Προγραμματισμός Υπολογιστών με C++

ΑΡΧΕΙΑ & ΡΕΥΜΑΤΑ

Γεώργιος Παπαϊωάννου (2013-20)
gepap@aueb.gr

Περιεχόμενο Παρουσίασης

- Περιγραφή:
  - Τύποι αρχείων
  - Ρεύματα
  - Άνοιγμα ρευμάτων
  - Εγγραφή και ανάγνωση δεδομένων προς/από ένα ρεύμα
  - Διαχείριση και κλείσιμο ρευμάτων
  - Παραδείγματα

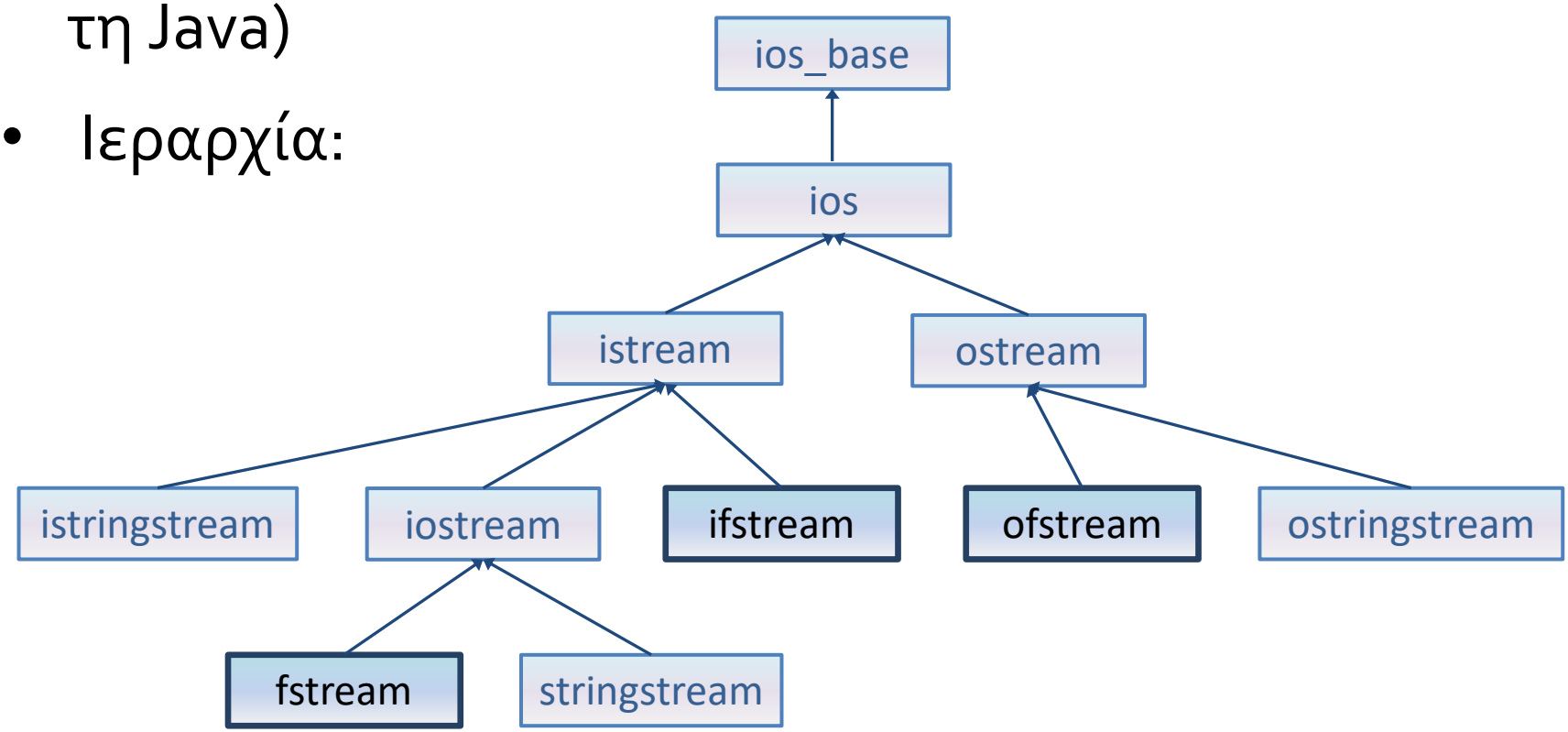
- Τελευταία ενημέρωση: Οκτώβριος 2020

Τα αρχεία στη C++

- Γενικά έχουμε 2 ειδών αρχεία:
  - Κειμένου
  - Δυαδικά (binary)
- Τα οποία, όπως και στη Java, μπορώ να ανοίξω είτε για ανάγνωση (read), είτε για εγγραφή (write/append), αλλά και για τα δύο
- Η C και η C++ μου προσφέρουν πολλούς τρόπους πρόσβασης στα αρχεία, από πολύ χαμηλό επίπεδο, μέχρι το ίδιο υψηλό με τη Java

Ρεύματα Εισόδου / Εξόδου

- Υψηλού επιπέδου αναπαράσταση αρχείων, σωληνώσεων και διαύλων επικοινωνίας (ίδια λογική με τη Java)
- Ιεραρχία:



Κωδικοποίηση Ονομάτων

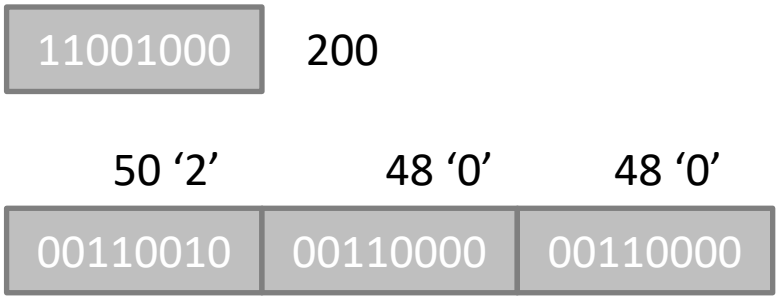
- [i/o] [f/string] stream
  - i/o: input / output. io ή τίποτα: κάνουν και για είσοδο και για έξοδο
  - f/string: Εξειδίκευση στον τύπο του stream: file / string.
- Ένα ρεύμα τύπου string αποθηκεύει και διαβάζει τα δεδομένα σε ένα buffer χαρακτήρων στη μνήμη
