

343 – Εισαγωγή στον Προγραμματισμό

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Ακαδημαϊκό Έτος 2015-2016

Χάρης Παπαδόπουλος
207δ, Β' όροφος
e-mail: `charis@cs.uoi.gr`

Ωρες Γραφείου:
Πέμπτη 11-13

Θ: διάλεξη (θεωρία)

Ε: Εργαστήριο

Q: Τεστ quiz

Ημερολόγιο Μαθήματος

Οκτώβριος 2015

Δ	Τ	Τ	Π	Π
			1	2
5	6	7	8	9 Θ
12	13	14	15	16 Θ
19 Ε	20	21	22	23 Θ
26	27	28	29	30 Θ

Νοέμβριος 2015

Δ	Τ	Τ	Π	Π
2 Ε	3	4	5	6 Θ
9 Ε	10	11	12	13 Θ
16 Q	17	18	19	20 Θ
23 Ε	24	25	26	27 Θ
30 Ε				

Δεκέμβριος 2015

Δ	Τ	Τ	Π	Π
	1	2	3	4 Θ
7 Ε	8	9	10	11 Θ
14 Q	15	16	17	18 Θ

Ιανουάριος 2016

Δ	Τ	Τ	Π	Π
4	5	6	7	8
11	12	13	14	15 Θ

Εβδομάδα	Θέματα	Υψη βιβλιογραφίας
Πα, 9 Οκτωβρίου	Εισαγωγικά μαθήματος & Δυσαιδική αναπαράσταση	[1]: 1.1, Παράρτημα 3 [2]: Κεφ. 1, Β, Δ
Πα, 16 Οκτωβρίου	Είσοδος/Εξοδος δεδομένων, τύποι δεδομένων & μεταβλητών	[1]: 1.2, 1.3, 1.4, 1.5, Παράρτημα 1 [2]: Κεφ. 2, Γ
Δε, 19 Οκτ	1 ^ο Εργαστήριο	
Πα, 23 Οκτωβρίου	Προεπεξεργαστής, αριθμητικοί και λογικοί τελεστές	[1]: 2.1, Παράρτημα 2 [2]: 4.11, 4.12, Α, ΣΤ
Πα, 30 Οκτωβρίου	Ροή ελέγχου: if/else, switch, for, while, do-while και ροή ελέγχου if/else	[1]: 2.2, 2.3 [2]: Κεφ. 4, Κεφ. 5
Δε, 2 Νοε	2 ^ο Εργαστήριο	
Πα, 6 Νοεμβρίου	Συναρτήσεις, εμβέλεια μεταβλητών και αναδρομή	[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2 [2]: Κεφ. 6
Δε, 9 Νοε	3 ^ο Εργαστήριο	
Πα, 13 Νοεμβρίου	Επανάληψη Εργαστηρίων	
Δε, 16 Νοε	1 ^ο Quiz	
Πα, 20 Νοεμβρίου	Επανάληψη με Παραδείγματα	
Δε, 23 Νοε	4 ^ο Εργαστήριο	
Πα, 27 Νοεμβρίου	Πίνακες (μονοδιάστατοι και πολυδιάστατοι)	[1]: 5.1, 5.2, 5.4 [2]: Κεφ. 7
Δε, 30 Νοε	5 ^ο Εργαστήριο	
Πα, 4 Δεκεμβρίου	Εφαρμογές σε ταξινομήσεις και αναζήτηση στοιχείων	[1]: Παράρτημα 4, 9.1, 9.2, 9.3 [2]: 6.7, 6.8, Κεφ. 18
Δε, 7 Δεκ	6 ^ο Εργαστήριο	
Πα, 11 Δεκεμβρίου	Αλφαριθμητικά και Συμβολοσειρές	[1]: 6.1, 12.1, 12.2, 12.4 [2]: Κεφ. 21, 17.1-17.10
Δε, 14 Δεκ	2 ^ο Quiz	
Πα, 18 Δεκεμβρίου	Εγγραφές, δομές και χρήση αρχείων	[1]: 5.3, 13.3 [2]: 7.7, 7.8, 8.6, Κεφ. 19
Πα, 15 Ιανουαρίου	Επανάληψη	

