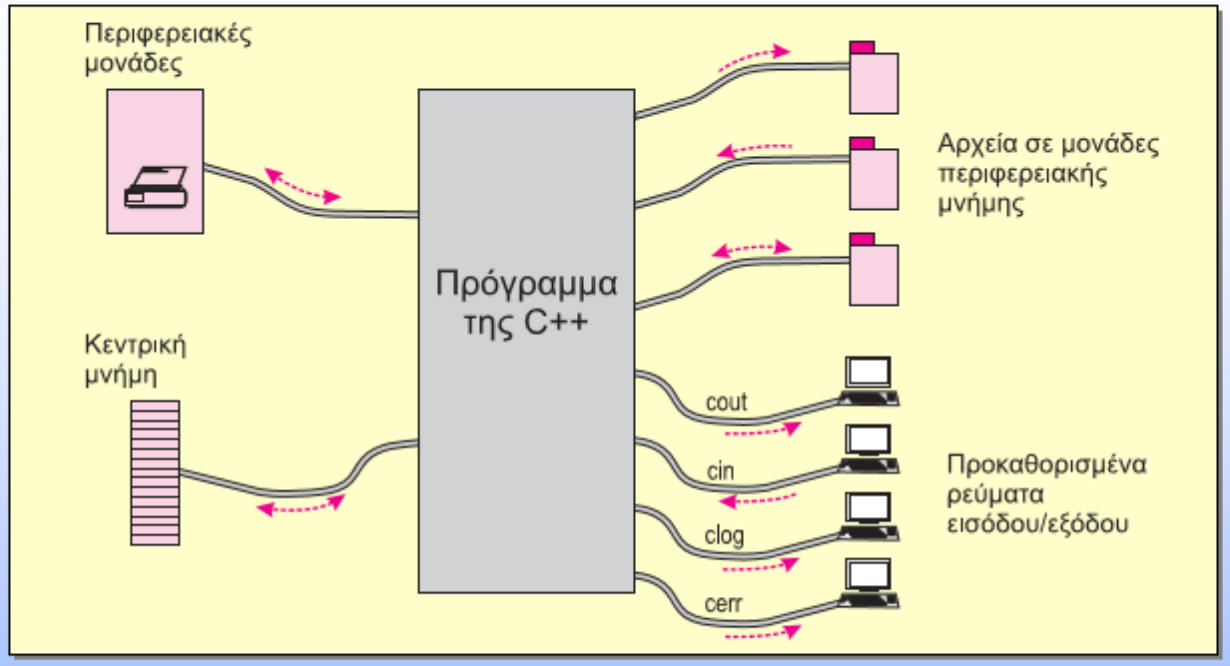
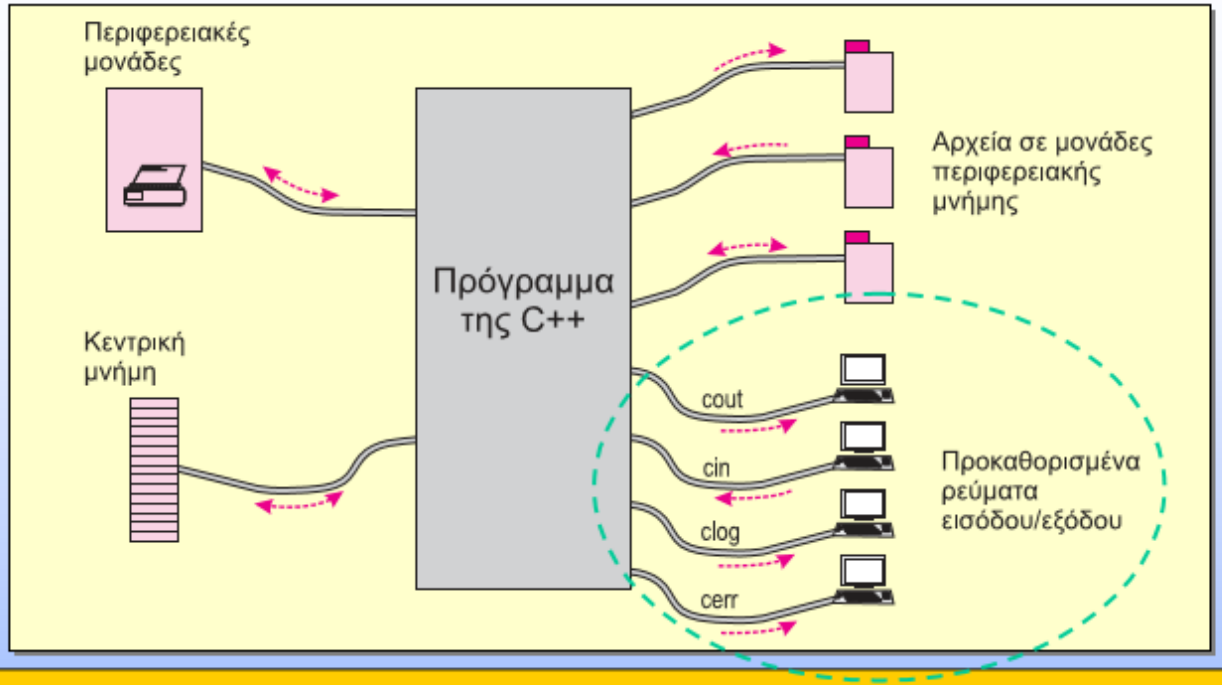
Ρεύματα εισόδου και εξόδου - Αρχεία

Ρεύματα εισόδου και εξόδου της C++ (input/output streams)



- Ένα ρεύμα εισόδου δέχεται bytes από μια περιφερειακή συσκευή, π.χ. από ένα αρχείο στο δίσκο, από το πληκτρολόγιο ή από άλλη συσκευή εισόδου.
- Ένα ρεύμα εξόδου στέλνει μια σειρά από bytes σε μια συσκευή εξόδου όπως στην οθόνη, σε ένα αρχείο, σε έναν εκτυπωτή κ.λπ.
- Υπάρχουν και ρεύματα εισόδου/εξόδου τα οποία μπορούν και να εισάγουν και να εξάγουν πληροφορίες από και προς τη συσκευή με την οποία είναι συνδεδεμένα (π.χ. αρχείο στο δίσκο, modem κ.λπ).
- Μπορούμε να φανταστούμε τα ρεύματα εισόδου και εξόδου της C++ σαν "κανάλια" μέσω των οποίων το πρόγραμμά μας ανταλλάσει πληροφορίες με τις περιφερειακές συσκευές του συστήματός μας.

Προκαθορισμένα αντικείμενα ρευμάτων της C++



Στη C υπάρχουν τέσσερα **προκαθορισμένα** ρεύματα (κανάλια) τα οποία συνδέουν τις τυπικές (standard) συσκευές ενός Η/Υ. Τα ρεύματα αυτά είναι **αντικείμενα** τα οποία ορίζονται στο **iosream** και είναι:

- **cout** Τυπική έξοδος (οθόνη)
- **cin** Τυπική είσοδος (πληκτρολόγιο)
- **cerr** Τυπική έξοδος για μηνύματα λάθους (οθόνη)
- **clog** Τυπική έξοδος για μηνύματα καταγραφής (οθόνη)

Πέρα από τα αντικείμενα-προκαθορισμένα ρεύματα εισόδου/εξόδου, στα προγράμματά μας μπορούμε να δημιουργήσουμε αντικείμενα-ρεύματα τα οποία μπορούν να συνδεθούν με οποιαδήποτε περιφερειακή συσκευή (π.χ αρχείο στο δίσκο μας, ο εκτυπωτής μας, ένα modem κ.λ.π)

Δυαδικά ρεύματα και ρεύματα χαρακτήρων

Η C++ διαθέτει δύο τρόπους **διαχείρισης** των ρευμάτων: Έναν **χαμηλού επιπέδου** (low-level) και έναν **υψηλού επιπέδου** (high level):

- Ο πρώτος, **χαμηλού επιπέδου**, τρόπος αντιμετωπίζει ένα ρεύμα σαν μια απλή **σειρά** από **bytes**, και η διαχείρισή του περιορίζεται στην εισαγωγή ή εξαγωγή συγκεκριμένου πλήθους bytes από ή προς το συγκεκριμένο ρεύμα.
- Ο δεύτερος, **υψηλού επιπέδου**, τρόπος, ο οποίος καλείται "μορφοποιημένη είσοδος/έξοδος" (formatted I/O), αντιμετωπίζει τα bytes ενός ρεύματος ως **χαρακτήρες**.