- Ένα ρεύμα τύπου αρχείου αποθηκεύει και διαβάζει τα δεδομένα στο σύστημα αρχείων

Η Αναπαράσταση των Ρευμάτων

- Τα ρεύματα μπορεί να περικλείουν σύνθετες δομές ελέγχου και συγχρονισμού, αλλά κατ ελάχιστο, εκφράζουν μια σειριακή δομή αποθήκευσης με:
  - Έναν προσδιοριστή τρέχουσας θέσης στο ρεύμα
  - Ένα σύνολο από σημαίες ελέγχου κατάστασης

Αναπαράσταση Δεδομένων στα Ρεύματα

- Έχει μεγάλη σημασία πως έχουμε αποθηκεύσει τα δεδομένα μας σε ένα ρεύμα, διότι καθορίζει τον τρόπο με τον οποίο περιμένουμε να τα διαβάσουμε και το αντίστροφο
- Για παράδειγμα, έναν αριθμό μπορούμε να τον αποθηκεύσουμε ως κείμενο, αλλά και με την δυαδική του αναπαράσταση:



Άνοιγμα Αρχείου ως Ρεύμα

- Ανοίγουμε ένα αρχείο μέσω ρεύματος εισόδου ή εξόδου αρχικοποιώντας ένα αντικείμενο τύπου fstream/ifstream/ofstream:

```
#include <fstream>

#using namespace std;

…

string filename = "log.data";

ofstream file(filename, ofstream::out | ofstream::binary);
```

Mode: «μάσκα» επιλογών

```
ifstream file2("input_report.txt" );
```

Mode: εδώ χρησιμοποιούμε προεπιλεγμένες επιλογές

Άνοιγμα Αρχείου για Ανάγνωση

```
ifstream ( const char* filename, ios_base::openmode mode );
```