Θ: διάλεξη (θεωρία)

Ε: Εργαστήριο

Q: Τεστ quiz

Ημερολόγιο Μαθήματος

Οκτώβριος 2015

Δ	Τ	Τ	Π	Π
			1	2
5	6	7	8	9 Θ
12	13	14	15	16 Θ
19 Ε	20	21	22	23 Θ
26	27	28	29	30 Θ

Νοέμβριος 2015

Δ	Τ	Τ	Π	Π
2 Ε	3	4	5	6 Θ
9 Ε	10	11	12	13 Θ
16 Q	17	18	19	20 Θ
23 Ε	24	25	26	27 Θ
30 Ε				

Δεκέμβριος 2015

Δ	Τ	Τ	Π	Π
	1	2	3	4 Θ
7 Ε	8	9	10	11 Θ
14 Q	15	16	17	18 Θ

Ιανουάριος 2016

Δ	Τ	Τ	Π	Π
4	5	6	7	8
11	12	13	14	15 Θ

Εβδομάδα	Θέματα	Υψη βιβλιογραφίας
Πα, 9 Οκτωβρίου	Εισαγωγικά μαθήματος & Δυσαική αναπαράσταση	[1]: 1.1, Παράρτημα 3 [2]: Κεφ. 1, Β, Δ
Πα, 16 Οκτωβρίου	Είσοδος/Εξοδος δεδομένων, τύποι δεδομένων & μεταβλητών	[1]: 1.2, 1.3, 1.4, 1.5, Παράρτημα 1 [2]: Κεφ. 2, Γ
Δε, 19 Οκτ	1 ^ο Εργαστήριο	
Πα, 23 Οκτωβρίου	Προεπεξεργαστής, αριθμητικοί και λογικοί τελεστές	[1]: 2.1, Παράρτημα 2 [2]: 4.11, 4.12, Α, ΣΤ
Πα, 30 Οκτωβρίου	Ροή ελέγχου: if/else, switch, for, while, do-while και ροή ελέγχου if/else	[1]: 2.2, 2.3 [2]: Κεφ. 4, Κεφ. 5
Δε, 2 Νοε	2 ^ο Εργαστήριο	
Πα, 6 Νοεμβρίου	Συναρτήσεις, εμβέλεια μεταβλητών και αναδρομή	[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2 [2]: Κεφ. 6
Δε, 9 Νοε	3 ^ο Εργαστήριο	
Πα, 13 Νοεμβρίου	Επανάληψη Εργαστηρίων	
Δε, 16 Νοε	1 ^ο Quiz	
Πα, 20 Νοεμβρίου	Επανάληψη με Παραδείγματα	
Δε, 23 Νοε	4 ^ο Εργαστήριο	
Πα, 27 Νοεμβρίου	Πίνακες (μονοδιάστατοι και πολυδιάστατοι)	[1]: 5.1, 5.2, 5.4 [2]: Κεφ. 7
Δε, 30 Νοε	5 ^ο Εργαστήριο	
Πα, 4 Δεκεμβρίου	Εφαρμογές σε ταξινομήσεις και αναζήτηση στοιχείων	[1]: Παράρτημα 4, 9.1, 9.2, 9.3 [2]: 6.7, 6.8, Κεφ. 18
Δε, 7 Δεκ	6 ^ο Εργαστήριο	
Πα, 11 Δεκεμβρίου	Αλφαριθμητικά και Συμβολοσειρές	[1]: 6.1, 12.1, 12.2, 12.4 [2]: Κεφ. 21, 17.1-17.10
Δε, 14 Δεκ	2 ^ο Quiz	
Πα, 18 Δεκεμβρίου	Εγγραφές, δομές και χρήση αρχείων	[1]: 5.3, 13.3 [2]: 7.7, 7.8, 8.6, Κεφ. 19
Πα, 15 Ιανουαρίου	Επανάληψη	