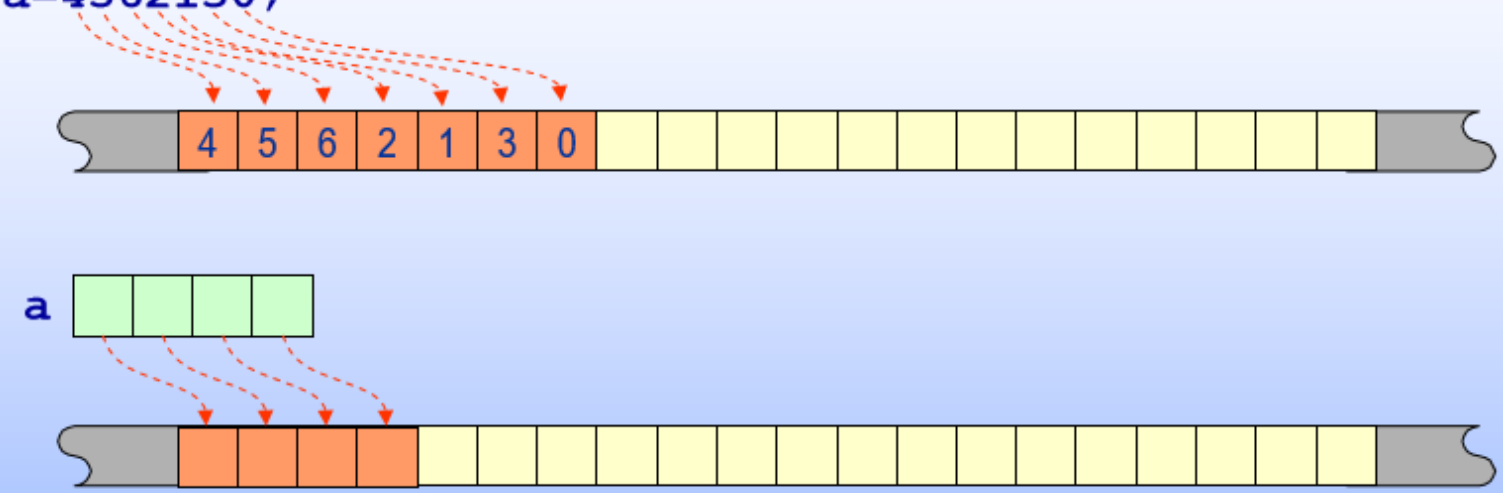
Τα ρεύματα "χαμηλού επιπέδου" αποκαλούνται **δυαδικά ρεύματα** (binary streams), ενώ τα "υψηλού επιπέδου" **ρεύματα χαρακτήρων** (text streams).
Τα **προκαθορισμένα** ρεύματα stdout, stdin, και stderr, είναι ρεύματα **χαρακτήρων**.

Εγγραφή/ανάγνωση σε αρχείο

Ένα **αρχείο** είναι μια σειρά από bytes. Η ανάγνωση ή εγγραφή ενός αρχείου μπορεί να γίνει **σειριακά** (ξεκινώντας από την αρχή ή από το τέλος του αρχείου), η με **τυχαία προσπέλαση** (σε οποιοδήποτε σημείο του αρχείου).
Ο **δείκτης θέσης** δείχνει το επόμενο byte το οποίο θα διαβαστεί ή θα εγγραφεί στο αρχείο.
Σε κάθε περίπτωση μετά από την ανάγνωση ή εγγραφή ενός αριθμού από bytes στο αρχείο, ο **δείκτης θέσης μετακινείται** ισάριθμες θέσεις **δεξιά**.

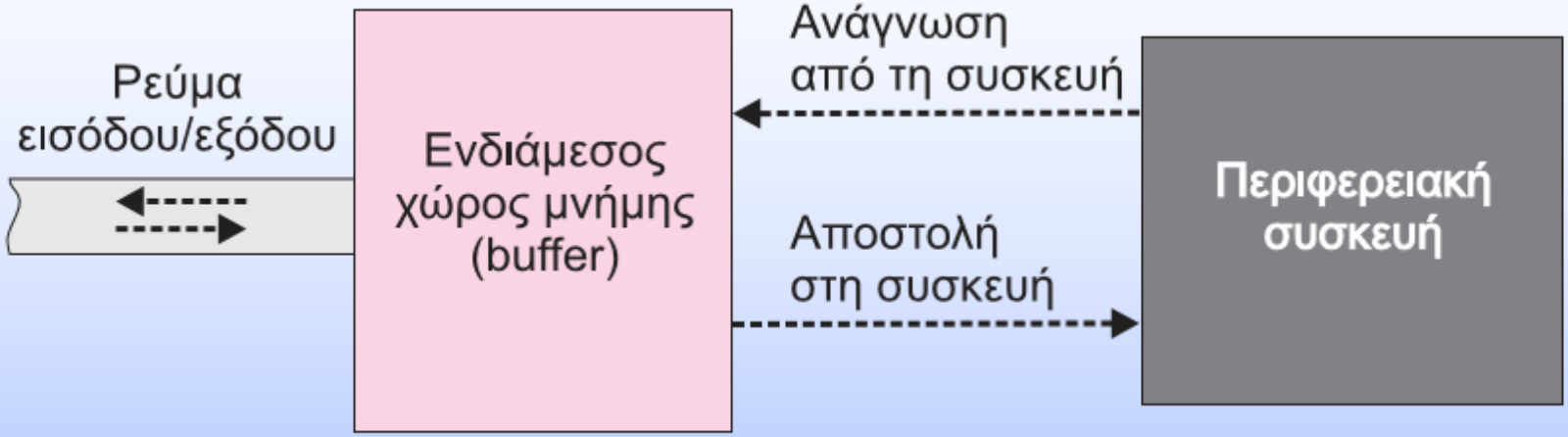
Αρχεία κειμένου και δυαδικά αρχεία

```
int a;
a=4562130;
```



- Σε ένα **αρχείο κειμένου** (text file) οι πληροφορίες εγγράφονται σε **μορφή χαρακτήρων**, όπως δηλαδή θα εμφανίζονταν στην οθόνη.
- Σε ένα **δυαδικό αρχείο** (binary file) εγγράφεται η εσωτερική απεικόνιση των πληροφοριών, δηλαδή τα bytes που συνθέτουν τη συγκεκριμένη πληροφορία (1 για char, 4 για int, 8 για double κ.λ.π).

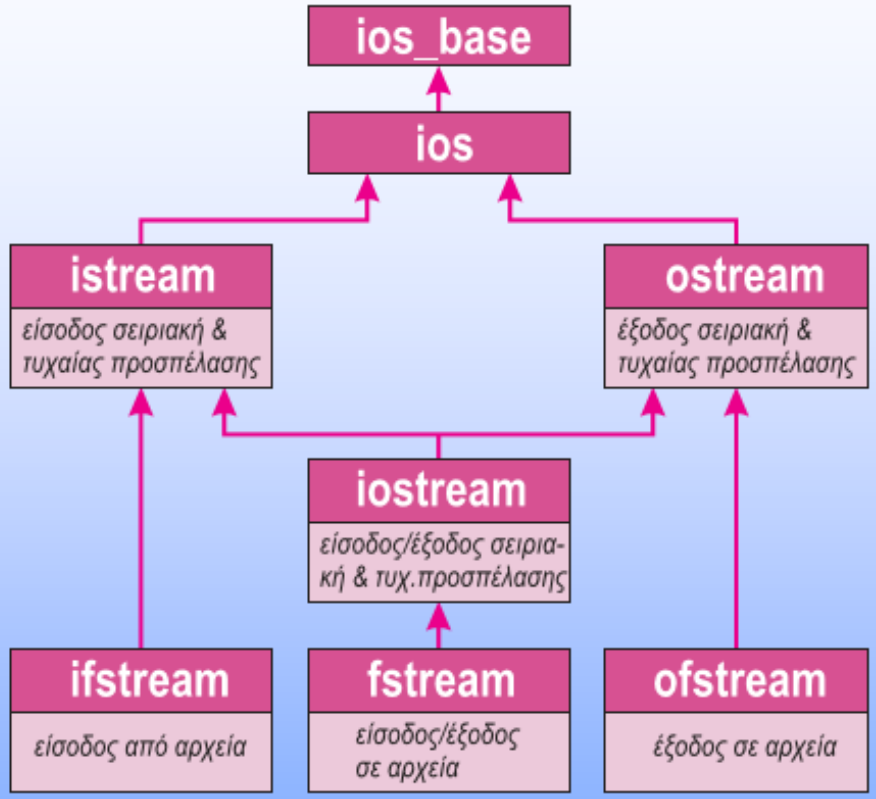
Περιοχή ενδιάμεσης αποθήκευσης (buffer)



Όταν γίνεται το διάβασμα ή η εγγραφή πληροφοριών από ή μέσα σε ένα αρχείο, χρησιμοποιείται πάντοτε μία περιοχή **ενδιάμεσης αποθήκευσης** (buffer).

