ΣΥΜΒΟΛΟΣΕΙΡΕΣ (Strings)

Ο ΑΤΔ Συμβολοσειρά

Μία συμβολοσειρά είναι μία ακολουθία χαρακτήρων

Βασικές πράξεις :

1. Δημιουργία
2. Μήκος
3. Ανάκτηση
4. Προσάρτηση
5. Διαγραφή

6. Αντιγραφή
7. Συνένωση
8. Αναζήτηση
9. Εισαγωγή
10. Απομάκρυνση
11. Αντικατάσταση
12. Σύγκριση

Υλοποίηση του ΑΤΔ Συμβολοσειρά με πίνακα

```
#define megisto_mikos ...
typedef struct
{
        int mikos;
        char xaraktiras[megisto_mikos];
}  typos_seiras;
```

```
void dimiourgia(typos_seiras *seira)
/*  Δημιουργεί μια κενή συμβολοσειρά */
{
        seira->mikos=0;
}


int mikos(typos_seiras seira)
/*  Υπολογίζει το πλήθος των χαρακτήρων της seira */
{
        return seira.mikos;
}
```

```
char anaktisi(typos_seiras seira, int i)
 /*  Αν 0<= i <mikos(seira), δηλαδή η θέση i ανήκει στη
συμβολοσειρά τότε επιστρέφει 1 το χαρακτήρα της θέσης i.*/
{
        return(seira.xaraktiras[i]);
}
```

```
int prosartisi(typos_seiras *seira, char xar)
{
 /*Βάζει στο τέλος της συμβολοσειράς τον χαρακτήρα xar.*/

    if (mikos(*seira)==megisto_mikos)
    {
        printf("Η συμβολοσειρά είναι πλήρης" );
        return 0;
     }
     seira->xaraktiras[seira->mikos]=xar;
     seira->mikos++;
     return 1;
}
```

```
int diagrafi(typos_seiras *seira)
/* Προ : Δημιουργία της seira.
   Μετα : Επιστρέφει τη seira χωρίς τον πρώτο χαρακτήρα της */
{
        int i;
        if (mikos(*seira) ==0)/* κενή seira */
         {
              printf (" Διαγραφή από κενή seira ");
              return 0;
         }
        for (i=0; i < seira->mikos-1; i++)
                  seira->xaraktiras[i]=seira->xaraktiras[i+1];
        seira->mikos--;
        return 1;
}
```

- Ολες οι πράξεις εκτός της διαγραφής έχουν πολυπλοκότητα O(1), δηλαδή εκτελούνται σε σταθερό χρόνο ανεξάρτητα από το μέγεθος της συμβολοσειράς.

- Η πράξη της διαγραφής έχει O(n) χρόνο για συμβολοσειρές μήκους n, γιατί κάθε χαρακτήρας πρέπει να μετακινηθεί κατά μία θέση αριστερά προκειμένου να καλυφθεί το κενό που δημιουργείται με τη διαγραφή του πρώτου χαρακτήρα.

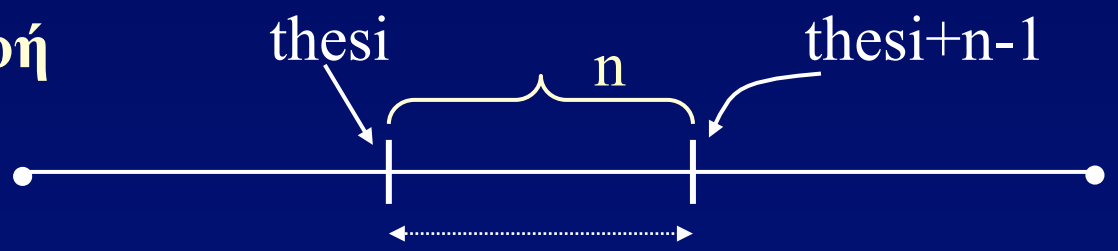
```
int antigrafi_seiras(typos_seiras arxiki, int thesi,
                     int n, typos_seiras *neaseira)

/* Προ :Δημιουργία συμβολοσειράς arxiki,  και thesi, n >= 0.
   Μετα : Επιστρέφει τη neaseira που περιέχει τους n χαρακτήρες της
          arxiki που αρχίζουν από τη thesi. */
{
 int i, teleytaios;
 teleytaios = thesi + n – 1;
 if (teleytaios  >= mikos(arxiki))
   {
        printf(" Δεν υπάρχουν οι χαρακτήρες που θέλετε να
                αντιγράψετε στην αρχική σειρά ");
        return 0;
   }
```

