Κεφάλαιο 16

ΠΑΡΑΡΤΗΜΑ

16.1 Εισαγωγή

16.2 Εντολές φυσικού επιπέδου σε Pascal

16.3 Εντολές φυσικού επιπέδου σε C/C++

16.4 Εντολές φυσικού επιπέδου σε C++

Κεφάλαιο 16

ΠΑΡΑΡΤΗΜΑ

16.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται μία σύντομη περιγραφή των εντολών χαμηλού επιπέδου, οι οποίες παρέχονται από τις γλώσσες προγραμματισμού Pascal και C/C++. Το βοήθημα αυτό δίνεται ώστε ο αναγνώστης να έχει συγκεντρωμένο το υλικό, που απατείται για την υλοποίηση των μεθόδων που εξετάσθηκαν στα προηγούμενα κεφάλαια. Θα ήταν χρήσιμο ο αναγνώστης να ανατρέξει σε σχετικά λεπτομερέστερα βιβλία και εγχειρίδια για να εμβαθύνει στα θέματα αυτά.

16.2 Εντολές φυσικού επιπέδου σε Pascal

Τύποι και δηλώσεις μεταβλητών αρχείων

Υπάρχουν 3 τύποι αρχείων:

- var FileVar: Text;
  Δηλώνει *αρχείο κειμένου* και περιέχει γραμμές χαρακτήρων ASCII.
- var FileVar: file of DataRecord;
  Δηλώνει *αρχείο καθορισμένου τύπου* και περιέχει δεδομένα ενός μόνο συγκεκριμένου τύπου.
- var FileVar: file;
  Δηλώνει *αρχείο μη καθορισμένου τύπου* και περιέχει οποιαδήποτε δεδομένα με οποιοδήποτε μήκος.

Ορισμός φυσικού αρχείου

Η αντιστοίχιση μίας μεταβλητής τύπου αρχείου με ένα πραγματικό αρχείο γίνεται με την εντολή:

- Assign(FileVar, 'file_name');

Το αλφαριθμητικό `file_name´ μπορεί να περιέχει και το μονοπάτι (path) του αρχείου. Η εντολή Assign καλείται μόνο μία φορά στο πρόγραμμα.

`Ανοιγμα αρχείου

Το άνοιγμα του αρχείου μπορεί να γίνει με 3 τρόπους:

- Reset(FileVar);
  Ανοίγει υπάρχον αρχείο για ανάγνωση, αρχίζοντας από την αρχή. Αν το αρχείο δεν υπάρχει, τότε προκαλείται λάθος.

- Rewrite(FileVar);
  Ανοίγει αρχείο για αποθήκευση, αρχίζοντας από την αρχή. Αν το αρχείο υπάρχει ήδη, τότε το σβήνει, αλλιώς το δημιουργεί.

- Append(FileVar);
  Ανοίγει αρχείο για αποθήκευση με προσάρτηση, δηλαδή αρχίζοντας από το τέλος του. Αν δεν υπάρχει, τότε προκαλείται λάθος.

Ειδικότερα για τα αρχεία καθορισμένου και μη καθορισμένου τύπου, το άνοιγμα με τη συνάρτηση Rewrite επιτρέπει και την ανάγνωση από το αρχείο.

Κλείσιμο αρχείου

`Ενα αρχείο κλείνει με την εντολή:

- Close(FileVar);

Ωστόσο, ακόμα και αν δεν υπάρχει ρητή εντολή κλεισίματος, κάθε αρχείο κλείνει αυτόματα στο τέλος του προγράμματος.

Ανάγνωση και αποθήκευση σε αρχεία κειμένου

Η ανάγνωση τιμών μπορεί να γίνει με 2 εντολές:

- Read(FileVar, Var1, Var2, ...);
  Διαβάζει όλες μεταβλητές Var1, Var2, ... από την τρέχουσα γραμμή του αρχείου.

