέρει τα πρώτα στοιχεία στο τέλος του vector.

Στην Segment::operate() έχουμε:

~Καλείται η exit(int) με το όχημα το οποίο είναι ready και έχει ως προορισμό την έξοδο του τμήματος στο οποίο βρίσκεται. Αν είναι επιτυχής, η μεταβλητή flag γίνεται true. Η Segment::exit(int) απλά διαγράφει το όχημα απ'τον πίνακα και του δίνει την τιμή NULL, ώστε να μην έχει κάποια τυχαία τιμή γιατί θα μας χρειαστεί στους ελέγχους.

->Η flag ελέγχει αν κάποιο όχημα μετακινήθηκε απ'τον πίνακα (είτε exit είτε pass). Αυτό ταυτόχρονα σημαίνει ότι υπάρχουν κενές θέσεις μεταξύ των οχημάτων με τιμή NULL. Εάν η flag είναι true, τότε καλείται η sort_vehicles(), η οποία έχει έναν μετρητή και δύο οδηγούς. Ο μετρητής count αυξάνεται κάθε φορά που συναντάμε όχημα ή κάθε φορά που μεταφέρουμε όχημα σε κενή θέση. Ο πρώτος οδηγός i σταματάει κάθε φορά που βρίσκουμε κενό κελί. Εκεί ξεκινάει ο δεύτερος j, ο οποίος μόλις βρει επόμενο όχημα το αντιγράφει μέσω copy constructor στην θέση του i και διαγράφει το όχημα που βρισκόταν στην θέση j, δημιουργώντας μια νέα κενή θέση. Με αυτόν τον τρόπο εξασφαλίζουμε ότι όλα τα οχήματα βρίσκονται μπροστά μπροστά στον πίνακα για τους υπολογισμούς που ακολουθούν.

~Καλείται η prev_seg->pass(i) με τα οχήματα του προηγούμενου τμήματος, τα οποία είναι ready και δεν έχουν ως προορισμό την έξοδο του τμήματος στο οποίο βρίσκονται. Αν είναι επιτυχής, η μεταβλητή flag2 γίνεται true και καλείται η sort_vehicles() όπως αναφέραμε ακριβώς από πάνω.

->Η pass(int), λειτουργεί για το παρόν τμήμα αλλά καλείται απ'το επόμενο. Για τον λόγο αυτό, παίρνει το πλήθος των οχημάτων που βρίσκονται στο επόμενο τμήμα και εφόσον αυτό είναι μικρότερο της χωρητικότητας, κάνει αντιγραφή το όχημα της θέσης που έλαβε σαν όρισμα, μέσω copy constructor, στην κενή θέση του πίνακα των οχημάτων στο επόμενο τμήμα. Αυξομειώνονται οι ποσότητες οχημάτων στα αντίστοιχα τμήματα και μετά την delete δίνουμε την τιμή NULL στο κελί που διαγράψαμε, ώστε να μην έχει κάποια τυχαία τιμή γιατί θα μας χρειαστεί στους ελέγχους.

~Μετά την pass(int) ακολουθεί η enter(int). Αρχικά βρίσκουμε πόσες κενές θέσεις έχουμε και αν αυτές δεν είναι μηδενικές προχωράμε στην είσοδο οχημάτων απ'τα διόδια, διαφορετικά καλούμε την Entrance::check_delays(int) με όρισμα 0, θα δούμε μετά γιατί.

->Η Segment::enter(int) παίρνει τις κενές θέσεις και καλεί την Entrance::operate(int, Segment*) με ορίσματα τις κενές θέσεις και τον εαυτό της, ώστε να μπορέσει μετέπειτα η είσοδος να έχει πρόσβαση στην συνάρτηση Segment::transfer_vehicle(Vehicle *).

Η Entrance::operate(int, Segment *) λειτουργεί ως εξής:

