ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ M/M/1 ΟΥΡΑΣ

Το πρόγραμμα ξεκινάει καθορίζοντας ότι η ουρά έχει μέγεθος 100 θέσεις. Στη συνέχεια καλεί τη συνάρτηση *main*, η οποία ανοίγει ένα αρχείο mm1.in για ανάγνωση των δεδομένων εισόδου και ένα αρχείο mm1.out για καταγραφή των δεδομένων εξόδου. Από το mm1.in λαμβάνουμε το μέσο χρόνο άφιξης (*mean_interarrival*), το μέσο χρόνο εξυπηρέτησης (*mean_service*) και τον αριθμό των απαιτούμενων εξυπηρετήσεων ή αλλιώς τον αριθμό των πελατών (*num_delays_required*).

Στη συνέχεια καλούμε τη συνάρτηση *initialize* για την αρχικοποίηση των μεταβλητών, που θα χρησιμοποιηθούν στη συνέχεια. Αυτή θέτει το χρόνο εκκίνησης του προγράμματος (*time*) ίσο με 0, την κατάσταση του διακομιστή (*server_status*) ως άεργη, τον αριθμό των εκχωρημένων στην ουρά αυτή τη στιγμή (*num_in_q*) ίση με 0, το χρόνο που διήρκησε το τελευταίο γεγονός (*time_last_event*) ίσο με 0, τον αριθμό των εξυπηρετημένων (*num_custs_delayed*) ίσο με 0, τη συνολική καθυστέρηση ή αλλιώς το συνολικό χρόνο εξυπηρέτησης (*total_of_delays*) ίση με 0.0, το εμβαδόν των εκχωρημένων στην ουρά (*area_num_in_q*) ίσο με 0.0, το εμβαδόν της κατάστασης του διακομιστή (*area_server_status*) ίσο με 0.0, το χρόνο για το επόμενο γεγονός άφιξης (*time_next_event[1]*) ίσο με το άθροισμα του τρέχοντος χρόνου (μηδέν ή κάποιος αριθμός πάρα πολύ κοντά στο μηδέν) και της συνάρτησης *expon* με παράμετρο το μέσο χρόνο άφιξης, δηλαδή *time_next_event[1] = time + expon(mean_interarrival)* και τέλος το χρόνο για το επόμενο γεγονός αναχώρησης (*time_next_event[2]*) ίσο με έναν πολύ μεγάλο αριθμό. Έτσι, ολοκληρώνεται η συνάρτηση *initialize* και επιστρέφουμε στη συνάρτηση *main*. Προτού, όμως, προχωρήσουμε στη συνάρτηση *main*, ας αναλύσουμε τη συνάρτηση *expon*.

Η συνάρτηση *expon* δέχεται ως παράμετρο το μέσο χρόνο άφιξης. Ύστερα από τη δημιουργία μιας τυχαίας τιμής από το 0 μέχρι το 1, εκχωρημένη σε μια μεταβλητή u, η συνάρτηση *expon* επιστρέφει την αρνητική τιμή του μέσου χρόνου αφίξεως, πολλαπλασιασμένη με το λογάριθμο της μεταβλητής *u*.

Μετά την ολοκλήρωση της συνάρτησης *initialize* μπαίνουμε σε μια λούπα που θα επαναλαμβάνεται όσο ο αριθμός των εξυπηρετημένων

(*num_custs_delayed*) είναι μικρότερος του αριθμού των απαιτούμενων εξυπηρετήσεων (*num_custs_required*). Με πιο απλά λόγια, όσο ο πελάτες που πρέπει να εξυπηρετηθούν είναι περισσότεροι από όσους έχουν εξυπηρετηθεί, τόσο θα επαναλαμβάνεται η λούπα.