συνέχεια

```
else
{
        dimiourgia(neaseira);
        /* Αντιγραφή των n  χαρακτήρων από τη thesi της seira
            στη neaseira*/
         for (i=0; i<n; i++)
                  prosartisi(neaseira, anaktisi(arxiki, thesi+i));
         return 1;
    }
}
```

- **Αντιγραφή**

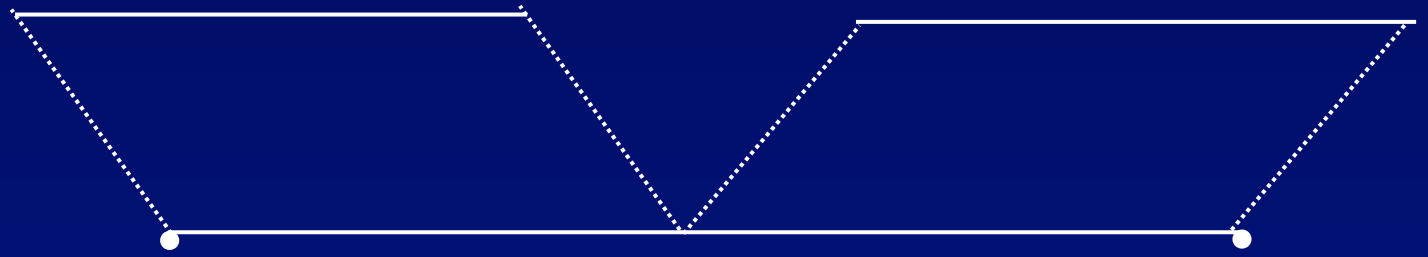


```
int synenosi_seiras(typos_seiras seira1, seira2, *neaseira)
/* Προ : Δημιουργία των συμβολοσειρών  seira1 και seira2.
   Μετα : Δημιουργεί μια νέα συμβολοσειρά από τη συνένωση των
      seira1 και seira2.*/
{
 int mks1, mks2, i, k;
 mks1=mikos(seira1);
 mks2=mikos(seira2);
 if (mks1+mks2 > megisto_mikos)
 {
        printf(" Οι συμβολοσειρές είναι πολύ μεγάλες και δεν
                μπορούν να συνενωθούν ");
        return 0;
 }
```