Ενότητα 24

ΔΟΜΕΣ

Δομές

- 2^η ομαδοποιημένη δομή δεδομένων:
struct
- Θυμίζουμε: "ομαδοποίηση"
 - Πίνακες: συλλογή από τιμές ίδιου τύπου
 - Δομή: συλλογή από τιμές διαφορετικών τύπων
- Τις χειριζόμαστε ως ένα αντικείμενο, όπως τους πίνακες
- Βασική διαφορά:
 - Πρέπει πρώτα να "ορίσουμε" την δομή
 - Πριν από την δήλωση οποιασδήποτε μεταβλητής

Τύποι δομών

- Ορίζουμε την δομή καθολικά (συνήθως)
- Δεν δεσμεύουμε μνήμη
 - Απλά δίνουμε έναν "τύπο" για το πώς η δομή θα μοιάζει
- Ορισμός:

```
struct CDAccountV1
{
    double balance;
    double interestRate;
    int term;
};
```

← όνομα της νέας δομής (ετικέτα)

← ονόματα για τα μέλη

Δήλωση μεταβλητών δομής

```
struct CDAccountV1
{
    double balance;
    double interestRate;
    int term;
};
```

- Με τον ορισμό της δομής μπορούμε τώρα να ορίσουμε νέες μεταβλητές αυτού του τύπου:

```
CDAccountV1 account;
```

- Ακριβώς όπως δηλώνουμε για απλούς τύπους
- Η μεταβλητή *account* τώρα είναι τύπου CDAccountV1
- Περιέχει "τιμές για τα μέλη"
 - Για κάθε "τμήμα" της δομής

Πρόσβαση στα μέλη της δομής

- Ο τελεστής τελεία . έχει πρόσβαση στα μέλη:

- account.balance
- account.interestRate
- account.term

```
struct CDAccountV1
{
    double balance;
    double interestRate;
    int term;
};
```

```
CDAccountV1 account;
```

- Καλούνται "μεταβλητές για τα μέλη"
 - Τα "τμήματα" της μεταβλητής δομής
 - Διαφορετικές δομές μπορούν να έχουν το ίδιο όνομα για τις μεταβλητές για τα μέλη
 - Δεν έχουμε σύγκρουση


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

struct CDAccountV1
{
    double balance;
    double interestRate;
    int term;
};
```

```
void getData(CDAccountV1& theAccount)
{
    cout << "Δώσε λογαριασμό: ";
    cin >> theAccount.balance;
    cout << "Δώσε επιτόκιο ";
    cin >> theAccount.interestRate;
    cout << "Δώσε αριθμό μηνών";
    cin >> theAccount.term;
}
```

```
void getData(CDAccountV1& theAccount);
```

```
int main( )
{
```

```
    CDAccountV1 account;
    getData(account);
```

```
    double rateFraction, interest;
```

```
    rateFraction = account.interestRate/100.0;
```

```
    interest = account.balance*(rateFraction*(account.term/12.0));
```

```
    account.balance = account.balance + interest;
```

```
    cout << "When your CD matures in "
```

```
        << account.term << " months,\n"
```

```
        << "it will have a balance of $"
```

```
        << account.balance << endl;
```

```
}
```

Παραδείγματα

```
struct StudentRecord
{
    int studentNumber;
    char grade;
};

int main()
{
    StudentRecord yourRecord;
    yourRecord.studentNumber = 10002;
    yourRecord.grade = 'A';
}
```

```
struct Automobile
{
    int year;
    int doors;
    double horsepower;
    char model[30];
};

int main()
{
    Automobile my;
    my.year = 2004;
    my.doors = 2;
    my.horsepower = 122;
    strcpy(my.model, "mini");
}
```

Παράλειψη ερωτηματικού στο τέλος

- Δεν επιτρέπεται η παράλειψη ερωτηματικού στο τέλος