ή

```
ifstream ();

open( const char* filename, ios_base::openmode mode );
```

- Το mode δίνει παραμέτρους για το άνοιγμα, όπως:
  - ios_base::binary : Ανοίγει το αρχείο ως δυαδικό αντί για αρχείο κειμένου
  - ios_base::app : Προσθέτει (appends) δεδομένα στο τέλος αντί να διαγράφει τα υπάρχοντα
  - ios_base::in / out : Δηλώνει αν στο αρχείο πρόκειται να γράψουμε ή να διαβάσουμε (ή και τα δύο). Ένα ρεύμα ifstream είναι εξ ορισμού ios_base::in αλλά μπορεί να είναι επιπρόσθετα και ios_base::out
  - Μπορώ να συνδυάσω παραμέτρους

Άνοιγμα Αρχείου για Εγγραφή

```
ofstream ( const char* filename, ios_base::openmode mode );
```

ή

```
ofstream ();

open( const char* filename, ios_base::openmode mode );
```

- Το mode δίνει παραμέτρους για το άνοιγμα, όπως:
  - ios_base::binary : Ανοίγει το αρχείο ως δυαδικό αντί για αρχείο κειμένου
  - ios_base::app : Προσθέτει (appends) δεδομένα στο τέλος αντί να διαγράφει τα υπάρχοντα
  - ios_base::in / out : Δηλώνει αν στο αρχείο πρόκειται να γράψουμε ή να διαβάσουμε (ή και τα δύο). Ένα ρεύμα ofstream είναι εξ ορισμού ios_base::out αλλά μπορεί να είναι επιπρόσθετα και ios_base::in
  - Μπορώ να συνδυάσω παραμέτρους

Κλείσιμο Ρεύματος Αρχείου

- Ειδικά για τα ρεύματα αρχείων (ifstream, ofstream, fstream), υπάρχει η μέθοδος close() που κλείνει και αποδεσμεύει το αρχείο από την εφαρμογή μας
- Είναι απαραίτητο για την ομαλή λειτουργία της εφαρμογής και της συνεργασίας της με άλλες να κλείνουμε τα ρεύματα αρχείων αμέσως μόλις διεκπεραιώνουμε μια εργασία με αυτά
- Το ίδιο κάνουμε και όταν συμβεί κάποιο σφάλμα και πρέπει να τερματίσουμε τη διαδικασία ανάγνωσης / εγγραφής

Έλεγχος Εγκυρότητας ενός Ρεύματος

- bool fail() : Επιστρέφει true αν συνέβη κάποιο σφάλμα κατά την πρόσβαση στο ρεύμα (π.χ. έκλεισε, διαγράφηκε το σχετικό αρχείο)
- bool eof() : Επιστρέφει true αν κατά την ανάγνωση του ρεύματος έχουμε φτάσει στο τέλος του
- bool good() : Επιστρέφει true όσο καμία από τις παραπάνω συνθήκες δεν έχει ενεργοποιηθεί → Το ρεύμα είναι έγκυρο

Εγγραφή Δεδομένων σε Δυαδικό Ρεύμα (ostream)

- Η εγγραφή δυαδικών δεδομένων γίνεται πάντα κατά bytes ή αλλιώς, με δεδομένα τύπου char (1 byte)
- Υπάρχουν δύο δυνατότητες:
  - ostream& put(char c) : γράψιμο ενός μεμονωμένου byte
  - ostream& write(const char* s, streamsize n) : γράψιμο ενός συνεχόμενου block από bytes (πίνακα)
- Η δεύτερη δυνατότητα είναι πολύ ταχύτερη αν έχουμε πολλά δεδομένα

Ανάγνωση από Δυαδικό Ρεύμα (istream)

- Η ανάγνωση δυαδικών δεδομένων γίνεται πάντα κατά bytes ή αλλιώς, με δεδομένα τύπου char (1 byte)
- Υπάρχουν (τουλάχιστο) τρεις δυνατότητες:
  - int get() : διάβασμα ενός μεμονωμένου byte

    istream& get(char& c)
  - istream& read(char* s, streamsize n) : διάβασμα το πολύ n bytes σε δεσμευμένο χώρο (υπάρχων πίνακας)
  - int peek() : διάβασμα ενός μεμονωμένου byte χωρίς να προχωρήσει η τρέχουσα θέση στο ρεύμα (in-place)!
  - … άλλες παραλλαγές της get
- Η δεύτερη δυνατότητα είναι πολύ ταχύτερη αν έχουμε πολλά δεδομένα