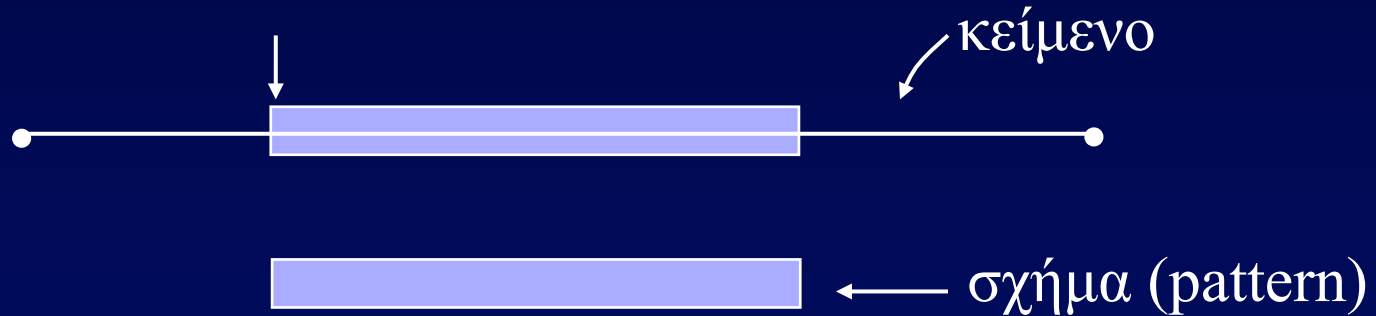
συνέχεια

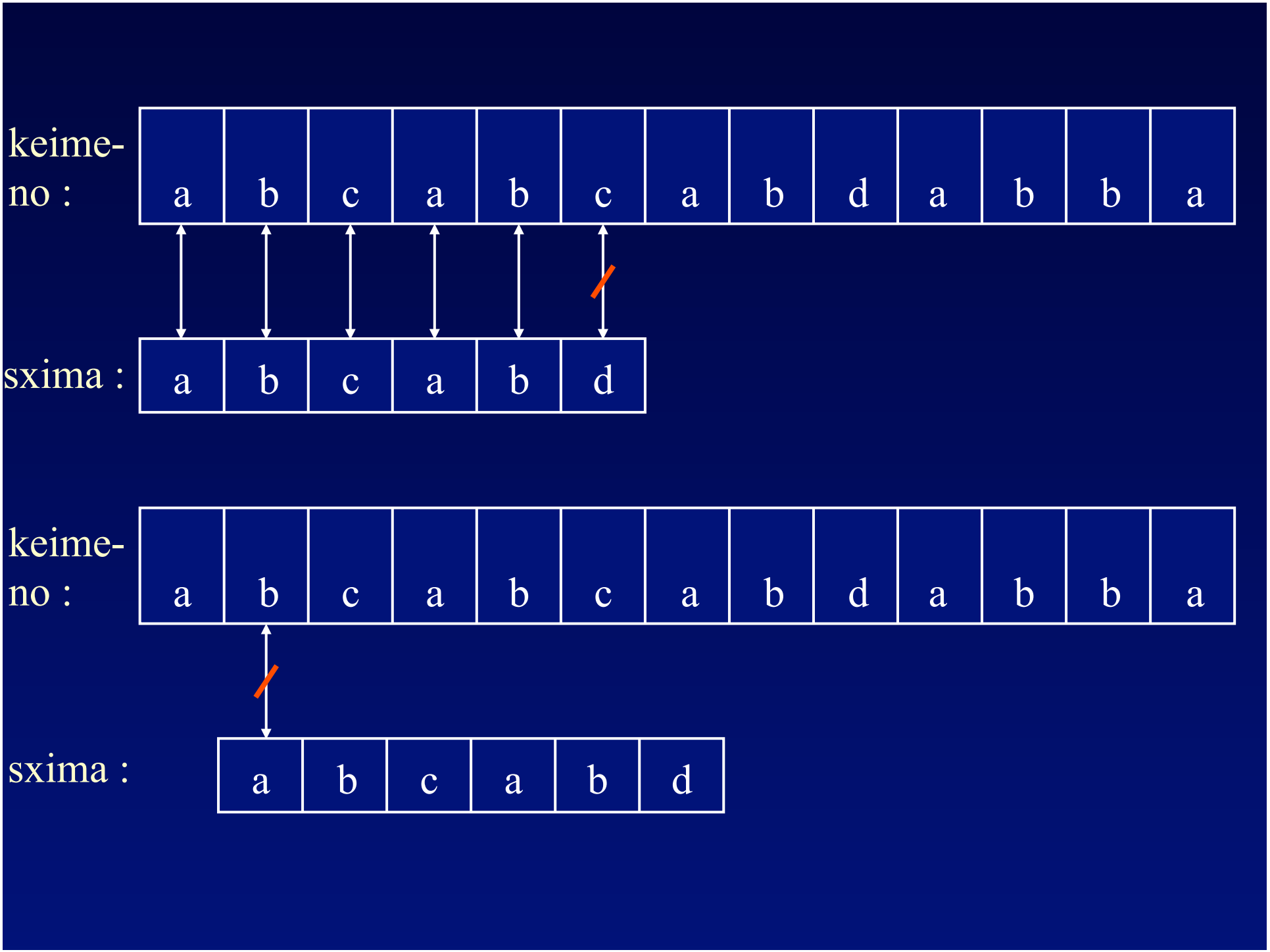
```
else
{
        /* Αντιγραφή της seira1 στη neaseira */
        k = antigrafi_seiras(seira1, 0, mks1, neaseira);
        if (k == 1)
        {
           /* Αντιγραφή της seira2 στη neaseira */
           for (i=0; i < mks2, i++)
                 k = prosartisi(neaseira, anaktisi(seira2, i));
        }
        return k;
   }
}
```