```
struct WeatherData
{
    double temperature;
    double windVelocity;
};
```

- Είναι απαραίτητο γιατί μπορείτε να δηλώσετε μεταβλητές δομής στο σημείο αυτό

```
struct WeatherData
{
    double temperature;
    double windVelocity;
} high, low;
```

Χρήση ιεραρχικών δομών

```
struct Date
{
    int day;
    int month;
    int year;
};

struct PersonInfo
{
    double height;
    double weight;
    Date birthdate;
};
```

- Δομές που μπορεί τα μέλη να είναι μικρότερες δομές
- Έχει σημασία η σειρά που δηλώνουμε τις δομές
 - πρώτα δηλώνεται η μικρότερη δομή
- Αν έχουμε μια μεταβλητή:

```
PersonInfo person1;
```

- Τότε εμφανίζουμε στην έξοδο:

```
cout << person1.birthdate.day;
```

Οι δομές σε ορίσματα σε συναρτήσεις

- Τα περνάμε όπως τα απλά δεδομένα
 - Παράμετρος με τιμή
 - Παράμετρος με αναφορά
 - Ή συνδυασμός
- Μπορούν επίσης να είναι ο επιστρεφόμενος τύπος μιας συνάρτησης
 - Επιστρεφόμενος Τύπος είναι μια δομή
 - Η εντολή return στον ορισμό της συνάρτησης στέλνει μια μεταβλητή δομής σε αυτόν που την κάλεσε

Απόδοση αρχικών τιμών σε δομές

- Μπορούν να αρχικοποιηθούν κατά τη δήλωση

```
struct Date
{
    int day;
    int month;
    int year;
};

Date dueDate = {12, 31, 2003};
```

- Η δήλωση παρέχει αρχικά δεδομένα σε όλα τα τρία μέλη της δομής

Παραδείγματα

```
struct Shoetype
{
    char style;
    double price;
};
```

```
Shoetype shoe1, shoe2;
shoe1.style = 'A';
shoe1.price = 9.99;
cout << shoe1.style
      << shoe1.price << endl;

shoe2 = shoe1;
shoe2.price = shoe2.price / 9;
cout << shoe2.style
      << shoe2.price << endl;
```

```
Shoetype discount(Shoetype oldshoe)
{
    Shoetype temp;
    temp.style = oldshoe.style;
    temp.price = 0.9 * oldshoe.price;
    return temp;
}
```

