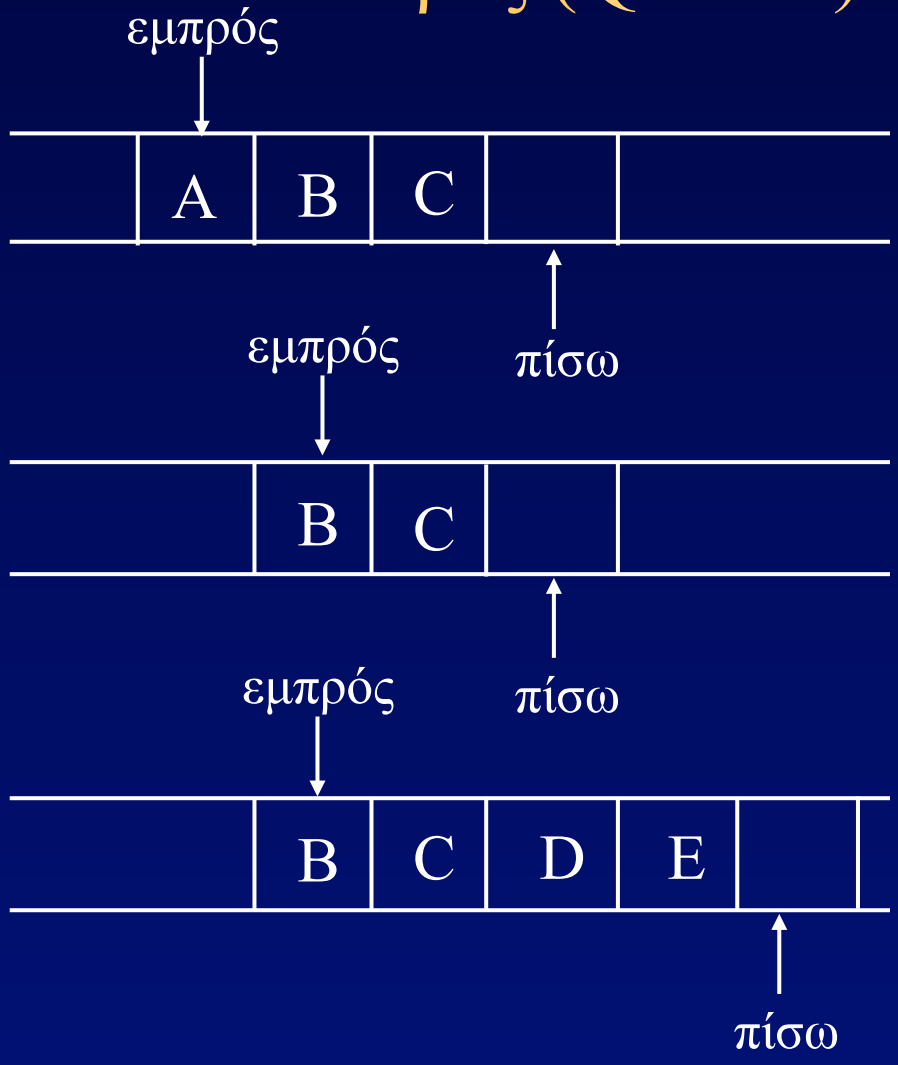
Ουρές (Queues)