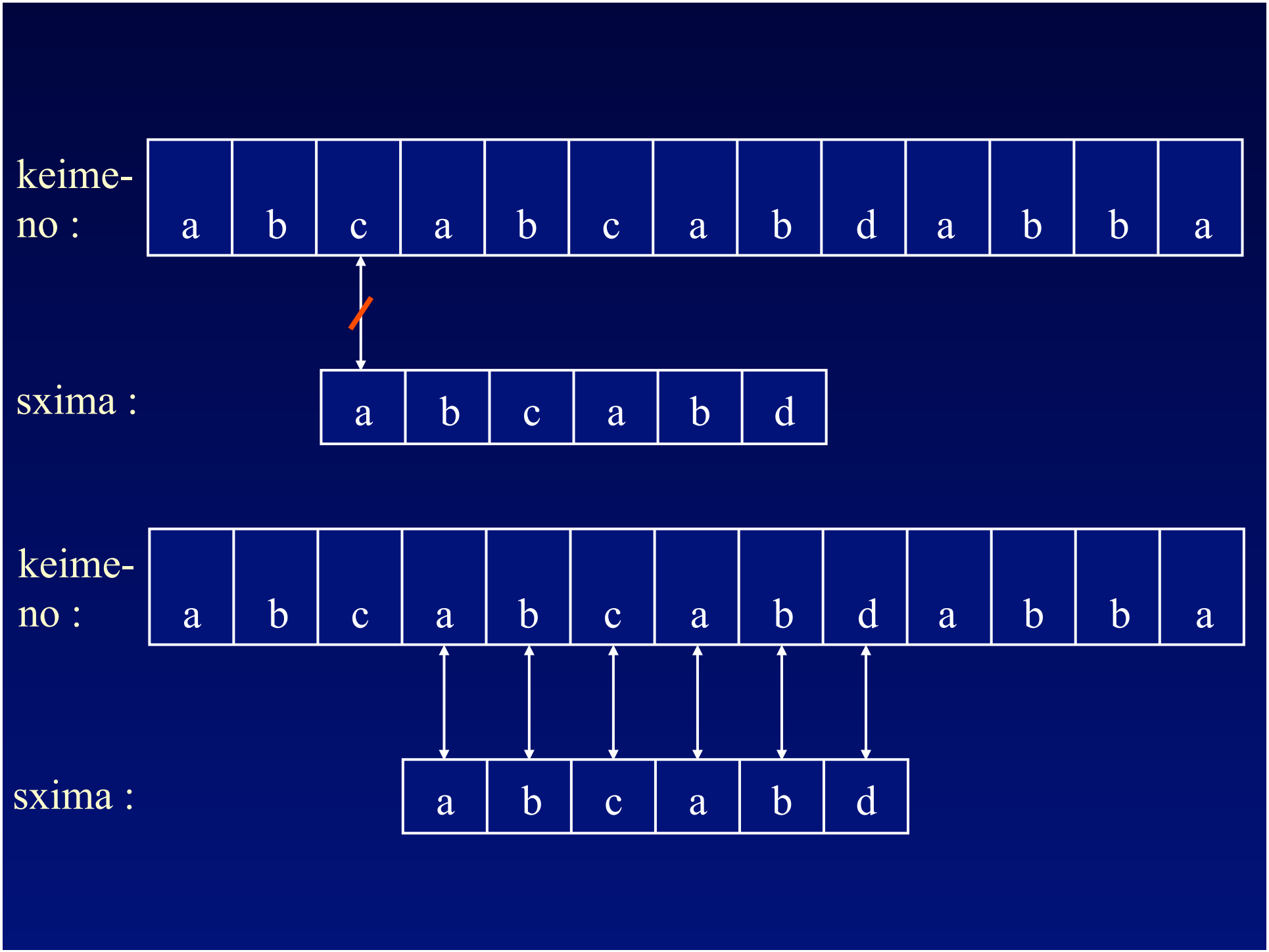
- **Συνένωση**



- **Αναζήτηση**




keime-
no :
a b c a b c a b d a b b a
sxima :
a b c a b d
keime-
no :
a b c a b c a b d a b b a
sxima :
a b c a b d

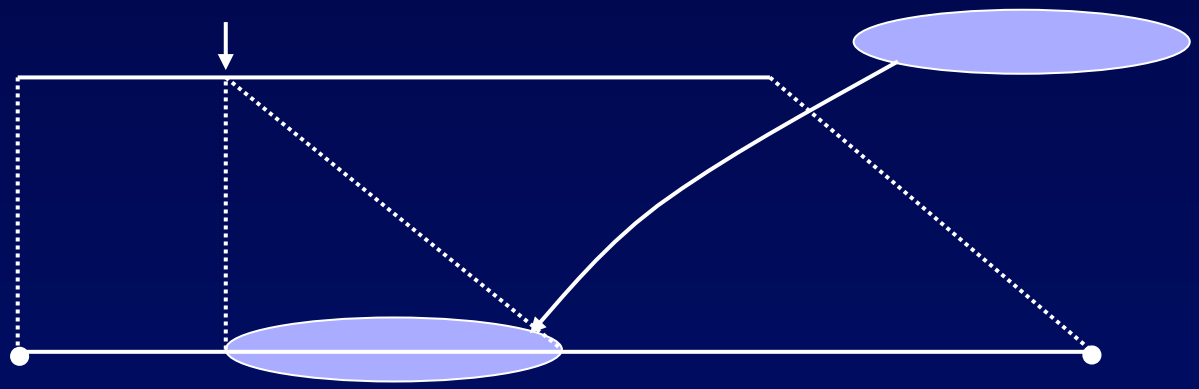

keime-
no :
a b c a b c a b d a b b a
sxima :
a b c a b d
keime-
no :
a b c a b c a b d a b b a
sxima :
a b c a b d

```
int anazitisi_seiras(typos_seiras sxima, keimeno)
/*Αναζητά τη πρώτη εμφάνιση της sxima μέσα στην keimeno*/
{
    int s, k, thesi, mkss, mksk;
    mkss=mikos(sxima);
    mksk=mikos(keimeno);
        if  (mkss==0)
            return 0;
        else
            if  (mkss > mksk)
                    return -1;
            else
            {
                    thesi=0; s=0; k=0;
```

