

**Ανοικτά Ακαδημαϊκά Μαθήματα**

**Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας**

**Αντικειμενοστρεφής Προγραμματισμός (Ε)**

**Ενότητα 7:** Streams

Κλειώ Σγουροπούλου

Τμήμα Μηχανικών Πληροφορικής ΤΕ

|  |  |
| --- | --- |
| Το περιεχόμενο του μαθήματος διατίθεται με άδεια Creative Commons εκτός και αν αναφέρεται διαφορετικά | Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους. |

Πίνακας Περιεχομένων

[Αντικειμενοστρεφής Προγραμματισμός- Εργαστήριο 7 3](#_Toc402880453)

[Streams 3](#_Toc402880454)

[Η κλάση ios 4](#_Toc402880455)

[Αρχεία 4](#_Toc402880456)

[Εγγραφή σε ένα αρχείο: 4](#_Toc402880457)

[Διάβασμα από το ίδιο αρχείο: 5](#_Toc402880458)

[Είσοδος/έξοδος χαρακτήρων (Character I/O) 5](#_Toc402880459)

[Δυαδική είσοδος/έξοδος (binary I/O) 6](#_Toc402880460)

[Χειρισμός αρχείων από το ίδιο το αντικείμενο 7](#_Toc402880461)

[Λάθη κατά τον χειρισμό αρχείων 8](#_Toc402880462)

# Αντικειμενοστρεφής Προγραμματισμός- Εργαστήριο 7

## Streams

Ως stream ονομάζουμε μια συνεχή ροή δεδομένων, από το πληκτρολόγιο, προς την οθόνη, ή από και προς ένα αρχείο. Στην C++ ένα stream αντιπροσωπεύεται από ένα αντικείμενο ή μια κλάση. Έχουμε χρησιμοποιήσει ήδη τα αντικείμενα cin και cout για την είσοδο από το πληκτρολόγιο και την έξοδο στην οθόνη. Υπάρχουν διαφορετικά streams για την αναπαράσταση διαφορετικών ειδών ροής δεδομένων, π.χ. η κλάση ifstream αντιπροσωπεύει ροές δεδομένων από αρχεία. Τα πλεονεκτήματα που μας παρέχει η χρήση των streams είναι η μεγαλύτερη απλότητα στην χρήση τους σε σύγκριση με τις συναρτήσεις της C, καθώς και η δυνατότητα υπερφόρτωσης τελεστών και συναρτήσεων (όπως οι

<< και >>).

Οι κλάσεις των streams έχουν μια κάπως περίπλοκη ιεραρχία, όπως φαίνεται στο παρακάτω σχήμα. Όλες σχεδόν οι κλάσεις κληρονομούν από την ios. Το αντικείμενο cin που ξέρουμε ως τώρα είναι ήδη ορισμένο αντικείμενο της κλάσης istream\_withassign ενώ αντίστοιχα το cout είναι αντικείμενο της κλάσης ostream\_withassign.

Η κλάση istream ορίζει συναρτήσεις όπως οι get(), getline(), read(), και τον overloaded operator >>, ενώ η ostream ορίζει τις put() και write(), και τον overloaded operator <<. Τέλος η iostream κλάση δημιουργείται από τις istream και ostream μέσω πολλαπλής κληρονομικότητας (multiple inheritance).

Τέλος, να σημειωθεί η διαφορά μεταξύ των κλάσεων istream και istream\_withassign (και αντίστοιχα μεταξύ ostream και ostream\_withassign). Οι κλάσεις \_withassign μοιάζουν πολύ με τις μητρικές τους, με την διαφορά ότι περιέχουν overloaded assignment operators ώστε τα αντικείμενά τους να μπορούν να αντιγραφούν (που χρειάζεται σε μερικές ειδικές όμως περιπτώσεις).

## Η κλάση ios

Η κλάση ios περιέχει τα περισσότερα χαρακτηριστικά που μας επιτρέπουν τον χειρισμό των ροών εισόδου/εξόδου, όπως είναι τα formatting flags και τα status error bits. Παραδείγματα:

#include <iostream> using namespace std; int main()

{

cout << "Hello world" << endl;

cout.width(15);

cout << "Hello world" << endl;

int var = 15;

cout << "Dekaexadikos: " << hex << var << endl;

}

## Αρχεία

Για το διάβασμα και το γράψιμο σε αρχεία χρησιμοποιούμε αντίστοιχα τις κλάσεις ifstream και ofstream, ενώ υπάρχει και η κλάση iofstream που κληρονομεί από αυτές τις δύο κλάσεις. Οι κλάσεις αυτές ορίζονται στο FSTREAM header file.