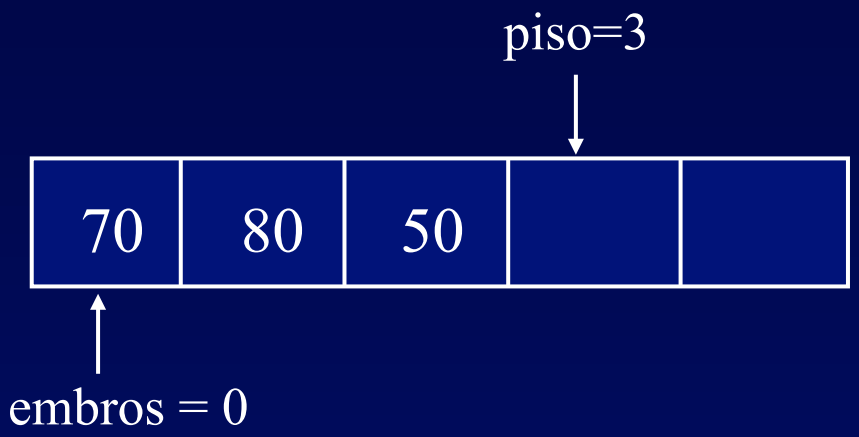
Ορισμός

Η ουρά είναι μια γραμμική λίστα στην οποία η διαγραφή ενός στοιχείου γίνεται στο ένα άκρο το οποίο καλείται **εμπρός** (front) και η εισαγωγή ενός στοιχείου γίνεται στο άλλο άκρο το οποίο καλείται **πίσω** (rear).

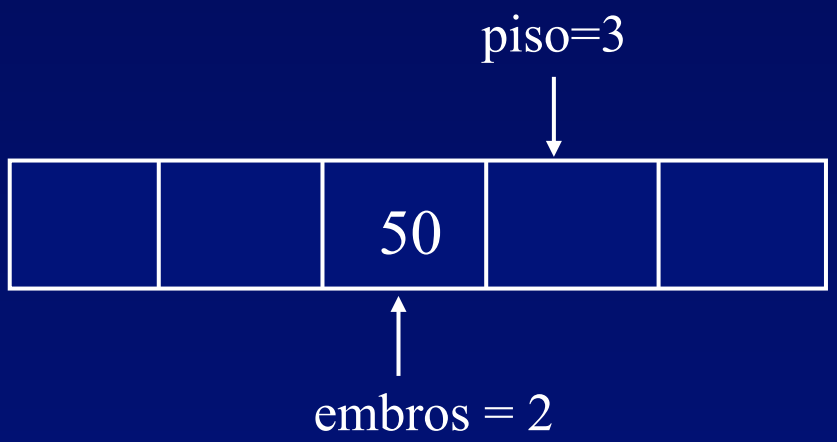
Βασικές πράξεις

Δημιουργία, Κενή, Πρόσθεση, Απομάκρυνση

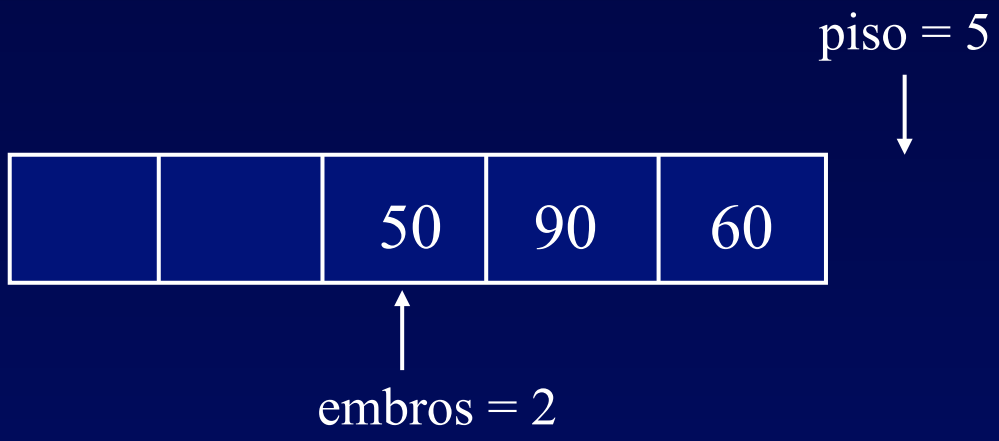
Υλοποίηση με πίνακα



Αν υποτεθεί ότι διαγράφονται δύο στοιχεία, τότε έχουμε :



Η πρόσθεση των 90 και 60 θα τροποποιήσει το σχήμα στο παρακάτω :