```
void readShoetype(Shoetype& newshoe1)
{
    cout << "Δώσε στυλ:";
    cin >> newshoe1.style;
    cout << "Δώσε τιμή:";
    cin >> newshoe1.price;
}
```

Ενότητα 25

ΧΡΗΣΗ ΑΡΧΕΙΩΝ

Ροές (streams)

- Τυπικά: διαβάζουμε δεδομένα από το πληκτρολόγιο και εκτυπώνουμε σε ένα τερματικό παράθυρο.
Ωστόσο:
 - Μπορούμε να διαβάζουμε από αρχεία
 - Μπορούμε να εκτυπώνουμε σε αρχεία
- Stream: Μια ροή από χαρακτήρες
- Ροή εισόδου (Input stream)
 - Μπορεί να έρθει από το πληκτρολόγιο
 - Μπορεί να έρθει από αρχείο
- Ροή εξόδου (Output stream)
 - Μπορεί να σταλθεί στην οθόνη
 - Μπορεί να σταλθεί σε αρχείο

Χρήση Ροών

- Ήδη χρησιμοποιούμε ροές
 - cin
 - Η ροή εισόδου ενώνεται με το πληκτρολόγιο
 - cout
 - Η ροή εξόδου ενώνεται με την οθόνη
- Μπορούμε να ορίσουμε άλλες ροές
 - Από ή προς αρχεία
 - Παρόμοια χρήση όπως με τα cin, cout

Χρήση ροών όπως με cin, cout

- Θεωρείστε:
 - Το πρόγραμμα ορίζει μια ροή (stream) `inStream` που έρχεται από κάποιο αρχείο:
`int theNumber;`
`inStream >> theNumber;`
 - Διαβάζει τιμή από την ροή `stream`, και την αναθέτει στο *theNumber*
 - Το πρόγραμμα ορίζει μια ροή (stream) `outStream` που στέλνεται σε κάποιο αρχείο
`outStream << "theNumber is " << theNumber;`
 - Γράφει την τιμή στη ροή `stream`, που πάει σε κάποιο αρχείο

Αρχεία

- Θα χρησιμοποιήσουμε αρχεία text
- Διάβασμα από αρχείο
 - Όταν το πρόγραμμα δέχεται κάποια είσοδο
- Εγγραφή σε αρχείο
 - Όταν το πρόγραμμα στέλνει κάποια έξοδο
- Ξεκινάει από την αρχή του αρχείου και καταλήγει προς το τέλος του αρχείου
 - Υπάρχουν και άλλοι τρόποι
 - Θα επεκταθούμε στο τέλος σε άλλους τρόπους προσπέλασης

Σύνδεση με αρχείο

- Πρώτα πρέπει να συνδέσουμε το *αρχείο* με μια *ροή*
- Για είσοδο:
 - Αρχείο → `ifstream` αντικείμενο
- Για έξοδο:
 - Αρχείο → `ofstream` αντικείμενο
- Οι κλάσεις `ifstream` και `ofstream`
Ορίζονται στη βιβλιοθήκη `<fstream>`
`#include <fstream>`

Βιβλιοθήκες αρχείων Ε/Ε

- Για να επιτρέψουμε τόσο για είσοδο από αρχείο όσο και για έξοδο σε αρχείο :

```
#include <fstream>
using namespace std;
```

ή

```
#include <fstream>
using std::ifstream;
using std::ofstream;
```

Δήλωση ροών

- Η ροή πρέπει πρώτα να δηλωθεί όπως μια τυπική μεταβλητή:

```
ifstream inStream;  
ofstream outStream;
```

- Μετά πρέπει να συνδεθεί με το αρχείο:

```
inStream.open("infile.txt");
```

- Καλείται "άνοιγμα αρχείου"
- Χρησιμοποιεί την συνάρτηση *open*
- Μπορούμε να ορίσουμε την ακριβή διαδρομή του αρχείου

Χρήση Ροών

- Μόλις δηλωθεί → χρησιμοποιείται κανονικά!

```
int oneNumber, anotherNumber;  
inStream >> oneNumber >> anotherNumber;
```

- Η ροή εξόδου γίνεται παρόμοια:

```
ofstream outStream;  
  
outStream.open("outfile.txt");  
outStream << "oneNumber = " << oneNumber  
          << " anotherNumber = "  
          << anotherNumber;
```

- Στέλνει τα αντικείμενα στο αρχείο εξόδου "outfile.txt"

Ονοματολογία αρχείων

- Προγράμματα και αρχεία
- Τα αρχεία έχουν δύο ονόματα στα προγράμματά μας

- Εξωτερικό όνομα αρχείου

```
inStream.open("infile.txt");
```

- Επίσης καλείται "φυσικό όνομα"
- Όπως το "infile.txt"
- Μερικές φορές λέγεται και "πραγματικό όνομα"
- Χρησιμοποιείται μόνο μια φορά (στο άνοιγμα/σύνδεση)

- Όνομα ροής

- Καλείται επίσης και "λογικό όνομα"
- Το πρόγραμμα χρησιμοποιεί το όνομα αυτό για όλα τα αρχεία που επεξεργάζεται

```
ifstream inStream;  
ofstream outStream;
```

Κλείσιμο αρχείων

- Τα αρχεία που ανοίγουν πρέπει να κλείνουν
 - Όταν το πρόγραμμα σταματάει να δέχεται είσοδο ή να στέλνει στην έξοδο
 - Απελευθερώνει τη ροή από το αρχείο