### Εγγραφή σε ένα αρχείο:

#include <fstream> //for file I/O

#include <iostream> using namespace std;

int main()

{

char ch = 'x'; int j = 77; double d = 6.02;

char str1[80] = "string"; //strings without char str2[80] = "file"; // embedded spaces

ofstream outfile("fdata.txt"); //create ofstream object outfile << ch //insert (write) data

<< j

<< ' ' //needs space between numbers

<< d

<< str1

<< ' ' //needs spaces between strings

<< str2;

cout << "File written\n";

return 0;

}

### Διάβασμα από το ίδιο αρχείο:

#include <fstream> //for file I/O

#include <iostream> using namespace std;

int main()

{

char ch; int j; double d;

char str1[80]; char str2[80];

ifstream infile("fdata.txt"); //create ifstream object infile >> ch >> j >> d >> str1 >> str2;

cout << ch << endl //display the data

<< j << endl

<< d << endl

<< str1 << endl

<< str2 << endl;

return 0;

}

Πρόταση:

Δοκιμάστε να αλλάξετε την μεταβλητή d στα δύο προηγούμενα παραδείγματα, από double σε char[80]. Τι θα συμβεί και γιατί;

## Είσοδος/έξοδος χαρακτήρων (Character I/O)

Για να γράψουμε και να διαβάσουμε χαρακτήρες από ένα αρχείο χρησιμοποιούμε τις συναρτήσεις get(char) και put(char) των κλάσεων ifstream και ofstream αντίστοιχα. Επίσης, μέσω της συνάρτησης seekg() μπορούμε να μετατοπίσουμε τον pointer που διατρέχει το αρχείο σε διάφορα σημεία (αρχή, τέλος, ή οπουδήποτε μέσα στο αρχείο).

#include <fstream>

#include <iostream>

using namespace std;

int main(int argc, char \*argv[])

{

if (argc != 3) {

cout << "Usage: CopyFile <from> <to>" << endl;

return 0;

}

ifstream fin(argv[1]); if (fin == 0) {

cout << "Error: Input file cannot be opened for reading!" << endl; return 10;

}

ofstream fout(argv[2]); if (fout == 0) {

cout << "Error: Output file cannot be opened for writing!" << endl;

return 10;

}

fin.seekg(0, ios::end); size\_t finsize = fin.tellg();

cout << "Input file size: " << finsize << endl;

fin.seekg(0, ios::beg); // ios::beg -> from the start of the file char c;

int percent = 0;

while (fin.get(c))

{

fout.put(c);

cout << "Copy Completed : " << 100\*percent/finsize << "\r"; percent++;

}

fin.close();

fout.close();

}

**Πρόταση:**

Δοκιμάστε να αλλάξετε τον παραπάνω κώδικα, ώστε η συνάρτηση get() να μην διαβάζει ένα χαρακτήρα την φορά αλλά ένα μπλόκ χαρακτήρων (π.χ. 512 bytes/χαρακτήρες). Χρησιμοποιήστε την συνάρτηση istream& istream::read(buf, bufsize) όπου buf είναι ένας πίνακας τύπου char, και το bufsize το πλήθος των χαρακτήρων που θα διαβαστούν με την μια.

Χρησιμοποιήστε την συνάρτηση int istream::gcount()μετά από κάθε read() ώστε να βρείτε πόσα bytes διαβάστηκαν τελικά (την τελευταία φορά δεν θα είναι 512 bytes αλλά λιγότερα).

## Δυαδική είσοδος/έξοδος (binary I/O)

Σε ένα αρχείο μπορούμε εκτός από κείμενο να αποθηκεύσουμε και δεδομένα οποιουδήποτε τύπου. Για παράδειγμα μπορούμε να αποθηκεύσουμε αντικείμενα κάποιας κλάσης, το ένα μετά το άλλο, σε ένα δυαδικό αρχείο.

#include <fstream> //for file streams

#include <iostream>

using namespace std;

//////////////////////////////////////////////////////////////// class person //class of persons

{

protected:

char name[80]; //person’s name

int age; //person’s age public:

void getData() //get person’s data

{

cout << "\n Enter name: "; cin >> name;

cout << " Enter age: "; cin >> age;

}

void showData() //display person’s data

{

cout << "\n Name: " << name;

cout << "\n Age: " << age;

}

};

int main()

{

char ch;

person pers; //create person object

fstream file; //create input/output file

//open for append

file.open("GROUP.DAT", ios::app | ios::out | ios::in | ios::binary );

do //data from user to file

{

cout << "\nEnter person’s data:";

pers.getData(); //get one person’s data

//write to file

file.write( (char\*)&pers, sizeof(pers) );

cout << "Enter another person (y/n)? ";

cin >> ch;

}while(ch=='y'); //quit on ‘n’

file.seekg(0); //reset to start of file

//read first person

while( file.read( (char\*)&pers, sizeof(pers) ) )

{

cout << "\nPerson:"; //display person

pers.showData(); //read another person

}

cout << endl;

system("pause"); return 0;

}

Στο παραπάνω πρόγραμμα, χρησιμοποιούμε την συνάρτηση seekg(0) για να οδηγήσουμε τον δείκτη του αρχείου *που διαβάζουμε* στην θέση μηδέν (αρχή του αρχείου). Αντίστοιχα χρησιμοποιούμε την seekp(..) για να πάμε τον δείκτη του αρχείου *στο οποίο γράφουμε* σε μια συγκεκριμένη θέση εντός του αρχείου.