- Readln(FileVar, Var1, Var2, ...);
  Διαβάζει όλες τις μεταβλητές Var1, Var2, ... από την τρέχουσα γραμμή του αρχείου. Μετά την ανάγνωση τοποθετεί το δείκτη αρχείου στην αρχή της επόμενης γραμμής.

Η Readln αγνοεί τυχόντα υπάρχοντα δεδομένα, που μεσολαβούν μέχρι την αρχή της επόμενης γραμμής.

Και οι δύο διαδικασίες ανάγνωσης από αρχείο κειμένου σταματούν την ανάγνωση από την τρέχουσα γραμμή, όταν αναγνωσθεί ο χαρακτήρας αλλαγής γραμμής (Eoln). Επίσης, σταματούν την ανάγνωση από το αρχείο, όταν αναγνωσθεί ο χαρακτήρας τέλους αρχείου (Eof). Οι δύο αντίστοιχες συναρτήσεις για την ανίχνευση αυτών των χαρακτήρων είναι:

- Eoln(FileVar);
  Αν ο χαρακτήρας που πρόκειται να αναγνωσθεί είναι ο χαρακτήρας αλλαγής γραμμής (Eoln), τότε επιστρέφει True,

- Eof(FileVar);
  Αν ο χαρακτήρας που πρόκειται να αναγνωσθεί είναι ο χαρακτήρας τέλους αρχείου (Eof), τότε επιστρέφει True.

Η αποθήκευση τιμών μπορεί να γίνει με δύο εντολές:

- Write(FileVar, Var1, Var2, ...);
  Γράφει τις μεταβλητές Var1, Var2, ...

- Writeln(FileVar, Var1, Var2, ...);
  Γράφει τις μεταβλητές Var1, Var2, ... και προσθέτει και το χαρακτήρα αλλαγής γραμμής (Eoln).

Η αποθήκευη με τις εντολές Write και Writeln δεν γίνεται αυτομάτως στο αρχείο αλλά σε απομονωτικές μνήμες, τα περιεχόμενα των οποίων γράφονται στο αρχείο με ευθύνη του λειτουργικού συστήματος. Για τη ρητή αποθήκευση των περιεχομένων των απομονωτικών μνημών στο αρχείο, χρησιμοποιείται η εντολή:

- Flush(FileVar);

Η εντολή Flush χρησιμοποιείται με αρχεία κάθε τύπου για την εξασφάλιση της αποθήκευσης υπολογισμών, έτσι ώστε σε περίπτωση λάθους σε επόμενο στάδιο, να έχει γίνει τουλάχιστον μερική καταχώρηση των αποτελεσμάτων στο αρχείο.

Ανάγνωση και αποθήκευση σε αρχεία καθορισμένου τύπου

Χρησιμοποιούνται οι διαδικασίες Read και Write με όμοιο τρόπο όπως στα αρχεία κειμένου. Όμως, στην περίπτωση αυτή δεν χρησιμοποιούνται οι διαδικασίες Readln και Writeln. Επίσης, μπορεί να χρησιμοποιηθεί η συνάρτηση Eof, αλλά όχι η Eoln.

Ανάγνωση και αποθήκευση σε αρχεία μη καθορισμένου τύπου

Η ανάγνωση και αποθήκευση στα αρχεία μη καθορισμένου τύπου γίνεται όχι κατά εγγραφές αλλά κατά τμήματα (blocks). Οι αντίστοιχες εντολές είναι:

- BlockRead(FileVar, Buffer, Count, NumRead);
- BlockWrite(FileVar, Buffer, Count, NumCount);

Η μεταβλητή Buffer περιέχει τα δεδομένα που θα αναγνωσθούν/αποθηκευθούν από/προς το δίσκο. Η μεταβλητή Count δηλώνει τον αριθμό των αντικειμένων, ενώ οι μεταβλητές NumRead και NumCount περιέχουν τον επακριβή αριθμό αντικειμένων που αναγνώσθηκαν και αποθηκεύθηκαν, αντίστοιχα. Σε περίπτωση κανονικής ανάγνωσης ή αποθήκευσης, οι τιμές αυτών των μεταβλητών θα είναι ίσες με την τιμή της Count, ενώ σε περίπτωση λάθους θα είναι διαφορετικές.