```
ifstream inStream;  
ofstream outStream;
```

```
inStream.close();  
outStream.close();
```

- Δεν παίρνουν ορίσματα
- Τα αρχεία κλείνουν αυτόματα όταν το πρόγραμμα τερματίζει
 - Ωστόσο πρέπει εμείς να τα κλείνουμε για να αποφεύγουμε λάθη εγγραφής/ανάγνωσης

Η συνάρτηση flush()

- Η έξοδος συνήθως "ενταμιεύεται (buffered)"
 - αποθηκεύεται προσωρινά πριν την εγγραφή της σε αρχείο
 - Η εγγραφή γίνεται σε "ομάδες"
- Συχνά μπορεί να θέλουμε να εξαναγκάσουμε την εγγραφή:
`outStream.flush();`
 - Η συνάρτηση *flush*, για όλες τις ροές εξόδου
 - Όλα τα δεδομένα εξόδου (buffered) γράφονται
- Όταν κλείνουμε το αρχείο τότε καλείται αυτόματα η συνάρτηση `flush()`

```
#include <fstream>
using namespace std;

int main( )
{
    ifstream inStream;
    ofstream outStream;

    inStream.open("infile.txt");
    outStream.open("outfile.txt");

    int first, second, third;
    inStream >> first >> second >> third;
    outStream << "The sum of the first 3\n"
        << "numbers in infile.txt\n"
        << "is " << (first + second + third)
        << endl;

    inStream.close( );
    outStream.close( );

    return 0;
}
```

infile.txt

1
2
3
4

outfile.txt

The sum of the first 3
numbers in infile.txt
is 6

Προσάρτηση σε αρχείο

- Ο τυπικός τρόπος που ανοίγουμε αρχείο ξεκινάει με κενό αρχείο
 - Ακόμα και όταν το αρχείο υπάρχει ήδη → τα δεδομένα χάνονται
- Άνοιγμα για προσάρτηση:

```
ofstream outStream;  
outStream.open("important.txt", ios::app);
```

- Αν το αρχείο δεν υπάρχει → το δημιουργεί
- Αν το αρχείο υπάρχει → προσαρτά στο τέλος
- το 2^ο όρισμα είναι η κλάση *ios* που ορίζεται ως σταθερά
 - στη βιβλιοθήκη `<iostream>`, `std namespace`

Εναλλακτικός τρόπος για άνοιγμα αρχείου

- Μπορούμε να δηλώσουμε το όνομα του αρχείου στη δήλωση
 - Το περνάμε ως ένα όρισμα

```
ifstream inStream;  
inStream.open("infile.txt");
```

Ισοδύναμο με:

```
ifstream inStream("infile.txt");
```

Παράδειγμα

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

int main( )
{
    cout << "Opening data.txt for appending.\n";
    ofstream fout;
    fout.open("data.txt", ios::app);

    fout << "5 6 pick up sticks.\n"
          << "7 8 ain't C++ great!\n";

    fout.close( );
    cout << "End of appending to file.\n";

    return 0;
}
```

data.txt (πριν)

```
1 2 bucket my shoe.
3 4 shut the door.
```

data.txt (μετά)

```
1 2 bucket my shoe.
3 4 shut the door.
5 6 pick up sticks.
7 8 ain't C++ great!
```

Έλεγχος για ύπαρξη αρχείου

- Το άνοιγμα αρχείου μπορεί να μην λειτουργήσει
 - Αν το αρχείο εισόδου δεν υπάρχει
 - Δεν υπάρχει άδεια εγγραφής στο αρχείο εξόδου
 - Άγνωστα αποτελέσματα
- Η συνάρτηση fail()
 - Κάλεσμα της fail() για έλεγχο σωστό ροών

```
inStream.open("stuff.txt");  
if (inStream.fail())  
{  
    cout << "File open failed.\n";  
    exit(1);  
}
```