- Για παράδειγμα, όταν γράφουμε πληροφορίες μέσα σε ένα αρχείο, αυτές καταχωρίζονται πρώτα στην περιοχή ενδιάμεσης αποθήκευσης και, όταν αυτή γεμίσει, τότε εγγράφονται πραγματικά στο αρχείο.
- Επίσης όταν διαβάζουμε πληροφορίες από ένα αρχείο συνήθως διαβάζονται περισσότερες από όσες ζητάμε και παραμένουν στην ενδιάμεση μνήμη σε περίπτωση μελλοντικής ζήτησης. Αυτό επιταχύνει τη διαδικασία ανάγνωσης από το αρχείο.

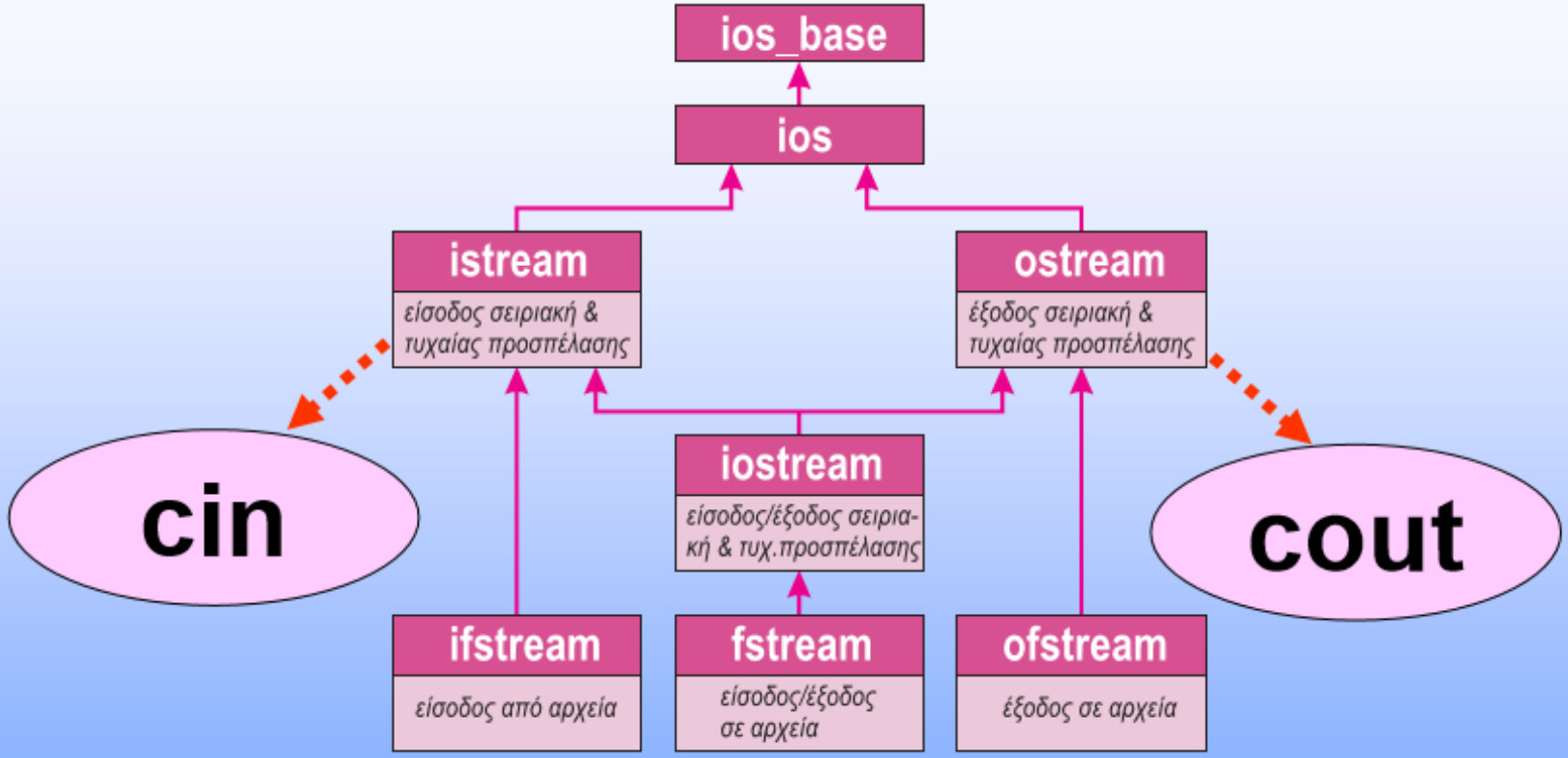
Κλάσεις ρευμάτων - Ιεραρχία κλάσεων



Στη C++ όλα τα ρεύματα αντιμετωπίζονται με τον ίδιο τρόπο, ανεξάρτητα με τη φυσική συσκευή του συστήματος με την οποία είναι συνδεδεμένα. Επομένως με τον ίδιο τρόπο με τον οποίο διαβάζουμε χαρακτήρες από ένα αρχείο, διαβάζουμε και από το πληκτρολόγιο ή από μια σειριακή θύρα.

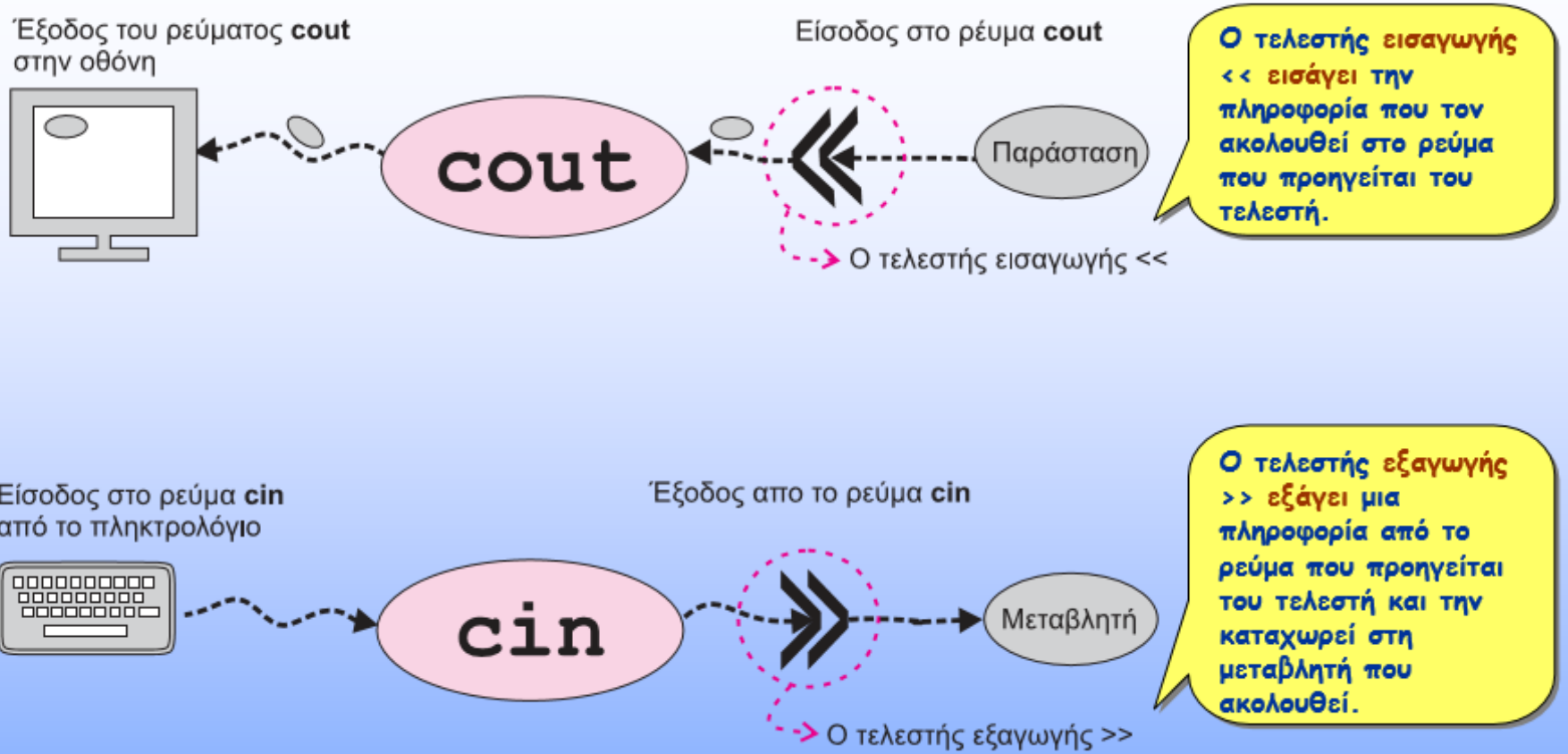
Στο σχήμα φαίνεται η ιεραρχία των κλάσεων που χρησιμοποιούνται για την διαχείριση των ρευμάτων. Στις βασικές κλάσεις **ios_base** και **ios** προσδιορίζεται το γενικό πλαίσιο χειρισμού των ρευμάτων. Από αυτές τις βασικές κλάσεις παράγονται όλες οι υπόλοιπες.