Οι συναρτήσεις seekg και tellg αφορούν αρχείο που έχει ανοιχτεί για διάβασμα, ενώ οι συναρτήσεις seekp και tellp αρχείο για γράψιμο.

## Χειρισμός αρχείων από το ίδιο το αντικείμενο

#include <fstream> //for file streams

#include <iostream>

using namespace std;

class person //class of persons

{

protected:

char name[40]; //person’s name

int age; //person’s age

public:

void getData(void) //get person’s data

{

cout << "\n Enter name: "; cin >> name;

cout << " Enter age: "; cin >> age;

}

void showData(void) //display person’s data

{

cout << "\n Name: " << name;

cout << "\n Age: " << age;

}

void diskIn(int); //read from file

void diskOut(); //write to file

static int diskCount(); //return number of

// persons in file

};

void person::diskIn(int pn) //read person number pn

{ //from file

ifstream infile; //make stream

infile.open("PERSFILE.DAT", ios::binary); //open it

infile.seekg( pn\*sizeof(person) ); //move file ptr

infile.read( (char\*)this, sizeof(\*this) ); //read one person

}

void person::diskOut() //write person to end of file

{

ofstream outfile; //make stream

//open it outfile.open("PERSFILE.DAT", ios::app | ios::binary); outfile.write( (char\*)this, sizeof(\*this) ); //write to it

}

int person::diskCount() //return number of persons

{ //in file

ifstream infile;

infile.open("PERSFILE.DAT", ios::binary);

infile.seekg(0, ios::end); //go to 0 bytes from end

//calculate number of persons return (int)infile.tellg() / sizeof(person);

}

int main()

{

person p; //make an empty person

char ch;

do { //save persons to disk cout << "Enter data for person:";

p.getData(); //get data

p.diskOut(); //write to disk

cout << "Do another (y/n)? ";

cin >> ch;

} while(ch=='y'); //until user enters ‘n’

int n = person::diskCount(); //how many persons in file? cout << "There are " << n << " persons in file\n";

for(int j=0; j<n; j++) //for each one,

{

cout << "\nPerson " << j;

p.diskIn(j); //read person from disk

p.showData(); //display person

}

cout << endl; return 0;

}

## Λάθη κατά τον χειρισμό αρχείων

Όταν ανοίγουμε ένα αρχείο που δεν υπάρχει, ή όταν δεν υπάρχει χώρος στον δίσκο για να γράψουμε υπάρξει πρόβλημα. Θα πρέπει λοιπόν το πρόγραμμά μας να είναι έτοιμο να τα αντιληφθεί και να αντιδράσει καταλλήλως. Η C++ μας παρέχει μια σειρά συναρτήσεων για τον έλεγχο των λαθών, όπως φαίνεται παρακάτω.

#include <fstream>

#include <iostream>

using namespace std;

int main()

{

ifstream file; file.open("test.dat");

if( !file )

cout << "\nCan’t open file";

else

cout << "\nFile opened successfully.";

cout << "\nfile = " << file;

cout << "\nError state = " << file.rdstate();

cout << "\ngood() = " << file.good(); cout << "\neof() = " << file.eof(); cout << "\nfail() = " << file.fail();

cout << "\nbad() = " << file.bad() << endl; file.close();

system("pause"); return 0; }

|  |
| --- |
| **Ανοικτά Ακαδημαϊκά Μαθήματα****Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας** |
| **Τέλος Ενότητας** |
| **Χρηματοδότηση*** Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
* Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Αθήνας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
* Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

 |

**Σημειώματα**

**Σημείωμα Αναφοράς**

Copyright ΤΕΙ Αθήνας, Κλειώ Σγουροπούλου, 2014. Κλειώ Σγουροπούλου. «Αντικειμενοστρεφής Προγραμματισμός». Ενότητα 7: «Streams». Έκδοση: 1.0. Αθήνα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: [ocp.teiath.gr](https://ocp.teiath.gr/).

**Σημείωμα Αδειοδότησης**

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

* που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
* που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
* που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

**Διατήρηση Σημειωμάτων**

* Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
* Το Σημείωμα Αναφοράς
* Το Σημείωμα Αδειοδότησης
* Τη δήλωση Διατήρησης Σημειωμάτων
* Το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει) μαζί με τους συνοδευόμενους υπερσυνδέσμους.