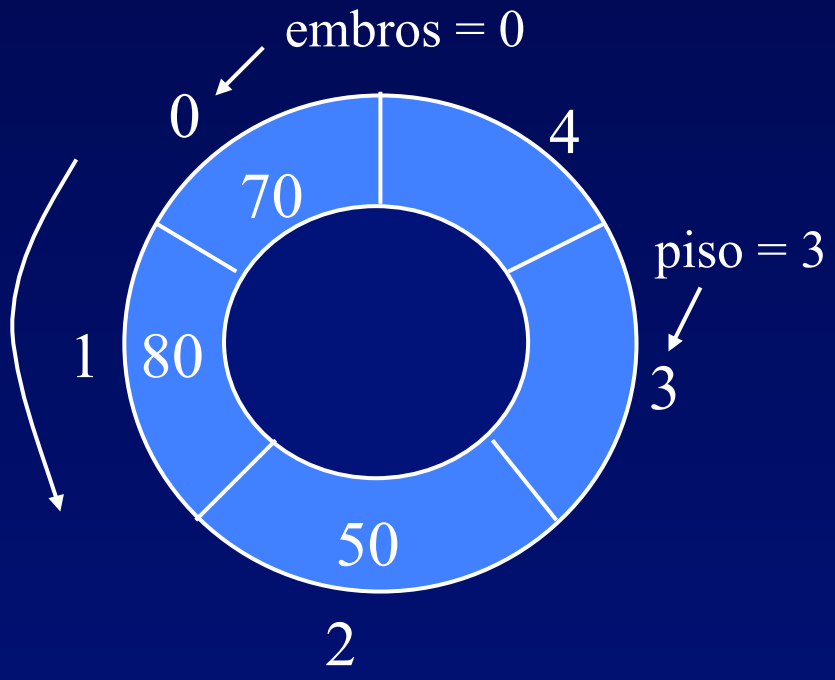


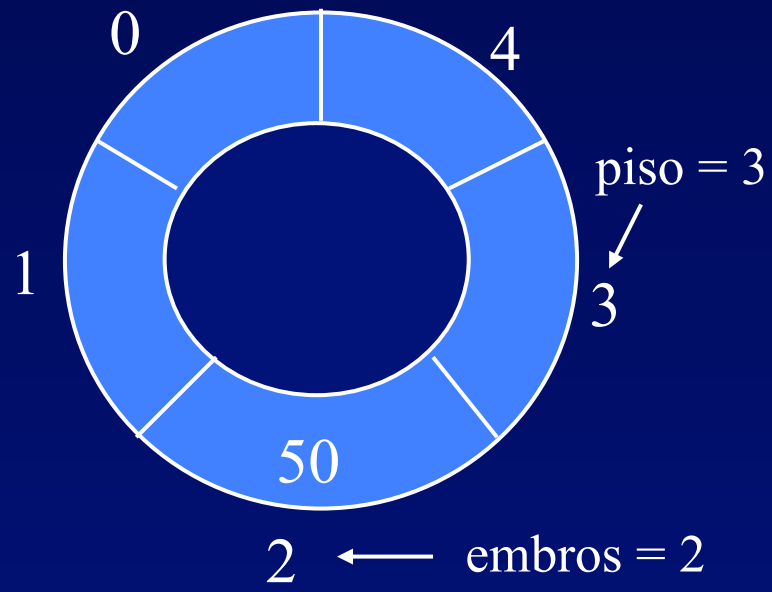
Πώς θα γίνει η πρόσθεση ενός επιπλέον στοιχείου;

O(n)