Κλάσεις και αντικείμενα ρευμάτων



Τα αντικείμενα **cout** και **cin** είναι αντικείμενα **ρευμάτων** των κλάσεων **istream** και **ostream** αντίστοιχα.

Τελεστές εισαγωγής και εξαγωγής



Ο τελεστής **εισαγωγής** << έχει **υπερφορτωθεί** για την κλάση **ostream** ώστε να εισάγει πληροφορίες σε ένα ρεύμα εξόδου. Ο τελεστής **εξαγωγής** >> έχει **υπερφορτωθεί** για την κλάση **istream** ώστε να εξάγει πληροφορίες από ένα ρεύμα εισόδου.

Μορφοποιημένη έξοδος/είσοδος ρευμάτων κειμένου

Τα ρεύματα κειμένου μπορούν να παραμετροποιηθούν με τρεις διαφορετικούς τρόπους. Η παραμετροποίηση ενός ρεύματος έχει ως αποτέλεσμα τη μορφοποιημένη έξοδο του ρεύματος ή ακόμη και τη μορφοποιημένη είσοδο στο ρεύμα. Η πλέον συχνή είναι η παραμετροποίηση ρευμάτων εξόδου η οποία επηρεάζει την εμφάνιση της εξόδου.

η μορφοποίηση ενός ρεύματος επιτυγχάνεται με ...

- Χειριστές
- Μεθόδους
- Σημαίες μορφοποίησης

Χειριστές μορφοποίησης ρευμάτων

Χειριστής	Εφαρμογή	I/O	Λειτουργία
endl	τώρα	O	Στέλνει ένα χαρακτήρα αλλαγής γραμμής ('\n') και εκκενώνει την ενδιάμεση μνήμη.
setw(n)	επόμενο	O	Καθορίζει το ελάχιστο πλάτος πεδίου που θα καταλαμβάνει στην οθόνη το επόμενο στοιχείο που θα σταλεί.
left	επόμενο	O	Αριστερή στοίχιση της εξόδου στο χώρο του πεδίου. Πρέπει να χρησιμοποιείται μετά από το χειριστή **setw(n)**.
right	επόμενο	O	Δεξιά στοίχιση της εξόδου στο χώρο του πεδίου. Πρέπει να χρησιμοποιείται μετά από το χειριστή **setw(n)**. Δεδομένου ότι η δεξιά στοίχιση είναι η προεπιλεγμένη, χρησιμεύει μόνο για να ακυρώσει μια αριστερή στοίχιση.
setfill(ch)	όλα	O	Χρήσιμος μόνο μετά από ένα **setw()**. Αν η τιμή δε χωράει ακριβώς στο πλάτος του πεδίου τότε ο χαρακτήρας ch χρησιμοποιείται για να συμπληρώσει τις υπόλοιπες θέσεις. Ο προεπιλεγμένος χαρακτήρας είναι ο χαρακτήρας του διαστήματος.
setprecision (n)	όλα	O	Καθορίζει το πλήθος των δεκαδικών ψηφίων μετά από την υποδιαστολή. Εφαρμόζεται σε όλους τους δεκαδικούς που θα σταλούν από εδώ και πέρα στο ρεύμα εξόδου.
fixed	όλα	O	Εμφανίζει τις δεκαδικές τιμές με τον **κλασικό τρόπο** (και όχι σε εκθετική μορφή). Αν δεν έχει καθοριστεί το πλήθος των δεκαδικών το θέτει ίσο με 6.
scientific	όλα	O	Εμφανίζει τις δεκαδικές τιμές σε **εκθετική μορφή**.
boolalpha noboolalpha	όλα	I/O	Λογικές τιμές μπορούν να εισάγονται, ή να εμφανίζονται, χρησιμοποιώντας τις λέξεις "true" και "false". Απενεργοποιείται με τον χειριστή **noboolalpha**.

Παράδειγμα χρήσεις χειριστών με το ρεύμα εξόδου cout

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    float a,b;
    string lex1,lex2;
    a=21.234;
    b=5467.1;
    cout << setw(10) << setprecision(3) << fixed << a << endl;
    cout << setw(10) << setprecision(3) << fixed << b << endl;
    lex1="Nikos";
    lex2="C++";
    cout << setw(10) << left << lex1 << endl;
    cout << setw(10) << right << lex2 << endl;
    return 0;
}
```

```
    21.234
  5467.100
Nikos
        C++
```

Στο παραπάνω πρόγραμμα, η χρήση των **χειριστών** μορφοποίησης επηρεάζει την έξοδο του αντικειμένου **cout** στην οθόνη.

Μέθοδοι μορφοποίησης ρευμάτων

Μέθοδος	Εφαρμογή	I/O	Λειτουργία
width()	τώρα	O	Επιστρέφει το τρέχον εύρος.
width(n)	επόμενο	O	Καθορίζει το νέο ελάχιστο πλάτος πεδίου που θα καταλαμβάνει στην οθόνη το επόμενο στοιχείο που θα σταλεί. Επιστρέφει το προηγούμενο πλάτος πεδίου.
fill()	τώρα	O	Επιστρέφει το χαρακτήρα πλήρωσης θέσης.
fill(ch)	όλα	O	Καθορίζει το νέο χαρακτήρα πλήρωσης θέσης. Επιστρέφει τον προηγούμενο χαρακτήρα.
precision()	τώρα	O	Επιστρέφει το πλήθος των δεκαδικών ψηφίων μετά από την υποδιαστολή.
precision(n)	όλα	O	Καθορίζει το πλήθος των δεκαδικών ψηφίων μετά από την υποδιαστολή.

Παράδειγμα χρήσεις μεθόδων με το ρεύμα εξόδου cout

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    double f = 3.14159123456;
    char frasi[40];
    cout << fixed;
    cout.precision(7);
    cout.width(15);
    cout.fill('#');
    cout << f << endl;
    cout.precision(3);
    cout << f << endl;
    return 0;
}
```

```
######3.1415912
3.142
```

Με αυτές τις τρεις μεθόδους, ορίζεται ότι η επόμενη έξοδος του ρεύματος **cout** θα εμφανιστεί με 7 δεκαδικά, σε χώρο 15 θέσεων, μα τη δίεση (#) ως χαρακτήρα πλήρωσης κενών θέσεων.

Στο παραπάνω πρόγραμμα, η χρήση των **μεθόδων** μορφοποίησης επηρεάζει την έξοδο του αντικειμένου **cout** στην οθόνη. Παρατηρούμε ότι η μέθοδος **precision()** εφαρμόζεται **μόνο** στην αμέσως επόμενη έξοδο.

Σημαίες μορφοποίησης ρευμάτων

Σημαία	Λειτουργία
boolalpha	Λογικές τιμές μπορούν να εισάγονται, ή να εμφανίζονται, χρησιμοποιώντας τις λέξεις "true" και "false".
showbase	Εμφανίζει τη βάση όλων των αριθμητικών τιμών.
showpoint	Εμφανίζει την υποδιαστολή και επιπλέον μηδενικά, ακόμα και όταν δε χρειάζονται.
showpos	Εμφανίζει το πρόσημο + πριν από θετικές αριθμητικές τιμές.
skipws	Αγνοεί κενά διαστήματα, χαρακτήρες tab και αλλαγής γραμμής όταν διαβάζει από ένα ρεύμα. Οι χαρακτήρες αυτοί ονομάζονται "Λευκά διαστήματα" και έτσι θα αναφέρονται στο εξής.
unitbuf	Αδειάζει την ενδιάμεση μνήμη μετά από κάθε εισαγωγή στο ρεύμα.
uppercase	Εμφανίζει το "e" της εκθετικής μορφής και το "x" της δεκαεξαδικής μορφής των αριθμών, με κεφαλαίους χαρακτήρες.