Το πλήθος των αντικειμένων που δηλώνεται με τη μεταβλητή Count ορίζεται με βάση το μέγεθος του block, που εξ ορισμού είναι 128 bytes. Για να γίνεται πιο κατανοητή η χρήση της ανάγνωσης από αρχεία μη καθορισμένου τύπου συνηθίζεται η αλλαγή του μεγέθους του block σε 1 byte. Αυτό γίνεται κατά το άνοιγμα του αρχείου με την προσθήκη μίας ακόμη παραμέτρου:

- Reset(FileVar, 1);
- Rewrite(FileVar, 1);

Τυχαία αναζήτηση σε αρχεία

Σε σειριακό αρχείο οι αναγνώσεις και οι αποθηκεύσεις γίνονται σε ακολουθία, από την αρχή ως το τέλος. Σε αρχείο τυχαίας προσπέλασης, η επόμενη ανάγνωση ή αποθήκευση μπορεί να γίνει σε οποιοδήποτε σημείο του αρχείου. Αυτό δεν μπορεί να γίνει για τα αρχεία τύπου κειμένου. Η διαδικασία Seek χρησιμοποιείται για την τοποθέτηση του δείκτη αρχείου στην επιθυμητή θέση. Η τρέχουσα τιμή του δείκτη αρχείου ανακτάται με τη συνάρτηση FilePos:

- Seek(FileVar, Offset);
- offset=FilePos(FileVar);

Αν το αρχείο έχει ανοιχθεί με την εντολή Append, τότε δεν μπορεί να γίνει τυχαία αναζήτηση. Επομένως, αν το αρχείο ανοιχθεί με την εντολή Rewrite και χρειάζεται προσθήκη δεδομένων στο τέλος του, μετακινείται πρώτα ο δείκτης στην θέση τέλους και μετά η αποθήκευση εκτελείται κανονικά. Η μετακίνηση γίνεται με την εντολή:

- Seek(FileVar, FileSize(FileVar));

Παράδειγμα

```
Program Example;

type
      RecordExm = Record
                     id : word;
                     value : real;
                  End;

var
      TextFile : Text;
      TypedFile : File of RecordExm;
      NonTypedFile1, NonTypedFile2 : File;

      Str80 : string[80];
      Rec : RecordExm;
      Buffer : array[ 1..512] of char;
      NumRead, NumWrite : word;
```

```
Begin

(* - - - - - - - - - - - - - - - - - - Αρχείο κειμένου - - - - - - - - - - - - - - - - - - *)

(* Ανάθεση αρχείου κειμένου *)
Assign(TextFile, 'example.txt');

(* Άνοιγμα αρχείου κειμένου για ανάγνωση *)
Reset(TextFile);

(* Ανάγνωση από το αρχείο κειμένου και κλείσιμό του *)
Readln(TextFile, Str80);
Close(TextFile);

(* Άνοιγμα του ίδιου αρχείου κειμένου για αποθήκευση. Δεν μεσολαβεί
δεύτερη κλήση της Assign *)
Rewrite(TextFile);

(* Αποθήκευση στο αρχείο κειμένου και κλείσιμό του *)
Writeln(TextFile, Str80); Close(TextFile);

(* - - - - - - - - - - - - - - Αρχείο καθορισμένου τύπου - - - - - - - - - - - - - - *)

(* Ανάθεση αρχείου καθορισμένου τύπου *)
Assign(TypedFile, 'example.dat');

(* Άνοιγμα αρχείου καθορισμένου τύπου για ανάγνωση και αποθήκευση *)
Rewrite(TypedFile);

(* Αποθήκευση μίας εγγραφής στο τέλος του αρχείου *)
rec.id = 100;
rec.val = 123.45;

Seek(TypedFile, FileSize(TypedFile));
Write(TypedFile, rec);

(* κλείσιμο αρχείου καθορισμένου τύπου *)
Close(TypedFile);

(* - - - - - - - - - - - - - Αρχεία μη καθορισμένου τύπου - - - - - - - - - - - - - *)

(* Άνοιγμα δύο αρχείων μη καθορισμένου τύπου για αντιγραφή των περιε-
χομένων του ενός στο άλλο *)

Assign(NonTypedFile1, 'source');
Assign(NonTypedFile2, 'destination');
```

```
Reset(NonTypedFile1);                          (* για ανάγνωση *)
Rewrite(NonTypedFile2);                        (* για αποθήκευση *)

Repeat
        BlockRead(NonTypedFile1, Buffer, SizeOf(Buffer), NumRead);
        BlockWrite(NonTypedFile1, Buffer, SizeOf(Buffer), NumWrite);
Until (NumRead=0) or (NumRead <> NumWrite);

(* Χρησιμοποιούνται οι τιμές των πραγματικών αριθμών bytes που αναγνώ-
σθηκαν και αποθηκεύθηκαν για να γίνει έλεγχος λάθους. Ο βρόχος τερματί-
ζει όταν δεν υπάρχουν άλλα δεδομένα εισόδου (NumRead=0) ή όταν συμβεί
λάθος κατά την αντιγραφή (NumRead <> NumWrite) *)

(* κλείσιμο αρχείων *)
Close(NonTypedFile1);
Close(NonTypedFile1);

End.
```