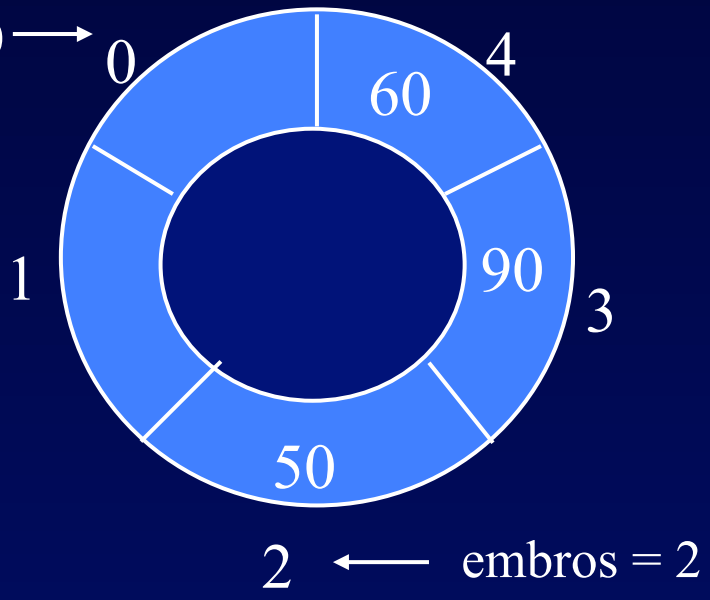
Η μετακίνηση των στοιχείων της ουράς μπορεί να αποφευχθεί αν φανταστούμε ότι ο πίνακας που φιλοξενεί την ουρά είναι **κυκλικός (circular)**, με το πρώτο στοιχείο του να έπεται του τελευταίου.

Διαγραφή των 70 και 80





Πρόσθεση των 90 και 60 :



Μετακίνηση του δείκτη piso :

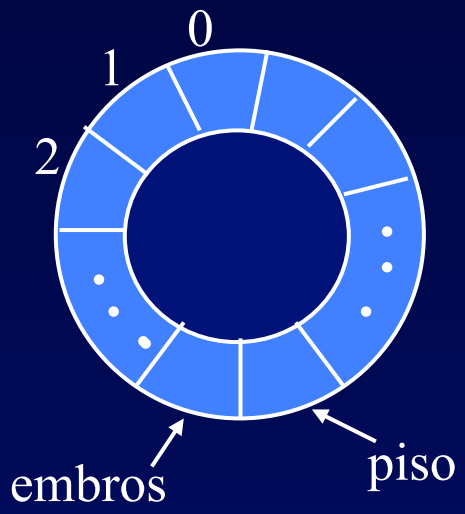
Αν piso == plithos-1 τότε
piso = 0
διαφορετικά
piso = piso+1

ή piso = (piso+1) % plithos

Μετακίνηση του δείκτη embros :

embros = (embros+1) % plithos

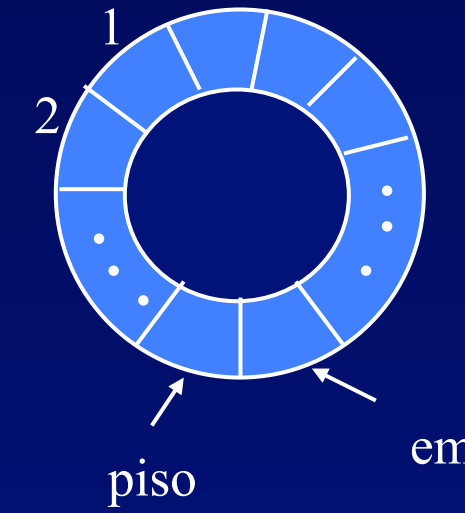
- Κενή Ουρά



τότε μετά τη διαγραφή του μοναδικού στοιχείου :

embros == piso

- Πλήρης Ουρά



Συνθήκη για την πλήρη ουρά :

(piso == plithos-1) AND (embros == 0)
OR
(embros == piso+1)

Εναλλακτικά:

(piso+1) % plithos == embros

Υλοποίηση

```
#define plithos ...
typedef  ...     typos_stoixeiou;
typedef  typos_stoixeiou     typos_pinaka[plithos];
typedef struct
        {
                int embros, piso;
                typos_pinaka  pinakas;
        } typos_ouras;

 typos_ouras    oura ;
```

Η υλοποίηση της δημιουργίας μιας κενής ουράς έχει σαν αποτέλεσμα οι δείκτες embros και piso να λάβουν την αρχική τιμή 0.

```
void dimiourgia(typos_ouras *oura)
/* Προ: Καμμία
   Μετά: Δημιουργία κενής ουράς.
*/
{
        oura->embros=0;
        oura->piso=0;
}
```

```
int keni(typos_ouras oura)
/* Μετά: Η συνάρτηση επιστρέφει 1 αν η oura είναι κενή
αλλιώς          επιστρέφει 0. */
{
        return ((oura.embros)==(oura.piso));
}
int gemati(typos_ouras oura)
/* Μετά: Αν η ουρά είναι γεμάτη τότε η συνάρτηση
επιστρέφει 1 αλλιώς επιστρέφει 0 */
{       int neo_piso = (oura.piso+1) % plithos;
        if (neo_piso == oura.embros)
            return 1;
         else
            return 0;
}
```