Άλλες Μέθοδοι Ανάγνωσης

Διάβασμα μιας «γραμμής» χαρακτήρων. Ως γραμμή νοείται μια ακολουθία από το πολύ n χαρακτήρες που τερματίζονται από τον ειδικό χαρακτήρα “end-of-line” (‘\n’) ή κάποιον άλλο χαρακτήρα που προσδιορίζει ο χρήστης (βλ. παρακάτω το όρισμα delim):

istream& getline (char* s, streamsize n);

istream& getline (char* s, streamsize n, char delim);

Μέθοδοι Διαχείρισης Ρεύματος

- std::streampos tellg()
- Μας επιστρέφει την τρέχουσα θέση μας μέσα στο ρεύμα
- istream& seekg (std::streampos pos)
- Μετακινούμε την τρέχουσα θέση ανάγνωσης / εγγραφής πριν από το στοιχείο στη θέση pos

Παράδειγμα Εγγραφής σε Αρχείο

```
#include <fstream>   // Χρειάζεται για τα ρεύματα αρχείων
#include <iostream>  // Χρειάζεται για την έξοδο κειμένου στη γραμμή εντολών
using namespace std;

void writeShortValue(string filename, unsigned short & value){
    ofstream file(filename, ios::binary);
    if (!file) {
        cerr << "Cannot open file" << endl;
        return;
    }

    file.write((char*)&value, 2); // Χρησιμοποίησε τη διεύθυνση του value ως
                                  // τη διεύθυνση ενός πίνακα 2 bytes.
                                  // Γράψε τα 2 bytes του short στο αρχείο

    if (file.fail()){
        cerr << "Could not write data" << endl;
    }

    file.close();
    return;
}
```

Παράδειγμα Ανάγνωσης από Αρχείο

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

unsigned short readShortValue(string filename){
    unsigned short value; // short: 2 bytes
    ifstream file(filename, ios::binary);
    if (!file) {
        cerr << "Cannot open file" << endl;
        return 0;
    }
    file.read((char*)&value, 2); // Χρησιμοποίησε τη διεύθυνση του value ως
                                 // τη διεύθυνση ενός πίνακα 2 bytes.
                                 // Διάβασε 2 bytes σε αυτό το χώρο στη μνήμη

    if (file.fail()){
        cerr << "Could not read data" << endl;
        return 0;
    }
file.close(); return value;
}
```

Παρατήρηση: Μίξη Κειμένου/Δυαδικών Δεδομένων

- Αν έχουμε ανοίξει ένα αρχείο ως ρεύμα κειμένου, συνεχίζουμε να μπορούμε να γράψουμε σε αυτό δεδομένα σε δυαδική μορφή ανά πάσα στιγμή και το αντίστροφο.

Τελεστές Εισαγωγής και Εξαγωγής Κειμένου (1)

- Στα ρεύματα, ανάλογα με το είδος τους (i/o) ορίζονται 2 τελεστές:
- >> (τελεστής εξαγωγής): Διαβάζει κείμενο από το αρχείο και το μετατρέπει στον κατάλληλο τύπο που ακολουθεί, ενδεχομένως κάνοντας συμβολική ανάλυση του κειμένου που διάβασε (π.χ. για μετατροπή του κειμένου «1234.05» στον αριθμό float 1234.05)
- << (τελεστής εισαγωγής): Αποθηκεύει σε μορφή κειμένου έναν από τους βασικούς τύπους

Τα Ρεύματα cout, cin και cerr

- Κληρονομούν από istream και ostream
- Το cout το χρησιμοποιούμε για να γράψουμε κείμενο (π.χ. αποτελέσματα) στην γραμμή εντολών
- Το cin το χρησιμοποιούμε για να διαβάσουμε δεδομένα από τη γραμμή εντολών
- Το cerr το χρησιμοποιούμε για να εξάγουμε μηνύματα λάθους στη γραμμή εντολών

Τελεστές Εισαγωγής και Εξαγωγής Κειμένου (2)

```
istream& operator>> (bool& val);
istream& operator>> (short& val);
istream& operator>> (unsigned short& val);
istream& operator>> (int& val);
istream& operator>> (unsigned int& val);
istream& operator>> (long& val);
istream& operator>> (unsigned long& val);
istream& operator>> (float& val);
istream& operator>> (double& val);
istream& operator>> (long double& val);
istream& operator>> (void*& val);
istream& operator>> (streambuf* sb );
istream& operator>> (istream& (*pf)(istream&));
...
```