Σύνοψη εντολών αρχείων στην Pascal

Ο επόμενος πίνακας συνοψίζει τις διαθέσιμες συναρτήσεις για τις διάφορες λειτουργίες επί των διαφόρων τύπων αρχείων.

Λειτουργία	Υποροντίνα	Κειμένου	Καθ. τύπου	Μη καθ. τύπου
Άνοιγμα	Append	Ναι	Όχι	Όχι
	Assign	Ναι	Ναι	Ναι
	Reset	Ναι	Ναι	Ναι
	Rewrite	Ναι	Ναι	Ναι
	Close	Ναι	Ναι	Ναι
Είσοδος	BlockRead	Όχι	Όχι	Ναι
	Read	Ναι	Ναι	Όχι
	Readln	Ναι	Ναι	Όχι
Έξοδος	BlockWrite	Όχι	Όχι	Ναι
	Write	Ναι	Ναι	Όχι
	Writeln	Ναι	Ναι	Όχι
Έλεγχος θέσης	Eof	Ναι	Ναι	Ναι
	Eoln	Ναι	Όχι	Όχι
Τυχαίας	FilePos	Όχι	Ναι	Ναι
Προσπέλασης	FileSize	Όχι	Ναι	Ναι
	Seek	Όχι	Ναι	Ναι

16.3 Εντολές φυσικού επιπέδου σε C/C++

Καθώς υπάρχει συμβατότητα μεταξύ των γλωσσών C και C++, οι εντολές της C μπορούν κάλλιστα να χρησιμοποιηθούν και στη C++. Στο Κεφάλαιο 16.4, θα δοθούν οι εντολές που αφορούν αποκλειστικά στη C++.

Τύποι και δηλώσεις μεταβλητών αρχείων

Η δήλωση μίας μεταβλητής αρχείου στη γλώσσα C προϋποθέτει τη χρήση του τύπου δεδομένων FILE. Πιο συγκεκριμένα, απαιτείται η δήλωση μίας μεταβλητής, που είναι δείκτης σε έναν τύπο FILE. Η δήλωση έχει ως εξής:

- FILE * f;

Η μεταβλητή f χρησιμοποιείται για να πραγματοποιηθούν προσπελάσεις στο αρχείο.