Παράδειγμα χρήσεις σημαιών μορφοποίησης

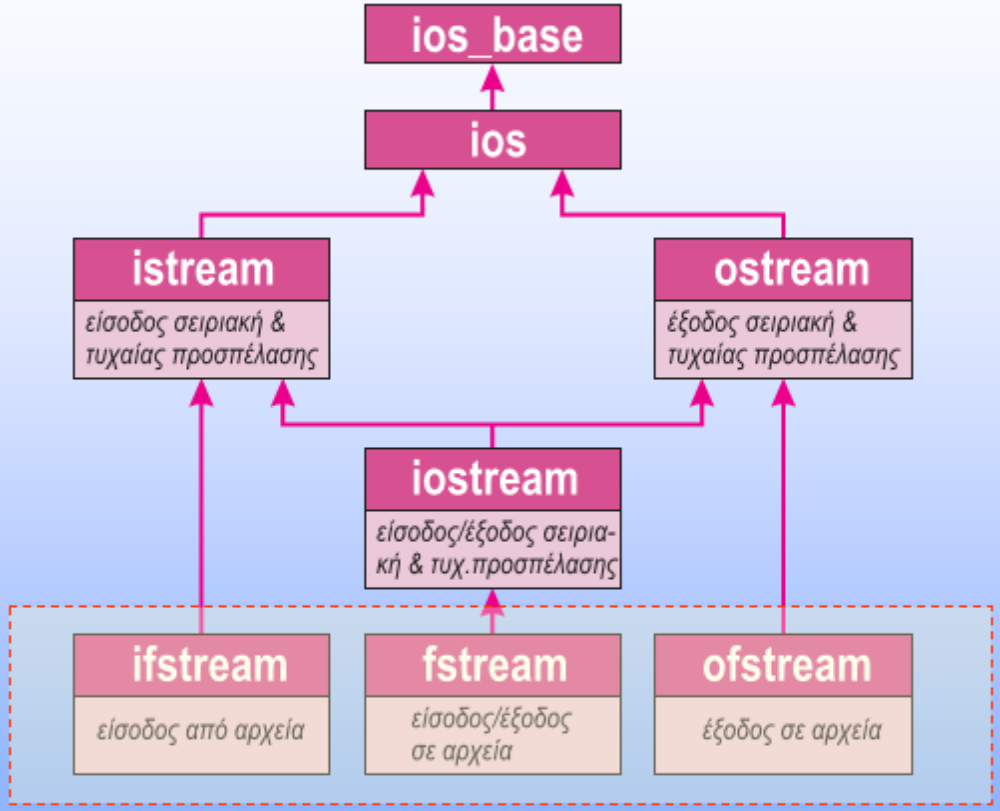
```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    bool t=true;
    cout.setf(ios::showpos);
    cout<<10<<endl;
    cout.unsetf(ios::showpos);
    cout<<10<<endl;
    cout<<t<<endl;
    cout.setf(ios::boolalpha);
    cout<<t<<endl;
    cin.setf(ios::boolalpha);
    cin>>t;
    cout<<t<<endl;
    return 0;
}
```

Δεδομένου ότι οι σημαίες μορφοποίησης ορίζονται στην κλάση **ios**, όταν αναφερόμαστε σε αυτές, πρέπει να χρησιμοποιούμε το προσδιοριστικό **ios** και τον τελεστή επίλυσης εμβέλειας ::. Π.χ. **ios::showpos**.

+10
10
1
true
false
false

Η ενεργοποίηση μιας σημαίας μορφοποίησης γίνεται με την εφαρμογή της μεθόδου **setf()** και η απενεργοποίηση της με την εφαρμογή της **unsetf()**. Δεν ξεχνάμε ότι οι δύο αυτές μέθοδοι εφαρμόζονται πάντα επάνω σε αντικείμενα ρευμάτων.

Κλάσεις ρευμάτων χειρισμού αρχείων

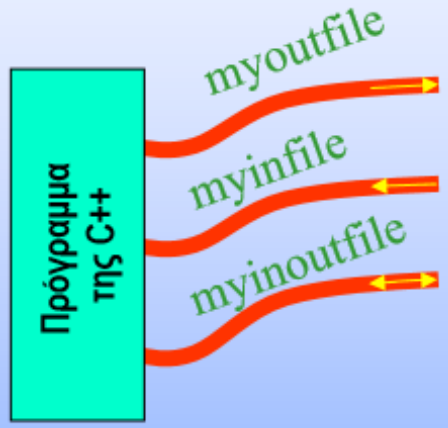


Στη C++ ο χειρισμός αρχείων γίνεται μέσω αντικειμένων-ρευμάτων των τριών κλάσεων **ifstream**, **fstream** και **ofstream**. Ανάλογα με το τι θέλουμε να κάνουμε χρησιμοποιούμε αντικείμενα της κατάλληλης κλάσης.

Δημιουργία αντικειμένων-ρευμάτων

Για να μπορέσουμε να διαχειριστούμε ένα αρχείο, η πρώτη μας ενέργεια είναι να δημιουργήσουμε ένα ρεύμα επικοινωνίας με σκοπό να συνδέσουμε αυτό το ρεύμα με κάποιο αρχείο του Η/Υ μας. Η δημιουργία ενός ρεύματος γίνεται απλά με τη δημιουργία ενός αντικειμένου μιας από τις τρεις κλάσεις που αναφέρονται στη προηγουμένη διαφάνεια.

```
ofstream myoutfile;

ifstream myinfile;

fstream myinoutfile;
```



Ο παραπάνω κώδικας δημιουργεί τρία αντικείμενα-ρευμάτων. Ένα για έξοδο σε αρχείο, ένα για είσοδο από αρχείο και ένα για είσοδο και έξοδο, με ονόματα **myoutfile**, **myinfile** και **myinoutfile** αντίστοιχα.
Τα ρεύματα αυτά **ΔΕΝ** έχουν συνδεθεί με κανένα αρχείο. Η σύνδεση ενός ρεύματος με ένα αρχείο γίνεται με τη μέθοδο **open()**.

Άνοιγμα αρχείου με τη μέθοδο open()

Η μέθοδος open() συνδέει ένα αντικείμενο-ρεύματος με ένα πραγματικό αρχείο στη περιφερειακή μνήμη του Η/Υ μας. Η μέθοδος εφαρμόζεται πάντα σε ένα υπάρχον αντικείμενο-ρεύματος

Το συντακτικό της **open()** είναι το εξής:

αντικείμενο_ρεύματος.open (όνομα_αρχείου, τρόπος);

Το **όνομα_αρχείου** προσδιορίζει το αρχείο που θα αντιστοιχιστεί με το συγκεκριμένο αντικείμενο ρεύματος, και η παράμετρος **τρόπος** καθορίζει τον τρόπο (mode) λειτουργίας του ρεύματος. Το **όνομα_αρχείου** μπορεί να είναι μια συμβολοσειρά, ένας πίνακας χαρακτήρων ή ένα αντικείμενο κλάσης **string**. Ο **τρόπος** μπορεί να είναι ένας συνδυασμός από τις παρακάτω σημαίες:

Τρόπος	Λειτουργία
ios::in	Ανοίγει το αρχείο για είσοδο (διάβασμα). Ο δείκτης θέσης του αρχείου τίθεται στην αρχή του αρχείου.
ios::out	Ανοίγει το αρχείο για έξοδο (εγγραφή). Ο δείκτης θέσης του αρχείου τίθεται στην αρχή του αρχείου.
ios::binary	Ανοίγει το αρχείο σε δυαδική μορφή (binary node).
ios::ate	Θέτει τον δείκτη θέσης στο τέλος του αρχείου.
ios::app	Προσθήκη στο τέλος του αρχείου. Χρησιμοποιείται μόνο σε ρεύματα εξόδου. Ο δείκτης θέσης του αρχείου τίθεται στο τέλος του αρχείου.
ios::trunc	Αδειάζει το αρχείο, το οποίο αποκτά μηδενικό μήκος.
ios::nocreate	Αν το αρχείο δεν υπάρχει, δεν επιτρέπεται να ανοιχτεί με τη μέθοδο open().
ios::noreplace	Αν το αρχείο υπάρχει, η προσπάθεια να ανοιχτεί με τη μέθοδο open() επιστρέφει τιμή λάθους.