```
ostream& operator<< (bool val);
ostream& operator<< (short val);
ostream& operator<< (unsigned short val);
ostream& operator<< (int val);
ostream& operator<< (unsigned int val);
ostream& operator<< (long val);
ostream& operator<< (unsigned long val);
ostream& operator<< (long long val);
ostream& operator<< (unsigned long long val);
ostream& operator<< (float val);
ostream& operator<< (double val);
ostream& operator<< (long double val);
ostream& operator<< (void* val);
ostream& operator<< (streambuf* sb );
...
```

Εισαγωγή Κειμένου σε Ρεύμα (1)

```
void writeFloat(string filename, float & value)
{
    ofstream file(filename);
    if (!file) {
        cerr << "Cannot open file" << endl;
    return;
    }

    setlocale(LC_ALL,"Greek");

    file << "Γράψαμε τον αριθμό " << value << endl;

    if (file.fail()){
        cerr << "Could not write data" << endl;
    }

    file.close();
    return;
}
```

Απαραίτητο για να γράψουμε σε Ελληνικά στο αρχείο (και στη γραμμή εντολών)

Ο αριθμός μορφοποιείται σε κείμενο και εγγράφεται στο αρχείο με βάση το locale του υπολογιστή

Μορφοποιημένο Κείμενο σε Ρεύμα

- Το αποτέλεσμα της διαδικασίας είναι ένα αρχείο με περιεχόμενο:

```
Γράψαμε τον αριθμό 2048.1
```

Κατά την μετατροπή του αριθμού σε χαρακτήρες, χρησιμοποιήθηκε συγκεκριμένη ακρίβεια αναπαράστασης, η οποία έκοψε κάποια δεκαδικά ψηφία!

- Για να αλλάξουμε την ακρίβεια αναπαράστασης των αριθμών στο ρεύμα, έχουμε ειδικές συναρτήσεις

Εξαγωγή Κειμένου από Ρεύμα

```
float readFloat(string filename) {
    float value;
    ifstream file(filename);
    if (!file) {
        return 0.0f;
    }
    setlocale(LC_ALL,"Greek");
    stringbuf buffer;
    file.get(buffer, 'ό');
    file.get(buffer, ' ');
    file >> value;
    cout << buffer.str() << " " << value;
    if (file.fail()){
        file.close(); return 0.0f;
    }
    file.close();
    return value;
}
```

Απαραίτητο για να καταλάβει ότι διαβάζει Ελληνικούς χαρακτήρες

Θα διαβάσουμε κείμενο άγνωστου μήκους. Ο stringbuf (string buffer) χρησιμεύει ακριβώς σε αυτό

Διαβάζουμε κείμενο μέσα στον buffer μέχρι να συναντήσουμε κάποιο μοναδικό σύμβολο πριν τον αριθμό

Επειδή το σύμβολο ‘ό’ δεν έχει εξαχθεί από το ρεύμα, πρέπει να διαβάσουμε κι άλλο ένα στοιχείο για να μπορέσει να διαβαστεί μετά ο αριθμός σωστά (“ 1000.1” και όχι “ό 1000.1”)

Συναρτήσεις Μορφοποίησης Κειμένου Ρευμάτων

- streamsize precision (streamsize prec);

  Αριθμός ψηφίων μετά την υποδιαστολή

- streamsize width (streamsize wide);

  Ολικό μήκος σε χαρακτήρες που καταλαμβάνει ο αριθμός

- char fill (char fillch);

  Ο χαρακτήρες με τον οποίο θα γεμίσει το κενό που μένει μεταξύ του πραγματικού μήκους του αριθμού και του προσδιορισμένου πλάτους του (συνάρτηση width)

Ειδική Μορφοποίηση Αριθμών (manipulators)

- Μπορούμε να προσδιορίσουμε τη σχετική τοποθέτηση του αριθμού μέσα στο χώρο του width:

```
int n = -77;
std::cout.width(6); std::cout << std::internal << n << '\n';
std::cout.width(6); std::cout << std::left << n << '\n';
std::cout.width(6); std::cout << std::right << n << '\n';
```