`Ανοιγμα αρχείου

Πριν από οποιαδήποτε λειτουργία με το αρχείο, πρέπει πρώτα να γίνουν οι απαραίτητες αρχικοποιήσεις, όπως για παράδειγμα ο ορισμός του ονόματος του αρχείου. Οι αρχικοποιήσεις πραγματοποιούνται με τη βοήθεια της συνάρτησης fopen. Η επακριβής σύνταξη της συνάρτησης έχει ως εξής:

- FILE *fopen(const char *fname, const char *access_mode);

όπου fname είναι το όνομα του φυσικού αρχείου και access_mode είναι τα χαρακτηριστικά προσπέλασης που θα έχει το αρχείο (για παράδειγμα, μόνο για ανάγνωση, ή για ανάγνωση/αποθήκευση, κλπ.). Οι διαφορετικές τιμές της μεταβλητής access_mode συνοψίζονται στον ακόλουθο πίνακα:

Τρόπος προσπέλασης	Λειτουργία
r	Ανοίγει το αρχείο μόνο για ανάγνωση.
w	Ανοίγει το αρχείο για αποθήκευση. Αν το αρχείο υπάρχει, τότε τα περιεχόμενά του καταστρέφονται.
a	Ανοίγει το αρχείο για προσάρτηση. Αν το αρχείο δεν υπάρχει, τότε δημιουργείται νέο αρχείο.
r+	Ανοίγει ένα υπάρχον αρχείο για ανάγνωση και αποθήκευση. Αν το αρχείο δεν υπάρχει, τότε επιστρέφεται κωδικός λάθους.
w+	Δημιουργεί ένα αρχείο και το ανοίγει για ανάγνωση και αποθήκευση. Αν το αρχείο υπάρχει, τότε τα τρέχοντα δεδομένα καταστρέφονται.
a+	Ανοίγει ένα αρχείο για ανάγνωση και προσάρτηση. Αν το αρχείο δεν υπάρχει, τότε δημιουργείται ένα νέο.

Οι τιμές αυτές χρησιμοποιούνται για αρχεία κειμένου (δηλαδή ASCII). Σε περίπτωση που θέλουμε να χρησιμοποιήσουμε δυαδικά (δηλαδή binary) αρχεία, τότε πρέπει να προσθέσουμε το χαρακτήρα b στις προηγούμενες τιμές (για παράδειγμα, rb, w+b, κλπ.).

Κλείσιμο αρχείου

Όταν έχει ολοκληρωθεί η εργασία με το αρχείο, πρέπει αυτό να κλείσει. Για το λόγο αυτό χρησιμοποιείται η συνάρτηση fclose.

- int fclose(FILE *f);

Αν το αρχείο έχει κλείσει με επιτυχία, τότε η fclose επιστρέφει μηδέν. Σε περίπτωση σφάλματος, η τιμή επιστροφής είναι ίση με τη σταθερά EOF, που ορίζεται στο αρχείο stdio.h.

Ανάγνωση και αποθήκευση αρχείων

Δύο είναι οι βασικές λειτουργίες σε ένα αρχείο, η ανάγνωση και η αποθήκευση. Οι βασικές συναρτήσεις είναι οι fread και fwrite, η σύνταξη των οποίων είναι ως εξής:

- size_t fread(void *buffer, size_t size, size_t count, FILE *f);

όπου τα ορίσματα δηλώνουν τα εξής. Η *buffer είναι ένας δείκτης στη μνήμη, όπου η fread αποθηκεύει τα δεδομένα που διαβάζει από το αρχείο, size είναι το μέγεθος του κάθε στοιχείου σε bytes, count είναι ο μέγιστος αριθμός στοιχείων που θα διαβασθούν από το αρχείο, και *f είναι δείκτης στο αρχείο.