Σε αυτή λοιπόν, αρχικά καλείται η συνάρτηση *timing*. Σε αυτή τη συνάρτηση θεωρούμε τον ελάχιστο χρόνο που χρειάζεται το επόμενο γεγονός για να ολοκληρωθεί (*min_time_next_event*) ίσο με ένα πολύ μεγάλο αριθμό. Θέτουμε ότι δεν υπάρχει επόμενο γεγονός για να συμβεί, οπότε εκχωρούμε την τιμή 0 στη μεταβλητή *next_event_type*. Εντοπίζουμε το είδος του επόμενου γεγονότος αν είναι άφιξη ή αναχώρηση από την ουρά. Αυτό γίνεται με μια λούπα που επαναλαμβάνεται δύο φορές. Την πρώτη φορά, εάν ο χρόνος που απαιτείται για την επόμενη άφιξη (*time_next_event[1]*) είναι μικρότερος του ελάχιστου χρόνου που χρειάζεται το επόμενο γεγονός για να ολοκληρωθεί (*min_time_next_event*), ο οποίος χρόνος υπενθυμίζεται ότι είναι ίσος με έναν πολύ μεγάλο αριθμό, δηλαδή εάν *time_next_event[1]* < *min_time_next_event*, τότε ο ελάχιστος χρόνος που χρειάζεται το επόμενο γεγονός για να ολοκληρωθεί (*min_time_next_event*) γίνεται ίσος με το χρόνος που χρειάζεται η επόμενη άφιξη, δηλαδή *min_time_next_event = time_next_event[1]*. Μετά σημειώνεται ότι το επόμενο γεγονός είναι άφιξη, εκχωρώντας την τιμή 1 στη μεταβλητή *next_event_type*. Τη δεύτερη φορά, εάν ο απαιτούμενος χρόνος για αναχώρηση (*time_next_event[2]*) είναι μικρότερος από τον ελάχιστο και απαιτούμενο χρόνο για το επόμενο γεγονός (*min_time_next_event*), δηλαδή εάν *time_next_event[2]* < *min_time_next_event)*, τότε ο ελάχιστος και απαιτούμενος χρόνος για το επόμενο γεγονός (*min_time_next_event*) γίνεται ίσος με τον απαιτούμενο χρόνο για την επόμενη αναχώρηση (*time_next_event[2]*). Μετά σημειώνεται ότι το επόμενο γεγονός είναι αναχώρηση, εκχωρώντας την τιμή 2 στη μεταβλητή *next_event_type*. Ελέγχουμε εάν υπάρχει επόμενο γεγονός (αναχώρηση ή άφιξη) διαφορετικά τερματίζεται το πρόγραμμα, διότι κάτι έγινε λάθος. Τέλος, η τωρινή χρονική στιγμή (*time*) είναι ίση με τον ελάχιστο χρόνο για το επόμενο γεγονός (*min_time_next_event*). Έτσι, τελειώνει η συνάρτηση *timing* και επιστρέφουμε στη συνάρτηση *main*.

Ακολούθως, καλείται η συνάρτηση *update_time_avg_stats*. Θέτουμε το χρονικό διάστημα από το τελευταίο γεγονός (*time_since_last_event*), ίσο με τη διαφορά του χρόνου που διήρκησε το τελευταίο γεγονός (*time_last_event*) από την τωρινή χρονική στιγμή (*time*), δηλαδή *time_since_last_event = time - time_last_event*. Θέτουμε το χρόνο που διήρκησε το τελευταίο γεγονός

(*time_last_event*) ίσο με την τωρινή χρονική στιγμή (*time*). Θέτουμε το εμβαδόν των εκχωρημένων στην ουρά (*area_num_in_q*) ίσο με το άθροισμα του τρέχοντος εμβαδού εκχωρημένων στην ουρά (*area_num_in_q*) και του γινομένου των εκχωρημένων στην ουρά αυτή τη στιγμή (*num_in_q*) επί το χρονικό διάστημα από το τελευταίο γεγονός (*time_since_last_event*), δηλαδή *area_num_in_q = area_num_in_q + num_in_q * time_since_last_event*.