Άνοιγμα αρχείου με τη μέθοδο open()

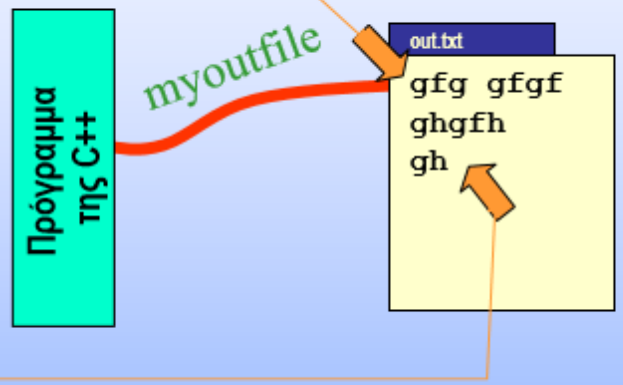
Η μέθοδος open() όταν εφαρμόζεται σε κάποιο αντικείμενο ρεύματος, χωρίς να αναφέρεται ο τρόπος ανοίγματος, ανοίγει το αρχείο με τον προκαθορισμένο τρόπο ανάλογα με τη κλάση του αντικειμένου-ρεύματος.

Ο δείκτης θέσης του αρχείου τοποθετείται στην αρχή του.
Οτιδήποτε εγγράφεται στο αρχείο θα αντικαθιστά τα υπάρχοντα.

```
ofstream myoutfile;
myoutfile.open("out.txt",ios::out);
```

```
ofstream myoutfile;
myoutfile.open("out.txt",ios::out|ios::app);
```

Ο δείκτης θέσης του αρχείου τοποθετείται στο τέλος του.
Οτιδήποτε εγγράφεται στο αρχείο θα προστίθεται στο τέλος του.



Οι σημαίες μπορούν να συνδυαστούν με τον bitwise τελεστή |. Για παράδειγμα **ios::out|ios::app** συνδυάζει τις δύο αυτές σημαίες με αποτέλεσμα να ανοίξει το αρχείο **out.txt** για έξοδο αλλά με προσθήκη από το τέλος του και μετά.

Προκαθορισμένοι τρόποι ανοίγματος της μεθόδου open()

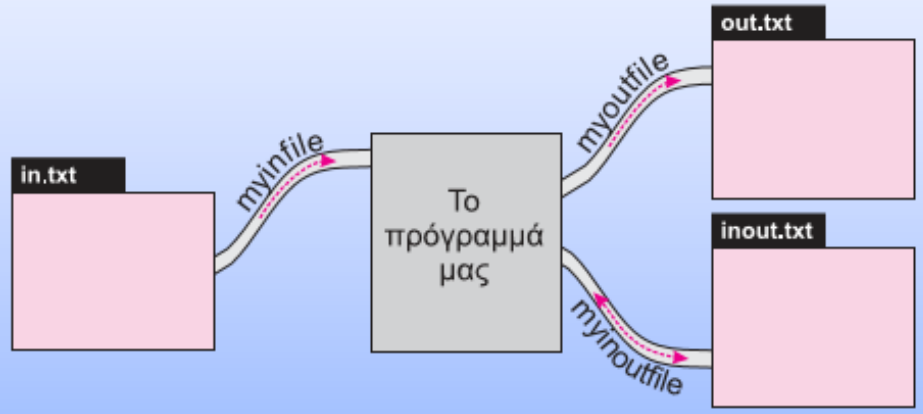
Όταν συνδέουμε ένα αρχείο, με ένα αντικείμενο μιας από τις κλάσεις ρευμάτων, με τη μέθοδο **open()**, ο καθορισμός της δεύτερης παραμέτρου **τρόπος**, είναι **προαιρετικός**. Στην περίπτωση που δεν καθορίζεται ο τρόπος λειτουργίας του ρεύματος, υπάρχει **προκαθορισμένος τρόπος**, ανάλογα με την κλάση του ρεύματος, ο οποίος αναφέρεται στον παρακάτω πίνακα:

Κλάση	Προκαθορισμένος τρόπος	Λειτουργία
ofstream	ios::out	Ανοίγει το αρχείο για είσοδο (διάβασμα). Ο δείκτης θέσης του αρχείου τίθεται στην αρχή του αρχείου.
ifstream	ios::in	Ανοίγει το αρχείο για έξοδο (εγγραφή). Ο δείκτης θέσης του αρχείου τίθεται στην αρχή του αρχείου.
fstream	ios::out | ios::in	Ανοίγει το αρχείο για είσοδο και έξοδο. Ο δείκτης θέσης του αρχείου τίθεται στην αρχή του αρχείου.

Άνοιγμα αρχείου με τη μέθοδο open()

Η μέθοδος open() όταν εφαρμόζεται σε κάποιο αντικείμενο ρεύματος, χωρίς να αναφέρεται ο τρόπος ανοίγματος, ανοίγει το αρχείο με τον προκαθορισμένο τρόπο ανάλογα με τη κλάση του αντικειμένου-ρεύματος.

```
ofstream myoutfile;
myoutfile.open("out.txt");

ifstream myinfile;
myinfile.open("in.txt");

fstream myinoutfile;
myinoutfile.open("inout.txt");
```



Ο παραπάνω κώδικας: 1) Δημιουργεί το αντικείμενο ρεύματος **myoutfile** κλάσης **ofstream** και το συνδέει με το αρχείο **out.txt** για έξοδο (ios::out). 2) Δημιουργεί το αντικείμενο ρεύματος **myinfile**, κλάσης **ifstream** και το συνδέει με το αρχείο **in.txt** για είσοδο (ios::in). 3) Δημιουργεί το αντικείμενο ρεύματος **myinoutfile**, κλάσης **fstream** και το συνδέει με το αρχείο **inout.txt** για είσοδο και έξοδο (ios::in|ios::out).

Κλείσιμο αρχείου με τη μέθοδο close()

Όταν η μέθοδος **close()** εφαρμοστεί σε ένα αντικείμενο ρεύματος αρχείου, **κλείνει** το συγκεκριμένο αρχείο. Η σύνταξη της μεθόδου **close()** είναι η ακόλουθη:

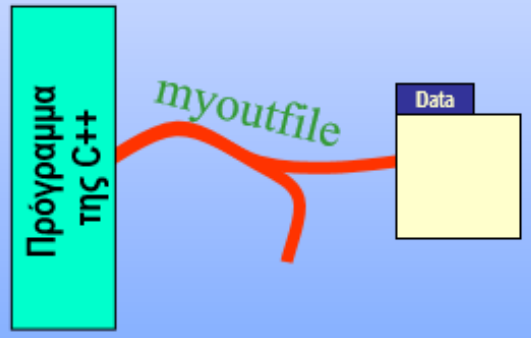
αντικείμενο_ρεύματος.close();

```
ofstream myoutfile;
myoutfile.open("Data");
..........
myoutfile.close();
```

Δημιουργεί ένα αντικείμενο ρεύματος για έξοδο σε αρχεία. Πρακτικά δημιουργεί ένα **ρεύμα** επικοινωνίας **myoutfile** **χωρίς** όμως να το **συνδέει πουθενά**!

Η μέθοδος **open()** συνδέει στο ρεύμα **myoutfile** ένα αρχείο κειμένου με όνομα **Data** για εγγραφή δεδομένων. Η διαδικασία αυτή λέγεται **άνοιγμα αρχείου**.

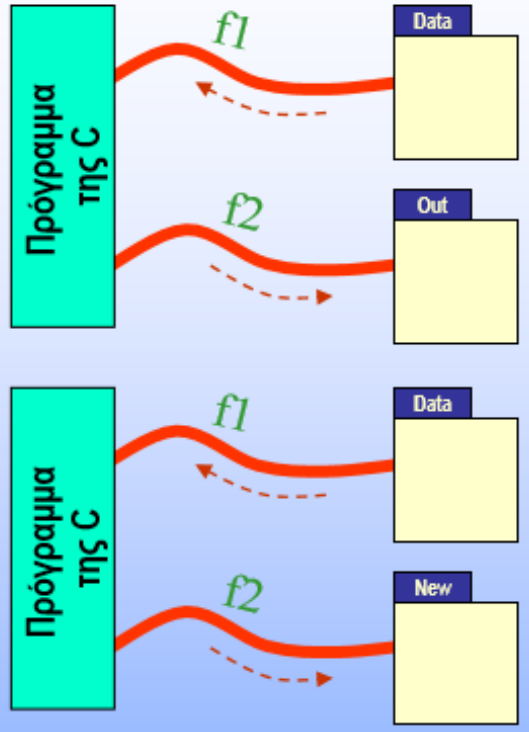
Κλείνει το αρχείο που ήταν συνδεδεμένο με το αντικείμενο ρεύματος **myoutfile**



Η μέθοδος **close()** γράφει στο αρχείο και όλα τα δεδομένα που βρίσκονται στην περιοχή **ενδιάμεσης** αποθήκευσης (buffer) πριν κλείσει το αρχείο.