~Στην συνάρτηση έχουμε τρεις μετρητές, τρεις ενδείξεις και έναν οδηγό. Ο οδηγός i προσπελάζει τον πίνακα των διοδίων. Ο μετρητής count αυξάνεται κάθε φορά που μεταφέρουμε ένα όχημα μέσα στον αυτοκινητόδρομο. Ο μετρητής kappa αναφέρεται στα οχήματα που μπορούν να εξυπηρετηθούν σε διόδια με υπάλληλο και αυξάνεται κάθε φορά που κάποιο εξυπηρετείται. Ο μετρητής twokappa αναφέρεται στα οχήματα που μπορούν να εξυπηρετηθούν σε ηλεκτρονικά διόδια και αυξάνεται κάθε φορά που κάποιο εξυπηρετείται. Η ένδειξη flag1 γίνεται false μόλις τελειώσουν τα οχήματα στα διόδια με υπάλληλο ή μόλις εξυπηρετηθούν Κ οχήματα. Η ένδειξη flag2 γίνεται false μόλις τελειώσουν τα οχήματα στα ηλεκτρονικά διόδια ή μόλις εξυπηρετηθούν 2Κ οχήματα. Η ένδειξη flag γίνεται false μόλις τελειώσουν τα οχήματα στα διόδια ή μόλις εξυπηρετηθούν 3Κ οχήματα. Το i αυξάνεται κυκλικά, οπότε μόλις φτάσει στο τελευταίο διόδιο ξαναγυρνάει στην αρχή.

~Καλείται η Toll::remove_Vehicle() και αν το αποτέλεσμά της είναι true δημιουργούμε μέσω copy constructor και της συνάρτησης Toll::get_veh_to_transfer() ένα αντίγραφο του οχήματος και το μεταφέρουμε στον πίνακα οχημάτων του τμήματος μέσω της Segment::transfer_vehicle(Vehicle *). Διαφορετικά, αν είναι false ελέγχουμε αν υπάρχουν άλλα οχήματα στα υπόλοιπα διόδια ώστε να προχωρήσουμε σε αυτά ή να λήξουμε αυτό το βρόχο. Το ίδιο γίνεται και στα ηλεκτρονικά διόδια στην από κάτω if. Τέλος καλείται η check_delays(int) με όρισμα τον μετρητή count.

->Η Toll::remove_vehicle() ελέγχει αν υπάρχουν οχήματα στον vector για αφαίρεση και αν ναι, αφαιρεί το τελευταίο στοιχείο του vector, το αποθηκεύει στην Vehicle * veh_to_transfer και επιστρέφει την τιμή true, διαφορετικά απλά επιστρέφει false. Η Toll::get_veh_to_transfer() απλά επιστρέφει την τιμή της private μεταβλητής veh_to_transfer.

->Η Segment::transfer_vehicle(Vehicle *) προσθέτει το νέο όχημα μέσω copy constructor στον πίνακα οχημάτων του τμήματος και αυξάνει το πλήθος του.

->Η Entrance::check_delays(int) δίνει στην μεταβλητή delays την τιμή true εάν τα οχήματα που πέρασαν είναι λιγότερα από αυτά που περιμένουν στα διόδια. Όταν την καλέσαμε απ'την Segment με τιμή 0 στην περίπτωση που δεν είχαμε κενές θέσεις για enter, τότε ξέραμε σίγουρα πως υπάρχουν οχήματα στην είσοδο, άρα θα είχαμε και σίγουρα την τιμή true.

~Η operate της Segment κλείνει με την κλήση της check_delays(). Η Segment::check_delays() έχει δύο ενδείξεις, μία για τις καθυστερήσεις γενικά (flag1) και μία για της καθυστερήσεις απ'τις pass (flag2). Η flag1 χρησιμοποιείται ως ένδειξη πως γενικά έχουμε καθυστερήσεις, ενώ η flag2 για να σταματήσει ο βρόχος στην δεύτερη συνθήκη. Στην πρώτη συνθήκη έχουμε ήδη έτοιμο το αποτέλεσμα απ'την Toll::remove_Vehicel(), οπότε απλά το καλούμε απ'την Entrance::get_delays(). Στην δεύτερη συνθήκη ελέγχουμε αν στο προηγούμενο τμήμα έχουν παραμείνει οχήματα σε κατάσταση ready και δεν έχουν ως προορισμό την έξοδο του προηγούμενου τμήματος. Όλα αυτά δεδομένου ότι δεν μιλάμε για το αρχικό τμήμα το οποίο δεν έχει προηγούμενο για να υφίσταται η pass. Αν καμία απ'τις δύο συνθήκες δεν μας δώσει την τιμή true, τότε εκτυπώνεται το τελευταίο μήνυμα.

Το πρόγραμμα ολοκληρώνεται με την κλήση των αντίστοιχων destructors.

Σχόλια:

Στο πρόγραμμα έχουμε αφήσει σε σχόλια αρκετές εκτυπώσεις, κυρίως constructors αλλά και άλλων λειτουργιών, τα οποία αποσχολιάζοντάς τα μπορεί ο χρήστης να δει την πλήρη ροή του προγράμματος. Σχολιάστηκαν ώστε το interface του terminal να χωρέσει όσο το δυνατό περισσότερους κύκλους προσομοίωσης και όχι μόνο…