Τέλος, θέτουμε το εμβαδόν της κατάστασης του διακομιστή (*area_server_status*) ίσο με το άθροισμα των τρέχοντος εμβαδού της κατάστασης του διακομιστή (*area_server_status*) και του γινομένου της κατάστασης του διακομιστή (*server_status*) επί το χρονικό διάστημα από το τελευταίο γεγονός (*time_since_last_event*), δηλαδή *area_server_status += server_status * time_since_last_event.*

Επιστρέφοντας στη συνάρτηση *main*, ελέγχουμε την τιμή της μεταβλητής *next_event_type* αν είναι ίση με 1 ή 2. Αν είναι 1, τότε συνεπάγεται ότι το επόμενο γεγονός είναι άφιξη στην ουρά, άρα καλούμε τη συνάρτηση *arrive*. Διαφορετικά, συνεπάγεται ότι το επόμενο γεγονός είναι αναχώρηση από την ουρά, άρα καλούμε τη συνάρτηση *depart*.

Στη συνάρτηση *arrive* αρχικά θέτουμε τον απαιτούμενο χρόνο για άφιξη (*time_next_event[1]*) ίσο με το άθροισμα του παρόντος χρόνου (*time*) και της συνάρτησης *expon* με παράμετρο το μέσο χρόνο άφιξης (*mean_interarrival*). Ελέγχουμε εάν ο διακομιστής είναι απασχολημένος. Εάν είναι, τότε προσαυξάνουμε κατά ένα τον αριθμό των εκχωρημένων στην ουρά (*num_in_q*) και ελέγχουμε εάν αυτός είναι μεγαλύτερος από τον αριθμό που χωράνε μέσα στην ουρά ή αλλιώς εάν έχουμε υπερχείλιση ουράς, δηλαδή εάν *num_in_q* > *Q_Limit*. Αν έχουμε υπερχείλιση σταματά το πρόγραμμα και αναγράφεται η χρονική στιγμή που ξεχείλισε η ουρά. Εάν δεν γίνει υπερχείλιση, δηλώνεται η χρονική στιγμή που έγινε η άφιξη για τον συγκεκριμένο εκχωρούμενο και ολοκληρώνεται η συνάρτηση. Ωστόσο, στην περίπτωση που ο διακομιστής δεν είναι απασχολημένος, θέτουμε την καθυστέρηση (*delay*) ίση με μηδέν, προσαυξάνουμε τον αριθμό των εξυπηρετούμενων (πελατών) (*num_custs_delayed*) και ορίζουμε την κατάσταση του διακομιστή, ως απασχολημένος (*server_status*). Τέλος, θέτουμε τον απαιτούμενο χρόνο για την επόμενη αναχώρηση (*time_next_event[2]*) ίσο με το άθροισμα του παρόντος χρόνου (*time*) και της συνάρτησης *expon* με παράμετρο το μέσο χρόνο εξυπηρέτησης (*mean_service*).

Αυτή είναι η λούπα, η οποία επαναλαμβάνεται όσο ο αριθμός των εξυπηρετημένων (*num_custs_delayed*) είναι μικρότερος του αριθμού των απαιτούμενων εξυπηρετήσεων (*num_custs_required*). Όταν βγούμε από αυτήν καλούμε τη συνάρτηση *report* που καταγράφει τη μέση καθυστέρηση στην ουρά ως το πηλίκο της διαίρεσης του συνολικού χρόνου καθυστέρησης (*total_of_delays*) δια του αριθμού των εξυπηρετημένων (*num_custs_required*). Καταγράφει, επιπλέον, το μέσο αριθμό εκχωρουμένων στην ουρά ως το πηλίκο της διαίρεσης του εμβαδού των εκχωρημένων στην ουρά (*area_num_in_q*) δια του παρόντος χρόνου (*time*). Επιπρόσθετα, καταγράφει τη χρησιμοποίηση του διακομιστή ως το πηλίκο της διαίρεσης του εμβαδού της κατάστασης του διακομιστή (*area_server_status*) δια του παρόντος χρόνου (*time*). Όλα αυτά καταγράφονται στο αρχείο mm1.out και το πρόγραμμα τερματίζει.