```
-   77
-77
   -77
```

- Μπορούμε να αλλάξουμε τη μορφή αναπαράστασης των αριθμών κινητής υποδιαστολής:

```
std::fixed
std::scientific
```

- Μπορούμε να αλλάξουμε τη βάση των αριθμών:

```
std::dec
std::hex
std::oct
```

Αρχεία στη C (1)

- Στη C, τα αρχεία τα διαχειριζόμαστε μέσω file pointers
- Γιατί χρειάζομαι και δεύτερο τρόπο για να χειρίζομαι αρχεία;
  - Συμβατότητα με μη αντικειμενοστραφή κώδικα
  - Μεγαλύτερη ταχύτητα (χαμηλότερου επιπέδου)
  - Απλούστερος!

Αρχεία στη C (2)

- Στη C, τα αρχεία τα διαχειριζόμαστε μέσω file pointers
- Βασικός τύπος «αρχείο»: FILE
  - Βρίσκεται στο header file stdio.h
  - Κάνω `#include <stdio.h>`

- Δήλωση ενός αρχείου:
  - `FILE *fp = nullptr;`

Άνοιγμα Αρχείου στη C

```
FILE * fp = fopen("c:\\temp\\myfile", "wt");
```

- Η fopen (file open), ανοίγει το αρχείο ποτ δίνεται στο πρώτο όρισμα με τρόπο προσπέλασης που δηλώνεται στο δεύτερο όρισμα:
  - w: write
  - r: read
  - a: append
  - b: binary
  - t: text

Κλείσιμο Αρχείου στη C

```
fclose( FILE * fp);
```

- Σημείωση:
  - Μετά την κλήση της fopen(), ελέγχουμε αν ορθά άνοιξε το αρχείο μας ζητώντας ο File pointer να είναι διάφορος του NULL:

```
if (fp==NULL)
   printf("error opening file\n");
```

Γράψιμο κειμένου σε αρχείο της C

```
FILE *fp;
fp = fopen("c:\\temp\a.txt","wt");
if (fp)
   fprintf(fp, "This is a No%d burger\n", 2);
fclose(fp);
```

- Εδώ, χρησιμοποιήσαμε μια παραλλαγή της printf, την fprintf
  - Συντάσσεται όπως και η printf
  - Παίρνει μία ακόμα παράμετρο (την πρώτη) που είναι ο FILE pointer

Διάβασμα κειμένου από αρχείο της C

- Και πάλι, χρησιμοποιούμε παραλλαγή της scanf, την fscanf για μορφοποιημένη εισαγωγή κειμένου

```
float a, b;
int num_args = fscanf(fp, "%f%*[ ,\t]%f\)", &a, &b);
```

- Άλλη χρήσιμη συνάρτηση που διαβάζει κατά μέγιστο n χαρακτήρες:

```
char buffer[100];
fgets(buffer,100,fp);
```

Δυαδική Εγγραφή σε Αρχεία της C

- `size_t fwrite ( const void * ptr, size_t size, size_t count, FILE * stream );`
- Παράδειγμα:

```
int main ()
{
   FILE * pFile;
   char buffer[] = { 'x' , 'y' , 'z' };
   pFile = fopen ("myfile.bin", "wb");
   fwrite (buffer , sizeof(char), sizeof(buffer), pFile);
   fclose (pFile);
   return 0;
}
```

Δυαδική Ανάγνωση από Αρχεία της C (1)

```
size_t fread ( void * ptr, size_t size, size_t count, FILE * stream );
```

- Παράδειγμα:

```
int main ()
{
     FILE * pFile;
     long lSize;
     char * buffer;
     size_t result;
     pFile = fopen ( "myfile.bin" , "rb" );
     if (pFile==NULL)
    {
          fputs ("File error",stderr);
          exit (1);
   }
```

Δυαδική Ανάγνωση από Αρχεία της C (2)

```
// Ας βρούμε το μέγεθος του αρχείου:
  fseek (pFile , 0 , SEEK_END);
  lSize = ftell (pFile);
  rewind (pFile);
  buffer = (char*) malloc (sizeof(char)*lSize);
  if (buffer == NULL)
  {
       fputs ("Memory error",stderr);
       exit (2);
  }
  result = fread (buffer,1,lSize,pFile);
  fclose (pFile);
  free (buffer);
  return 0;
}
```