Άνοιγμα περισσότερων αρχείων

```
....
ifstream f1;
ofstream f2;
f1.open("Data");
f2.open("Out");
....

....
f2.close();
f2.open("New");
....
```



Στο συγκεκριμένο παράδειγμα, ανοίγονται δύο αρχεία (το Data και το Out) και **συνδέονται** αντίστοιχα με τα ρεύματα **f1** και **f2**. Από εδώ και πέρα ο **χειρισμός** των δύο αρχείων **γίνεται μέσω** των **αντικειμένων-ρευμάτων** f1 και f2.
Ακόλουθα, με τη μέθοδο close(), **κλείνουμε** το αρχείο Out και **αποσυνδέεται** το ρεύμα **f2** και αμέσως μετά το **ίδιο** ρεύμα **f2 συνδέεται** με το αρχείο **New**.

Μορφοποιημένη είσοδος/έξοδος σε αρχεία κειμένου

Η μορφοποιημένη είσοδος/έξοδος σε αρχεία κειμένου (text files) γίνεται ακριβώς με τον ίδιο τρόπο όπως με τα προκαθορισμένα αντικείμενα ρευμάτων **cin** και **cout**, με χρήση των **τελεστών εισαγωγής** (<<) και **εξαγωγής** (>>). Η διαφορά είναι ότι, αντί να χρησιμοποιήσουμε τα προκαθορισμένα αντικείμενα ρευμάτων, θα χρησιμοποιήσουμε αντικείμενα ρευμάτων μιας από τις κλάσεις **ofstream**, **istream** και **fstream**, τα οποία προηγουμένως έχουν συνδεθεί με κάποιο αρχείο.

```
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;
int main()
{
    int i=10,a=20;
    ofstream myfile;
    myfile.open("data");
    myfile << i << " " << a << endl;
    myfile << 30 << endl <<"telos"<<endl;
    myfile.close();
    return 0;
}
```

Το πρόγραμμα ανοίγει ένα αρχείο εξόδου με όνομα **data**, στο οποίο εγγράφει τους αριθμούς 10, 20, 30 καθώς και τη συμβολοσειρά "telos". Ακόλουθα κλείνει το αρχείο!

data

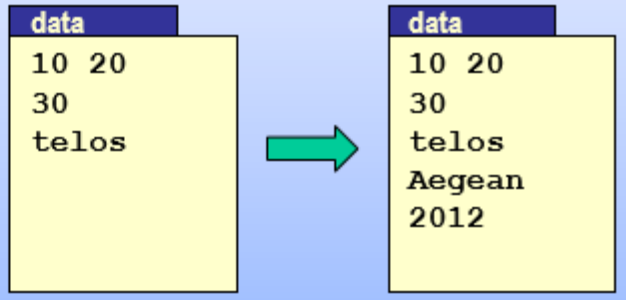
```
10 20
30
telos
```

Για να μπορούμε να χρησιμοποιήσουμε τις κλάσεις ρευμάτων σε αρχεία, θα πρέπει να συμπεριλάβουμε στο πρόγραμμά μας το αρχείο κεφαλίδας **fstream** στο οποίο δηλώνονται. Για παράδειγμα:
#include <fstream>

Μορφοποιημένη έξοδος σε αρχεία κειμένου

```
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;
int main()
{
     ofstream myfile;
     myfile.open("data",ios::app);
     myfile << "Aegean" << endl;
     myfile << 2012;
     myfile.close();
     return 0;
}
```

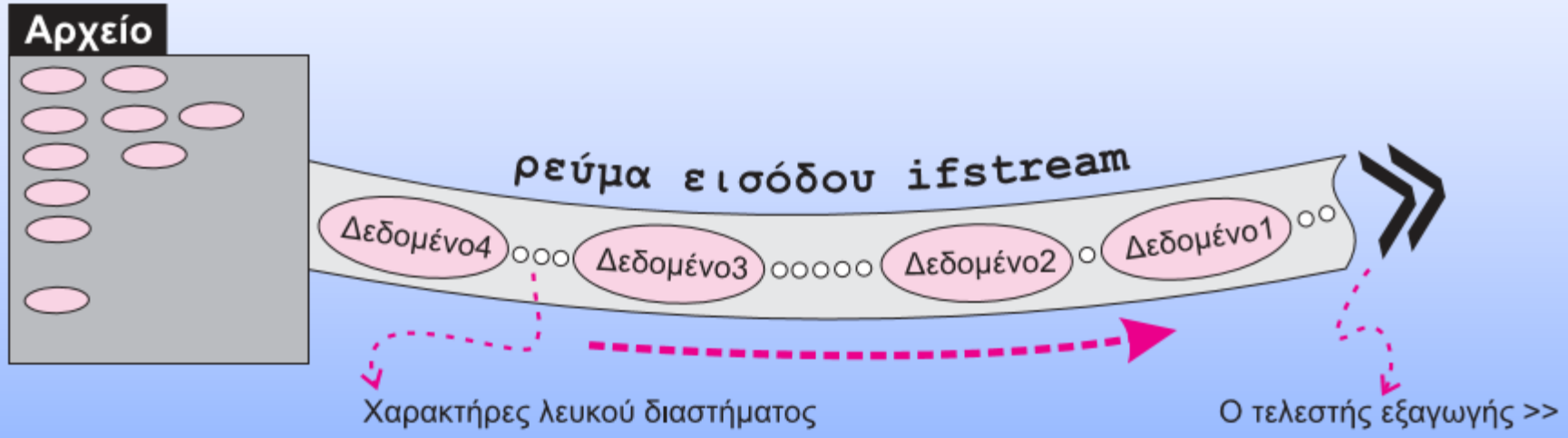
Το πρόγραμμα ανοίγει ένα αρχείο εξόδου με όνομα **data**, για **προσθήκη** δεδομένων (ios::app). **Μετά το τέλος του αρχείου** προσθέτει τη συμβολοσειρά "Aegean" και τον αριθμό 2012!



Στην περίπτωση που κάποιο υπαρκτό αρχείο ανοίξει χωρίς τη σημαία **ios::app**, τότε η εγγραφή του αρχίζει από την αρχή του αρχείου και όλα του τα δεδομένα χάνονται ακόμα και αν δεν εγγράψουμε τίποτα μέσα του!
Στην περίπτωση που το αρχείο δεν υπάρχει, τότε η μέθοδος **open()** - με ή χωρίς τη σημαία **ios::app**, δημιουργεί το αρχείο.

Μορφοποιημένη είσοδος από αρχεία κειμένου

Η μορφοποιημένη είσοδος από ένα αρχείο επιτυγχάνεται με τον τελεστή εξαγωγής >>. Φανταζόμαστε ένα τέτοιο ρεύμα εισόδου σαν ένα κανάλι το οποίο παρέχει δεδομένα τα οποία χωρίζονται μεταξύ τους με χαρακτήρες λευκού διαστήματος (white space characters). Οι χαρακτήρες αυτού είναι το κενό, ο χαρακτήρας αλλαγής γραμμής και ο χαρακτήρας tab.

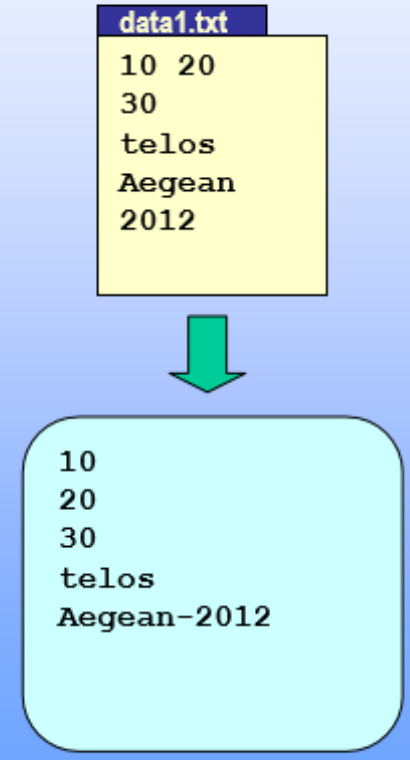


Κάθε φορά που εφαρμόζεται ο τελεστής εξαγωγής >>, εξάγεται από το ρεύμα εισόδου το αμέσως επόμενο δεδομένο που προέρχεται από το συνδεδεμένο με το ρεύμα αρχείο. Όταν η C++ εξάγει δεδομένα (με τον τελεστή εξαγωγής >>) από ένα κανάλι εισόδου, αγνοεί όλους τους χαρακτήρες λευκού διαστήματος που προηγούνται του κάθε δεδομένου.