συνέχεια

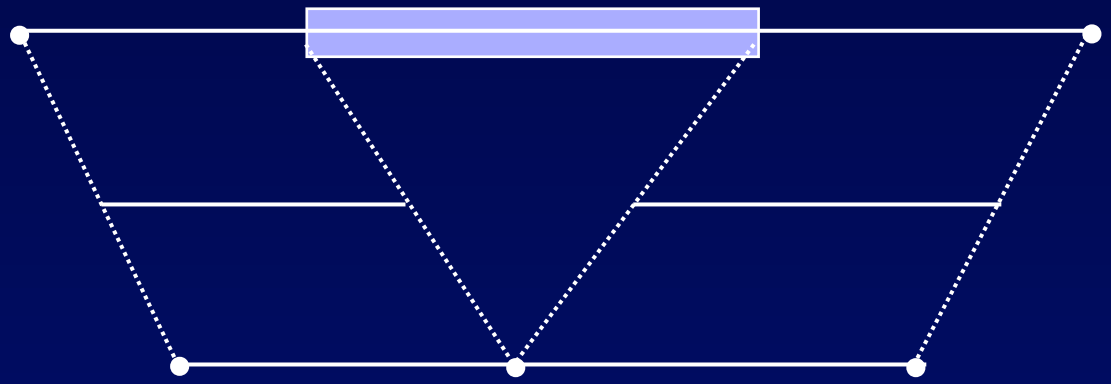
```
        while ((s<mkss) && (k<mksk))
            if  (anaktisi(sxima,s) == anaktisi(keimeno,k))
                    {s++; k++; }
            else
            {
                    thesi++;
                    k=thesi;
                    s=0;
            }
        if  (s>mkss-1)
            return thesi;
        else
            return -1;
    }
}
```

- **Εισαγωγή**



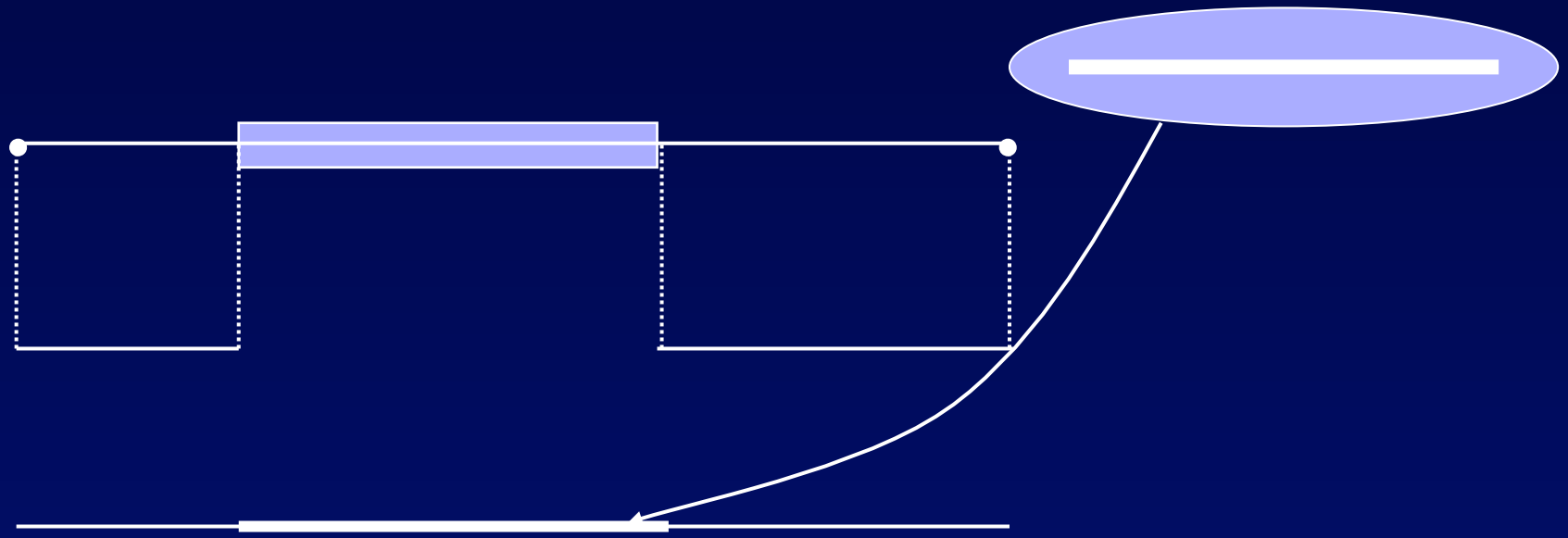
```
int eisagogi_seiras(typos_seiras *seira1, *seira2, int thesi)
/*Εισάγει τη seira1 μέσα στη seira2 στη θέση thesi*/
{
   typos_seiras proti, teleytaia;
   if ((0 <= thesi) && (thesi < mikos(*seira2)))
   {
        antigrafi_seiras(*seira2, 0, thesi, &proti);
        antigrafi_seiras(*seira2, thesi, mikos(*seira2) - thesi, &teleytaia);
        synenosi_seiras(proti, seira1, &proti);
        synenosi_seiras(proti, teleytaia, seira2);
        return 1;
   }
   else
        return 0;
}
```