- size_t fwrite(const void *buffer, size_t size, size_t count, FILE *f);

όπου και πάλι τα ορίσματα δηλώνουν τα εξής. Η *buffer είναι ένας δείκτης στη μνήμη όπου η fwrite διαβάζει τα δεδομένα που θα αποθηκευτούν στο αρχείο, size είναι το μέγεθος του κάθε στοιχείου σε bytes, count είναι ο μέγιστος αριθμός στοιχείων που θα αποθηκευθούν στο αρχείο, και *f είναι δείκτης στο αρχείο FILE.

Ανάγνωση και αλλαγή θέσης αρχείου

Μερικές φορές είναι χρήσιμο να γνωρίζουμε σε ποιά θέση βρισκόμαστε μέσα στο αρχείο. Η θέση αυτή γίνεται γνωστή με τη βοήθεια της συνάρτησης ftell:

- long ftell(FILE *f);

Αν η εκτέλεση είναι επιτυχής, τότε η ftell επιστρέφει έναν ακέραιο τύπου long, ο οποίος περιέχει τον αριθμό των bytes που η τρέχουσα θέση απέχει από την αρχή του αρχείου. Σε περίπτωση λάθους η ftell επιστρέφει -1.

Για να μετακινήσουμε την τρέχουσα θέση μέσα στο αρχείο χρησιμοποιούμε τη συνάρτηση fseek:

- int fseek(FILE *f, long offset, int origin);

όπου offset είναι η μετατόπιση της νέας θέσης σε bytes από την αρχική θέση, και origin είναι μία σταθερά που δείχνει τη θέση από την οποία υπολογίζεται η μετατόπιση.

Στο αρχείο stdio.h ορίζονται οι ακόλουθες σταθερές που μπορούν να χρησιμοποιηθούν στη θέση της μεταβλητής origin:

- SEEK_SET για την αρχή του αρχείου,
- SEEK_CUR για την τρέχουσα θέση στο αρχείο, και
- SEEK_END για το τέλος αρχείου.

`Αλλες συναρτήσεις

Η συνάρτηση fprintf χρησιμοποιείται για φορμαρισμένη αποθήκευση σε αρχείο κειμένου, όπως ακριβώς χρησιμοποιείται και η printf. Στην περίπτωση της fprintf πρέπει να ορίσουμε σε ποιό αρχείο θα εργασθούμε.

- int fprintf(FILE *f, const char *format_string);

Η συνάρτηση fscanf χρησιμοποιείται για φορμαρισμένη ανάγνωση από αρχείο κειμένου, όπως ακριβώς χρησιμοποιείται και η scanf. Στην περίπτωση της fscanf πρέπει να ορίσουμε σε ποιό αρχείο θα εργασθούμε.

- int fscanf(FILE *f, const char *format_string, ...);

Η συνάρτηση fputs χρησιμοποιείται για την αποθήκευση μίας σειράς χαρακτήρων σε ένα αρχείο κειμένου.

- int fputs(const char *string, FILE *f);

Η συνάρτηση fgets χρησιμοποιείται για την ανάγνωση μίας σειράς χαρακτήρων από ένα αρχείο κειμένου.

- char *fgets(char *string, int maxchar, FILE *f);

Η μακροεντολή feof χρησιμοποιείται για να προσδιορίσουμε αν βρισκόμαστε στο τέλος του αρχείου ή όχι. Αν έχουμε φθάσει στο τέλος αρχείου, τότε η feof επιστρέφει μία μηδενική τιμή, διαφορετικά επιστρέφει μηδέν.

- int feof(FILE *f);

Παράδειγμα

```
include <stdio.h>

main()
{
        FILE *f1;
        FILE *f2;
        size_t size;
        size_t count;
        char string[100];
        struct { int x; int y;} mystruct;

        /* άνοιγμα αρχείου κειμένου myfile.txt για ανάγνωση */
        f1 = fopen("myfile.txt", "r");

/* άνοιγμα δυαδικού αρχείου myfile.bin για ανάγνωση και αποθήκευση */
        f2 = fopen("myfile.bin", "r+b");

        /* αρχικοποίηση */
        size = 1;
        count = 100;

        /* ανάγνωση 100 χαρακτήρων από το f1 */
        fread((void *) string, size, count, f1);
```

```
        /* αρχικοποίηση */
        mystruct.x = 100;
        mystruct.y = 200;
        size = sizeof(mystruct);
        count = 1;

        /* εγγραφή δομής mystruct στο f2 */
        fwrite((void *) &mystruct, size, count, f2);

        /* κλείσιμο αρχείων */
        fclose(f1);
        fclose(f2);
}
```

