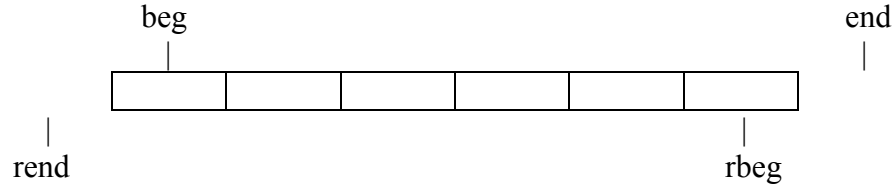
1. Να γράψετε ένα πρόγραμμα όπου
α) ορίσετε δυναμικα έναν πινακα ακεραιων με **n** στοιχεία,
β) ορισετε δυο δεικτες ο πρωτος με ονομα **beg** δειχνει στο πρωτο στοιχειο του πινακα ο δευτερος με ονομα **end** δειχνει στο επομενο στοιχειο μετα το τελευται στοιχειο του πινακα,
γ) ορισετε δυο δεικτες ο πρωτος με ονομα **rbeg** δειχνει στο τελευταιο στοιχειο του πινακα ο δευτερος με ονομα **rend** δειχνει στο προηγούμενο στοιχείο πριν το πρωτο στοιχειο του πινακα.
πχ. σε πινακα με 6 στοιχεια οι προαναφερομενοι δεικτες πρεπει να εχουν την εξής εικόνα.



Ορισετε δύο επαναλήπτες (δείκτες) με ονοματα **it** και **rit** ως εξής

```
typedef int* iter;
iter it;
iter rit;
```

Με την βοηθεια του επαναλήπτη **it** και των δεικτων **beg** και **end** (δεν αλλαζουν παραμένουν στις θεσεις τους) να εισαγεται τυχαιους αριθμους από το 1-100 στον πινακα. (Χρηση ψευδοκώδικα *K1*)

K1. for(it=arxi; it!=telos; ++it)
// eisagogi me xrisi tou it

Με την βοηθεια του επαναλήπτη **rit** και των δεικτων **rbeg** και **rend** (δεν αλλαζουν παραμένουν στις θεσεις τους) να εμφανίσεται τα στοιχεία του πινακα από το τέλος προς την αρχή. (Χρηση ψευδοκώδικα *K2*)

K2. for(rit=rarxi; rit!=rtelos; **--rit**)
// emfanisi me xrisi tou rit

Ελέγξετε με το παρακατω κωδικα

```
int a[] = { 3, 9, 34, 22, 62 };
int* beg = a;
int* end = a + 5;
int* rbeg = a + 4;
int* rend = a - 1;

typedef int* iter;
iter it; // Δήλωση επαναλήπτη it
iter rit;

for( it = beg; it != end; ++it)
     cout << *it << " ";
cout << endl;
```

2. Διδεται η ταξη

```
class iter {
    int* p;
```

```
    public:


class iter {
    int* p;
    public:
    // Δομητες
    iter() { p = 0; } ; // Εξ ορισμού δομητής

    // Υπερφορτωση τελεστών
    iter& operator=(int *x) { p = x; return *this; }
    iter& operator++() { ++p; return *this;}
    bool operator!=(const int* ptr) { return p!= ptr; }
    int& operator*() { return *p;}
};
```

Για την τάξη **iter** υλοποιησετε τους δομητές

```
    iter();
    iter(int* x);
    iter(const iter& mit);
```

και τις συναρτήσεις υπερφόρτωσης των τελεστών

```
    int& operator[](int i);
    bool operator==(const int* ptr);
    iter& operator++(int); // μεταθεματικός τελεστής ++
```

ώστε να είναι δυνατον να χρησιμοποιηθούν

```
    it[i],   it == end, it++
```

Μπορείτε να χρησιμοποιήσετε την **main** της άσκησης 1 για να ελέγξετε την ορθότητα των λειτουργειών (συναρτήσεις) της ταξης **iter** (διαγραψετε μόνο την γραμμή που υπαρχει η **typedef**). Πρεπει να παρετε το ιδιο αποτέλεσματα με αυτά της άσκησης 1.

3. Στον ψευδοκώδικα Κ2 (στην ασκηση 1) ο επαναλήπτης **rit** συμπεριφέρεται **σαν** ένας αντιστροφος επαναλήπτης. Για να γίνει ενας αντιστροφος επεναληπτης πρεπει να ορισθεί ως

```
    rev_iter rit; // ένα αντικείμενο (με ονομα πχ. rit)  τύπου rev_iter.
```

οπου **rev_iter** είναι μια ταξη σχεδον όμοια με την **iter** μονο που πρεπει να δουλεύει ο παρακάτω ψευδικώδικας Κ3 δηλ. πρέπει να δίνει το ίδιο αποτελεσμα με τον Κ2.

K3. for(rit=rarxi; rit!=rtelos; ++**rit**)
// emfanisi me xrisi tou rit

Αντιγραψετε την ταξη **iter** (και τις μεθόδους) και αλλαξετε το ονομα σε **rev_iter**. Κανετε τις απαραίτητες αλλαγές στις συναρτήσεις της τάξης ετσι ώστε αντικείμενα τυπου **rev_iter** να είναι αντιστροφοι επεναλήπτες.

```
    for( rit = rbeg; rit != rend; ++rit)
         cout << *rit << " ";
```