- **Διαγραφή**



```
int diagrafi_seiras(typos_seiras *seira, int thesi, int mks)
/*Διαγράφει mks χαρακτήρες από τη συμβολοσειρά seira
ξεκινώντας από τη θέση thesi.*/
{
   typos_seiras proti,teleytaia;
   if ( mks > mikos(*seira) – thesi )
   {
      printf(“ Δεν είναι δυνατή η διαγραφή “);
      return 0;
   }
   antigrafi_seiras(*seira, 0, thesi, &proti);
   thesi+=mks;
   antigrafi_seiras(*seira, thesi, mikos(*seira)-thesi, &teleytaia);
   synenosi_seiras(proti, teleytaia, seira);
   return 1;
}
```

- **Αντικατάσταση**



```
int antikatastasi_seiras(typos_seiras seira1, seira2, *seira)
/*Αντικαθιστά την συμβολοσειρά seira1 με την συμβολοσειρά
        seira2 στην πρώτη εμφάνισή της στη συμβολοσειρά seira*/
{
    int thesi; int mks;
    thesi=anazitisi_seiras(seira1,*seira);
    if (thesi<0)
        return 0;
    else
        if ( mikos(*seira) - mikos(seira1) + mikos(seira2) > megisto_mikos)
        {
            printf (" Δεν είναι δυνατή η αντικατάσταση");
            return 0;
        }
    mks=mikos(seira1);
    diagrafi_seiras(seira, thesi, mks);
    eisagogi_seiras(seira2, seira, thesi);
    return 1;
}
```

```
char sygrisi(typos_seiras seira1, seira2)
/*Συγκρίνει τη seira1 και τη seira2. Επιστρέφει το χαρακτήρα '<' αν
seira1 < seira2, '=' αν seira1 = seira2, και '>' αν seira1 > seira2. Δύο
συμβολοσειρές είναι ίσες αν έχουν το ίδιο μήκος και όλοι οι χαρακτήρες
τους συμπίπτουν. Αν όλοι οι χαρακτήρες συμπίπτουν μέχρι το μήκος της
μικρότερης συμβολοσειράς, τότε αυτή είναι η 'μικρότερη' */
{
    int i,ises;
    i=0;
    ises=1;
    while ( ises && (i<mikos(seira1)) && (i<mikos(seira2)))
        if (anaktisi(seira1,i) == anaktisi(seira2,i))
            i++;
```

συνέχεια

```
        else
        {
            ises=0;
            if(anaktisi(seira1, i) < anaktisi(seira2, i))
                    return '<';
                    else      return '>';
        }
        if (ises)
            if (mikos(seira1) == mikos(seira2))
                    return '=';
            else
                    if (mikos(seira1) < mikos(seira2))
                        return '<';
                    else return '>';
}
```

```
void diavase_seira(typos_seiras *seira)
/*Διαβάζει χαρακτήρες και τους τοποθετεί στη seira*/
{
        char xar;
        dimiourgia(seira);
        xar = getchar( );
        while (xar != '\n' )
        {
                prosartisi(seira, xar);
                xar = getchar( );
        }
}
```

```
void typose_seira(typos_seiras seira)
/*Τυπώνει τους χαρακτήρες της seira*/
{
        int i, mks;
        mks = mikos(seira);
        for (i=0; i < mks; i++)
        putchar(anaktisi(seira, i));
}
```