Η πρόσθεση ενός στοιχείου στην ουρά γίνεται στη θέση που δείχνει ο δείκτης piso, ο οποίος μετά την πρόσθεση προχωρά μια θέση προς την αντίθετη κατεύθυνση των δεικτών του ρολογιού.

```
void prosthesi(typos_ouras *oura, typos_stoixeiou stoixeio)
/* Προ : Δημιουργία ουράς. Η ουρά δεν είναι γεμάτη.
  Μετά: το stoixeio προστίθεται στο πίσω μέρος της ουράς */
{
        if (gemati(*oura))
            printf (" Η ουρά είναι γεμάτη");
        else
        {
              oura->pinakas[oura->piso]= stoixeio;
              oura->piso= (oura->piso+1) % plithos;
         }
}
```

Η απομάκρυνση ενός στοιχείου από την ουρά γίνεται από το εμπρός μέρος. Απομακρύνεται δηλαδή το στοιχείο που δείχνει ο δείκτης embros.

```
 void apomakrynsi(typos_ouras *oura,  typos_stoixeiou *stoixeio)
/* Προ : Η ουρά δεν είναι κενή.
 Μετά: Ανάκτηση και απομάκρυνση του πρώτου στοιχείου της ουράς. Η τιμή του καταχωρείται στο stoixeio. */
{       if (keni(*oura))
              printf(" Η ουρά είναι κενή");
         else
          {
              *stoixeio=oura->pinakas[oura->embros];
              oura->embros=(oura->embros+1) % plithos;
         }
}
```

Υλοποίηση ουράς με λογική μεταβλητή

Στην υλοποίηση αυτή χρησιμοποιείται μια λογική μεταβλητή προκειμένου να ελεγχθεί αν μια ουρά είναι κενή ή γεμάτη. Οι δηλώσεις είναι τώρα οι εξής:

```
#define plithos  …
typedef  ...  typos_stoixeiou;
typedef typos_stoixeiou     typos_pinaka[plithos];
typedef struct
{
        int embros, piso, adeia;
        typos_pinaka  pinakas;
} typos_ouras;
```

Λαμβάνοντας υπόψη τις παραπάνω δηλώσεις, οι υλοποιήσεις των βασικών πράξεων για τον ΑΤΔ ουρά τροποποιούνται ως εξής:

```
void dimiourgia(typos_ouras *oura)
/*Προ : Καμμία.
    Μετά: Η oura είναι μια κενή ουρά. */
 {
        oura->embros=0;
        oura->piso=0;
        oura->adeia=1;
  }
```

```
int keni(typos_ouras oura)
/* Μετά: Η συνάρτηση επιστρέφει 1 αν η oura είναι κενή
αλλιώς επιστρέφει 0. */
{
        return (oura.adeia);
}
```

```
int gemati(typos_ouras oura)
/* Προ : Δημιουργία ουράς.
   Μετά: Αν η ουρά είναι γεμάτη τότε επιστρέφει 1 διαφορετικά
   επιστρέφει 0 */
{
        return ((oura.embros==oura.piso) && !keni(oura));
}
```

```
void prosthesi(typos_ouras *oura, typos_stoixeiou stoixeio)
/* Προ : Δημιουργία ουράς.
    Μετά: Εισάγεται στο πίσω μέρος της ουράς η τιμή του stoixeio.
*/
{
        if (gemati(*oura))
                printf(" Η ουρά είναι γεμάτη");
        else
         {
                oura->pinakas[oura->piso]=stoixeio;
                oura->piso=(oura->piso+1) % plithos;
                oura->adeia=0;
          }
}
```

```
void apomakrynsi(typos_ouras *oura, typos_stoixeiou *stoixeio)
/* Προ : Δημιουργία ουράς.
  Μετά: Απομάκρυνση και ανάκτηση του πρώτου στοιχείου της
ουράς.*/
{
        if (keni(*oura))
              printf(" Κενή ουρά");
         else
         {
              *stoixeio=oura->pinakas[oura->embros];
               oura->embros=((oura->embros+1) % plithos);
               oura->adeia = (oura->embros==oura->piso);
        }
}
```