Μορφοποιημένη είσοδος από αρχεία κειμένου - Παράδειγμα

Το παρακάτω πρόγραμμα ανοίγει το αρχείο **data1.txt** για ανάγνωση και το συνδέει στο ρεύμα **myfile**. Διαβάζει από το ρεύμα **myfile** (δηλαδή από το αρχείο), ένα-ένα τα δεδομένα, χρησιμοποιώντας τον τελεστή εξαγωγής >> και τα εμφανίζει στην οθόνη. Η χρήση του τελεστή εξαγωγής είναι ακριβώς ίδια όπως τον έχουμε χρησιμοποιήσει μέχρι τώρα με το αντικείμενο ρεύματος **cin**.

```
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;
int main()
{
     int a,b,c;
     string lex;
     ifstream myfile;
     myfile.open("data1.txt");
     myfile >> a >> b >> c;
     cout << a << endl << b << endl << c << endl;
     myfile >> lex;
     cout << lex << endl;
     myfile >> lex >> a;
     cout << lex << "-" << a << endl;
     myfile.close();
     return 0;
}
```



Η μέθοδος eof()

Η μέθοδος **eof()** εφαρμόζεται σε αντικείμενα ρευμάτων και επιστρέφει τιμή ψέμα (0) όταν δεν έχουμε φτάσει στο **τέλος** ενός **αρχείου**, και τιμή διάφορη του 0 και σε κάθε άλλη περίπτωση. Το συντακτικό της είναι:

```
int main()
{
     float a;
     ifstream myfile;
     myfile.open("data.txt");
     while (!myfile.eof())
     {
          myfile >> a;
          cout <<a<<endl;
     }
     myfile.close();
}
```

Ο βρόχος **while** εκτελείται ενόσω δεν έχουμε φτάσει στο τέλος του αρχείου.

Εξαγωγή ενός αριθμού από το ρεύμα, και καταχώρισή του στη μεταβλητή **a**. Εμφάνιση του αριθμού στην οθόνη.

data.txt

```
10 20
30
5 3 6 89
234
56 78
1245
```

Κλείνει το αρχείο

Τις περισσότερες φορές, όταν ανοίγουμε ένα αρχείο εισόδου για ανάγνωση δεδομένων, δε γνωρίζουμε το πλήθος των δεδομένων που περιέχει, αλλά απλά θέλουμε να τα διαβάσουμε όλα από την αρχή μέχρι το τέλος. Το παραπάνω πρόγραμμα χρησιμοποιεί την συνάρτηση-μέλος **eof()** για να διαβάσει όλους τους αριθμούς από την αρχή μέχρι το τέλος ενός αρχείου με όνομα **data.txt**.

Συνοψίζοντας ...

- Ο τρόπος με τον οποίο η C++ επικοινωνεί με τις περιφερειακές μονάδες του Η/Υ μας είναι μέσω των αντικειμένων κλάσεων ρευμάτων.
- Στη C++ όλα τα ρεύματα αντιμετωπίζονται με τον ίδιο τρόπο, ανεξάρτητα με τη φυσική συσκευή του συστήματός με την οποία είναι συνδεδεμένα.
- Τα αντικείμενα cin και cout είναι δύο από τα προκαθορισμένα αντικείμενα ρευμάτων της C++. Χρησιμοποιούνται αντίστοιχα για την είσοδο από το πληκτρολόγιο και την έξοδο στην οθόνη.
- Η μορφοποιημένη είσοδος/έξοδος επιτυγχάνεται με τις σημαίες μορφοποίησης, τους χειριστές και τις συναρτήσεις-μέλη που εφαρμόζονται στα αντικείμενα ρευμάτων.
- Οι περισσότερες λειτουργίες των σημαιών μορφοποίησης υλοποιούνται επίσης και με τη χρήση χειριστών.
- Ο χειρισμός ενός αρχείου στη C++ γίνεται μέσω ενός ρεύματος το οποίο συνδέεται με το συγκεκριμένο αρχείο. Η συνάρτηση-μέλος open() χρησιμοποιείται για το άνοιγμα του αρχείου και τη σύνδεση του με το ρεύμα.
- Η C++ επιτρέπει τόσο μορφοποιημένη όσο και μη μορφοποιημένη είσοδο/έξοδο σε αρχεία. Για τη μορφοποιημένη είσοδο/έξοδο χρησιμοποιούνται οι τελετές εξαγωγής και εισαγωγής.
- Για τη μη μορφοποιημένη είσοδο χρησιμοποιούνται συναρτήσεις-μέλη όπως π.χ η getline().
- Συναρτήσεις-μέλη όπως η eof() κ.α. χρησιμοποιούνται για τον έλεγχο και τον χειρισμό των σφαλμάτων.

Παράδειγμα 1

- Το επόμενο πρόγραμμα ζητάει αριθμούς και τους εμφανίζει σε δεκαεξαδική μορφή με κεφαλαίους χαρακτήρες και με ένδειξη της βάσης τους. Το πρόγραμμα σταματάει όταν δοθεί ο αριθμός 0

```cpp
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

int main()
{
    int ar;
    while (true)
    {
        cout<<"Δώσε αριθμό:";
        cin >> ar;
        if (ar==0) break;
        cout    <<"Ο δεκαεξαδικός του "<<ar
                <<" είναι ο "<<hex<<showbase
                <<uppercase<<ar<<endl;
    }
    return 0;
}
```

Παράδειγμα 2

- Το επόμενο πρόγραμμα ζητάει λέξεις και τις καταχωρίζει σε ένα αρχείο κειμένου με όνομα lexis. Το πρόγραμμα σταματάει όταν δοθεί ως λέξη μια συμβολοσειρά με 4 αστερίσκους ****

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
    string lexi;
    ofstream myfile;
    myfile.open("lexeis");
    do
    {
        cin>>lexi;
        if (lexi=="****") break;
        myfile<<lexi<<endl;
    }while (true);
    myfile.close();
    return 0;
}
```

Παράδειγμα 3

- Υποθέστε ότι το αρχείο arithmoi.txt περιέχει έναν άγνωστο αριθμό από ακεραίους αριθμούς. Το παρακάτω πρόγραμμα βρίσκει τον μεγαλύτερο και τον μικρότερο από αυτούς τους αριθμούς

```cpp
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

int main()
{
    float ar,max,min;
    ifstream myfile;
    myfile.open("arithmoi.txt");
    if (!myfile)
    {
        cout<<"Πρόβλημα στο άνοιγμα του αρχείου"<<endl;
        return 1;
    }
    myfile>>ar;
    max=min=ar;
    while (myfile>>ar)
    {
        if (ar>max) max=ar;
        if (ar<min) min=ar;
    }
    myfile.close();
    cout<<"MAX="<<max<<endl;
    cout<<"MIN="<<min<<endl;
    return 0;
}
```

Παράδειγμα 4

- Έστω ότι έχουμε το αρχείο sxoleio (περιέχει τα στοιχεία 100 μαθητών ενός σχολείου σε έναν πίνακα αντικειμένων μιας κλάσης δομής επώνυμο, τάξη, μέσος όρος, ηλικία). Το επόμενο πρόγραμμα ζητάει από τον χρήστη να πληκτρολογήσει το επώνυμο ενός μαθητή, εντοπίζει την εγγραφή του μαθητή μέσα στο αρχείο και εμφανίζει τα υπόλοιπα στοιχεία του. Αν δεν εντοπίσει τον μαθητή με το επώνυμο που δόθηκε, εμφανίζει σχετικό μήνυμα

```cpp
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

int main()
{
    char ch,file_in[20],file_out[20];
    ifstream fin;
    ofstream fout;
    cout<<"Δώσε όνομα αρχείου εισόδου:";
    cin>>file_in;
    fin.open(file_in,ios::binary);
    if (!fin)
    {
        cout<<"Πρόβλημα στο άνοιγμα αρχείου εισόδου\n";
        return 1;
    }
    cout<<"Δώσε όνομα αρχείου εξόδου:";
    cin>>file_out;
    fout.open(file_out,ios::binary);
    if (!fout)
    {
        cout<<"Πρόβλημα στο άνοιγμα αρχείου εξόδου\n";
        return 2;
    }
    while (fin.get(ch)) fout.put(~ch);
    fin.close();
    fout.close();
    return 0;
}
```