Έλεγχος για το τέλος αρχείου (EOF)

- Χρήση βρόχου για επεξεργασία έως ότου το τέλος αρχείου
 - Συνήθης πρακτική
- Δύο τρόποι για έλεγχο EOF:
 - Η συνάρτηση eof()
 - Διαβάζει κάθε χαρακτήρα έως EOF
 - eof() συν/ση επιστρέφει bool
 - Η διαδικασία διαβάσματος επιστρέφει bool τιμή! (inStream >> next)
 - Η έκφραση επιστρέφει true αν το διάβασμα έγινε επιτυχώς
 - Επιστρέφει false αν επιχειρήσει να διαβάσει πέρα από το τέλος του αρχείου

```
inStream.get(next);  
while (!inStream.eof())  
{  
    cout << next;  
    inStream.get(next);  
}
```

```
double next, sum = 0;  
while (inStream >> next)  
    sum = sum + next;  
cout << "sum:" << sum;
```

Τα ονόματα αρχείων ως μεταβλητές

- Η διαδικασία ανοίγματος ροής
 - Το όρισμα στη συνάρτηση open() είναι τύπου string
 - Μπορεί να είναι σταθερά " " ή μεταβλητή

```
char fileName[16];  
ifstream inStream;  
cout << "Enter file name: ";  
cin >> fileName;  
inStream.open(fileName);
```

- Παρέχει περισσότερη ευελιξία στο πρόγραμμα

```
#include <fstream>
#include <iostream>
#include <cstdlib> //for exit
using namespace std;

int main( )
{
    ifstream inStream;
    ofstream outStream;

    inStream.open("infile.txt");
    if (inStream.fail( ))
    {
        cout << "Error.\n";
        exit(1);
    }
    outStream.open("outfile.txt");
    if (outStream.fail( ))
    {
        cout << " Error.\n";
        exit(1);
    }
}
```

```
int first, second, third;
inStream >> first >> second >> third;
outStream << "The sum of the first 3\n"
           << "numbers in infile.txt\n"
           << "is "
           << (first + second + third)
           << endl;

inStream.close( );
outStream.close( );
}
```

```

#include <fstream>
#include <iostream>
#include <cstdlib> //for exit
using namespace std;

void addplusplus(istream& inStream, ostream& outStream);

int main( )
{
    ifstream fin;
    ofstream fout;

    fin.open("cad.txt");
    if (fin.fail( ))
    {
        cout << "Error.\n";
        exit(1);
    }
    fout.open("cppad.txt");
    if (fout.fail( ))
    {
        cout << " Error.\n";
        exit(1);
    }

    addPlusPlus(fin, fout);

    fin.close( );
    fout.close( );
}

```

cad.txt

C is one of the world's most modern programming languages. There is no language as versatile as C, and C is fun to use.

```

void addPlusPlus( istream& inStream,
                  ostream& outStream )
{
    char next;

    inStream.get(next);
    while (! inStream.eof( ))
    {
        if (next == 'C')
            outStream << "C++";
        else
            outStream << next;

        inStream.get(next);
    }
}

```

Σύνοψη

- Οι ροές συνδέονται με αρχεία με την λειτουργία `open()`
- Η συνάρτηση `fail()` ελέγχει για επιτυχημένη ανάγνωση/εγγραφή
- Οι τύποι ροών μπορούν να είναι παράμετροι σε συναρτήσεις
 - Πρέπει να είναι παράμετροι με αναφορά

Καλή Μελέτη

- **Βιβλιογραφία**

[1] W. Savitch, Πλήρης C++, Εκδόσεις Τζιόλα, 2011

[2] H. Deitel and P. Deitel, C++ Προγραμματισμός 6η Εκδοση, Εκδόσεις Μ. Γκιούρδας, 2013

Ύλη βιβλιογραφίας

[1]: 6.1, 12.1, 12.2, 12.4

[2]: Κεφ. 21, 17.1-17.10