16.4 Εντολές φυσικού επιπέδου σε C++

Η είσοδος/έξοδος στη C++ μπορεί να γίνει με χρήση **ρευμάτων** (streams). Με τον τρόπο αυτό γίνεται μετατροπή των αντικειμένων σε σύνολα bytes. Η είσοδος/έξοδος σε αρχείο με streams γίνεται με συναρτήσεις της βιβλιοθήκης fstream.h. Στη συνέχεια, αντί του όρου stream χρησιμοποιείται ισοδύναμα ο όρος `αρχείο΄ για λόγους συμβατότητας με τα προηγούμενα.

`Ανοιγμα αρχείου

`Ενα αρχείο ανοίγει για είσοδο με την εντολή:

- ifstream inFile("file_name");

και για έξοδο με την εντολή:

- ofstream outFile("file_name");

Οι κλάσεις ifstream και ofstream είναι απόγονοι της κλάσης fstream. Για χρήση περισσότερων επιλογών κατά το άνοιγμα του αρχείου, δηλώνονται αντικείμενα της κλάσης fstrem με τον ακόλουθο constructor:

- fstream file("file_name", mode);

όπου ο τρόπος ανοίγματος εξαρτάται από την τιμή της παραμέτρου mode, η οποία μπορεί να είναι συνδυασμός με | (με λογικό Η) των εξής επιλογών:

- ios::app
  Άνοιγμα για προσάρτηση, δηλαδή για προσθήκη στο τέλος του αρχείου.

- ios::in
  Άνοιγμα για είσοδο. Αν το αρχείο υπάρχει, τότε δεν καταστρέφονται τα περιεχόμενά του.

- ios::out
  Άνοιγμα για έξοδο. Αν το αρχείο υπάρχει, τότε καταστρέφονται τα περιεχόμενά του και γράφονται τα νέα.

- ios::trunc
  Άνοιγμα με καταστροφή των περιεχομένων του αρχείου. Μπορεί να συνδυασθεί, για παράδειγμα με το ios::in (ios::in | ios::trunc) για άνοιγμα για είσοδο με ταυτόχρονη καταστροφή των περιεχομένων του αρχείου.

- ios::nocreate
  Δηλώνει ρητά άνοιγμα χωρίς δημιουργία αρχείου σε περίπτωση που αυτό δεν υπάρχει (ταυτόχρονα προκαλεί και λάθος).

Τέλος, μία βασική επιλογή κατά το άνοιγμα του αρχείου αφορά στο αν αυτό είναι αρχείο κειμένου ή δυαδικό αρχείο. Αυτό γίνεται με την τιμή της παραμέτρου mode:

- ios::binary

η οποία δηλώνει δυαδικό αρχείο, ενώ αν δεν υπάρχει η τιμή, τότε πρόκειται για αρχείο κειμένου.

Κλείσιμο αρχείου

Το κλείσιμο ενός αρχείου γίνεται με την εντολή:

- file.close();

Ακόμη και αν δεν υπάρχει η ρητή εντολή κλεισίματος, το αρχείο κλείνει από το destructor του.

Ανάγνωση και αποθήκευση σε αρχεία

Για την ανάγνωση ενός χαρακτήρα από αρχείο χρησιμοποιείται η εντολή:

- file.get(ch);

όπου ch είναι μεταβλητή τύπου char. Αν συμβεί λάθος κατά την ανάγνωση, τότε επιστρέφει NULL. Για την ανάγνωση συγκεκριμένου αριθμού bytes υπάρχει η εντολή:

- file.read(pch, nCount);

όπου pch είναι μεταβλητή τύπου char*, ενώ η μεταβλητή nCount είναι τύπου int και δηλώνει τον αριθμό των bytes. Η εντολή read χρησιμοποιείται κυρίως για δυαδικά αρχεία.