Υλοποίηση ουράς με μετρητή

Στην περίπτωση αυτή χρησιμοποιείται ένας μετρητής για τον υπολογισμό των θέσεων στην ουρά. Οι δηλώσεις για μια ουρά έχουν τη μορφή :

```
#define plithos ...
typedef  ...  typos_stoixeiou;
typedef typos_stoixeiou   typos_pinaka[plithos];
typedef struct
{
        typos_pinaka pinakas;
        int metritis, embros, piso;
} typos_ouras;
typos_ouras  oura;
```

Στη συνέχεια παρουσιάζονται οι υλοποιήσεις των βασικών πράξεων για μια ουρά:

```
void dimiourgia(typos_ouras *oura)
{
        oura->metritis=0;
        oura->embros=0;
        oura->piso=0;
}
```

```
int keni(typos_ouras oura)
/*Προ : Η ουρά έχει δημιουργηθεί.
 Μετά : Η συνάρτηση επιστρέφει 1 ή 0 ανάλογα αν η ουρά
          είναι κενή ή όχι*/

{
        return (oura.metritis == 0);
}
```

Επίσης είναι χρήσιμο να είναι γνωστό πότε η ουρά είναι γεμάτη.

```
int gemati(typos_ouras oura)
/*Προ : Η oura έχει δημιουργηθεί.
 Μετά: Η συνάρτηση επιστρέφει  1 ή 0 ανάλογα αν η oura
        είναι γεμάτη ή όχι*/

{
        return (oura.metritis == plithos);
}
```

```
void prosthesi(typos_ouras *oura, typos_stoixeiou stoixeio)
/*Προ : Η oura έχει δημιουργηθεί.
   Μετά : Το stoixeio έχει τοποθετηθεί στο πίσω μέρος της oura*/
{
        if (gemati(*oura))
              printf(" Η ουρά είναι γεμάτη");
         else
         {
               oura->metritis++;
               oura->pinakas[oura->piso]=stoixeio;
               oura->piso=(oura->piso+1) % plithos;
        }
}
```

```
void apomakrynsi(typos_ouras *oura, typos_stoixeiou
*stoixeio)
/*Προ : Η oura έχει δημιουργηθεί και δεν είναι κενή.
  Μετά : Το πρώτο στοιχείο της ουράς απομακρύνθηκε και η
  τιμή του καταχωρήθηκε στο stoixeio*/
{
        if (keni(*oura))
                printf(" Η ουρά είναι κενή");
        else
         {
                oura->metritis--;
                *stoixeio=oura->pinakas[oura->embros];
                 oura->embros=(oura->embros+1) % plithos;
        }
}
```

Εφαρμογή : Προσομοίωση ουράς αναμονής

Μελέτη συμπεριφοράς μιας ουράς σε ένα ταμείο

Δεδομένα προσομοίωσης

- Πιθανότητα άφιξης ενός πελάτη εντός ενός λεπτού
- Χρόνος (σταθερός) εξυπηρέτησης ενός πελάτη
- Συνολικός χρόνος προσομοίωσης