Για την αποθήκευση ενός χαρακτήρα σε αρχείο υπάρχει η εντολή:

- file.put(ch);

όπου ch είναι μεταβλητή τύπου char. Αν συμβεί λάθος κατά την αντιγραφή, τότε επιστρέφει NULL. Για την αποθήκευση συγκεκριμένου αριθμού bytes υπάρχει η εντολή:

- file.write(pch, nCount);

όπου pch είναι μεταβλητή τύπου char* και η μεταβλητή nCount είναι τύπου int και δηλώνει τον αριθμό των bytes. Και η εντολή write χρησιμοποιείται κυρίως για δυαδικά αρχεία.

Για τον έλεγχο αν ο δείκτης του αρχείου βρίσκεται στο τέλος του αρχείου, χρησιμοποιείται η εντολή:

- file.eof( );

η οποία επιστρέφει μία μη μηδενική τιμή, αν ο δείκτης είναι στο τέλος του αρχείου.

Τυχαία αναζήτηση σε αρχεία

Για τη μετακίνηση του δείκτη σε αρχείο, που είναι αντικείμενο της κλάσης ifstream (δηλαδή, αρχείο για είσοδο), υπάρχει η εντολή:

- seekg(offset, dir);

Αν το αρχείο είναι αντικείμενο της κλάσης ofstream (δηλαδή, αρχείο για έξοδο), τότε υπάρχει η εντολή:

- seekp(offset, dir);

Η παράμετρος offset είναι τύπου streamoff, που είναι ισοδύναμος με τον τύπο long. Η παράμετρος dir και στις δύο συναρτήσεις δηλώνει την κατεύθυνση προς την οποία γίνεται η αναζήτηση και μπορεί να πάρει τις εξής τιμές:

- ios::beg
  Εκτελεί αναζήτηση από την αρχή του αρχείου.

- ios::cur
  Εκτελεί αναζήτηση από την τρέχουσα θέση στο αρχείου.

- ios::end
  Εκτελεί αναζήτηση από το τέλος του αρχείου.

Η τιμή του δείκτη αρχείου γίνεται για τα αρχεία κλάσης ifstream με την εντολή:

- filePos = tellg();

ενώ για τα αρχεία κλάσης ofstream με την εντολή:

- filePos = tellp();

Η μεταβλητή filePos είναι τύπου long.

Παράδειγμα

```
#include <fstream.h>
#include <iostream.h>                    // για μηνύματα στην οθόνη

void main()
{
        /* Αντιγραφή των περιεχομένων ενός αρχείου σε άλλο */
        ifstream source("inputFileName");
        ofstream destination("outputFileName");


        char ch;
        while (source.get(ch))
                destination.put(ch);

        if (!source.eof())
                cout<<"Λάθος κατά την αντιγραφή";

        /* Ο βρόχος while εκτελείται όσο η συνάρτηση get δεν επιστρέφει
        τιμή NULL. Μετά το τέλος του βρόχου γίνεται έλεγχος αν το αρχείο
        έχει φτάσει στο τέλος με τη συνάρτηση eof. Αν δεν συμβαίνει
        αυτό τότε προκαλείται λάθος, γιατί αυτό σημαίνει ότι δεν αντιγρά-
        φηκαν όλα τα περιεχόμενά του */

        /* Ανάγνωση περιεχομένων αρχείου και προβολή των θέσεων στις
        οποίες υπάρχουν κενά */

        ifstream tfile("text");

        while ( !tfile.eof() )
        {
                long here = tfile.tellg();
                tfile.get(ch);
                if (ch == " ")
                        cout << "\nPosition " << here << " is a space";
        }
}
```