Παράσταση των αντικειμένων (δεδομένων). Ο ταμίας παριστάνεται με μια απλή μεταβλητή τύπου int, της οποίας η τιμή θα είναι ίση με το χρόνο εξυπηρέτησης όταν ένας πελάτης μετακινείται στο ταμείο και ελαττώνεται κατά 1 κάθε λεπτό που ο πελάτης παραμένει στο ταμείο. Όταν ο μετρητής αυτός λάβει την τιμή 0, ο ταμίας είναι ελεύθερος και θα πρέπει να έλθει στο ταμείο ο επόμενος πελάτης.
Αυτό που ενδιαφέρει να γνωρίζουμε για κάθε πελάτη είναι ο χρόνος αναμονής του στην ουρά.. Έτσι κάθε πελάτης μπορεί επίσης να παρασταθεί με μια απλή μεταβλητή τύπου int. Με την εισαγωγή του πελάτη στην ουρά ο μετρητής αυτός τίθεται ίσος με 0 και αυξάνεται κατά 1 για κάθε λεπτό παραμονής.
Τέλος, για τον υπολογισμό της μέσης τιμής του χρόνου αναμονής, θα πρέπει να είναι γνωστό το πλήθος των πελατών και ο συνολικός χρόνος αναμονής όλων των πελατών.

Το ακόλουθο πρόγραμμα εκτελεί την προσομοίωση της ουράς με ένα ταμείο

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/* Σε αυτό το σημείο ο χρήστης εισάγει τον ορισμό του τύπου
δεδομένων typos_ouras και τις αντίστοιχες συναρτήσεις */
main()
{
        typos_ouras oura; /* ουρά πελατών */
        float pithanotita_aphiksis; /* πιθανότητα άφιξης
                                       ενός πελάτη σε ένα λεπτό */
        int xronos_eksipiretisis; /* χρόνος για την εξυπηρέτηση
                                     ενός πελάτη */
        int xronos_prosomoiosis; /* συνολικός
                                    χρόνος προσομοίωσης */
```

συνέχεια

```
int xronos; /* το ρολόϊ της προσομοίωσης */
int enapomenon_xronos; /* χρόνος που
απομένει για το τέλος της εξυπηρέτησης  ενός πελάτη */

int arithmos_pelaton; /* πλήθος των
        πελατών που εξυπηρετήθηκαν */
int xronos_anamonis; /* συνολικός
                                χρόνος αναμονής */
int xronos_eisodou; /* η ώρα που
                                εισήλθε ο πελάτης στην ουρά */

float mesos_xronos; /* μέσος χρόνος αναμονής */

float random;
```

συνέχεια

```
scanf("%d %f %d",&xronos_prosomoiosis,
                 &pithanotita_aphiksis, &xronos_eksipiretisis);
printf("Η προσομοίωση θα διαρκέσει %4d λεπτά.\n",
                                        xronos_prosomoiosis);
 printf("Η πιθανότητα άφιξης ενός πελάτη σε ένα λεπτό
είναι");
 printf("%4.2f.\n",pithanotita_aphiksis);
 printf("Η διάρκεια εξυπηρέτησης ενός πελάτη είναι: ");
 printf("%d λεπτά.\n",xronos_eksipiretisis);
 fflush (stdin);
```

συνέχεια

```
dimiourgia (&oura);
xronos = 0;
enapomenon_xronos = 0;
arithmos_pelaton = 0;
xronos_anamonis = 0;
srand(time(NULL));

while (xronos < xronos_prosomoiosis)
    {
        random=( (float) rand() ) / (float) RAND_MAX ;
      if ( random < pithanotita_aphiksis )
            prosthesi(&oura, xronos);
```

συνέχεια

```
        if (enapomenon_xronos == 0)
        /* ελεύθερος ταμίας*/
            if (!keni(oura))      /* υπάρχει πελάτης */
                {
                apomakrynsi(&oura, &xronos_eisodou);
              xronos_anamonis += (xronos - xronos_eisodou);
               arithmos_pelaton++;
               enapomenon_xronos =xronos_eksipiretisis;
            }
        xronos++;
        if (enapomenon_xronos > 0)
              enapomenon_xronos--;
} /* while */
```

συνέχεια

```
if (arithmos_pelaton == 0)
        mesos_xronos = 0.0;
else
        mesos_xronos =((float)xronos_anamonis) /
((float)arithmos_pelaton);
        printf("Εξυπηρετήθησαν %1d πελάτες”,arithmos_pelaton);
        printf(“Ο μέσος χρόνος αναμονής ήταν %4.2f λεπτά.\n",
mesos_xronos);
}
```